DIGITAL DATA PROCESSING AND COMPUTATIONAL DESIGN FOR LARGE AREA MASKLESS PHOTOPOLYMERIZATION

A Thesis
Presented to
The Academic Faculty

by

Anirudh V. Rudraraju

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mechanical Engineering

Georgia Institute of Technology
December 2013

DIGITAL DATA PROCESSING AND COMPUTATIONAL DESIGN FOR LARGE AREA MASKLESS PHOTOPOLYMERIZATION

Approved by:

Dr. Suman Das, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. David Rosen
School of Mechanical Engineering
Georgia Institute of Technology

Dr. John Halloran
School of Material Science
University of Michigan

Dr. Yan Wang
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Suresh K Sitaraman
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Massimo Ruzzene
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: July 31, 2013

To my family.

ACKNOWLEDGEMENTS

First and Foremost, I would like to thank my advisor, Dr. Suman Das, for his constant support through out the tenure of my PhD. Without his steadfast support, guidance and vision, this thesis would be not be possible. I would also like to extend my sincerest gratitude to my reading committee, composed of Dr. John Halloran, Dr. David Rosen, Dr. Suresh K Sitaraman, Dr. Yan Wang and Dr. Massimo Ruzzene, whose inputs have been very valuable in improving the quality of this thesis.

Secondly, I would like to thank my fellow Direct Digital Manufacturing group members, both former and current, Dr. Dajun Yuan, Dr. Siddharth Ram Attreya, Dr. Chan Yoon, Kiran Kambly, Tao Wu, Raphael Alabi, Mathew Conrad, Rohan Bansal, Shaun Eshragi and Justin Gambone. I have developed and learned a lot, both personally and professionally, through their camaraderie over the long span of my graduate studies. I would like to especially thank Ranadip Acharya, a fellow group member, for extending his expertise on Ansys CFX and helping me set up the film cooling cooling simulations. I would also like to extend my gratitude to Hassan Rashidi, also a fellow group member, for helping me prepare LAMP suspensions with varying compositions used in the cure width studies.

Thirdly, I would like to thank all the group members at the University of Michigan-Ann Arbor, especially Dr. Vladka Tomeckova and Dr. Susan Gentry, who were working on the LAMP suspension formulation and the firing process of green ceramic parts. Meetings with them from time to time helped me gain a better understanding of the comprehensive scope of the LAMP project. I would also like to thank the sponsor, Dr. Bill Coblenz at the DARPA Defense Sciences, whose constant support ensured financial assistance for the majority of my graduate studies thereby allowing

me to focus on research in a stress free manner.

Lastly, I would like to express my sincerest gratitude to my family, for their endless patience and support through many trials and tribulations over the course of my study which helped me keep focused on the goal. I would also like to thank the many friends I had made at Georgia Tech, whose fun filled company had helped me manage the rigors of graduate studies and made the process so much more enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Large Area Maskless Photopolymerization (LAMP) is a novel layer-based manufacturing technology currently being developed at Georgia Tech in collaboration with the University of Michigan-Ann Arbor and PCC Airfoils. It is intended for the fabrication of integrally-cored ceramic molds, with complex internal geometries, that are used in the investment casting of high-pressure turbine blades. Unlike most layer-based manufacturing technologies that produce prototype parts in plastic, LAMP is developed with the goal of producing functional ceramic components that can withstand the rigors of the high temperature processes involved in the single-crystal casting of turbine blades. The complex internal geometries and the stringent requirements on the physical properties of the parts to be produced poses several challenges. Consequently, the LAMP technology development initiative is a comprehensive one involving work in several research domains to address these challenges. The work presented in this dissertation addresses the digital data processing and computational needs of the LAMP process.

Several data processing schemes have been developed as part of this dissertation to enable the LAMP technology. STL files, which are meshed approximations of the part geometry, have become the *De facto* standard of the additive manufacturing (AM) industry. However, owing to the complex part geometries in LAMP, such an approximation of the geometry is not affordable. Therefore, a novel error-tolerant direct slicing approach using ACIS kernel has been developed to slice the native CAD geometry and output extremely high resolution (1500 dpi) bitmap images of the slice contour. STL file slicing algorithms that are faster than the approaches previously

presented in literature have also been implemented. Furthermore, a suite of post processing algorithms, like error-checking, part placement and tiling etc. that work on the slice image data have been implemented.

In addition to the data processing schemes that enable basic functionality of the LAMP process, several computational schemes to further improve the part quality have been presented in this dissertation: (1) A novel volume-deviation based approach for adaptively slicing CAD models has been implemented to alleviate the "stair-stepping" effect on parts produced through LAMP and other AM processes in general. The approach presented here is less computationally demanding than the methods presented in the literature while producing reasonably good results. (2) A gray-scaling and dithering scheme has been implemented on the slice images as yet another means to alleviate the stair-stepping effect. This approach takes into account, the effects of gray scale factor on the curing characteristics of the material system when computing gray scale intensities unlike previous approaches. However this approach only works in reducing the stair-stepping effect on surfaces with a downward facing (w.r.t the build direction) normal vector. (3) An experimental study to characterize the side curing behavior of the LAMP suspension with the objective of slice image compensation prior to build in order to produce the dimensionally accurate parts is also presented.

A new approach for supporting geometries that result in unsupported features or "floating" islands during part builds is presented. The approach presented here works on native CAD geometry while the previously presented methods mostly work on the simpler STL meshes. Moreover, prior approaches cannot be applied to the LAMP process due to the inability to remove these support structures after build completion. Consequently, a different approach that is suitable to the needs of LAMP process is implemented. Since the implanted supports cannot be removed post-build, a potential multi-functional design strategy for meeting the dual objectives of successful part builds as well good cooling performance is discussed.

Finally, a discussion on the impact of LAMP in the manufacturing of next-generation turbine blade designs is presented. Many novel cooling schemes to improve the performance of turbine blades in high temperature engine operating conditions have been proposed in the literature. However, it is extremely challenging to fabricate them using current manufacturing methods. The LAMP process with its layer-by-layer approach can alleviate these manufacturing limitations thereby opening up doors for new design possibilities. To demonstrate these advanced capabilities, a few novel film cooling hole geometries, that are difficult to fabricate conventionally are analyzed for their cooling performance and fabricated through the LAMP process. Details of the performance analysis of these geometries and the fabrication results are presented.

CHAPTER I

INTRODUCTION

Large Area Maskless Photopolymerization (LAMP) is a novel Additive Manufacturing (AM) technology that is being developed at Georgia Tech in collaboration with the University of Michigan-Ann Arbor and PCC airfoils. LAMP technology is aimed at disrupting the current state-of-the-art in investment casting of high pressure (HP) turbine blades by alleviating the need for tooling and manual labor. In doing so, it leads to significant cost savings and substantial reduction in lead times for fabrication of new prototype designs. Moreover, it also empowers designers with more freedom by eliminating the constraints that are associated with conventional manufacturing techniques. The work presented in this thesis addresses some of the challenges encountered in the development of this technology.

LAMP is an AM process, among many others, that intends to make functional ceramic components. It has many novel features, partly due to the contributions presented in this thesis, that make it superior to some of the existing AM technologies and enable it in addressing the challenges associated with fabricating complex ceramic molds for casting HP turbine blades. An introduction to the various classes of AM processes in existence is given in Section 1.1. A survey of relevant literature in AM of Ceramics is also included. The limitations of these processes and the advantages of LAMP over these processes are also discussed.

A component of the work presented in this thesis is intended at leveraging LAMP technology to build novel airfoil designs that are currently extremely challenging to fabricate using conventional manufacturing techniques. Consequently, in order to give a better appreciation of the limitations of conventional manufacturing techniques,

a brief overview of the current state of the art in airfoil design and manufacturing is given in Section 1.2. Following the discussion on the current state-of-the-art, an overview of the various advantages of LAMP over conventional manufacturing techniques and the possibilities that LAMP opens up for advanced designs are also discussed.

1.1 Direct Digital Manufacturing

Direct Digital Manufacturing (DDM) or AM is a method of building three dimensional parts from their digital representations by "growing" them one layer at a time. There are several different AM techniques in existence which can be broadly classified based on the techniques they use to grow the 3D parts into the following types : extrusion-based, granular-based, lamination-based and photopolymerization-based techniques. Extrusion-based techniques use a moving head that extrudes material onto to a substrate to build each layer. Granular-based techniques start with a bed of powder and use either heat, radiation or a binder to define each layer in the bed of powder. Lamination-based techniques use conventional machining techniques to make each of the layers which are then bonded to each other to get the three dimensional object. Lastly, photopolymerization-based techniques use a light source to form layers in photo-curable resins.

Within each of these broad classifications there are several technologies and some of the most popular of these are shown in Figure 1.

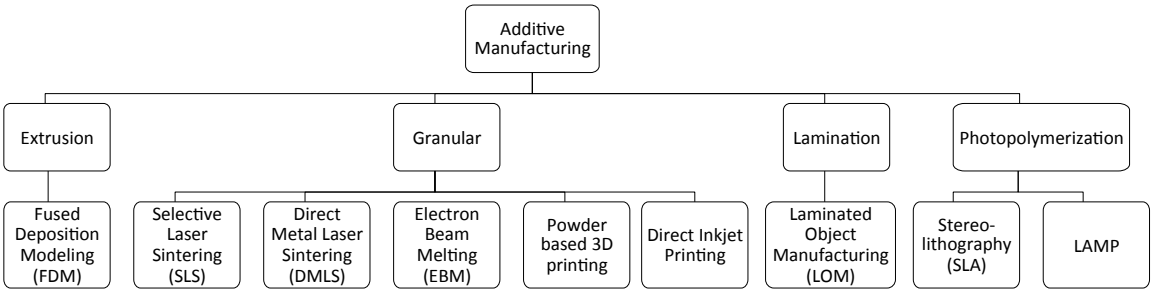


Figure 1: Various AM processes.

As shown in the figure, LAMP uses a photopolymerization based technique to produce three dimensional ceramic parts.

1.1.1 Literature Review

There have been several previous efforts in the past to build ceramic parts within each of the various classes of AM technologies . A brief review of these works is given in this section. Citations to these works are segregated based on the specific class of AM technique used and presented accordingly.

1.1.1.1 Extrusion:

In extrusion-based techniques, an extrusion head (also known as liquefier) is used in order to deposit each layer of the part onto a substrate. The basic schematic of an extrusion-based technique called fused deposition modeling (FDM) is shown in Figure 2.

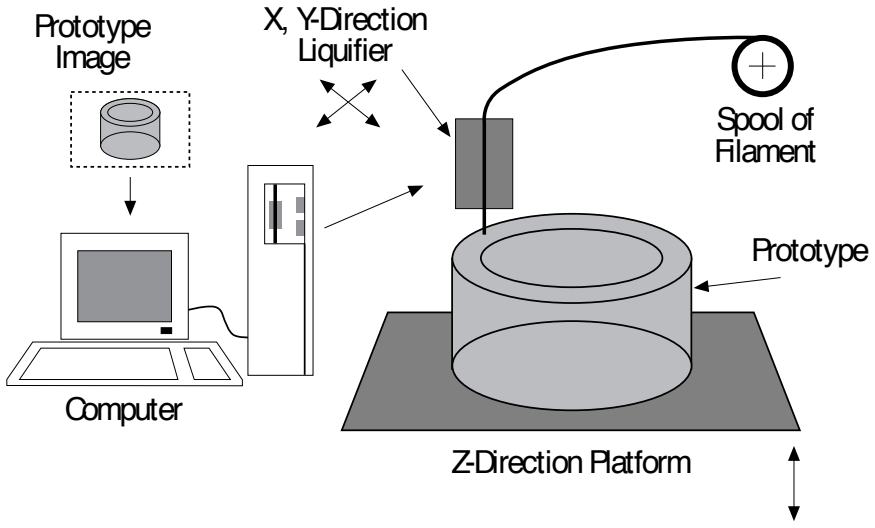


Figure 2: Schematic of the Fused Deposition Modeling Process [30].

In this process, a digital representation of the part is fed to a controlling computer where it is processed by various algorithms to extract the cross-sectional information of each layer into a format that can be fed to the extrusion head. A spool of filament

(usually of a polymer) is fed to the head where it is melted using heating elements which are embedded in the head. The molten material is then extruded on to a substrate according to the cross-sectional information received from the computer to deposit each layer. Once a layer is deposited, the platform moves down and subsequent layers are deposited. By repeating this process for every layer of the part, a three-dimensional object is grown.

One of the earliest variants of this process called the fused deposition of ceramics (FDC), intended at producing ceramic parts, was proposed by Agarwala et al. [31]. In their process, a mixture of *in-situ* reinforced silicon nitride (Si_3N_4) powder and polymer or wax based binder systems was used as the filament material instead of a conventional polymer filament in the FDM process. Using this filament, a "green" ceramic part is obtained by the FDM process which is a composite of ceramic and polymer. This part is then taken through a binder burnout process in order to remove the polymer binder and a sintering process to obtain the final component. Subsequently, the process has been extended to handle silica, piezoelectric and alumina powders to produce parts of various kinds of ceramic materials [32, 33]. Efforts on using this process as well as an indirect approach to deposit piezoelectric ceramics like lead zirconium titanate (PZT) have been reported by Safari et al. [30, 34, 35]. In the indirect approach, a polymer component, which is a negative of the intended ceramic component design, is first fabricated and the ceramic part is obtained via lost mold technique. Extensive work (on both direct and indirect extrusion techniques) focused on making controlled porosity ceramic structures used for various applications such as photonic crystals, sensors, tissue engineering scaffolds etc. has also been reported [36–41]. Safari et al. reported a multi ceramic deposition process which can deposit upto four different ceramic materials simultaneously onto the same substrate [42, 43].

1.1.1.2 Granular:

Direct ink jet printing technologies are a major class of granular-based techniques. One kind of direct ink jet printing technologies start with a bed of powder. A binder is then selectively deposited in order to bond the powder to make each layer. This process is repeated layer after layer to get a green part which is then sintered to obtain the final ceramic component. Another kind of direct 3d printing technologies print droplets of ink, which is usually either a wax-based or a solvent-based ceramic suspension, directly onto a substrate. Wax-based suspensions are deposited onto a cold substrate where the wax solidifies to form the layer whereas in solvent-based techniques, the solvent evaporates and leaves a ceramic residue to form each layer. The parts thus obtained are then sintered to yield the final component. Blazdell et al. reported some of the earliest work in solvent-based ink jet printing of zirconia parts [44]. Several efforts to deposit various ceramic materials like alumina and PZT using the solvent based ink jet process have been reported [45–51]. Reis et al. reported a wax-based alumina suspension ink jet printing process [52–54].

Laser sintering-based technologies are, apart from direct ink jet printing, the other major class of granular based techniques. A schematic of a typical laser sintering technique called SLS (Selective Laser Sintering) is shown in Figure 3. As shown in the schematic, these kinds of processes start with a bed of powder (usually metal or polymer) that is placed on a moving piston. A laser beam, scanning over the powder bed according to the paths fed from the CAD model cross-sectional information fuses the powder particles to form each layer. Once a layer is formed, a roller delivers fresh powder from the delivery system to deposit a new layer. This process is repeated until the all the layers of the object have been created to obtain the final component. Several works reported in the literature have extended this process to make ceramic parts. Subramanian et al. [56] and Griffin et al. [33] used a ceramic powder and a polymer binder mixture to produce "green" parts which are then heat-treated in order

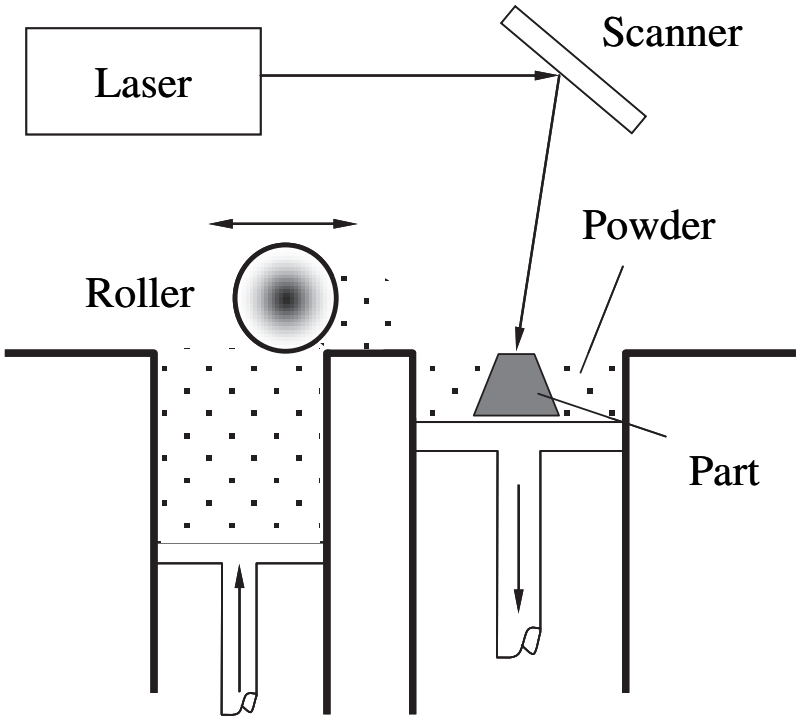


Figure 3: Schematic of the Selective Laser Sintering Process [55].

to burn out the binder and sinter the ceramic powder to obtain final ceramic parts. Heinrich [57], Regenfuß [58, 59] demonstrated the use of this process on alumina-silica mixtures. Coulon and Aubry [60] and Tang [61] reported on a direct solid-state laser sintering process for alumina powder without the use of any polymer binders. Bertrand et.al [62] reported on a similar binder-less process for zirconia. Shishkovsky et al. [63] and Wilkes et al. [64] reported a laser melting process for a eutectic alumina-zirconia mixture.

1.1.1.3 Lamination:

Lamination-based technologies are a hybrid between additive and subtractive methods. A schematic of laminated object manufacturing (LOM), which is one of the most popular technique in this category, is shown in Figure 4.

As shown in the schematic, a sheet of material, usually paper, is fed between two rollers onto a platform. The scanner guides a laser beam across the sheet of material

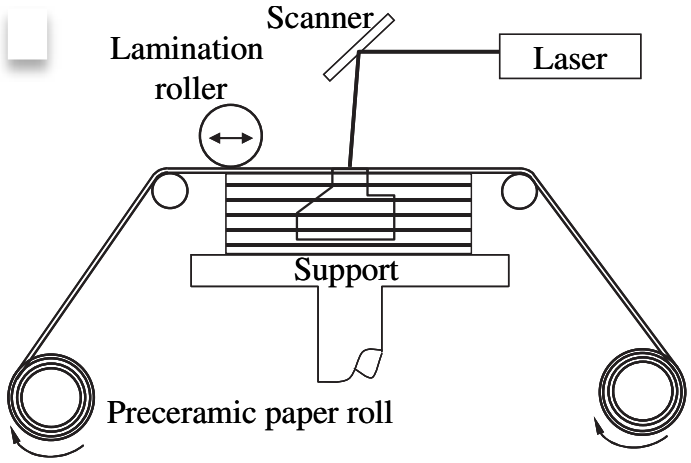


Figure 4: Schematic of the Laminated Object Manufacturing (LOM) Process [65].

based on the cross-sectional information of the CAD model in order to cut out the required contour. Once the cross-section is cut out on the sheet of material, it is then bonded onto the previous layers on the platform using a thermoplastic adhesive which is activated by heat or pressure applied by the lamination roller. This process is repeated to make and bond all the layers of the part successively in order to get a three-dimensional component. Several researchers have attempted to extend this technique to process ceramics instead of paper as in the conventional process. Daufenbach et al. [66] and Griffin et al. [67, 68] at Lone Peak Engineering Inc. replaced the paper with sheets of alumina to produce ceramic alumina parts. Klosterman et al. [69] proposed a novel curved layer laminated object manufacturing process for making silicon carbide parts as well as a flat layer process for the same material system [70]. Weinsel et al. [71] reported on using LOM for fabrication of dense biomorphous silicon-silicon carbide structures. Travitzky et al. reported on the development of preceramic paper-derived silicon-silicon carbide and alumina-silicon carbide papers for use in LOM [65, 72]. Schindler and Roosen [73] proposed a low pressure, low temperature lamination process (unlike the high temperatures and pressures used in conventional LOM) that uses commercially available low temperature

co-fired ceramic green tapes for making various 3-dimensional ceramic devices.

1.1.1.4 Photopolymerization:

Photopolymerization-based technologies start with a vat of photo-curable Suspension and form each cross-sectional layer by polymerizing the Suspension with a light source. A schematic of the Stereolithography (STL) process, which is one of the most popular technologies in this genre, is shown in figure 5.

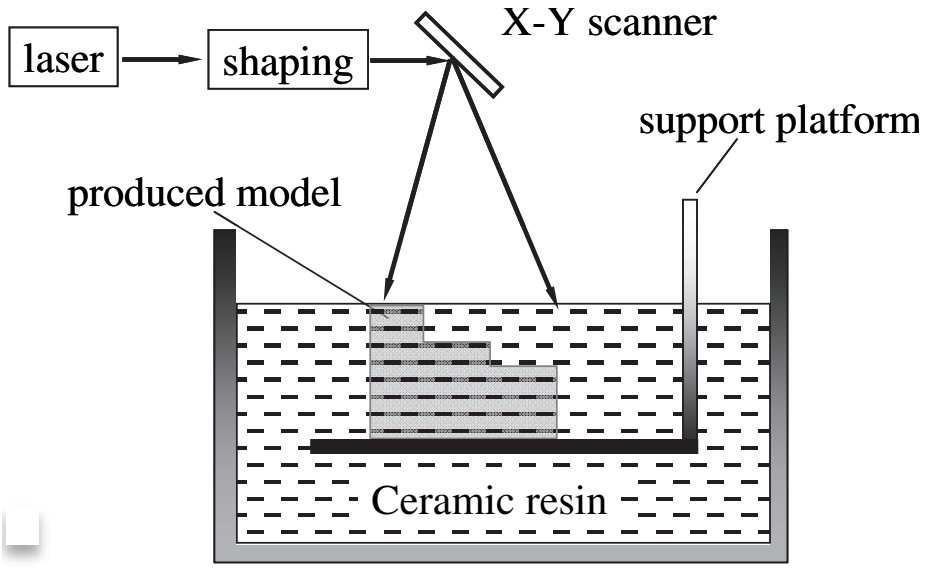


Figure 5: Schematic of the Stereolithography (STL) Process [74].

As shown in the schematic, the process starts with an "L-platform" dipped inside a vat of photo-curable Suspension (usually polymer). A laser beam, guided by the scanner mirror according to the cross-sectional information provided from the CAD part, scans over the Suspension to expose and solidify the desired regions. Once the layer is exposed, the platform lowers down into the vat by a distance equal to the layer thickness being used in the build and a re-coating arm (not shown in the figure) sweeps over in order to lay down a uniform layer of fresh Suspension for the next layer to be exposed. This process is repeated until all the layers along the height of the part are cured to obtain the final 3-dimensional component. Several works

that extend this process to make ceramic parts instead of plastic ones were reported in the literature. Griffith and Halloran [74] were one of the first to use ceramic powder-loaded photo-curable resin suspensions to make ceramic parts. These ceramic suspensions are used in the stereolithography process to make "green" parts which are then taken through a binder burnout and sintering cycle in a furnace to produce fully functional ceramic parts. In their work, Griffith and Halloran [74] reported results with silica, alumina and silicon nitride suspensions for making investment casting molds and structural ceramic parts. Chartier et al. [75] and Hinczewski et al. [76] have reported work on ceramic stereolithography with alumina suspensions. Jang et al. [77] reported on the preparation and characterization of barium titanate suspensions for use ceramic stereolithography. Delhote et al. [78] reported work on incorporating high performance ceramics like BZT ($Ba_3ZnTa_2O_9$) suspension for making high quality resonant structures and band pass filters. Zhang et al. [79] reported a ceramic SLA process for making micro-sized alumina parts. Bertsch et al [80] used an integral stereolithography process, where an entire area is exposed using a spatial light modulator instead of raster scanning line by line with a laser, to make micro-sized alumina parts thus greatly improving the speed of the process.

1.1.2 Limitations

As discussed in the previous section, while several AM processes have been proposed for building ceramic parts, they have various limitations. These limitations are even more apparent especially in light of the challenges of making ceramic investment casting molds for casting high-pressure turbine blades. An overview of these various limitations is given in this section.

Speed is a severe limitation in many of the technologies as they use a laser beam to vector scan, i.e, scan point-by-point or line-by line (or equivalently a material deposition head to deposit material line by line) to pattern each layer. The technology

aimed at producing investment casting molds must be capable of high throughput (i.e, make multiple parts on a large build area) and it would take a very long time to be able to achieve this by vector scanning.

Scalability is another major limitation of many of these technologies. Many of the processes discussed in the previous section are tailored to work best at specific length scales. Processes are designed carefully depending on whether the length scales of the parts they intend to produce are macro or micro sized. High pressure turbine blades have overall outer dimensions which are in the macro domain (in the order of inches) while consisting of minute internal features with micro sized dimensions (in the order of a few hundred microns). Such a broad range of length scales within the parts coupled with the large areas of exposure for making multiple parts in a single build poses severe challenges in many of these technologies. For instance, in stereolithography or selective laser sintering, a galvanometer mirror is used to position the laser for writing each layer. As the build area becomes larger and larger, the minimum feature resolution that these technologies can make goes down drastically as the mirror's discrete number of positions will be mapped over larger and larger areas. Simultaneously, the minimum achievable focal spot size goes up thereby further decreasing resolution.

The material system used in these technologies is also not entirely suited for the investment casting of HP turbine blades. HP turbine blades are cast using a special high temperature (upwards of 1500^0C) casting process with controlled temperature gradients to obtain single crystal castings for high creep strength. These huge temperature gradients place enormous stresses on the ceramic molds and consequently they have to have the right amount of crystallinity and other flexural and mechanical properties to withstand the metal process. Any process intending to make these molds must take into account these demands on mechanical properties while designing the material system.

Lastly, the need for support structures for successfully building geometries that lead to unsupported or "floating" islands, cripples many of these AM techniques. As parts are grown layer by layer, due to the nature of the geometry, new features grow after some layers into the build which dont have any cured region underneath to support them. A more detailed explanation of this problem and the need for support structures is given in Chapter 4. In order to tackle this problem, many technologies grow extra support features along with the part which are then removed during post processing after the build. However, in the case of integrally cored molds for investment casting, all the core features are completely encapsulated within the outer shell, removal of support structures post build is not possible. Hence, any process that is proposed for this application needs to address this challenge.

LAMP technology, being developed specifically for the purpose of investment casting of HP turbine blades, addresses many of these challenges. An overview of the basic process and the advantages of LAMP over other ceramic forming technologies specifically for the purposes of this application is given in the following sections.

1.1.3 LAMP Process

A schematic of the LAMP process is shown in Figure 6. LAMP process starts with the material re-coating system spreading a fresh layer of ceramic powder loaded photocurable resin suspension onto the build platform. A UV light source, in conjunction with a Digital Micro-Mirror device (DMD), is used to cure the suspension selectively to make each layer of the part. The first few layers are cured into a thin mesh that provides breakable support to the parts being built on top of them. A Digital Micro-Mirror device (DMD) is a micro-electro-mechanical device that consists of several tiny mirrors than can be actuated independently. A controlling computer processes the CAD geometry of the desired part and feeds the cross-sectional information to the DMD chip in the form of high-resolution bit map images. From the image information,

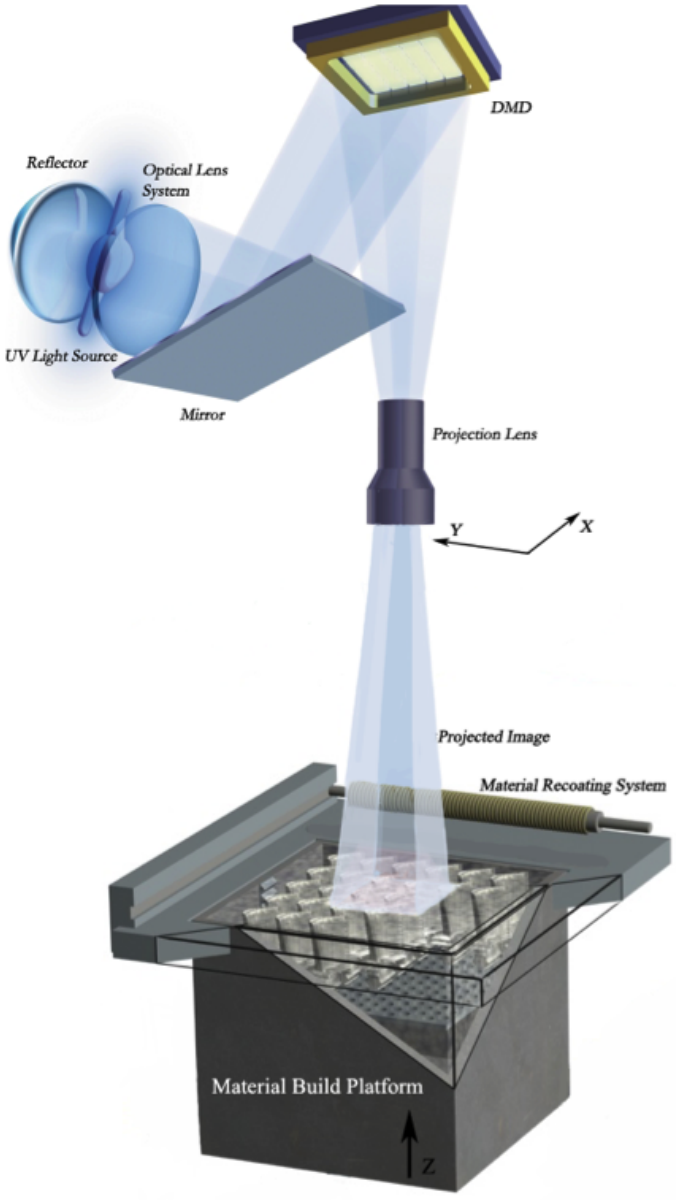


Figure 6: Schematic of the LAMP Process.

the DMD chip actuates each of the several mirrors accordingly to project the image on to the photocurable resin surface to cure the required shape of the cross-section. The UV light source, the DMD chip and the projection optics are mounted on a gantry system which enables the patterning of areas much larger than the projection area of a single DMD exposure. Once each layer is exposed properly, the platform moves down by the layer thickness (typically 100 microns but can be made arbitrarily

low) and the material re-coating system lays down a fresh layer of suspension for the exposure of next layer. This process is repeated for all the layers along the height of the part to be built in order to obtain the "green" ceramic components. These green parts are then taken through a binder burnout and sintering cycle during which they are fired in a furnace using controlled temperature ramps to eliminate the polymer binder and sinter the ceramic to final desired density, crystallinity and mechanical properties suitable for investment casting. Upon completion of the sintering cycle, the ceramic molds are ready for mold cluster assembly and casting.

1.1.4 Advantages

Since LAMP technology has been developed with the specific intent of making ceramic molds for investment casting of HP turbine blades, it has features embedded in it that addresses many of the challenges discussed in Section 1.1.2. This makes it the most suitable candidate, out of all the various AM technologies reported in the literature, for this application.

The limitations of speed and throughput are addressed by the use of integral projection method. Since LAMP uses the DMD chip as a spatial light modulator, it exposes entire areas instead of raster scanning line-by-line like the previously reported technologies. Although, some techniques presented in the literature used an integral projection system (eg. Bertsch et al. [80]), LAMP advances this one step further by encapsulating the projection optics on a gantry system. This makes LAMP highly scalable to larger build areas without any loss of the minimum feature size that can be resolved. Also, due to fact that the DMD chip used consists of extremely tiny micro-mirrors of the order of $17\mu m$, feature resolutions much higher than conventional AM techniques can be obtained. Moreover, the material system is carefully designed in order to ensure that the molds obtained possess the mechanical properties demanded

by the casting process. Additionally, as will be discussed in Chapter 4, a multi-functional design approach is utilized to design the support structures to address the issue of floating islands during the build. The supports thus generated, not only aid in the successful fabrication of the part, but also aid in enhancing the cooling performance of the design or at the very least do not adversely affect it.

1.2 Airfoils Design & Manufacturing

The LAMP process, apart from alleviating the limitations of the current state-of-the-art in investment casting of HP turbine blades, is ultimately intended to push the envelope in terms of what is possible in the design and fabrication of next-generation of blade designs. To place the potential of LAMP into perspective, a brief overview of the current state-of-the-art in the design and fabrication of HP blades is given in this section.

1.2.1 Typical HP Turbine Blade Designs

Gas turbines are the preferred source of power generation for many aerospace, marine and land based propulsion systems. Stringent demands are placed on their efficiency and thrust-to-weight ratio especially in the aerospace sector since every additional pound of weight costs several million dollars to ferry around over the life span of an aircraft. Recently, more and more land based gas turbine manufacturers are also increasingly working towards improving their efficiencies in order to save money and energy. As a result there has been a tremendous push in the last couple of decades towards exploring all avenues where efficiency can be improved.

Gas turbines operate on Brayton cycle. They take in large volumes of air, compress it to high pressures and temperatures, mix it with fuel and ignite the air-fuel mixture in order to extract power. One of the important parameters in the thermodynamic equations determining efficiency and specific power output (power output per unit mass flow rate) is the temperature difference that can be achieved across

the engine. Hence, higher thermodynamic and operating efficiencies can be achieved by making the operating temperatures higher. As a consequence, there has been a huge surge in the turbine inlet gas temperatures over the years to the point that in contemporary engines, it is much higher (about 600-800^0C higher!) than the melting point of the materials used to make the turbine blades themselves. Although, increasing the operating temperatures leads to higher operating efficiencies, it poses severe challenges. Turbines consist of several hundreds of turbine blades that rotate at very high speeds (on the order of 10000+ rpm) on a series of coaxial disks. Operating stresses are very high and even marginal increases in operating temperatures leads to substantial reductions in the blade life. Moreover, considering the fact that a typical gas turbine engine is expected to operate for long periods of time with minimal maintenance (typical 30 years!), such inordinate increases in operating temperatures have pushed the boundaries of blade design in order to keep up with the operating life expectations. Thus in modern turbine engines, it is imperative to incorporate efficient and innovative cooling schemes to meet any reasonable expectations on reliability and life expectancy.

Simple cooling strategies first began to take shape around the 1960s which gave way to further sophistications in the 80s and the 90s as the entry gas temperatures began to rise. The schematic of a typical state-of-the-art turbine blade with cooling schemes is illustrated in Figure 7. Turbine blade cooling schemes can be broadly classified into two categories: internal and external cooling. The air used for turbine airfoil cooling is bled from the compressor outlet. Internal cooling is achieved by circulating this cooling air through the internal passages of the blades as shown in Figure 7. The mid-span of the airfoil usually has channels decorated with ribs (or trip strips) along the passage in order to trip the boundary layer into a turbulent state thereby increasing the convective heat loss from the surface. The internal wall of the leading edge is usually cooled by impinging some of this air through holes cut in the

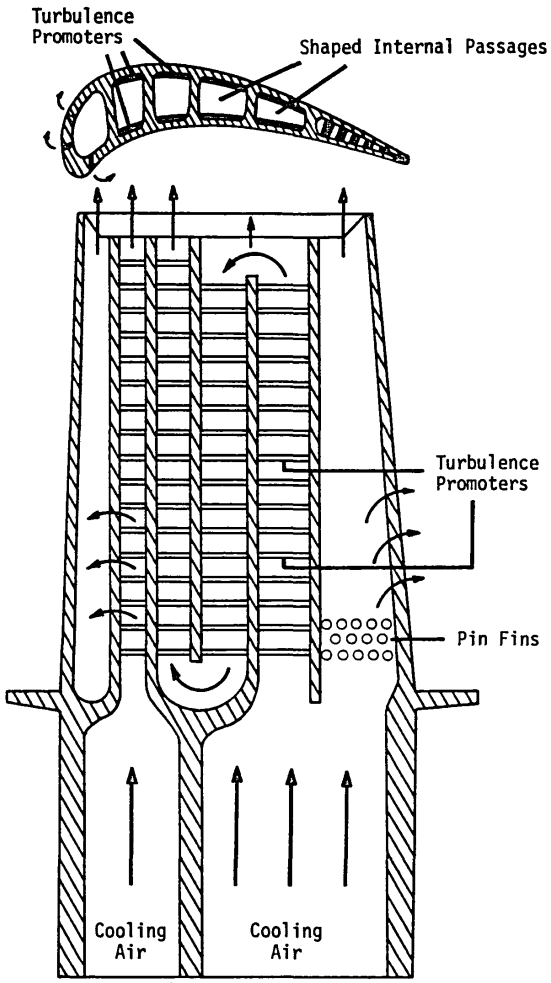


Figure 7: Schematic of a typical internally cooled turbine blade [81].

walls of these passages. The trailing edge is usually too narrow to accommodate any such passages and hence has dimpled surfaces (or surfaces with pin fins) to enhance the heat transfer as cooling air is bled through them and often out through slots in the trailing edge. External cooling is usually achieved by bleeding some portion of the cooling air through holes drilled on the outer wall of the turbine blade such that the coolant can form a conformal film on the external blade surface thereby protecting it from the hot gases and prolonging its life. Yet another popular, but not so prevalent, cooling scheme for achieving external cooling is via transpiration. The outer wall of the blade is made with a fine wire mesh through which the cooling air can transpire

thereby cooling it from the hot gases. However, constraints in terms of structural rigidity and blockage of the fine pores by particles in the coolant flow have prevented this technology from widespread usage. Thermal barrier coatings are also used to provide heat resistance on the external surface.

The surge in inlet gas temperatures in the quest for higher efficiencies has clearly made the design of HP turbine blades very complex. Manufacturing these complex designs is a challenge. A brief overview of the current state of the art in the manufacturing process of these blades is given in the next section.

1.2.2 State-of-the-Art in HP Turbine Blade Manufacturing

HP turbine blades are typically made of nickel-based "superalloys" and other exotic metals which cannot be mechanically worked. Hence, the primary process for fabricating these blades with complex internal cooling passages is investment casting. Investment casting, or lost-wax casting, is one of the oldest known metal forming techniques. Archaeologists have dated objects that were investment cast all the way back to between 3000 B.C. The process gets its name due to the fact that it uses sacrificial wax patterns that are invested in ceramic suspension for mold preparation. The aerospace industry forms the single biggest sector in the investment casting industries capturing up to 65% of the market according to some studies [82]. As the turbine blade designs have become increasingly complex, turbine blade manufacturers have had to constantly evolve and improve the process to be able to reproduce the designs accurately in the cast parts.

A schematic of the current state-of-the-art investment casting process for casting HP turbine blades is shown in Figure 8. As shown in the figure, the first step in the process is injection molding of the ceramic cores. Special dies and tools are used to fabricate the core patterns which form the negative geometry of the desired internal cooling structures of the turbine blades. Once the cores are fabricated, they

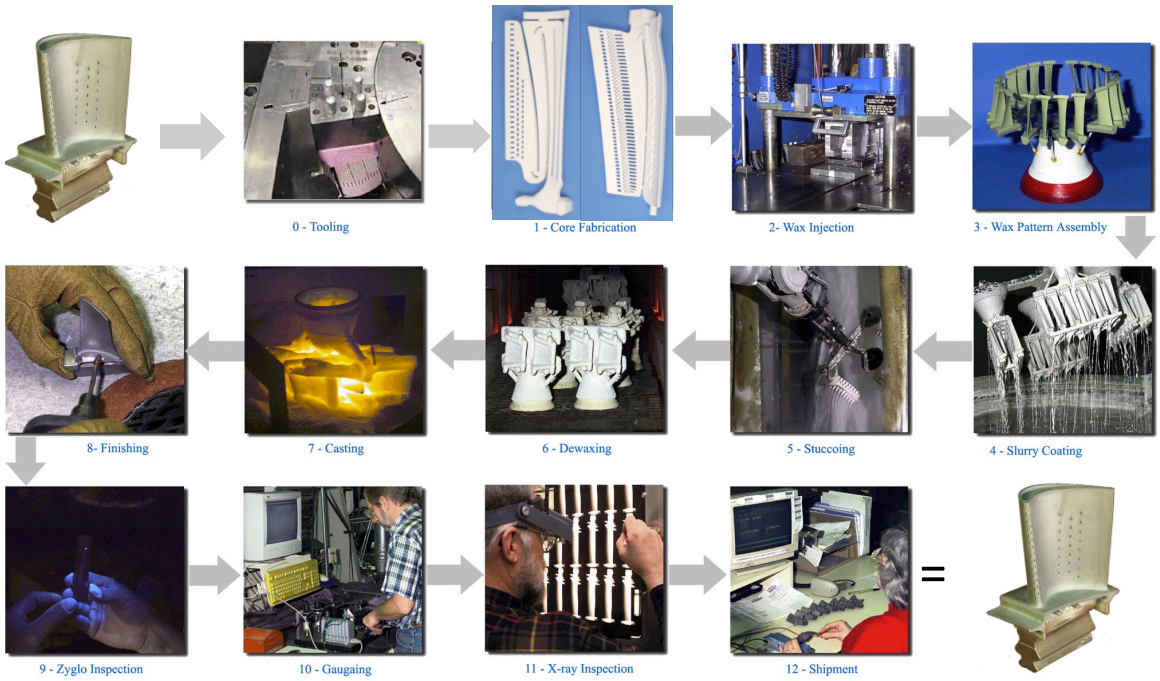


Figure 8: Various steps involved in the Investment Casting of HP turbine blades (Courtesy Alcoa Howmet).

are precisely assembled in a master mold pattern and wax is injected to form a turbine blade preform. In order to precisely maintain core positioning, pinning wires are pressed through the wax to butt against the cores within the preform. These wax preforms are then invested with ceramic suspension by taking them through the suspension coating and stuccoing process steps. The ceramic molds thus obtained are then taken through a heating cycle to melt out the wax and create a cavity for the metal to be cast. The de-waxed molds are then preheated and molten metal is poured and cast to obtain the turbine blades. These turbine blades are then taken through a series of finishing and inspection operations to check for casting defects and dimensional tolerances. Once they are checked for defects, they are then taken through further external machining operations where processes such as laser drilling, electrodischarge machining (EDM) and water jet are used to drill film cooling holes and other surface features.

Although the investment casting process for fabricating these internally cooled turbine blades is fairly evolved, it is still has major limitations. An overview of the difficulties in this process is given in the next section. A discussion on how LAMP can alleviate these limitations, to not only improve the yield and precision in the manufacturing of current designs but also enable the fabrication of next generation blade designs, is also presented.

1.2.3 Possibilities with LAMP

As discussed in the previous section, the investment casting process for fabricating these turbine blades has a lot of tooling and several steps involved. Human labor is also involved in several of these steps. As a result, the process is expensive and time consuming. It is also not amenable to design changes as new sets of tools and dies have to be made for every design change. This leads to high costs and long lead times (about 6-24 months) before a new design can be reliably fabricated. The yield of the process is also quite low due to the scrap that results in several of these processes. It is estimated that about 90% of the scrap and defects result in the mold making process which includes the first seven processes shown in Figure 8 up to the de-waxing stage. These defects include several forms of core shift, glue joint failures and handling breakage at various stages of mold preparation process. The cost of the overall process and consequently of the turbine blades increases substantially because of the low yields. It is estimated that for low yields, the cost of making integrally cored molds can take up to 50% of the overall cost of process.

Apart from the defects and dimensional variations produced from the casting process, further variations are introduced during the post process machining and surface finishing operations. For example, film cooling holes are usually drilled by the percussion laser drilling method where the molten metal is carried off as a vapor and forms a plasma in a trapped region. As a result, laser drilling results in a

somewhat irregular internal hole exit shape, diameter and finish. Film cooling holes are ideally expected to have a shallow exit angle with respect to the surface tangent and long aspect ratios with length-to-diameter (L/D) ratios, typically of the order of 5. Such shallow exit angles and large aspect ratios are very challenging to achieve through laser drilling as well. The laser might just bounce off the surface at shallow drilling angles and might not have enough penetration to achieve the long aspect ratios while maintaining uniformity of shape along the length of the whole. Such challenges invariably lead to variations which can have a substantial impact on the blade performance and life A recent study [83] on the effects of various manufacturing tolerances of these features on their cooling performance has shown that a variability of about 5-10% in their dimensions might lead to a 40-50^0C increase in blade operating temperature, depending on the specific feature. Such inordinate increases in operating temperatures might reduce the service life of the blade by up to a third of its expected life span.

Yet, despite all the discussion on current manufacturing challenges, it should be realized that contemporary cooling schemes are not even in their optimal configurations. For the most part, the current schemes have been developed through repeated experimental trials in simulated engine conditions without a complete understanding of the complex flow physics occurring in these cooling features. Although the flow physics are not yet completely understood, further advances in numerical methods, computing power and development of sophisticated experimental flow measurement techniques have led to significant advances in understanding some of the dominant features in these flows, which have in turn been translated into more novel and better performing cooling schemes. Several researchers have investigated such novel schemes both computationally and experimentally. For instance, novel film cooling configurations [26–28], internal channels with various cross-sections and trip strip configurations [20–22], impingement cooling configurations [23–25] have all been investigated.

More discussion on these and some illustrations are given in Chapter 5.

Many of these novel schemes are extremely challenging and in some cases almost impossible to fabricate using the current manufacturing methods. Yet, implementation of some, if not most of these schemes is inevitable if any realistic progress is to be made towards producing higher thrust/ efficiency engines.

LAMP, with its layer-by-layer build approach, opens new possibilities into the design and fabrication of precisely such complex cooling schemes that are difficult or impossible to manufacture currently with conventional techniques. The ability to computationally design these schemes offers precise control on dimensional tolerances while also reducing design to prototype time drastically. Due to significantly fewer processing steps, greater accuracy and repeatability through an all-digital approach, LAMP also increases the yield of successful turbine blade casting significantly over any other conventional process.

LAMP not only enables the fabrication of some of the schemes currently under investigation, but it also opens up opportunities for schemes with a higher level of complexity that have not been envisioned thus far. On a fully optimized LAMP process, building more complex features will become relatively straightforward. This could include arbitrarily positioned three dimensional internal passages and channels with varying cross-sections, film cooling holes with sculpted walls that would deter adverse flow structures, or perhaps outer blade walls overlaid with special mesh or wire structures that would enable transpiration cooling while also preserving the structural integrity of the blade.

Some of the work presented in this dissertation intends to design, fabricate and test precisely some of these complex cooling schemes as an illustration of the impact LAMP could have and also to serve as a foundation for future work along this direction. As part of the preliminary work in this dissertation, some such complex and novel film cooling configurations have been investigated through multi-physics simulations and

the results have been observed to be encouraging. A more detailed discussion on this is given in Chapter 5.

1.3 Thesis Outline

The thesis consists of a total of six chapters. An brief outline of the content covered in each of these chapters is given here.

Chapter 1 gave a basic introduction to various classes of additive manufacturing technologies available in the market currently. It also gave a detailed literature review of the various AM technologies within each of these classes that have been adapted to produce ceramic components. A detailed outline of the LAMP process was also given and the key differences and advantages over the previous technologies were highlighted. Furthermore, a brief discussion on the state-of-the-art in the investment casting and cooling schemes design of HP turbine blades was given to showcase the possibilities that LAMP technology opens up for building next generation airfoils with superior performance.

Chapter 2 gives a detailed discussion on the various digital data processing algorithms that have been developed to enable the LAMP process. Several algorithms like direct slicing of the CAD parts, STL file slicing, post processing operations like error checking, part placement and tiling, image level geometry modification etc. have been implemented. The details on these algorithms, a brief relevant literature review and the key differences of these approaches from prior work is presented.

Chapter 3 presents a few computational schemes that have been developed for improving part quality. Three different schemes are presented. A novel adaptive slicing scheme using volume deviation as a metric has been implemented to reduce stair

stepping effects. Gray scaling effects on LAMP suspension have been presented and leveraged to reduce stair stepping effects on surfaces with downward facing normals. Finally, cure width studies to characterize the side scatter and shrinkage behavior of the LAMP suspension have been conducted and presented with the ultimate goal of image compensation prior to build to get the most accurate parts. A review of the relevant literature within each of these components has been presented and an interpretation of the results obtained in this work is given.

Chapter 4 addresses the need for support structures in the LAMP process. A review of the prior work was presented and noted that these approaches cannot be applied for the LAMP process. A new approach for identifying features that generate floating geometries and generating internal supports structures to connect these islands to the surrounding geometry is presented. Finally, since the structures built in this manner cannot be removed post-build, unlike in other layered manufacturing applications, the potential for a multi-functional support design to meet the dual objectives of successful part builds as well tolerable cooling performance is presented.

Chapter 5 gives an overview of the typical cooling schemes currently used in HP turbine blades and the potential for novel cooling designs that are currently non-manufacturable but possible to fabricate through a layered manufacturing process such as LAMP. The application of film cooling is selected and some novel hole geometries have been analyzed and shown to give superior cooling performance than the simple geometry currently in use. The approach and the methodology used for these simulations and an interpretation of the results is given.

Chapter 6 summarizes the work presented in this thesis, lists the unique contributions made and provides the scope for potential future research along these directions.

CHAPTER II

DIGITAL DATA PROCESSING SCHEMES FOR LAMP

2.1 Introduction

The LAMP technology development effort at Georgia Tech is a comprehensive one and consequently it involves work in a number of different areas such as polymerization kinetics, materials processing, machine design and process control etc. A summary of the various technical areas that need to be investigated for successful LAMP process development is given in Figure 9.

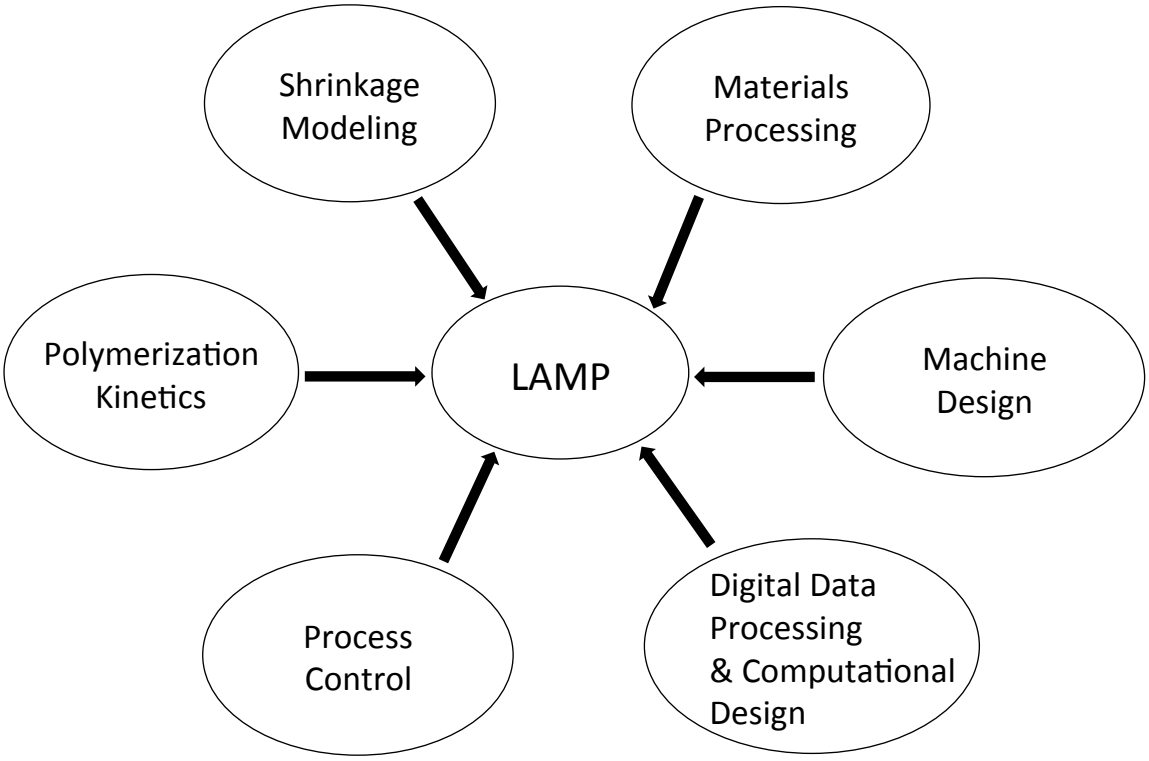


Figure 9: Various research areas for LAMP technology development.

While work on the various components of the LAMP technology development

initiative is being simultaneously pursued in a multi member team, this thesis specifically addresses the Digital Data Processing and Computational Design component. Several algorithms have been developed specifically for the LAMP process which give it some novel capabilities. Details of these algorithms as well as prior work and unique contributions of this work is presented in the following sections.

2.1.1 Motivation

LAMP process, being so unique in its exposure technique and aiming to make high precision components such as turbine blades, dictates some unique data processing capabilities. The DMD chip it uses for exposure, requires very high resolution bitmap images (at 1500 dpi or more depending on the kind of chip used; for comparison, conventional high quality prints are 300 dpi) as input for conducting exposure. Consequently, all of the CAD slice data is required to be output in this format. Most commercial slicing programs output slice data in the form of contours or scan patterns specific to the prototyping process for which they are designed and do not offer much flexibility in the modification of output data to suit the needs of the LAMP process. Moreover, they work mostly only on STL (Stereolithography file, an industry standard format) files which are simply a tessellated approximation of the original CAD model. They do not represent the native CAD geometry thus compromising on feature accuracy and resolution. Since the ultimate goal of the LAMP process is to build high-precision components like internally cooled turbine blades, this compromise of feature resolution is unacceptable. Hence, new data processing algorithms that can efficiently handle the native CAD files need to be developed. These algorithms should also be capable of handling the ubiquitous STL file format as well and should be capable of generating output data specific to the needs of LAMP.

Additionally, owing to the high feature resolutions (16-17 microns), slice images need to be generated at very high resolutions (1500 dpi). This can generate enormous

amounts of data. Each uncompressed slice image is typically of the order of several megabytes, and at this scale the slice data for an entire part can run into several terabytes. The LAMP process, as it currently stands, uses 100 micron thick layers at 1500dpi. For a typical HP turbine blade mold which has dimensions of 3 inches by 3 inches by 5 inches, it would take 18.75 gigabytes of data storage for a LAMP Alpha machine and 625 gigabytes of storage for a LAMP Beta machine. The Alpha machine is a preliminary LAMP prototype machine with a smaller build area of 4 inches by 4.5 inches whereas the Beta machine is the pre-commercial machine with a much larger build area of 23 inches by 23 inches. In the future, it is intended to build parts with 25 microns layer thickness and 2400dpi resolution for higher precision and in this scenario it would take approximately 194 gigabytes and 6.5 terrabytes of data for the alpha and beta machines respectively. Details of these data calculations in each of these build scenarios is given in Table 1. Having data on this order of magnitude

Table 1: LAMP data size estimates.

	Scenario 1	**Scenario 2**
Layer Thickness	100 μm	25 μm
Resolution	1500 dpi	2400 dpi
No. of layers	1250	5000
Data size for one part	3.125 GB	32.4 GB
No. of parts per build	Alpha Machine: 6	Alpha Machine: 6
	Beta Machine: 200	Beta Machine: 200
Total Data per build	Alpha Machine: 18.75 GB	Alpha Machine: 194.4 GB
	Beta Machine: 625 GB	Beta Machine 6.48 TB

poses severe storage issues and also hinders fast data transfer between the controlling computer and the LAMP machine. Thus, efficient data compression schemes and innovative data transfer techniques also need to be investigated. Moreover, once the data for each part is prepared, several post processing algorithms like error checking, part layout and tiling etc. need to be implemented in order to successfully build the parts.

Once the basic data processing flow is set up, there is scope for implementing several computational design schemes that can improve the part quality of LAMP well beyond the conventional additive manufacturing techniques. For example, implementing correction schemes to compensate for shrinkage and side-scattering, and gray-scaling to reduce stair stepping effect etc. Details of the various data processing schemes developed for LAMP as part of this dissertation and the corresponding prior work reported in literature is presented in this chapter. The next chapter deals with the computational schemes developed to improve part quality.

2.2 Direct Slicing

As discussed before, the LAMP process is intended to fabricate high-precision internally cooled turbine blades and hence cannot afford the coarse tessellated geometry approximation of STL files. The native CAD geometry needs to be processed to output the slice data used for building these components. A direct slicing algorithm for accomplishing this, is implemented using a geometric kernel called ACIS. The details of this algorithm are presented in this section. A brief summary of the direct slicing work previously reported in literature is presented first.

2.2.1 Literature Review

Several different direct slicing works using different approaches have been reported in the literature. Guduri et al. [84] proposed a direct slicing method for slicing a constructive solid geometry (CSG) representation of a part. Vuyyuru et al. [85] directly sliced solid models built by SDRC Corp's I-DEAS and segmented NURBS-based (Non-Uniform rational B-splines) contour curves. Rajagopalan et al. [86] also presented a similar approach to slice NURBS-based models using I-DEAS modeling sofware. Starly et al. [87] reported work on directly slicing STEP-based NURBS models. Zhao et al. [88] and Cao et al. [89] implemented direct slicing using AutoCAD. Luo and Ma [90] proposed a method for the direct slicing of surface patches and biarc

fitting of the cross-sectional data. Their work was implemented in AutoCAD modeling environment as well. Jameison and Hacker [10] used Parasolid, which is the kernel of Unigraphics, in order to achieve direct slicing. Chen et al. [91] and Chang [92] implemented direct slicing of Powershape models. Zhu and Yu [93] implemented a dexel based direct slicing approach. West et al. [94], Suh and Wozny [5], Marsan and Dutta [95], Kulkarni and Dutta [6] presented work primarily on other process planning tasks like part orientation, adaptive slicing etc. but have reported the use of ACIS kernel (A commercially available C++ CAD library of functions) for performing the necessary slicing operation.

In spite of the substantial work reported in the literature on direct slicing, all of these works tout the direct slicing approach as a fix for all problems STL files pose due to the errors in their geometry. Almost all the approaches assume the native CAD geometry is devoid of any errors and hence prefer direct slicing over STL slicing. This assumption of error-free CAD models is only true when the parts are relatively simple, and is typically not true for more complex "real world" geometries. There could be numerous errors in the CAD geometry both due to designer error as well as CAD translations needed to import the geometry into a specific file structure. Numerous works have been proposed in the literature that address the issue of tolerantly slicing erroneous STL files but to the best of the author's knowledge, none have been reported on tolerantly slicing direct CAD geometries. In the case of LAMP, the part geometries of turbine blade molds are extremely complex (refer to Figure 14 in Section 2.2.3 for an illustration), and hence both the designer errors as well as the CAD translation errors are present in the model. The direct slicing approach presented in this thesis aims to tolerantly slice an erroneous CAD model using the ACIS kernel, which is novel.

2.2.2 Direct Slicing Algorithm

The ACIS kernel is a commercially available C++ CAD library marketed by Spatial Corp, a subsidiary of Dassault Systems. It offers robust APIs (Application Programming Interface) and function calls for most of the basic CAD operations. These APIs have been integrated into the slicing software to produce CAD slices. The resulting CAD slices are then rasterized to obtain the bitmaps used for exposure. The algorithm, on a basic level, is outlined in Figure 10.

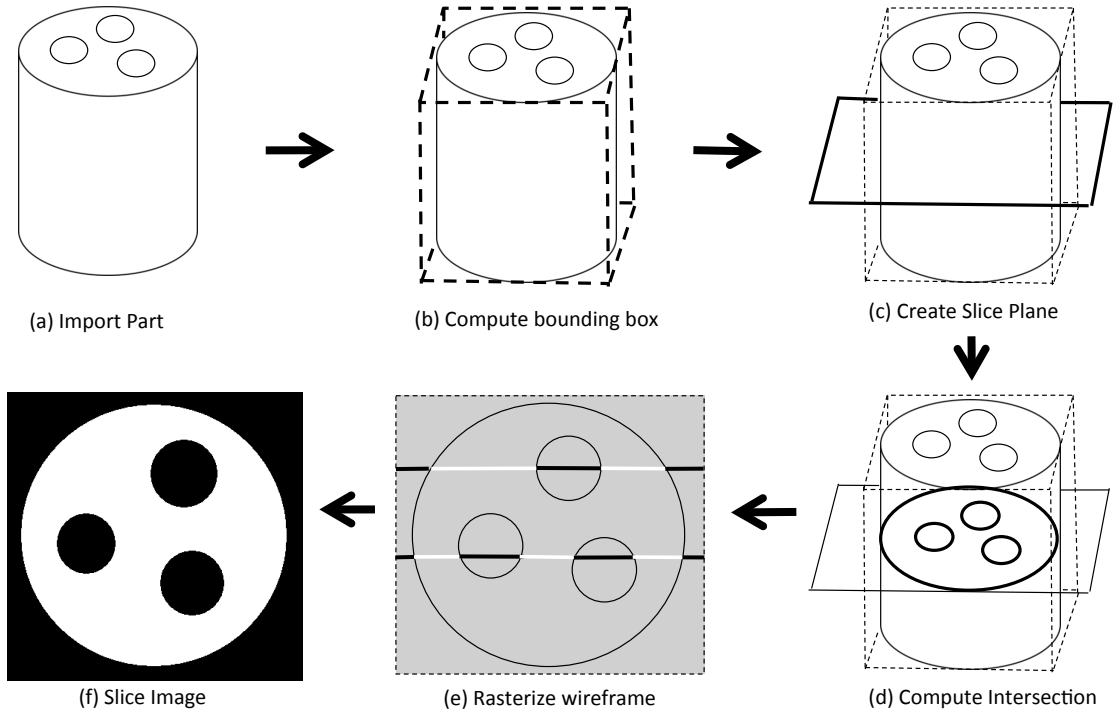


Figure 10: Direct slicing algorithm: (a) Original CAD part loaded into the program. (b) Bounding box computed. (c) Slicing plane created. (d) Intersection between the part and the slicing plane computed. (e) Intersection wire rasterized to create bitmap images. (f) Slice bitmaps obtained and compressed in CCITT FAX4.

The original CAD mold that needs to be sliced is first loaded into the algorithm using ACIS's load functions (Fig. 10a). ACIS libraries can only work with the SAT

file format and hence CAD files in other formats need to be converted first into the SAT format either by using commercial CAD softwares or by using ACIS's inbuilt file format translation functions. During the translation, numerical or topological inaccuracies can creep into the part. In severe cases, error checking and correction schemes need to be implemented. Once the part has been loaded, its bounding box is computed to obtain an estimate of the size of the bitmaps that would be generated (Fig. 10b). A slicing plane is then created and intersected with the part using ACIS's Boolean APIs to get an intersection wire (Fig. 10c). Once the intersection wire is obtained, it is rasterized to obtain the bitmaps (Fig. 10d). This essentially involves shooting rays for each row of pixels in the image and computing the intersection points with the intersection wire. Pixel values are then filled with alternating white and black segments in between each of these intersection points as shown in (Fig. 10e). The bitmap images obtained are then compressed using CCITT fax4 lossless compression scheme that compresses the data by three orders of magnitude without any loss (Fig. 10f). (CCITT fax4 is an industry standard lossless compression scheme for efficiently compressing 1-bit TIFF images. A good explanation of the CCITT compression schemes are given in [96]). These bitmaps are then fed to the post processing algorithms as discussed in subsequent sections.

In order to accomplish the tasks discussed in the basic algorithm outline shown in Figure 10, several different steps need to be completed. A more detailed view of these steps is shown in the pseudo code for direct slicing, Algorithm 1.

Algorithm 1 Direct Slicing.

```
set ACIS parameters like native resolution
set build parameters like layer thickness and slice image resolution
import CAD File to SAT format using ACIS Interops module
perform error checking on part
if part free of errors then
    get bounding box of the part, i.e., (X_min, Y_min, Z_min) & (X_max, Y_max, Z_max)
    adjust bounding box dimensions
    compute the height and width of slice image in pixels
    create and allocate memory for a 'characterBuffer' array
    for each 'Z' location along the build direction do
        create a plane at the specific 'Z' location
        for each body in the CAD slice do
            intersect body with the plane created to get a wire graph of the cross section
            store the cross-section wire graph to a 'crossSectionList'
        end for
        for each cross-section wire graph in the 'crossSectionList' do
            extract all the edges and store in an 'edgeList'
        end for
        for each row of pixels in the image do
            create a ray
            for each edge in the 'edgelist' do
                compute intersection of the edge with the ray
                if ray is not tangent to the edge then
                    store the intersection point in an 'intersectionList'
                end if
```

Algorithm 1 Direct Slicing Cond...

```
            sort points in 'intersectionList' w.r.t their 'X' coordinates
            for each intersection point in the 'intersectionList' do
                compute the corresponding pixel number in the image
                store the pixel number to a 'pixelNumberList'
            end for
            create temporary 'integerBuffer' array
            for each pixel in the current row of pixels do
                create a boolean variable 'color' and initiate to '0'
                fill 'integerBuffer' array with the value of 'color'
                toggle 'color' value when you hit a number in 'pixelNumberList'
            end for
            for every 8 values in 'integerBuffer' array do
                compute the decimal sum
                identify the corresponding ASCII character
                store it in the corresponding row of 'characterBuffer' array
            end for
        end for
    end for
    use 'characterBuffer' array information to create the slice image
    save the slice image to a multipage tiff
else
    display error message to user
    exit slicer program
end if
```

After importing the CAD file and performing error checking on the part, it's

bounding box is computed to yield the minimum and maximum extents of the part, i.e., $(X_{min}, Y_{min}, Z_{min})$ and $(X_{max}, Y_{max}, Z_{max})$. The size of the slice image (in pixels) is determined by these extents and the resolution (dpi) required. It is a convention in the image processing field to round up the image width to a value that is an integer multiple of 32 bits (4 bytes) and hence, these bounding box extents need to be adjusted. Assuming (without loss of generality) that the build direction is along 'Z' axis, the image size can be computed from these adjusted extents as follows:

$$Image\ Width = \frac{X_{max} - X_{min}}{Resolution} \quad (1)$$

$$Image\ Height = \frac{Y_{max} - Y_{min}}{Resolution} \quad (2)$$

Once the image size is determined, an ASCII character array denoted by 'characterBuffer' is created and dynamically allocated in order to store the necessary information for each slice. In binary (black and white) bitmap images, each pixel requires only 1-bit of memory. There is no provision in C++ to access each bit of memory individually. So, sets of values ('0's and '1's; '0' denoting black and '1' denoting white) of eight pixels are read at a time and the ASCII character corresponding to their decimal sum is stored in the appropriate location of the 'characterBuffer' array. After memory has been allocated for the image, the part is then sliced at the corresponding 'Z' height location by calling the 'api_planar_slice' API to produce an intersection contour. Details of the various ACIS APIs can be found at their documentation portal [97]. If the part file has multiple bodies, each of these bodies is sliced as well and the resulting cross section contours are stored in a list denoted by 'crossSectionList'. Having computed all of the planar intersections, all the edges from each of these contours are extracted and stored in an 'edgeList'. Once the 'edgeList' is populated, it is then time for computing the necessary color information for creating the bitmap image. The "exterior"of the part is denoted by black whereas the "interior" of the part is denoted by white. In order to determine if a point is in the

interior or exterior of the part, its membership with respect to the part needs to be established as discussed by Tilove [98]. A point's membership with respect to the interior can be established by originating a non "osculating" (touch without crossing) curve from the desired point and letting it propagate to infinity (with the assumption that a point infinitely far away is exterior to the part) while counting the number of times it intersects with the part. If the number of intersections is odd (even), then the point is interior (exterior) to the part. This method of identifying the "interior" from the "exterior" is illustrated in Figure 11. As can be clearly seen, the ray originating

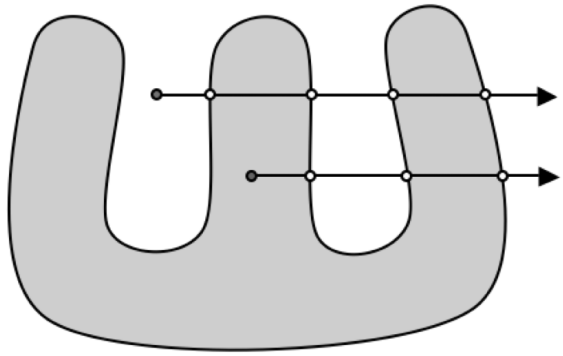


Figure 11: Determining membership of a point w.r.t part interior [99].

from the exterior point makes an even number (four) of intersections while the ray originating from the interior point makes an odd number (five). It is fairly evident that the same logic applies if the ray were to start at infinity and terminate at a point whose membership is to be determined. More importantly, it can be observed that if a ray starting at infinty were to cut across a part, it membership toggles between interior and exterior of the part every time it intersects with the part boundary. For the purposes of making the slice image, this fact is taken advantage of. Rays starting at the left end of the bounding box are created for each row of pixels in the image and their intersection points with each of the edges in 'edgeList' is computed. As previously mentioned, these rays have to be non "osculating" for this method to work and hence the computed intersection points are only stored to the 'intersectionList' if

the ray is non-tangential to the edge with which the intersection point is computed. Once the 'intersectionList' is created, an 'integerBuffer' array is created to store the pixel color data. A boolean 'color' variable (a variable that can only take values of '0' and '1') is created and initiated to '0' to start with (as the start point of the ray is always external to the part). The process of marching along the ray is simulated by counting the pixels as we move from left to right in the row. The value of the 'color' variable is toggled every time the pixel number exceeds the 'X' coordinates of one of the points in 'intersectionList'. It is vital that this progression along the ray is counted in terms of pixels and not the absolute floating point distance traveled along the ray with respect to size of each pixel. The size of each pixel is a floating point number (0.00066667 inches for a 1500dpi image) and counting the distance traveled with respect to this size instead of the integer pixel numbers will lead to floating point errors and will result in rough edges in the slice image as shown in Figure 12.

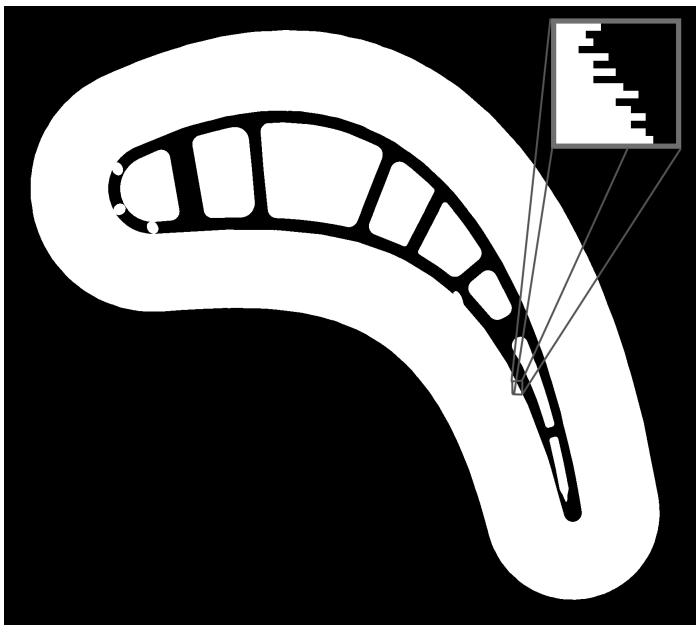


Figure 12: Floating point error; Inset shows the resulting rough edges.

Such rough edges will greatly diminish the surface quality of the part and hence need to be avoided at all costs. Once the integer values of pixels are properly filled,

the 'integerBuffer' array is then converted to the corresponding ASCII characters to fill the 'characterBuffer' array for the image as discussed previously. The 'character-Buffer' array gets completely filled when this process of intersection and collecting the pixel color information is completed for all the rows of the image at which point it can be used to make the compressed TIFF image.

2.2.3 Geometric Complexity & Error Tolerance

It was discussed in Section 2.2.1 that much of the previous work reported in the literature claims direct slicing to be free of errors while STL file slicing to be ridden with them and that this assertion is true only when the part geometries are relatively simple. This section provides more details on this claim and also elaborates on the error tolerant strategy used in the direct slicing approach presented in the previous section. Figure 13 illustrates the test parts used in much of the previously reported work in the direct slicing literature. As can be seen, most of the test test parts are

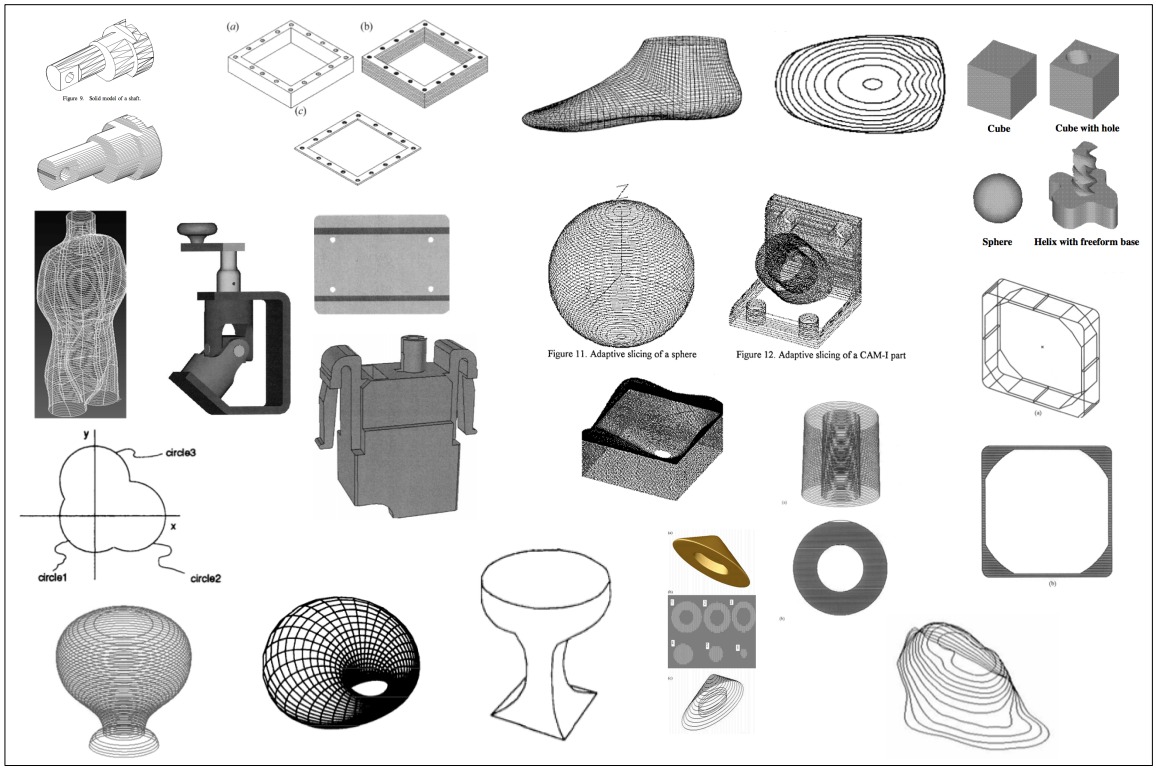


Figure 13: Test parts used in direct slicing literature.

single volume solids with a few that a have one or two voids in them. In comparison, Figure 14 shows the CAD model of a typical internally cooled HP turbine blade. It is deliberately shown in the wire frame view to give a better appreciation of the geometric complexity involved. Every edge in the figure represents an interface where two or

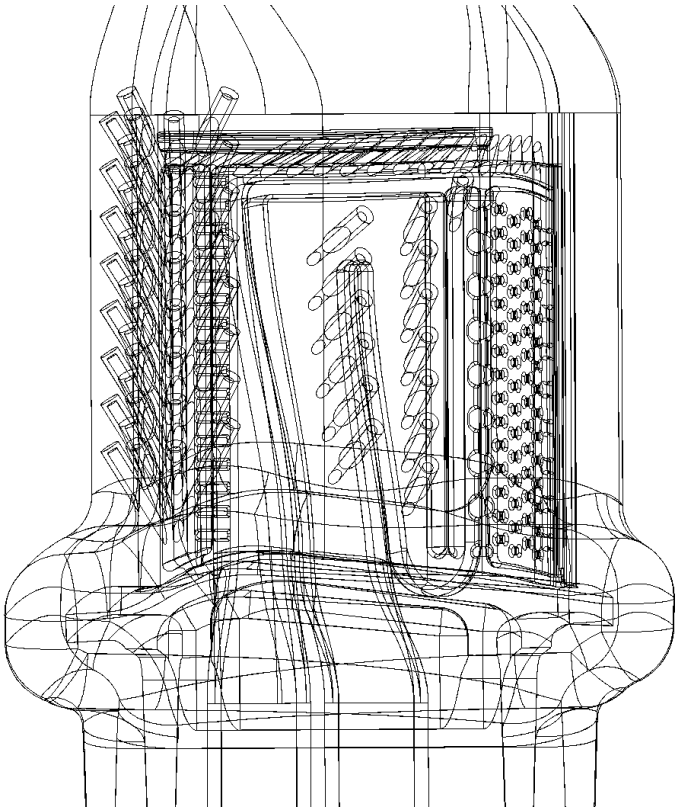


Figure 14: CAD model of a HP turbine blade mold in wire frame view.

more NURBS surfaces meet. Upon comparing the test parts previously shown with the HP blade mold, the order of magnitude difference in the geometric complexities involved in LAMP parts is evident. There is so much scope for errors with parts of such high complexity. Tiny gaps in the model due to CAD translations or non manifold geometry due to improperly defined surface intersections at the interfaces by the designer are very common. In order to successfully slice the model, the direct slicing algorithm described in the previous section is implemented to be tolerant to these errors.

It was observed that errors occur in two steps of the direct slicing algorithm discussed in the previous section. The first step is slicing which involves computation of the intersection between the slice plane and the CAD object. Due to the improperly defined geometry at complex regions of the mold, in some instances the slicing operation fails to produce a wireframe intersection curve (step (c) in Figure 10). The second step that contributes to errors is rasterization (step (d) in Figure 10) which involves computing the intersections between rays and the wire frame intersection curve. Any gaps in the model manifest as gaps in the wire frame curve and these intern manifest as stray lines in the rasterized image (refer to Figure 28(a) for an illustration of these stray lines). It was observed that the errors produced in these two steps are very sensitive to a parameter known as *ACIS Resolution* which defines how sensitive ACIS kernel is to the inherent topological errors in the CAD model. Figure 15 gives a plot of the number of stray lines observed in each image from a stack of hundred consecutive slices produced at two different ACIS resolutions. It can be

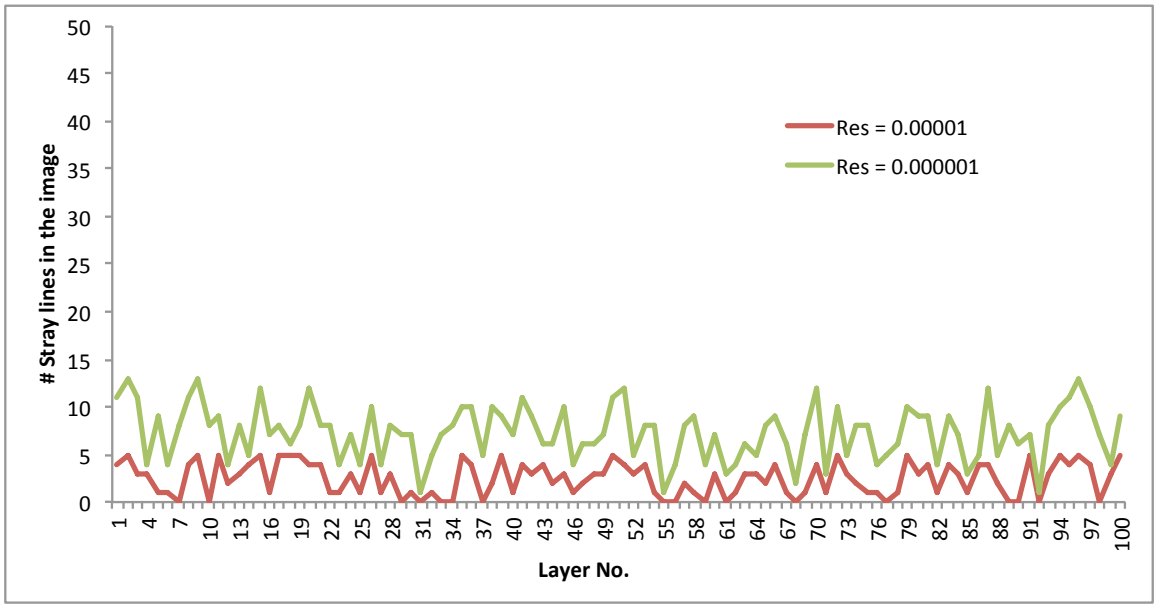


Figure 15: Plot showing the number of errors occuring in each layer for a stack of 100 layers for two different ACIS resolutions.

clearly seen that consistently more number of errors are produced in each slice layer

at the lower resolution (lower ACIS tolerance to topology errors) value. Not only does the ACIS Resolution parameter influence the slice image errors in rasterization, but it was also observed to influence the errors produced in the slicing step. When the slicing operation fails, in some instances it was observed that by lowering the ACIS resolution (thus making the kernel more tolerant) and re-slicing, the wire frame intersection curves can still be computed.

Keeping this fact in mind, in order to tolerantly slice the erroneous CAD geometry, two levels of error tolerance are embedded in the direct slicing algorithm. The first is in the CAD space which is aimed to alleviate the problems induced in the slicing step. If the slicing operation fails, the algorithm dynamically lowers the ACIS resolution to try and compute an intersection curve. If this fails, the slice plane location is perturbed by a small amount along the the height of the part (usually by $\pm 10\mu m$) and the slicing operation is conducted again at this new slice plane location with various resolutions in order to explore the possibility of successfully computing a wire frame slice. In extreme instances, if the slicing operation still fails, the corresponding images cannot be created and these are then borrowed from the output obtained by slicing an STL representation of the same part (STL file slicing approaches are discussed in Section 2.3).

The second level of error tolerancing is in the image space. As was previously discussed, having successfully computed the wire frame intersection curve, stray lines can sometimes result in the output slice images during to the rasterization step of the direct slicing process. In order to alleviate this problem, an error checking algorithm that automatically detects and corrects these stray lines with a stack of slice images as input is implemented. Details of this algorithm are given in Section 2.4.1. Note that this error checking algorithm can work on a stack of slice images irrespective of whether they are produced by the direct slicing algorithm or the several STL file slicing algorithms (will be discussed in Section 2.3). Hence, this second level of error

tolerancing, apart from making the direct slicing algorithm error tolerant, makes all the STL file slicing algorithms error tolerant as well.

2.2.4 Time Complexity & Computational Time

A rough estimate of the time complexity of the direct slicing algorithm and the computational times required to slice a few representative parts is given in this section. The time complexity of the algorithm can be estimated as follows: Assuming there are N surfaces in the part, for every layer in the part:

1) N surface intersections need to computed with the slicing plane. This amounts to N operations requiring roughly constant time.

2) Next, considering the worst case scenario, for each row of pixels in the image, N ray-edge intersections need to be computed. This amounts to another $Image\ Height * N$ operations.

3) Once the intersection points are computed, the proper integer pixel values need to be determined for each pixel in the slice image. These need to be later converted to character values to properly create the 1-bit TIFF image. These two operations will together amount to $C * Image\ Height * Image\ Width$ operations where C is some constant.

So, by adding together the number of operations for each of the steps described above, we can arrive at the time complexity for the algorithm as shown below:

$$T(N) = \#Layers * \{N + Image\ Height * \{N + C * Image\ Width\}\} \qquad (3)$$

Where, N denotes the number of surfaces in the part and C is a constant. So, it can be seen that the computation time for the algorithm roughly scales linearly with the number of layers in the part (i.e., directly proportional to the height of the part and inversely proportional to the Layer thickness) and the number of surfaces in the part. Although the expression in Equation 3 predicts the slicing time to scale quadratically with respect to the Image Height and Width, the number of operations are much

more dependent on the image height than the width of the image. Therefore, the slicing time roughly scales linearly with respect to the image height and is almost independent of the image width (linear in reality but with a very small slope) and therefore also scales linearly with respect to the output resolution (DPI, dots per inch) of the slice image.

Tables 2 gives the computational times taken to compute each slice for various output image resolutions. Figures 16 gives a trend line of how these computational times scale up. For the full fledged internally cooled HP turbine blade mold shown in Figure 14, it takes about 2 minutes to compute each slice and about a day and a half to compute the entire stack of slice images along the length of the part.

Table 2: Time taken to compute one slice image at various output resolutions for the generic turbine blade mold.

DPI	Image Width	Image Height	Slice Time (s)
300	632	1066	24.963
600	1265	2132	42.866
900	1898	3199	60.388
1200	2531	4265	80.34
1500	3163	5332	100.077
1800	3796	6398	118.458
2100	4429	7465	140.901
2400	5062	8531	159.843

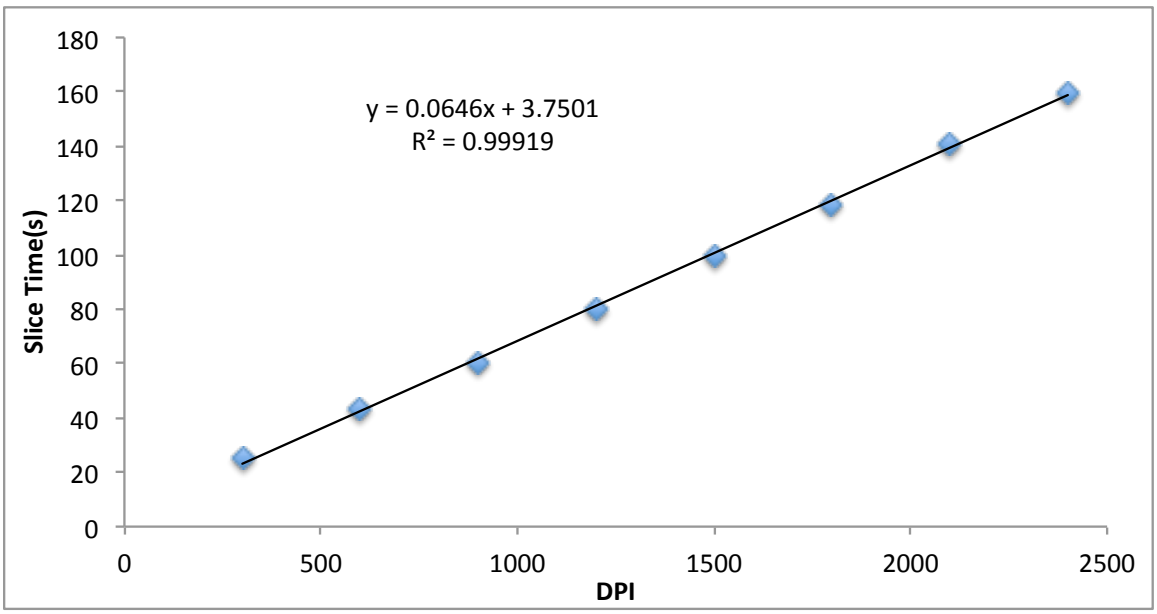


Figure 16: Time to compute one slice scaling with DPI (dots per inch).

2.3 STL Slicing

Since STL files are the De facto industry standard and the direct slicing approach using ACIS cannot handle these files, several alternate methods have been investigated. The implementation details and a review of corresponding literature for some of these approaches is given in the following sections.

2.3.1 POVRAY Rendering

POVRAY is an open source ray tracing software that is routinely used in the computer graphics industry to render high resolution photo realistic images. The schematic in Figure 17 shows the concept of ray tracing.

A small routine was used to convert an STL file into a POVRAY recognizable mesh object. The object is then sliced with a thin cuboid (thickness equal to slice thickness) using POVRAY's Boolean functions. The resulting intersection body is then positioned and ray traced as shown in the Fig. 2. The positions of the light sources and the camera are adjusted so as to obtain the slices at the correct scales.

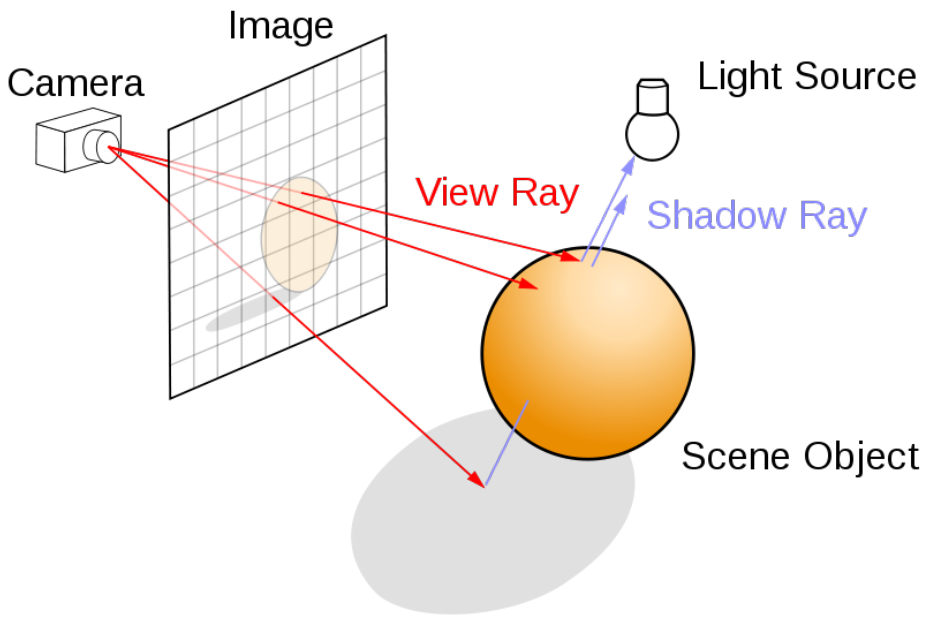


Figure 17: Concept of Ray Tracing.

An ''inside vector" was defined to identify the inside from the outside. When the scene is rendered, the camera shoots a ray for every pixel in the image to determine if it is inside the slice object or outside thereby rendering the slice bitmap image. The images obtained were then compressed using CCITT FAX4 scheme and sent to the DMD chip for exposure.

A major advantage of using this method to render the slice images is its simplicity in implementation. The povray rendering engine has all the facilities for computing the ray intersections during rendering. Therefore, minimal effort is required on the part of the developer to render the slices. But it was observed that in order to get very high resolution slice images, as is the case with LAMP, it was taking prohibitively long to render the images to a desired level of accuracy. Hence, other methods for slicing STL files have been investigated as discussed in the following sections.

2.3.2 Reconstructing SAT Files

One of the major weaknesses of the STL file format is the lack of topological information. As mentioned before, STL files are just a random collection on triangular facets

with no edge or vertex connectivity information embedded in it. The basic structure of a sample STL file is shown below:

```
solid cube_corner
  facet normal 0.0 -1.0 0.0
    outer loop
      vertex 0.0 0.0 0.0
      vertex 1.0 0.0 0.0
      vertex 0.0 0.0 1.0
    endloop
  endfacet
  facet normal 0.0 0.0 -1.0
    outer loop
      vertex 0.0 0.0 0.0
      vertex 0.0 1.0 0.0
      vertex 1.0 0.0 0.0
    endloop
  endfacet
  facet normal 0.0 0.0 -1.0
    outer loop
      vertex 0.0 0.0 0.0
      vertex 0.0 0.0 1.0
      vertex 0.0 1.0 0.0
    endloop
  endfacet
  facet normal 0.577 0.577 0.577
    outer loop
      vertex 1.0 0.0 0.0
```

```
      vertex 0.0 1.0 0.0
      vertex 0.0 0.0 1.0
    endloop
  endfacet
endsolid
```

As can be seen, each of the facets and their respective vertex coordinates and normal vectors are listed successively in an arbitrary order. This poses severe constraints in efficiently performing several vital operations on them. First, slicing the model becomes difficult and time consuming. This aspect is discussed in more detail in the section on direct STL slicing, Section 2.3.3. In addition to slicing, performing error checking operations like identifying missing facets or gaps in the model, computing integral properties like mass, center of gravity, volume etc. (which are important for various process planning operations like part orientation, build area packing etc.), and manipulation of part geometry become very difficult as well. With the ultimate objective of alleviating these limitations, an algorithm is implemented to reverse engineer a CAD file (or an SAT file) from the input STL file with all the topological information embedded in it. Doing this will aid in using the robust APIs of ACIS to perform the slicing operations and other vital operations like error checking, calculating the integral properties, manipulation of part as previously stated.

In order to reverse engineer an SAT file, the BRep(Boundary Representation) data structure of ACIS needs to built from ground up. Figure 18 shows the various elements of the data structure that need to be created and populated. Every body in the the BRep data structure is first divided into 'Lumps'. Lumps may or may not be disconnected and are present for the sole reason of simplifying the geometry to make the CAD algorithms more efficient. Each Lump consists of a list of disconnected closed objects called 'Shells'. Each shell consists of a list of 'Faces' that bound the space defined by the shell. Each 'Face' consists of a series of 'Loops'. A Loop is a

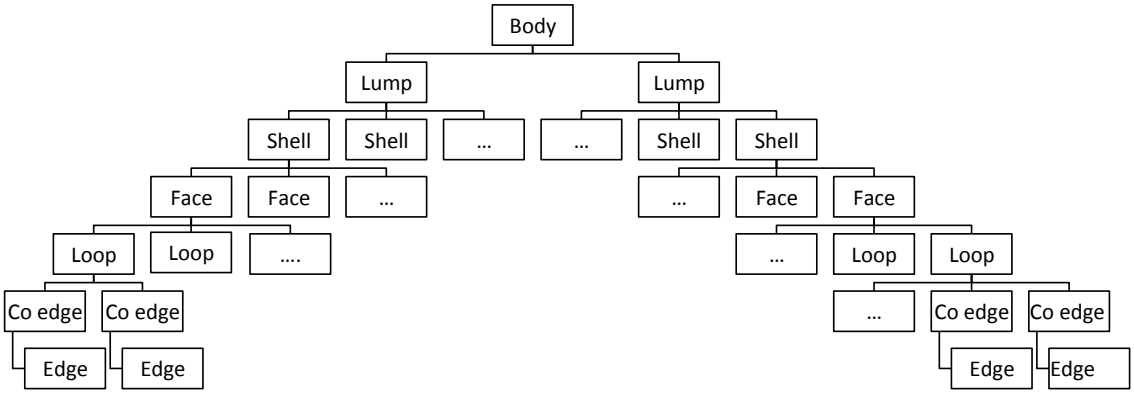


Figure 18: ACIS BRep Data Structure.

ring of end-to-end connected 'Co-edges'. A Co-edge is a topological entity associated with every 'Edge' in the model and is used to store edge connectivity information in the part. If two faces meet at an edge, the corresponding co-edges from the loops of each of these faces refer to the same edge and this how the modeling kernel identifies adjacency information between them. More information about the various elements in the data structure and how the topological information is stored in it can be found in [100].

To populate the data structure and create an SAT file from a triangular mesh, the following steps need to performed in that order:

1) For every listed facet, vertices need to be created. Co-edges connecting these vertices need to be created as well and looped in the right 'sense', i.e, clockwise or counter clockwise.

2) From the face adjacency information, edges need to be created and corresponding co-edges from the adjacent faces need to point to the same edge.

3) Again, from the face adjacency information, all the interconnected faces need to be grouped into shells. At the end of the grouping, each shell should have a list of all the faces that are interconnected but do not touch or intersect with any of the faces of the other shells.

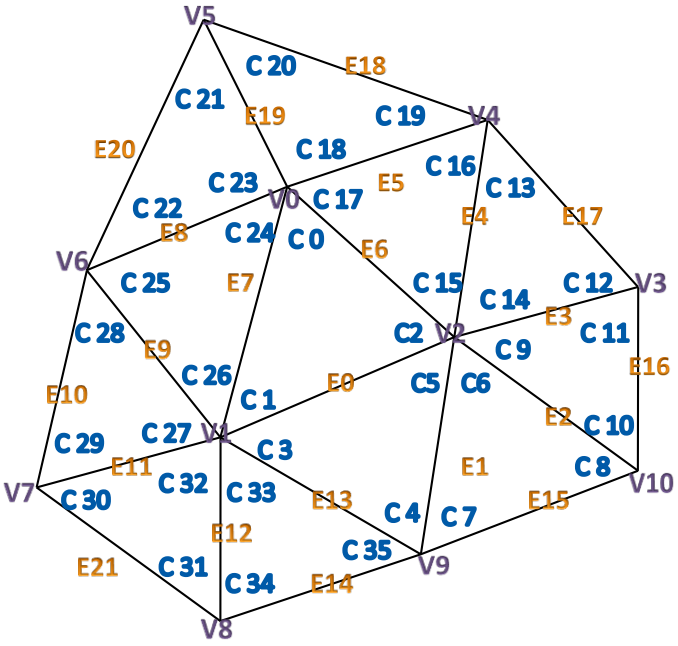
4) These shells can be arbitrarily grouped into lumps and the whole body can be

created from the resulting list of lumps.

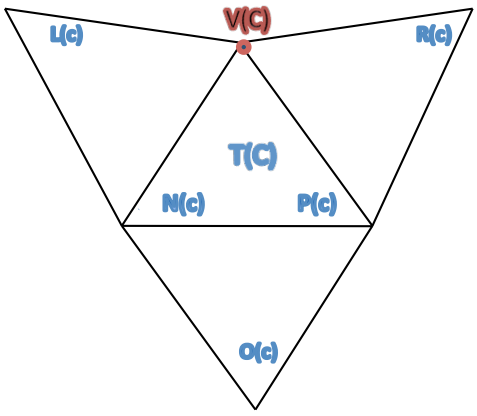
From the above discussion, it is evident that, to reverse engineer an SAT file all of the topological information like vertex and edge connectivity and facet adjacency needs to be first extracted from the STL file. This is quite a challenging task and special data structures need to implemented to accomplish it. There are a few data structures and algorithms reported in the literature to recover the topology information from a random list of facets like the STL file. Stephen Rock and Michael Wozny [101] presented an algorithm for extracting the adjacency information and for implementing basic error detection/ correction. There are others in the computer graphics and visualization domain that address this issue [102–104]. Many of these algorithms use complex multidimensional data structures for sorting three dimensional data which are difficult to implement. J Rossignac et al. [99, 105] of Georgia Tech proposed a simple integer array based data structure called the corner table data structure that is an elegant solution for working with triangular meshes. This data structure was primarily used by them to compress triangular meshes in computer graphics applications. The basic ideas of the data structure have been used and extensions have been made to make it applicable for our purpose of reconstructing SAT files from STL files. Details of the data structure and the overall algorithm are presented next.

2.3.2.1 Corner Table Data Structure & Implementation Details

The Corner table data structure stores all of the topology and connectivity information in two simple integer arrays. The schematic in Figure 19 shows the nomenclature and the integer arrays that hold the information. The region around a vertex in a facet is loosely referred to as a 'Corner'. The vertex that corresponds to that corner is referred to as 'v(c)'. The corner opposite to the current corner 'c' is referred to as 'o(c)'. The left and the right corners are respectively referred to as 'l(c)' and 'r(c)'.



(b) Sample Triangular Mesh



(a) Nomenclature

c	V[c]	O[c]	E[c]
0	0	4	0
1	1	16	6
2	2	25	7
3	1	8	1
4	9	0	0
5	2	34	13
6	2	--	15
7	9	11	2
8	10	3	1
9	2	--	16
10	10	13	3
11	3	7	2
12	3	17	4
13	4	10	3
14	2	--	17
15	2	20	5
16	4	1	6
17	0	12	4
18	0	--	18
19	4	22	19
20	5	15	5
21	5	26	8
22	6	19	19
23	0	--	20
24	0	29	9
25	6	2	5
26	1	21	8
27	1	--	10
28	6	31	11
29	7	24	9
30	7	35	12
31	8	28	11
32	1	--	21
33	1	--	14
34	8	5	13
35	9	30	12

(c) V&O tables for sample triangular mesh

Figure 19: Corner table data Structure: (a) Nomenclature. (b) sample triangular mesh with indexed corners, vertices and triangles. (c) Vertex table 'V[c]' and Opposites Table 'O[c]'. (Note: in a manifold triangular mesh, there will not be any empty cells in the O[c] array as shown here. This is the case here because the array has been populated for a small portion of the mesh only).

The next and previous corners are given by 'n(c)' and 'p(c)' respectively (assuming

the vertices are listed in a counterclockwise manner). The triangle to which the current corner belongs is referred to as 't(c)'. [Refer to Figure 19(a)]. The two integer arrays that store the connectivity information are the Vertex array 'V[c]' and the Opposite array 'O[c]' as shown in Figure 19(c). For any given corner 'c', the corresponding vertex and opposite corner indices can be obtained from the Vertex array 'V[c]' and Opposite array 'O[c]' respectively. Once these two arrays are populated, the adjacency information is available. For example, starting from a random corner 'c', we can access the left triangle by querying t(o(p(c))), the right triangle by t(o(n(c))) and the opposite triangle by t(o(c)). In order to reconstruct an SAT file, the edge connectivity information is also required. Hence this data structure is extended for our purpose by also constructing an edge array 'E[c]'. It would store the edge index of the edge opposite to a given corner 'c'. If these three arrays are computed, then all the topological information required to reconstruct the SAT file can be recovered. Implementation details on how to compute each of these arrays are discussed next.

Constructing V[c]:

'V[c]' can be constructed by implementing a cell structure where the space enclosed in the bounding box of the part is divided into discrete cells. As the facets are read from the STL file, each of their vertices are first checked for redundancy before they are given an index and stored in V[c]. As each vertex is read, its cell number is computed. The distance between this vertex and the vertices already present in its cell and the immediate neighboring cells is computed. If it is less than a set tolerance value, the vertex is discarded and the index of the vertex closest to it is recorded in V[c]. If not, a new vertex index is created and stored in V[c]. Continuing this process for all the vertices of all the facets in the STL file, we will get a fully completed V[c]. Figure 20 shows a schematic of the cell structure used to identify redundant vertices and Figure 21 shows a flow chart of the algorithm used to fill V[c].

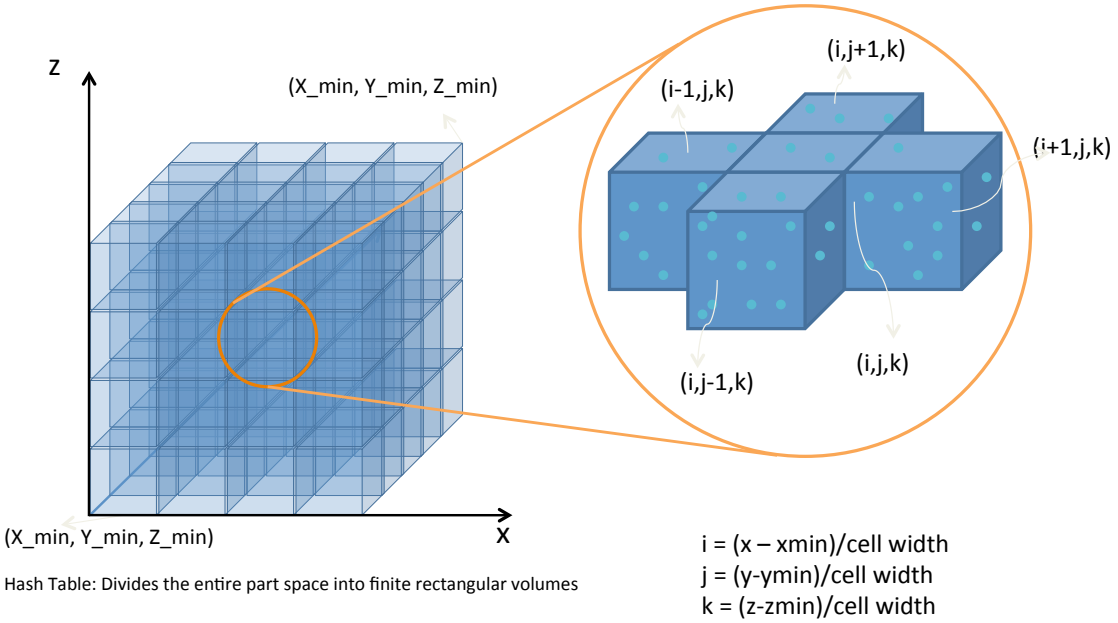


Figure 20: Schematic of the cell structure used to identify redundant vertices.

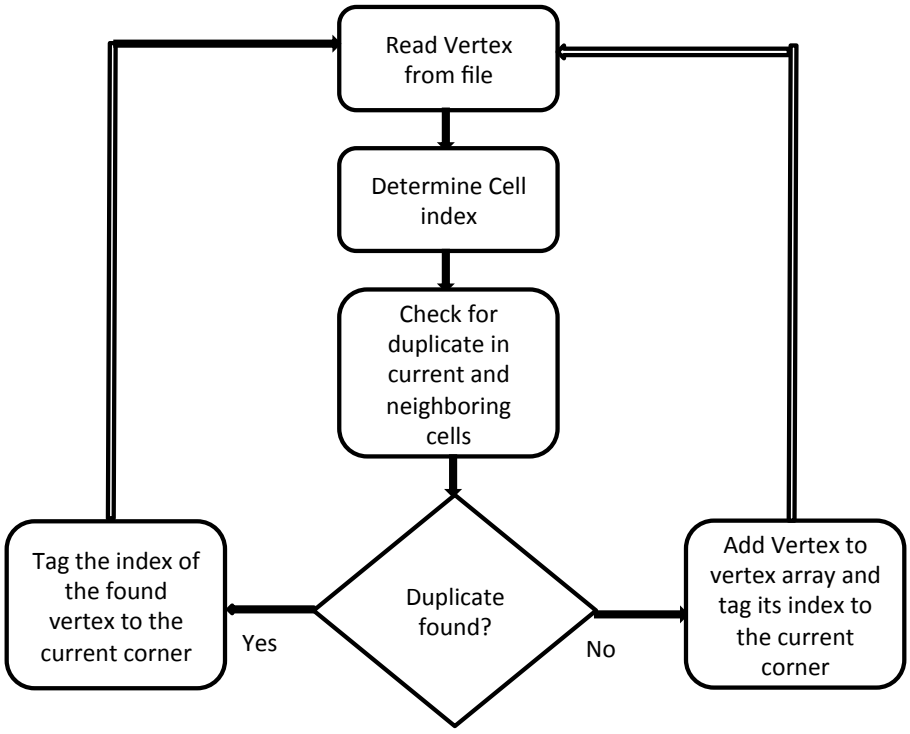


Figure 21: Flow chart of the Algorithm for filling V[c].

Constructing O[c]:

'O[c]' can be constructed by first identifying all the corners associated with a vertex and revolving around each vertex, marking the opposite corners. This is done by first populating a temporary data structure called 'bins'. Each node in 'bins' corresponds to a unique vertex in the mesh. For each corner 'c' in the part, the minimum vertex index among the vertex indices corresponding to the next and previous corners of 'c' is identified. The triplet of $(min\{V[n(c), V[p(c)]\}, max\{V[n(c)], V[p(c)]\}, c)$ is stored in the $min\{V[n(c), V[p(c)]\}^{th}$ node of 'bins'. This is essentially grouping all the edges originating from the vertex in its corresponding node in 'bins' and the corner 'c' pointing to the edge. After doing this for every corner in the mesh, each node in 'bins' will point to all edges originating from a vertex and the corners pointing to those edges. Once 'bins' is fully populated, it is easy to check for corners pointing to the same edge originating from the same vertex. If such a pair of corners exits then each corner is marked as the opposite of the other. Doing this for all the edges in each of the nodes in 'bins', 'O[c]' can be fully populated. A pseudo code for doing this is shown in Algorithm 2.

Algorithm 2 Constructing O[c].

```
1: create and allocate memory for bins
2: for each corner c in the mesh do
3:     e1 ← min{V[n(c), V[p(c)]}
4:     e2 ← max{V[n(c), V[p(c)]}
5:     bins[e1] ← (e1, e2, c)
6: end for
7: for each node in bins do
8:     for any pair of triplets (e1, e2, c) and (e1′, e2′, c′)
9:     if e1 = e1′ and e2 = e2′ then
10:        O(c) ← c′
11:        O(c′) ← c
12:    end if
13: end for
```

Constructing E[c]:

Once 'V[c]' and 'O[c]' are constructed, 'E[c]' can be easily constructed as follows. First an empty array 'E' is created and intialized to null. For every corner 'c' in the mesh, check if either or both of 'p(c)' and 'o(p(c))' are not pointing to any edge. If one of them is not pointing to an edge, assign the edge index of the edge pointed by the other. If both of them are not pointing to any edge, then create a new index for the edge corresponding to the two vertices of 'c' and 'n(c)' and store this edge index in 'E[c]' for the two corners 'n(c)' and 'o(p(c))'. Doing this for every corner in the mesh, the 'E[c]' table can be fully populated. A pseudo code for constructing 'E[c]' is given in Algorithm 3

Algorithm 3 Constructing E[c].

```
1: create and allocate memory for an array E
2: for each corner c in the mesh do
3:     E[c] ← NULL
4: end for
5: edgeIndex ← 0
6: for each corner c in the mesh do
7:     if E[p(c)] = NULL and E[o(p(c)] = NULL then
8:         create a new edge with vertices V(c) and V(n(c))
9:         E[p(c)] ← edgeIndex
10:        E[o(p(c))] ← edgeIndex
11:        edgeIndex ← edgeIndex + 1
12:    else if E[p(c)] = NULL then
13:        E[p(c)] ← E[o(p(c))]
14:    else
15:        E[o(p(c))] ← E[p(c)]
16:    end if
17: end for
```

Identifying Shells:

Having constructed V[c], O[c] and E[c], a simple function called 'swirl' can be implemented in order to identify the number of disjoint shells in the mesh. First, an array called 'Shell' with a length equal to the number of facets in the mesh is initiated and set to null. Each node in shell points to the shell number of a facet. For every corner 'c' in the mesh, if its corresponding facet 't(c)' is not assigned to a shell number in 'Shell[t(c)]' the number of shells is incremented by one and the Swirl function is called for the corner 'c'. Swirl function is a recursive function which calls itself. When it is called for a specific corner, it first sets the shell number for the facet and calls

itself on the left ('l(c)') and right corners('r(c)') of 'c'. Through this process of calling itself recursively, it tags all the interconnected facets with a shell number and this process continues until all the facets are tagged with a shell number. At the end of the routine, the number of disjoint shells in the mesh and the list of facets belonging to each shell can be identified. A pseudo code for doing this is shown in Algorithm 4.

Algorithm 4 Identifying the number of disjoint shells in a mesh using 'Swirl' function.

```
1: create and allocate memory for an array 'shell'
2: for each facet 't' in the mesh do
3:     shell[t] ← NULL
4: end for
5: #shells ← 0
6: for each corner 'c' in the mesh do
7:     if shell[t(c)] = NULL then
8:         swirl(c, #shells)
9:         #shells ← #shells + 1
10:    end if
11: end for
12: function SWIRL(c, k)
13:    if shell[t(c)] = NULL then
14:        shell[t(c)] ← k
15:        swirl(l(c), k)
16:        swirl(r(c), k)
17:    end if
18: end function
```

In this way, once all the arrays 'V[c]', 'O[c]', 'E[c]' are computed and the number of disjoint shells identified, all the required topological information for reconstructing

a SAT file from the STL mesh is recovered. Using this information, the SAT file can be constructed by populating the BRep data structure as previously discussed. An algorithm with these ideas has been implemented and STL files were successfully sliced with the ACIS kernel. In addition to slicing, other operations like error checking, geometry modification, calculation of integral properties like center of gravity, volume etc. have also been successfully performed.

2.3.2.2 Limitations

Although, the approach of recovering the topological information has several advantages as outlined before, it does have its limitations. One of the severe limitations crippling this method is the excessive size of the resulting SAT files. This is due to the fact that a number of excess entities like edges, coedges, vertices, faces etc need to be created to store the topological information in place of much fewer entities in the case of a native CAD representation. For example, if the STL file of a sphere consisting of N facets were to be reconstructed into a SAT file, it would now have N surface planar patches and several edges and vertices in place of just the one surface if it was represented in its native CAD format. Due to all of these excess entities, the SAT file size is several times larger than that for a native CAD file representing the same geometry, and this file size scales linearly with the number of faces. Table 3 gives an estimate of the SAT file sizes generated for a few sample meshes. Figure 22 gives a trend of this scaling with respect to the number of facets.

Table 3: SAT file sizes produced for a few sample STL meshes.

# Facets	SAT File Size (KB)
12	12
24	24
540	593
2376	2935
2376	2935
3872	4763
6162	7668

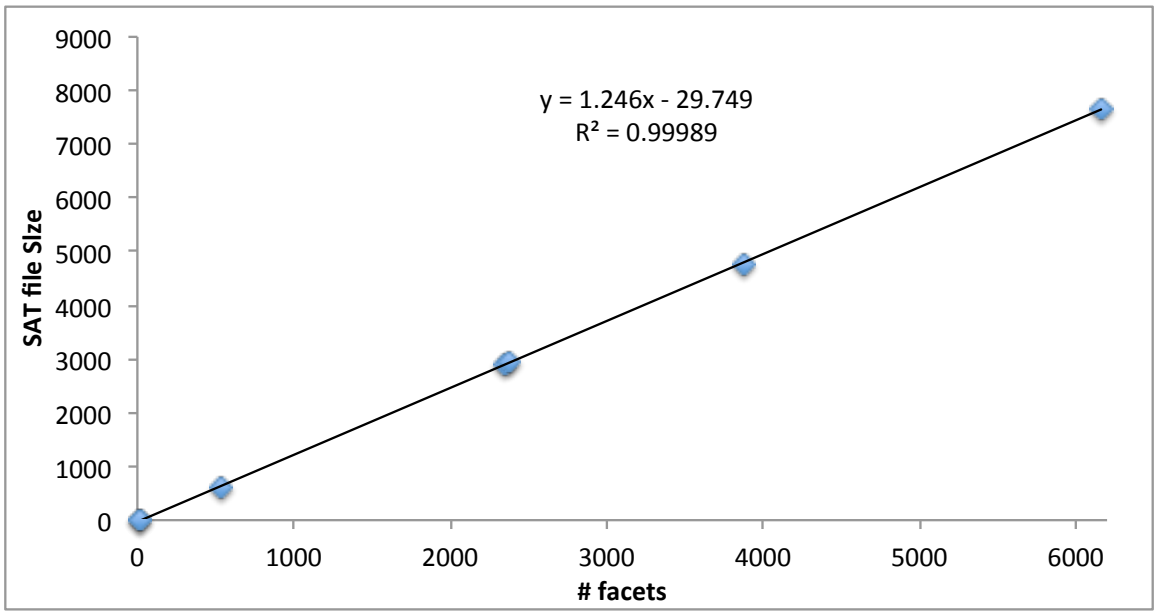


Figure 22: SAT file sizes scaling with # of Facets.

For the complex CAD model of the mold for an internally cooled turbine blade, it is imperative to have an STL mesh of upwards of 5.5 million facets. At such large facet counts, this approach produces prohibitively large file slices to work with (estimated to be 6.85 GB from the curve fit in Figure 22). Hence, this approach has been not integrated into the LAMP data processing flow and instead more direct approaches to read and slice STL files have been investigated. Details of these approaches are give next.

2.3.3 Direct STL Slicing

Owing to the huge SAT file sizes being generated using the previous approach, an algorithm for directly reading and slicing STL files without any intermediate conversions has been implemented. Since an STL file is a list of facets in random order, a major issue in efficiently slicing it is the lack of an ability to quickly identify those facets that lie in the intersection region from the rest of the facets in the file. Hence, a data structure of some kind needs to be implemented for this purpose. Although the STL file format has become the ubiquitous standard for representing CAD geometry in additive manufacturing and several companies or research groups have implemented methods to slice them, there is surprisingly little work reported in the literature about the various data structures used to efficiently slice them. Tata et al. [106] proposed a facet grouping strategy for this purpose. Luo et al. [107] proposed yet another strategy for identifying these facets quickly.

The algorithm presented here is very similar to [107] with minor modifications to suit the needs of LAMP process. The schematic in Figure 23 illustrates the procedure for identifying the intersecting facets at an arbitrary Z-height (slicing assumed to be along the z-direction without loss of generality). In order to identify the intersecting facets, first each facet's maximum and minimum z-coordinates are computed and stored in memory . Then, for a given slicing plane, first all the facets whose minimum z-coordinate is lesser than the slice plane height are selected (Figure 23(b)). Out of these selected facets, only those whose maximum z-coordinate is greater the slice plane height are identified and retained while the rest are discarded. This way, only those facets that intersect with the given slicing plane are isolated from the rest of the facets in the file (Figure 23(c)).

In order to do this, a data structure consisting of linked lists is implemented. Figure 24 shows a schematic of this data structure. It consists of a primary linked list sorted in the increasing order of z-values. Each node in this linked list consists of its

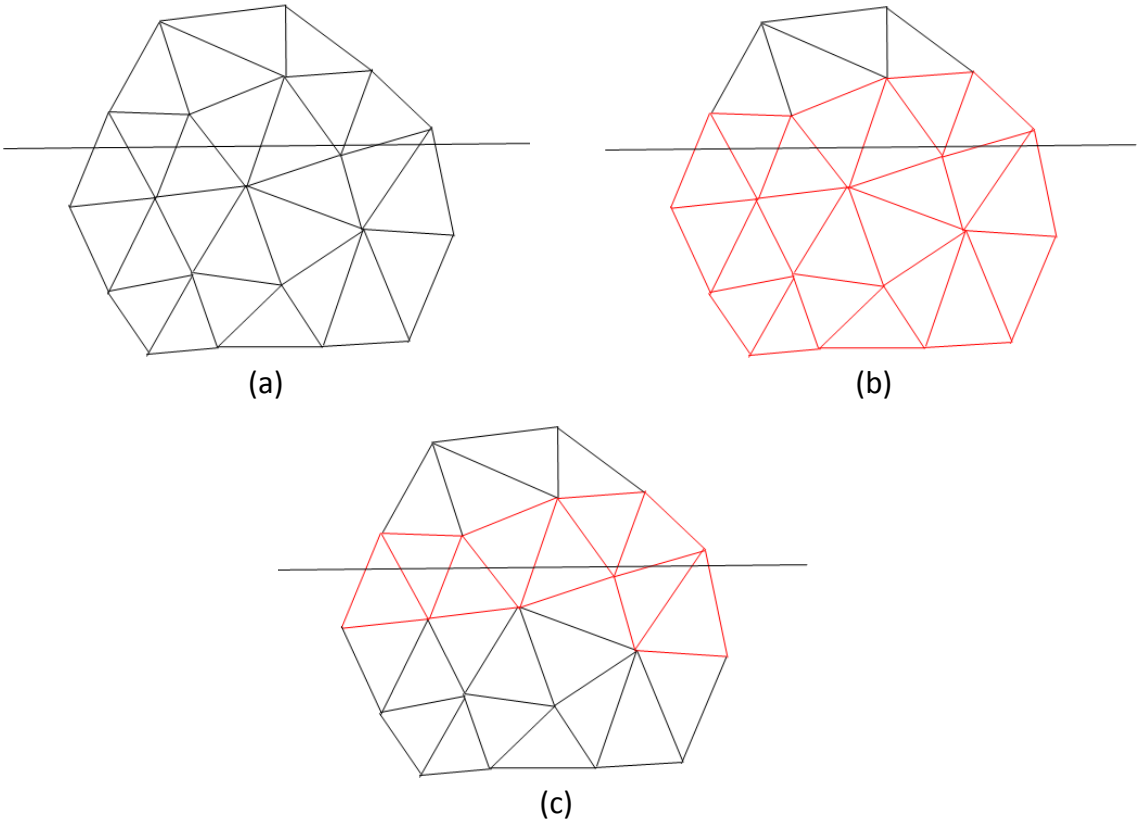


Figure 23: Direct slicing of STL files: (a) Sample triangular mesh and slicing plane. (b) Facets with minimum z-coordinate lesser than the slice plane height selected. (c) Of the facets selected in (b), only those facets with maximum z-coordinate greater than the slice plane height are kept and the rest discarded

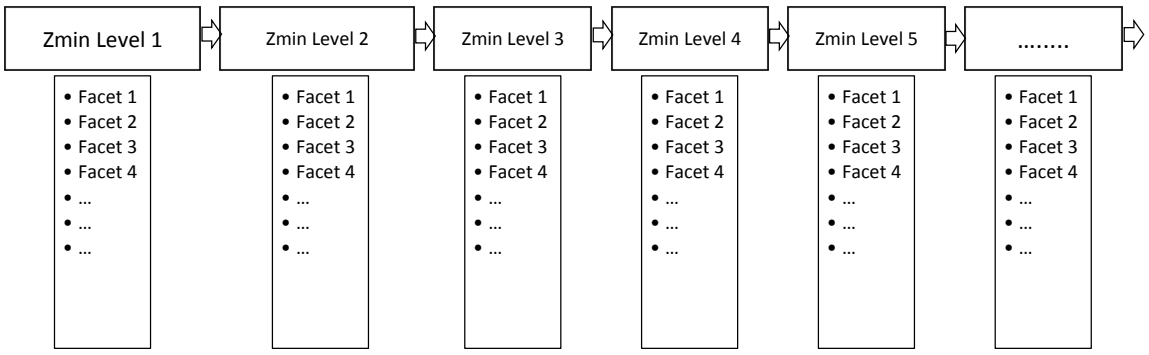


Figure 24: Data Structure used for Direct Slicing of STL files.

specific z-value and a pointer to a secondary list that contains all facets with the same minimum z-coordinate value as the z-value of that node. Once all the facets in the given STL file are populated in this data structure, it is straightforward to implement the rest of the operations required to accomplish the steps depicted in Figure 23.

For each slice, once the facets in the intersection region are identified, a simple parametric intersection is computed between the facets and the slice plane to yield the various edges of the intersection wire. If each of the edges of a facet are represented as a straight line in parametric form as shown in Equation 4 (where subscripts 'i' & 'j' denote two different vertices of a facet), the parameter value at the intersection between the edge and the slice plane at height 'Z' can be computed as shown in Equation 5.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_i + t \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}_j - \begin{bmatrix} x \\ y \\ z \end{bmatrix}_i \right) \tag{4}$$

$$t_Z = \frac{Z - z_i}{z_j - z_i} \tag{5}$$

If the so computed parameter value 't_Z' is between 0 and 1, the co-ordinates of the intersection point are computed as shown in Equation 6 and stored.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{intPoint} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_i + t_Z \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}_j - \begin{bmatrix} x \\ y \\ z \end{bmatrix}_i \right) \tag{6}$$

Each facet yields two intersection points when it intersects with the slice plane. The other intersection point is also computed similarly and these two points together make an edge of the cross-section wire. Since the facets are listed in a random order, the wire edges are also computed in a random order. In conventional contour planning operations, these edges need to be sorted and the intersection loops need to be constructed. However, for the LAMP process, it is sufficient to generate a bitmap image

of the slice. This can be directly accomplished by shooting rays for each row of pixels and computing intersection points. Then, using these points, the pixel values can be filled. In order to efficiently identify the edges that intersect with a particular ray a similar data structure like the one used for intersecting facets with a slice plane is used, except that these edges are now sorted based on their minimum y-coordinates rather than the z-coordinates. Once the edges are identified, the intersection points are computed parametrically in a similar manner as facet-plane intersections were computed. Once the intersection points are obtained, the process of computing the bitmap data is exactly the same as the one discussed in section 2.2.2. The images thus obtained are saved in CCITT fax4 format and fed to the post-processing algorithms.

Implementation Details:

The pseudo code for the algorithm implemented to accomplish the sequence of operations described in the previous section is shown in Algorithm 5. To begin with, the STL file of a given name is loaded into the program. STL files can be of two types: ASCII and Binary. ASCII STL files contain all the facet information listed in plain text and can be opened in any standard text editor. But for meshes with large facet counts, ASCII STL file sizes become prohibitively large. Binary STL files, on the other hand, store all the information in a binary format instead of plain text and hence are much smaller in size. Since the typical STL meshes encountered in LAMP have very large facet counts (upwards of 5 million), the slicing algorithm has been implemented to specifically slice binary STL files.

Binary STL files begin with an 80 byte block of memory known as the header which contains any file specific information. Following the 80 byte block, there is a block of 4 bytes which contains the number of facets in an unsigned integer format. Following the first 84 bytes of memory in the file, each facet information is stored in blocks of 50 bytes. Every facet block of 50 bytes consists of 12 bytes to store

the normal vector (4 each for the three direction cosines) and 36 bytes to store each of the three vertices. The rest of the 2 bytes of the 50 byte block for each facet is usually empty but can be used to store special attribute information like color in some applications.

Once the STL file is loaded into the program, the number of facets in the file is read from the 4 byte block of memory following the header and assigned to $\#facets$. Starting from zero, for every $i < \#facets$, the normal vector and vertex coordinate information of the corresponding facet is read using the function $readFacet$ and stored

Algorithm 5 Direct Slicing of STL files.

load STL file of given name
determine $\#facets$ in the file
$i \leftarrow 0$
$zList \leftarrow NULL$
for $i < \#facets$ **do**
    $bufferFacet \leftarrow readFacet(i)$
    update the max and min bounds of the part
    $addFacet(bufferFacet, zList)$
end for
using max and min bounds, compute image size and allocate memory
$eList \leftarrow NULL$
for each Z location along the build direction **do**
    $facetsToSlice \leftarrow isolateFacets(zList, Z)$
    $crossSectionWireEdges \leftarrow slice(facetsToSlice, Z)$
    $addEdge(crossSectionWireEdges, eList)$
    for each row of the image **do**
        $Y \leftarrow$ ycoord value corresponding to the row
        $intersectingEdges \leftarrow isolateEdges(eList, y)$
        for each edge in *intersectingEdges* **do**
            create a horizontal ray at Y from the left edge of the bounding box
            compute intersection points between ray the edge
        end for
        compute *integerBuffer* from the intersection points
        compute *characterBuffer* from *intBuffer*
    end for
    use information in *characterBuffer* to write the slice image to disk
end for

Algorithm 6 Reading a facet from binary STL file.

```
1: bufferFacet ← NULL
2: function readFacet(i)
3:     facetStartLocation ← 84 + 50 * i
4:     bufferFacet.N_x ← readByte(facetStartLocation)
5:     bufferFacet.N_y ← readByte(facetStartLocation + 4)
6:     bufferFacet.N_z ← readByte(facetStartLocation + 8)
7:     for n := 0 to 2 do
8:         bufferFacet.V[i]_x ← readByte(facetStartLocation + 12 + 12 * n)
9:         bufferFacet.V[i]_y ← readByte(facetStartLocation + 12 + 12 * n + 4)
10:        bufferFacet.V[i]_z ← readByte(facetStartLocation + 12 + 12 * n + 8)
11:    end for
12:    return bufferFacet
13: end function
```

in a temporary variable called *bufferFacet*. The pseudo code for the function $readFacet$ is shown in Algorithm 6. This function, in essence, seeks the file to the correct memory location corresponding to the i^{th} facet and reads the corresponding bytes of information within each facet block of memory and populates the temporary variable *bufferFacet*. Each facet i begins at the $(84 + 50 * i)^{th}$ byte from the beginning of the file as there are 80 bytes for the header, 4 bytes for $\#facets$ and 50 bytes for each of the $i - 1$ facets before the i^{th} facet. Once the facet starting location is identified and assigned to *facetStartLocation*, the rest of the operations read the corresponding bytes of information using the function *readByte* and the x, y, z coordinates of the normal vector and each of the vertices are populated.

In this manner, having read a facet from the file, the next step in Algorithm 5 is to update the min & max bounds of the part and to populate the facet in the data structure discussed in the previous section and shown in Figure 24. This is

accomplished by passing each facet that is read from the file to the function *addFacet*. The pseudo code for this function is shown in Algorithm 7.

Algorithm 7 Populating a facet in the Data Structure.

```
1:  function addFacet(facet , zList)
2:      traverse through zList
3:      look for matching zList node corresponding to facet.minZ
4:      if found then
5:          add facet to the list pointed by the matching zList node
6:      else
7:          create a new node toAdd
8:          toAdd.Z ← facet.minZ
9:          toAdd.facetList ← facet
10:         add toAdd to the zList in the proper sorted location
11:     end if
12: end function
```

The given facet's minimum Z-coordinate value is identified and the *zList* is traversed to find a node whose Z value matches the minimum Z-coordinate of the facet. If such a node is found, the given facet is added to the list pointed by the node. In the event that such a node is not found, a new node variable denoted by *toAdd* is created with its Z value equal to the facet's minimum Z-coordinate and with its facet list pointing to the give facet. This new node is then added to the *zList* in the appropriate location so it stays sorted in the increasing order with respect to the Z values of the nodes.

With the aid of these two functions *readFacet* and *addFacet*, by the end of the first loop in Algorithm 5, the *min* and *max* bounds of the part would be determined and the data structure of facets discussed in Figure 24 would be populated. The next loop then makes use of the data structure thus populated in order to slice the model

and output the images. For a give Z location of the slice plane, first the facets falling in the intersection zone of the plane are identified using the function *isolateFacetes* and stored in the list *facetsToSlice*. The pseudo code for isolating the facets is shown in Algorithm 8

Algorithm 8 Isolating facets in the intersection zone.

```
1: function isolateFacets(zList , Z)
2:     facetsToSlice ← NULL
3:     for each node in zList do
4:         if z value of the node is less than Z then
5:             traverse through each facet in the facet list pointed by the node
6:             look for facets whose maxZ > Z
7:             if such a facet found  then
8:                 add facet to facetsToSlice
9:             end if
10:        else
11:            break
12:        end if
13:    end for
14: end function
```

As can be seen from the pseudo code, in order to isolate the required facets to intersect, first each node in *zList* is read. If the Z value corresponding to the node is less than the Z height of the slice plane, each of the facets in the facet list pointed by the node is parsed. If a facet with maximum Z value of greater than the slice plane height is found, it is added to *facetsToSlice* list. After each of the nodes whose Z values are less than the slice plane height have been parsed in this manner, the *isolateFacets* function returns back the facets collected in *facetsToSlice*.

Having isolated the facets in the intersection zone of the slice plane, Algorithm 5

then computes the intersection of these facets with the slice plane using the function *slice*. These intersections are computed parametrically as described in the previous section. Having computed the intersection edges, which form the contour of the part cross-section, the next step is to generate an image from them. In order to accomplish this, rays are created for each row of the image and their intersection with the contour edges are computed. In order to compute these intersections efficiently, the edges are in turn populated in a data structure very similar to the one used for populating the facets. The edges are arranged into bins based on their minimum Y-coordinate instead of the Z-coordinate in case of the facets. This data structure for sorting edges is denoted by *eList* is Algorithm 5. The function *addEdge* is used to populate the edges in this data structure and its implementation is very similar to *addFacet* function previously described. After *eList* is populated, the process of computing intersections between the rays and the edges is the same as the one used for computing intersections between the slice plane and facets. For each ray, the intersecting edges are isolated using *isolateEdges* whose implementation is similar to *isolateFacets*. Once the intersecting edges are identified, the intersection points are computed parametrically. After the intersection points are computed, the process of creating bitmap data is exactly same as the one described in Section 2.2.2. First, a temporary *intergerBuffer* array is populated and later converted to ASCII character array *charBuffer* which is then used to save the bitmap image.

Time Complexity & Computational Time:

A rough estimate of the time complexity and the time it takes to finish the slicing operations is given in this section. If N denotes the number of facets in an STL mesh, in order to populate the slicing data structure discussed in the previous sections, the following operations need to be completed for every facet read: 1) Scan through the each node in *zList* to identify a matching node and 2) Scan through to the end of the

facet list pointed by the matching node to add the facet. Each of these two operations take, in the worst case scenario, N time steps. Since these two operations need to be performed for every facet in the file, the time complexity for populating the data structure is $O(N^3)$.

Polynomial time complexities like N^3 are usually acceptable but if the N in consideration is large, the computational time becomes excessively long. A rough estimate of the computational time for slicing STL files of various mesh sizes using this algorithm is shown in Table 4. The computational time scaling with respect to number of facets is shown in Figure 25

Table 4: Slicing time for various mesh sizes.

# Facets	Indexing Time (s)
81080	11.475
102268	17.902
114818	22.356
160296	44.893
191172	68.579
219584	109.218
337128	613.151
438434	1464.206
531370	2476.122
678358	4656.452
756446	6075.474

As can be seen, for a file with large number of facets, the slicing time runs into several days. In the case of LAMP, because of the geometric complexity of integrally cored HP turbine blade molds, the mesh sizes of the STL files go upwards of 5.5M. For such a huge file, it takes more than 4 days to slice which is prohibitively large. Hence a new STL slicing algorithm is implemented to cut down the slicing time. Details of this algorithm and the new data structure implemented are given in the next section.

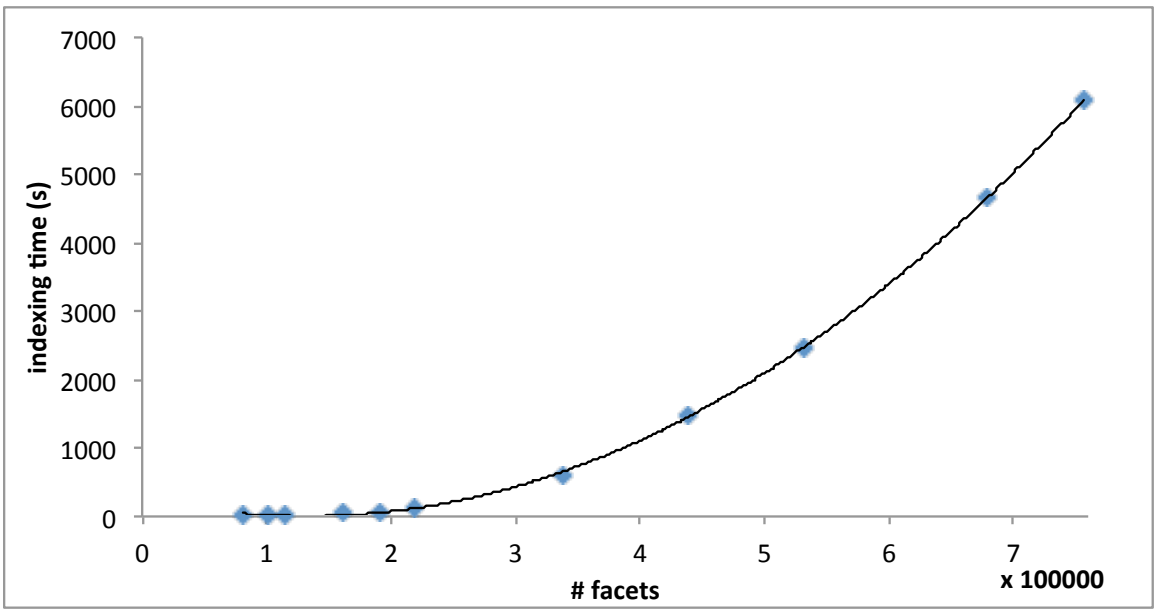


Figure 25: Slicing time scaling w.r.t mesh size.

2.3.4 Accelerated Direct STL Slicing

Since, for large STL meshes, it takes excessively long to populate the data structure described in the previous section, a new approach is investigated in order to cut down the slicing time. Upon close examination, it is evident that the key pieces of information in order to create the slice images, are the points of intersection of the rays with the cross-section wire edges. So instead of spending significant compute time in populating the facets in the previously described data structure, it suffices to compute and store these intersection points as each facet is read from the file. Additionally, if these facets are stored in an array structure rather than in linked lists as was previously done, data access can be much quicker as arrays have $O(1)$ complexity for indexing and searching while linked lists which have $O(N)$ complexity for the same (i.e, random access vs sequential access). The data structure used to store these intersection points is shown in Figure 26.

Z \ Y	Row 1	Row 2	Row 3	Row 4	.....
Slice 1	Points(1,1)	Points(1,2)	Points(1,3)	Points(1,4)	.....
Slice 2	Points(2,1)	Points(2,2)	Points(2,3)	Points(2,4)	.....
Slice 3	Points(3,1)	Points(3,2)	Points(3,3)	Points(3,4)	.....
Slice 4	Points(4,1)	Points(4,2)	Points(4,3)	Points(4,4)	.....
.....	.....	.....	.....	.....	.....

Figure 26: Data Structure for storing intersection points.

Each row in the data structure represents a Z height corresponding to each slice i in the part. Each column represents a Y level corresponding to a row j of the slice image. Each location (i, j) in the data structure contains all the intersection points corresponding to the j^{th} row of the i^{th} slice image.

As each facet is read from the file, depending on its maximum and minimum Z extents, it is possible to determine all the slice numbers that this particular facet will contribute edges to. For each of these edges corresponding to each of these slice numbers, it is also possible to determine all the rows of the respective slice images they contribute intersection points to. Based on this information, as every facet is read, all the intersection points contributed by this facet to each of the slices are computed and stored in the corresponding locations in the data structure shown in Figure 26. Once the intersection points contributed by all the facets in the file are computed and stored, slice images can be prepared by using a similar procedure as discussed in the previous sections.

Implementation Details:

The pseudo code for implementing this new algorithm for slicing STL files is shown in Algorithm 9. The bounding box coordinates of the part are first created, either by doing a linear scan of the whole part or by directly inputing the information in the program from the CAD model. The minimum and maximum vertices of the bounding

box are denoted by $(minX, minY, minZ)$ and $(maxX, maxY, maxZ)$ respectively. Next, memory for the 2D array data structure denoted by $zYMatrix$ is allocated and initiated to $NULL$. As each facet f is read from the file, the range of slices that this facet f contributes edges to is computed. $minLayer$ and $maxLayer$ denote the layer numbers of the lowermost and uppermost slices in this range. For each slice i starting from $minLayer$ to $maxLayer$, the facet is sliced at the corresponding slice height and the resulting edge is stored in a variable denoted by e. For each edge e, the range of image rows to which this edge contributes an intersection point are computed. The lowermost and uppermost rows of this range are denoted by $minRow$ and $maxRow$ respectively. For each row j between $minRow$ and $maxRow$, the intersection point of the edge e and the ray corresponding to the row j is computed and stored in a variable denoted by $intPoint$. These intersections of the facet with the slice plane and the edge with the ray are computed parametrically as described in the previous section. The $intPoint$ thus computed is populated in the $(i, j)^{th}$ cell of the 2D array $zYMatrix$. Once this $zYMatrix$ is completely populated after one linear scan of all the facets in the list, the slice images can be created in a similar manner as described in Section 2.2.2. For each slice image i, in order to compute the pixel information for each row j, the variables *integerBuffer* and *characterBuffer* are populated based on the intersection points retrieved from $(i, j)^{th}$ cell of $zYMatrix$. Once the *characterBuffer* array is completely filled for all rows of the slice, it can be used to write the slice image to disk.

Algorithm 9 Improved direct slicing of STL files.

compute bounding box of the part
$(minX, minY, minZ) \leftarrow$ minimum extents of the part
$(maxX, maxY, maxZ) \leftarrow$ maximum extents of the part
$zYMatrix \leftarrow NULL$
for each facet f in the file **do**
    $bufferFacet \leftarrow readFacet(f)$
    $minLayer \leftarrow floor\left(\frac{bufferFacet.minZ - minZ}{LayerThickness}\right) + 1$
    $maxLayer \leftarrow floor\left(\frac{bufferFacet.maxZ - maxZ}{LayerThickness}\right)$
    for each slice i; $i := minLayer$ **to** $maxLayer$ **do**
        $e \leftarrow sliceFacet(bufferFacet, i)$
        $minRow \leftarrow floor\left(\frac{maxY - e.maxY}{pixelSize}\right) + 1$
        $maxRow \leftarrow floor\left(\frac{maxY - e.minY}{pixelSize}\right)$
        for $j := minRow$ **to** $maxRow$ **do**
            $intPoint \leftarrow intersectEdge(e, j)$
            $zYMatrix[i][j] \leftarrow intPoint$
        end for
    end for
end for
for each slice i; $i := 0$ **to** $\#Layers$ **do**
    for each row j; $j := 0$ **to** $imageHeight$ **do**
        $intPoints \leftarrow zYMatrix[i][j]$
        compute *integerBuffer* using *intPoints*
        compute *characterBuffer* using *integerBuffer*
    end for
    use information in *characterBuffer* to write the slice image to disk
end for

Time Complexity & Computational Time:

A rough estimate of the time complexity and the computational time taken to slice STL files using this algorithm is given here. In order to index all the intersection points in the 2D matrix, it would take $N * Z * Y * C$ operations, where N denotes the number of facets in the file, Z denotes the slice range ($maxLayer - minLayer$), Y denotes the row range ($maxRow - minRow$), and C denotes some independent constant. In the worst case scenario, Z and Y can both be equal to N and the time complexity reduces to $O(N^3)$ just like the linked list STL slicing algorithm discussed in the previous section. However, in typical scenarios, both Z and Y are much smaller than N. The constant C is also very small since an array structure is used instead of a linked list. Hence, in most typical scenarios, the algorithm behaves like it is $O(N)$ complexity with a very small constant value and hence is much faster than the linked list algorithm discussed in the previous section. An estimate of indexing times using this 2D Matrix approach to slice typical STL files with various facet counts and the corresponding times for the linked list algorithm discussed in the previous section are shown in Table 5 and Figure 27.

Table 5: Slicing time for various mesh sizes.

# Facets	Indexing Time for Linked Lists(s)	Indexing Time for 2D Matrix
81080	11.475	14.147
102268	17.902	14.197
114818	22.356	14.55
160296	44.893	15.466
191172	68.579	16.157
219584	109.218	16.895
337128	613.151	18.69
438434	1464.206	19.715
531370	2476.122	21.754
678358	4656.452	24.045
756446	6075.474	24.921

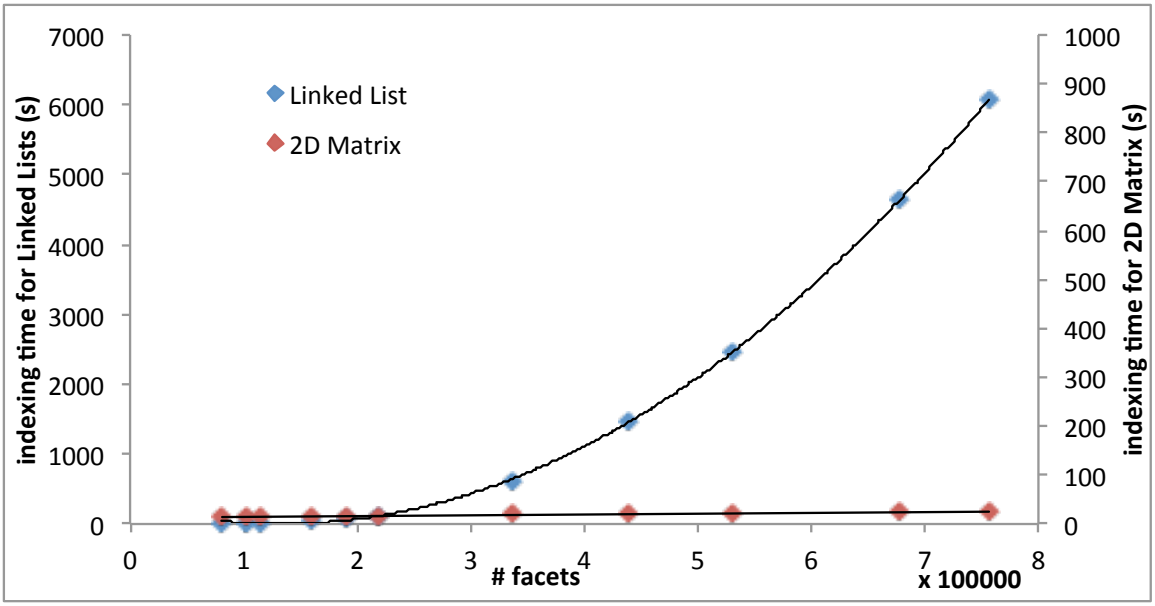


Figure 27: Slicing time scaling w.r.t mesh size.

As can be seen, the savings in slicing time are substantial using this approach. It takes just 30 minutes to slice a 5.5M facet STL file versus the 4 days it to slice the same file using the linked list approach. This leads to enormous time savings in preparing build ready images for each new part design.

Although, this new approach reduces the computational time, it does have its limitations. Firstly, since the facets are just read once and discarded from memory, once the slice thickness and image resolution are set, it is impossible to change them dynamically during the execution as the facet information cannot be retrieved. In the linked list approach, since all the facets are indexed in the data structure, there is always flexibility to change the layer height and image resolution dynamically for applications like adaptive slicing. However, this shortcoming can be alleviated in other ways. Since the slicing time itself is very short, multiple stacks of slice images can be created at multiple layer thicknesses and the proper images from each of these stacks can be selected for utilizing variable layer thicknesses in applications like adaptive slicing.

Secondly, this new approach uses far more memory than the linked list approach since it stores all the intersection points of all the rays with each of the slices in the part whereas in the linked list approach, just the facets are stored in memory. As an example, for the 5.5M triangle part, the new approach takes about 2.5 gigabytes of memory versus just 300 megabytes of memory using the linked list approach. Since memory is very cheap in recent times, this limitation is not a significant hurdle. Hence, considering the pros and cons of each of the approaches of STL slicing discussed in this section, the approach in this section is the preferred method to slice extremely high resolution STL files for LAMP.

2.4 CAD Data Preparation for LAMP

The slice images produced through STL slicing and Direct CAD Slicing in particular require further post-processing before they are ready for use in a LAMP build. The details of these various post-processing operations and algorithms are given in this section.

2.4.1 Error Checking & Correction

One of the mandatory post-processing operations that needs to be performed on the slice images is checking for errors and validity of the images. These errors are caused by gaps that creep into the model either due to errors in the CAD model or due to translation between different CAD formats as described in Section 2.2.3. During the slicing operation, in the process of creating the slice images, rays are created for each row of the image to identify the correct pixel color values as previously described. Any ray that coincides with one of these gaps that creep into the CAD model will result in the omission of an intersection point that should have been computed in an error-free file. This omission will result in the pixel color value not toggling at the corresponding location and this will in turn manifest as a stray line in the slice image. For illustration purposes, a particularly bad instance of these stray line errors

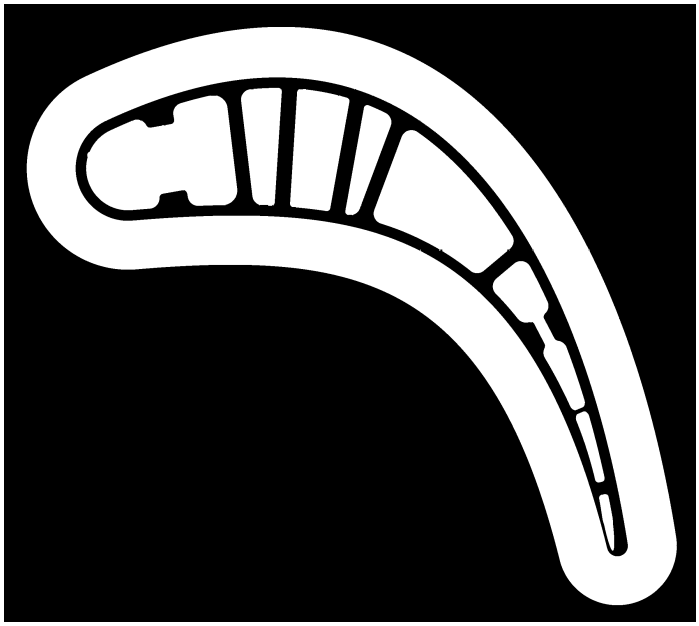
is shown in Figure 28(a). Such stray lines need to detected and corrected before the images can be used for part builds on the LAMP machine. For this purpose, an error checking and correction algorithm has been implemented. The algorithm takes an erroneous slice like the one shown in Figure 28(a) and outputs a corrected slice shown in Figure 28(b).

In order to identify these stray lines, the algorithm scans through each row of the image and checks for pixels that are sandwiched by pixels of an opposite color, i.e, pixels that have a different color value than the ones present on the rows immediately above and below the current row that is being searched. Once it finds such pixels, it flips their values to match the color values of the top and bottom rows to correct the stray lines. This approach works for correcting only stray lines that are 1-pixel thick but can be extended to detect lines that multiple pixels wide. The details of the extended algorithm along with the pseudo code for detecting stray lines that are multiple pixels wide is given next.

Implementation Details:

The pseudo code for the error correction operation is shown in Algorithm 10. The given stack of slice images that need to checked are first loaded into the program. As each image in read from the stack, its pixel data is first converted from ASCII representation to integer representation for easy data manipulation and stored in an integer array *integerBuffer*. If an intersection point was missed in the slicing operation and a stray line caused, then the last pixel of the row containing the stray line will have a white pixel on a surrounding black backdrop. This fact is exploited in identifying erroneous rows that need to be corrected. This is accomplished by checking for pixels with a color value of 1 (i.e, white) in the last column of pixels in *integerBuffer*. If such a white colored pixel is identified, it means that the corresponding row needs to

(a) Error Slice.



(b) Corrected Slice.

Figure 28: Error Correction.

be corrected for a stray line. Having identified the row at which a stray line occurs, the next step is to determine the width this stray line. So, having identified the row at which a stray line originates, the color values of end pixels in the rows immediately following the identified row are checked. If consecutive rows are found to have white

pixel values, then the width of the stray line is more than one. The number of consecutive rows are counted and the stored in the variable *width*. Having identified the beginning row number and the width of a stray line, this information is then passed to the function *correctRow*.

Algorithm 10 Error Checking Algorithm.

```
1: load the stack of images to correct
2: for each image in the stack do
3:     characterBuffer ← pixel data of the image
4:     integerBuffer ← convert ASCII data in characterBuffer to integer data
5:     for each row j of the image do
6:         width ← 0
7:         if the last pixel of row j in integerBuffer = 1 (i.e, white) then
8:             width ← width + 1
9:             j ← j + 1
10:            while last pixel of row j in integerBuffer = 1 do
11:                width ← width + 1
12:                j ← j + 1
13:            end while
14:            correctedIntegerBuffer ← correctRow(integerBuffer, j − width, width)
15:        end if
16:    end for
17:    correctedCharacterBuffer ← convert correctedIntegerBuffer to ASCII data
18:    output the corrected image to disk using correctedCharacterBuffer
19: end for
```

The pseudo code for the function *correctRow* is shown in Algorithm 11. It is just an extended version of the logic described previously for correcting one pixel wide stray lines. Three counters i, j and k are used in the algorithm. Counter k keeps

track of the row number in the image corresponding to each row in a multi-pixel wide stray line that is being corrected. Counter i keeps track of the number of rows above a particular row in the stray line that needs to be checked for color information. Similarly, counter j keeps track of the number of rows below a particular row in the stray line that needs to be checked for color information. For each pixel in each row of the stray line, the color of the corresponding pixel in the row that is i rows above and j rows below is checked. If it is the same but different from the color of the current pixel in the row that is being corrected, the pixel value is flipped.

Algorithm 11 row correction function.

```
1: function correctRow(integerBuffer, row, width)
2:     for i := 1 to width do
3:         j = width − i + 1
4:         k = row + i − 1
5:         for each pixel in row k do
6:             if color i rows up = color j rows down ≠ color on current row then
7:                 flip the current pixel color
8:             end if
9:         end for
10:    end for
11: end function
```

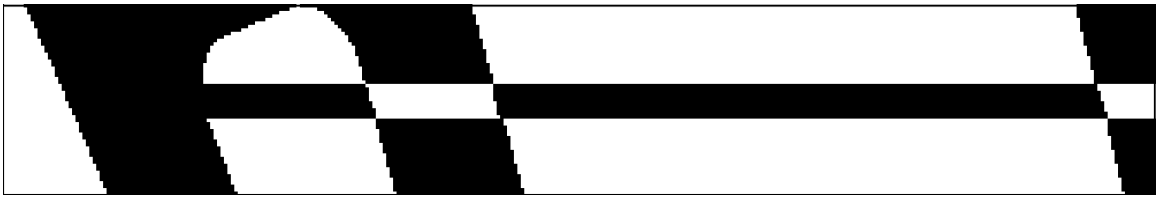
For example, lets assume the beginning row number of a stray line is thirty (i.e, $row = 30$). Lets also assume that the width of the stray line is three (i.e, $width = 3$). Hence in this example, if we were correcting the first row in the stray line (first iteration of the loop), the color values of the pixels one row above three rows below should be checked for equality and if they are equal but different from the color value of pixels in the first row that is being corrected, then the pixel color value needs to be flipped. The counter values are correspondingly set, i.e, for the first iteration

(correcting first row in the stray line), $k = 30$ (beginning row number of the stray line), $i = 1$ (one row above) and $j = 3$ (three rows below). Similarly for correcting the second row in the stray line ($k = 31$), the pixel values two rows ($i = 2$) above and two rows below ($j = 2$) need to be checked for color information and so on.

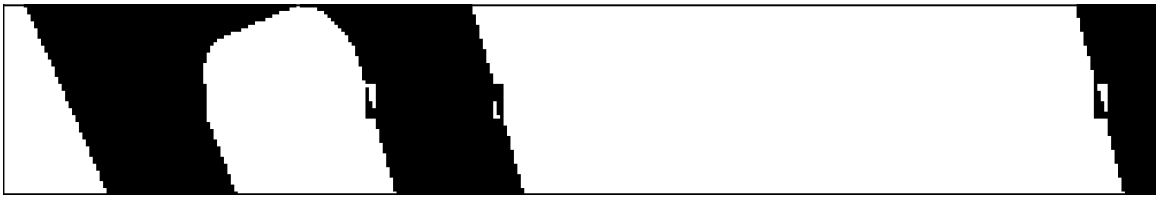
All of these error checking operations are based on the assumption that the collective width of these consecutive stray lines is much less than the minimum feature size in the CAD part that is to be sliced. For the blade designs that are currently being built by LAMP have minimum features sizes of about 500 microns, i.e, approximately 30 pixels in size at 1500 dpi. The widest stray lines observed in the slices were five pixels wide which is much less than the minimum feature size of 30 pixels and hence can be corrected with reasonable accuracy.

The other limitation of this algorithm is that, when the stray lines are too wide, the corrected rows do not conform to the boundaries of the part where the color toggles from black to white or vice versa. The edges of the part in the corrected rows become vertical instead of smoothly connecting with the rest of the contour. This phenomenon is shown in Figure 29(b) for a sample stray line shown in Figure 29(a). When the stray lines are thin (less than 5 pixels wide) the inaccuracy caused is negligible but as they get wider, it needs to be corrected for. This problem can be fixed by constructing a spline (a Hermite cubic spline with $C1$ continuity for example) to close the contour in a continuous way and the color toggling points in each of the rows in the stray line can be computed from the so constructed spline. This way smoothness of the edge contours can be maintained as shown in Figure 29(c) while correcting stray lines that are arbitrarily wide. Although, it can correct a stray line of any arbitrary width, if the stray line is too wide (width of the stray line approaching minimum feature size in the part) then rather than passively correcting it in the image, the geometry of the original CAD part needs to be repaired for accurate slices.

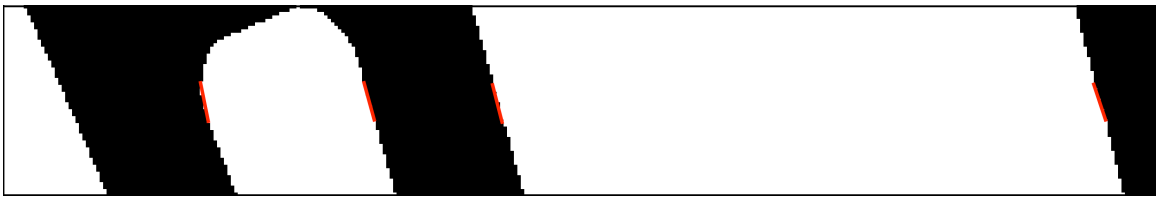
In this way, the function *correctRow* can correct stray lines given the starting



(a) Multi-pixel wide stray line.



(b) Improper correction with out a curve construction.



(c) Proper correction with a curve construction

Figure 29: Rectifying multi-pixel wide rows with stray lines.

row number of the stray line in the slice image and its width. Returning back to Algorithm 10, once all the stray lines identified in the integer data of the slice image (*integerBuffer*) are corrected, it is then converted back to ASCII data and saved back to disk. In this way, all the slice images in the stack are checked and corrected for errors and by the end, a stack of error free images are obtained which are taken through other post processing operations as described in the following sections.

2.4.2 Tiling and Part Placement

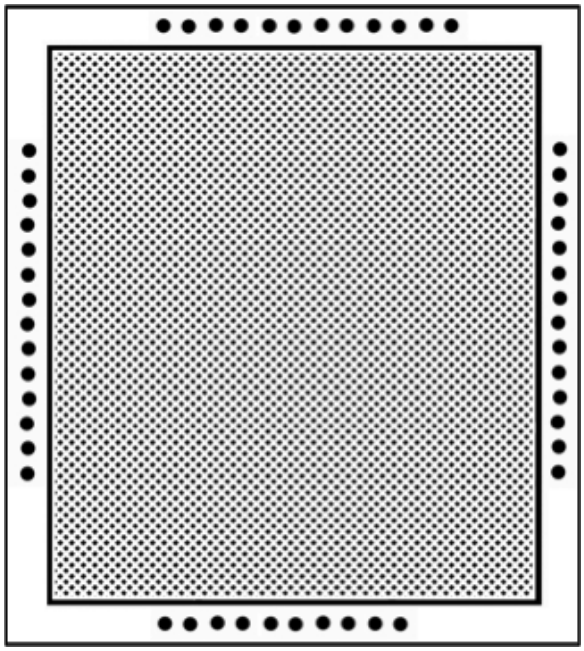
The corrected images as obtained above then need to be tiled properly on an image template for proper part placement in the build area. Figure 30(a) shows a typical input image to the tiling code. A mesh structure as shown in Figure 30(b) is used as the background on which this input slice is tiled. The mesh structure has been optimized after considerable experimentation to prevent the uncured Suspension in the empty regions from sloshing around in the build tank during recoating of a fresh

layer of Suspension by the blade. The algorithm automatically computes the maximum extents of the slice image, determines the number of parts that can be built within the build area, lays them out at the correct coordinates and creates break lines along the mesh structure for easy removal of parts after the build is complete. The final build-ready images produced by the code look like the one shown in Figure 30(c). The code runs through the entire stack of the slice images to produce a build-ready stack that is then fed to the LAMP machine.

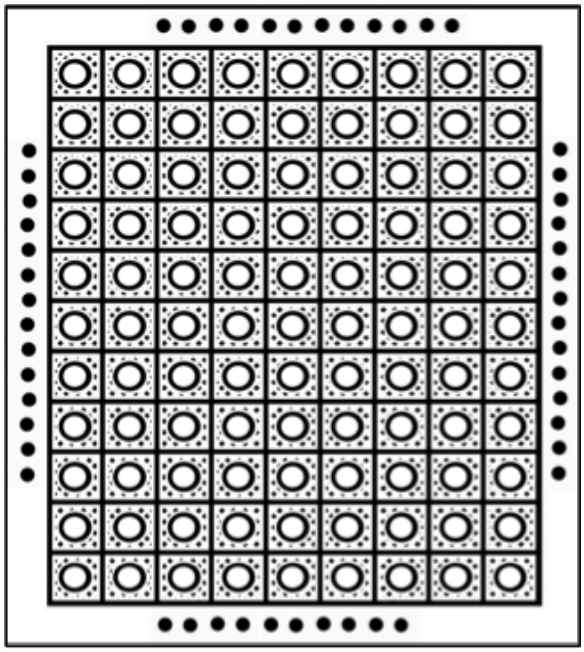
2.4.3 Image Level Geometry Modification

Depending on the level of complexity of the CAD model, slicing and processing the stack of build images can take a substantial amount of time. For turbine mold geometry, it currently takes two days to prepare the data. Sometimes, during evaluation of various CAD designs, there arises a need to build several designs with minor differences in the features. In such cases, it would be beneficial to implement algorithms that can take a base design and implement the minor feature additions and modifications directly at the image level to generate the build images for each of the intended CAD configurations. This would save enormous amounts of time and effort as it avoids re-slicing the entire CAD model. Such algorithms are currently under development on need-by-need basis.

(a) Single Slice.



(b) Background Template.



(c) Tiled Image.

Figure 30: Tiling.

2.5 Summary

A detailed overview of the various data processing algorithms developed for enabling the basic functionality of the LAMP process was given in this chapter. The algorithms presented belong to two categories: slicing and post-processing. A brief summary of the work presented in this chapter is given in this section.

Owing to the geometric complexity of the parts encountered in the LAMP process, a direct slicing approach using the ACIS kernel was implemented to slice the native CAD geometry instead of the conventional STL slicing approach ubiquitously used in the additive manufacturing industry. Prior direct slicing approaches presented in the literature tout direct slicing to be the cure for all ills posed by STL files. Direct slicing is claimed to be error free and fast while STL slicing is claimed to be prone with errors and time consuming. It was shown in this chapter that this claim is only true while working with simple geometries as is the case with much of the previously reported work. When the geometries are complex, direct slicing approach produces more errors and consumes more time than STL file slicing. The direct slicing algorithm presented in this chapter is tolerant to such errors.

Although direct slicing approach is the preferred method to produce slice data for the LAMP process, STL file slicing algorithms were also implemented owing to the prevalence of this file format. Multiple approaches to slice files of this format were implemented. The STL slicing approach using POVRAY, a graphics rendering engine, was easy to implement but not accurate enough for the purposes of LAMP. An approach to reconstruct topology information from an STL mesh using an extended version of the corner table data structure [99, 105] was implemented. However, for complex parts like the ones built in LAMP, the SAT files sizes resulting from this method proved to be too large to handle. Therefore, a more direct approach of reading the facet data and sorting it into data structures similar to the ones reported in literature was implemented. While these data structures yielded reasonably quick

slicing times for small parts, for high facet counts of the order of 5.5M that is typically required for the LAMP process, they take too long to process (4 or more days). Thus a much faster approach which bypasses the facet data sorting operation is implemented and found to reduce the processing time by several orders of magnitude (from 4 or more days to about an hour).

Following the slicing operations, the output data needs to further processed before it is ready for LAMP builds. For this purpose, several post processing like error checking, part placement and tiling, image level geometry modification etc. were implemented and the details of these were presented. In summary, the work presented in this chapter establishes the basic data processing flow required to produce successful builds using the LAMP process.

CHAPTER III

COMPUTATIONAL SCHEMES FOR IMPROVED PART QUALITY

3.1 Introduction

The data processing algorithms described in Chapter 2 were developed to meet the basic requirements of the LAMP process. In addition to these algorithms required for the basic LAMP process functions, several computational design schemes have also been implemented to improve the quality of parts produced by the LAMP process. Details of these algorithms are given in this chapter. The algorithms described in this chapter aim to solve two fundamental problems that arise in LAMP as well as in many other layered manufacturing techniques. The first of these is the problem of "stair-stepping" and the resulting surface roughness that is manifested in 3D-printed parts. When parts are built layer-by-layer, the slice contours are computed at discrete Z-values along the height of the part and the three-dimensional geometry in between the successive layers is approximated by a 2.5D slab obtained by extruding the slice contours down by the layer thickness amount. This effect is illustrated in Figure 31.

As can be seen from the figure, the boundary of the CAD model is not accurately reproduced in the additively manufactured part. In order to solve this problem and accurately reproduce the CAD model surface, one approach that is widely reported in the literature is adaptive slicing. There have been several different approaches for adaptively slicing STL meshes and CAD models presented in the literature. In this thesis, a new simpler method for adaptively slicing CAD models is presented. A brief survey of the prior work in this field and the details of the new adaptive slicing algorithm are presented in Section 3.2.

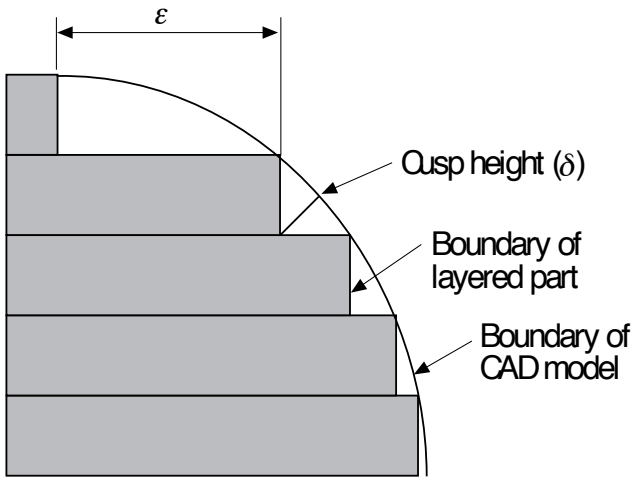


Figure 31: "Stair-Stepping" effect occurring in layered manufacturing [1].

Another approach that can be used to solve the "stair-stepping" problem involves using some novel techniques to alter the cure depth locally within each layer to accurately represent the inter-layer 3D geometry of the CAD part. A few approaches for doing this have been proposed in the literature recently. A gray-scaling and dithering approach is used in this thesis to accomplish the same. A brief survey of the prior work and the details of the gray-scaling and dithering approach are presented in Section 3.3.

In addition to the "stair-stepping" problem, another fundamental problem addressed in this chapter aims to solve is the dimensional accuracy of the built parts. As previously described, LAMP intends to build molds and cores for casting HP turbine blades that have very stringent restrictions on dimensional tolerances. Hence the dimensions of the parts produced through LAMP need to be at least as accurate as those used in current foundry practice. In order to accomplish this objective, experimental studies have been conducted on how the part dimensions vary with respect to some important LAMP build parameters. Details of these experiments and a discussion of the results are presented in Section 3.4.

3.2 Adaptive Slicing

In additive manufacturing, parts are usually built with a constant layer thickness. As previously mentioned, the three-dimensional part geometry is approximated by stacking these 2.5D layers which leads to the "Stair-Stepping" effect illustrated in Figure 31. This effect becomes even more prominent at locations where the CAD surface is highly slanted (i.e, inclining away more steeply from the build direction; there would be no stair stepping effect for vertical surfaces that are oriented with the build direction). Hence, in order to mitigate this problem, a variable layer thickness is used instead of using a constant one. This layer thickness is varied based on the local part geometry at each Z-location along the height of the part being built. This approach of varying layer thickness based on the part geometry is known as adaptive slicing. Figure 32 illustrates the difference between conventional slicing and adaptive slicing while slicing a hemispherical part.

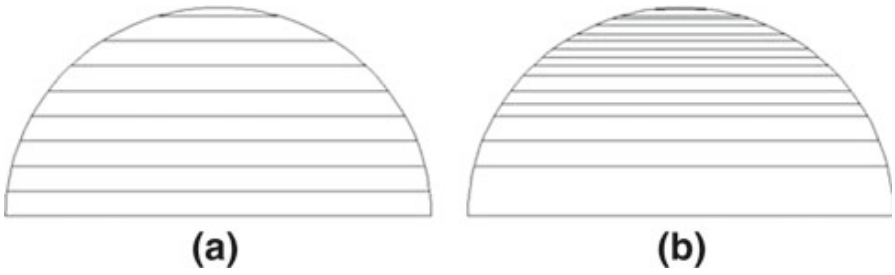


Figure 32: Illustration of slicing approach (a)Uniform Slicing (b) Adaptive Slicing, [11].

A substantial amount of prior work has been reported in the field of adaptive slicing. A survey of this work is given next.

3.2.1 Literature Review

3.2.1.1 Adaptive slicing of STL meshes

Dolenc and Makela [108] were one of the first to introduce the concept of adaptive slicing. They defined a concept called "cusp height" as a measure of the deviation

of the part geometry caused by the stair stepping effect. Cusp height denotes the maximum length of the deviation along a vector normal to the facet surface in each layer and the maximum allowable layer thickness is determined by placing an upper bound on this deviation. Figure 33 shows the illustrates the concept of cusp height.

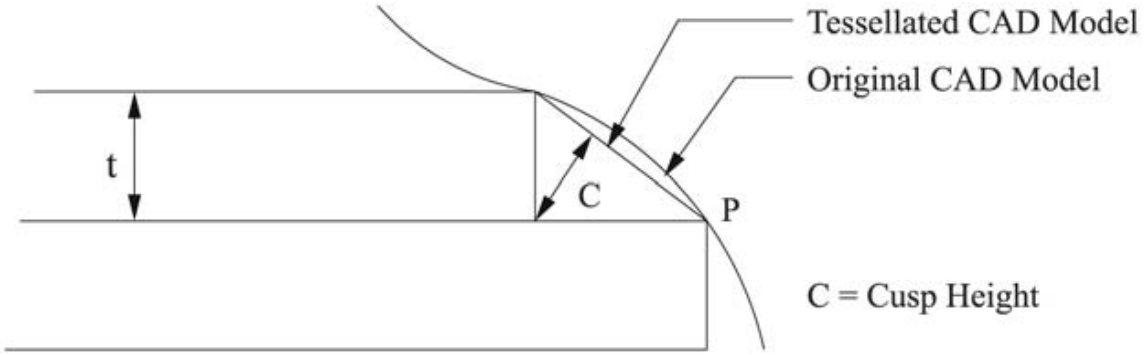


Figure 33: Illustration of Cusp height, [2].

For a tessellated STL mesh, the methodology for calculating cusp height and the adaptive layer thickness as discussed by Dolenc and Makela [108] is shown below. If the normal vector at any point P is given by

$$N = (N_x, N_y, N_z) \tag{7}$$

then the corresponding cusp vector C is given by

$$C = cN \tag{8}$$

where c is the magnitude of the cusp vector, i.e, cusp height. If the maximum allowable cusp height is set as C_{max} then,

$$|C| = c \leq C_{max} \tag{9}$$

The layer thickness t is obtained by using the following expression:

$$t = \begin{cases} \frac{C_{max}}{N_z} & \text{if } N_z \neq 0 \ \& \ t_{min} \leq \frac{C_{max}}{N_z} \leq t_{max} \\ t_{min} & \text{if } \frac{C_{max}}{N_z} < t_{min} \\ t_{max} & \text{if } \frac{C_{max}}{N_z} > t_{max} \end{cases} \tag{10}$$

where t_{min} and t_{max} are the minimum and maximum layer thicknesses respectively, that can be successfully implemented in a specific additive manufacturing technique. In this manner, Dolenc and Makela [108] computed the layer thickness t to adaptively slice an STL mesh.

Several other researchers used this cusp height concept for achieving various goals through adaptive slicing. Sabourin et.al. [109] implemented adaptive slicing using stepwise uniform refinement where a tessellated CAD model was first divided into uniform, horizontal slabs of thickness equal to the maximum acceptable layer thickness, t_{max}. Out of these horizontal slabs, those slabs that do not satisfy the maximum cusp height requirement ($c < C_{max}$) are further subdivided into finer slabs having uniform thickness. Interpolation is used for slice thickness determination. In order to reduce the likelihood of missing high-curvature regions, a slab was examined both from its bottom slice looking upward and from its top slice looking downward and the succeeding slice is subdivided unlike the previous method used by Dolenc [108] where they only examined the bottom slice. The basic measure of slice thickness in Sabourin's work was same as Equation 26. The number of subdivisions (uniform thickness) to be made on a required slab are obtained as follows:

$$\alpha_{slab} = int \left[\frac{t_{max}}{C_{max}} \ max(N_{z\ bottom}, N_{z\ top}) \right] \tag{11}$$

$$\alpha_{slab} \in [1, \alpha_{max}], \alpha_{slab} = int \left(\frac{t_{max}}{t_{min}} \right) \tag{12}$$

where $N_{z\ bottom}$ and $N_{z\ top}$ are the sets of unit normal z components for P_i across

the bottom and top levels of a particular slab respectively. The resulting uniform layer thickness within a particular slab is therefore given by

$$t = \frac{t_{max}}{\alpha_{slab}} \tag{13}$$

Tyberg and Bohn [3] have used the cusp height concept to achieve local adaptive slicing of parts. More often than not, in CAD parts there are multiple features with varying curvatures at the same slice height and each of them would require a different slice height to satisfy the maximum cusp height criterion. Hence, instead of using the minimum layer thickness dictated by the feature with maximum curvature for each layer, each feature is sliced independently of the other based on its unique cusp height requirements and thereby achieve local adaptive slicing. First, a tessellated CAD model is sliced into uniformly thick slabs using the maximum thickness available in an additive manufacturing system. The resulting contours belonging to the slabs top and bottom slices are then matched using topological information to form a set of sub-slabs. Finally, each sub-slab is independently divided into a distinct number of thinner layers based on the vertical slope of its surface or surfaces, measured along the contours. Different tests namely orientation, multiple part, proximity, direct, indirect and virtual tests are carried out to establish vertical connectivity between the two contours. After sub-slabs identification, each one is further subdivided independently into an integer number of uniform layers using the stepwise uniform refinement procedure proposed by Sabourin [109]. Figure 34 shows typical parts sliced by this method.

It can be seen from Figure 34 that the first build consists of a single part (Figure 34(a)), containing a base which branches into three different features of varying curvature. The second build also consists of a single part (Figure 34(b)), having vertical surfaces and three different features. The build shown in Figure 34(c) is a combination of builds shown in Figure 34(a) and 34(b). Each build was sliced and

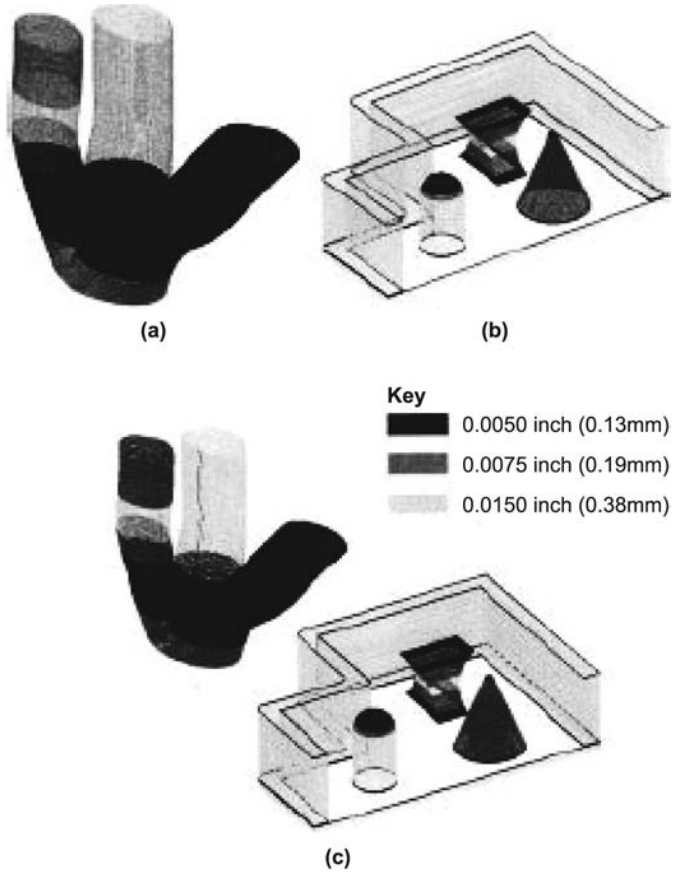


Figure 34: Typical parts sliced by local adaptive slicing, [3].

fabricated with the three-layer thickness. These layer thickness are shown in Figure 34 with different shades. It is concluded by them that the build time can be reduced up to 73, 42 and 55 percent for build shown in Figure 34(a)-(c), respectively, using local adaptive slicing.

Sabourin et al. [4] proposed an adaptive slicing algorithm that aims to achieve accurate exterior surface finish in a part with finer layers while building the interiors of the part quickly by using thicker layers. Hence, the interior layers are not of the same thickness as the exterior layers, but several times thicker as shown in Figure 35.

The process of fabrication in 3D space using thick and thin layers is coordinated by first slicing the tessellated CAD model into thick horizontal slabs of uniform thickness. These slabs are then processed from the bottom to the top of the part and the contours

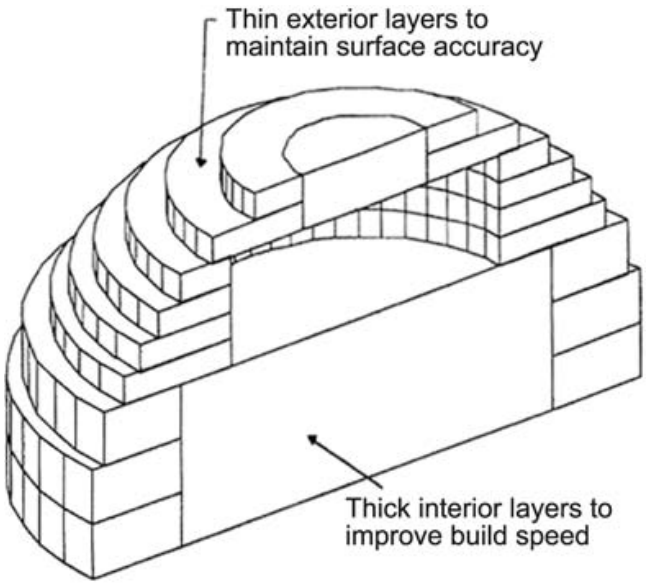


Figure 35: Concept of accurate exterior and fast interior slicing, [4].

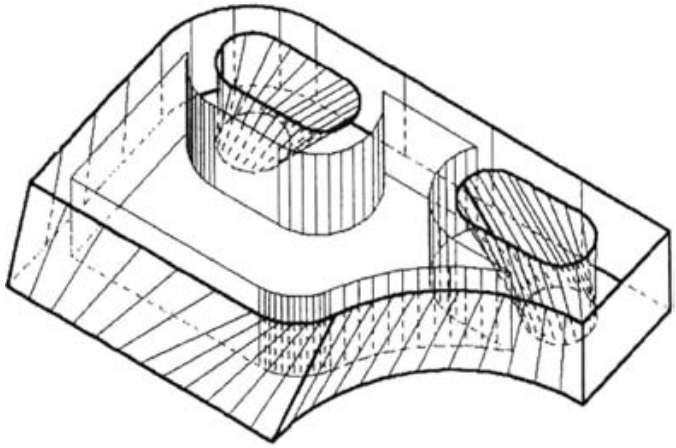


Figure 36: Contours offseted inward to implement accurate exterior and fast interior slicing procedure, [4].

on the top and bottom of each slab are offset inward (shown in Figure 36) into the part in the horizontal plane to form a new set of contours that separate the exterior and interior regions within the slab. These two sets of regions can be built in a sequence. First, the exterior region is built with thin layers and then the interior regions are backfilled with thick layers to complete the fabrication of the part. Accurate exterior and fast interior slicing procedure is important as it could reduce the build time of

the prototype from 50 to 80 percent.

Tata et al. [106] proposed yet another strategy for adaptively slicing STL files by defining and making use of a new parameter called surface complexity. They defined Surface Complexity as $S = |tan\theta|$, where θ is the acute angle between the slice axis (Z) and its projection on the surface. The layer thickness for a zero complexity surface is chosen such that the fastest build time can be achieved i.e. the maximum allowable fabrication thickness for a additive manufacturing system. For complex surfaces having nonzero complexity the *criterion of maximum deviation* ($t * tan\theta$) and *chord length* ($\frac{t}{cos\theta}$) are considered as constraints for slicing. Maximum deviation criterion is expected to be useful when it is necessary to contain the deviation of a layered model within the predetermined value. Chord length as a criterion is expected to be useful, when the deviation of a layered model is compared with the original CAD model. Backtracking is introduced to meet the sudden changes in surface complexity levels. A comparison of the number of layers obtained and estimated build time reduction by adaptively slicing the same file using each of these criteria when compared to uniform slicing is given. A fourth criterion based on the volumetric error was also postulated. A layered model can lose or gain volume over the tessellated model based on whether the layer error is positive or negative. The volumetric error is a function of perimeter (p), layer cross sectional area (A), layer thickness (t) and surface complexity (S). Hence, it was noted that maintaining the volumetric error constant between layers is not possible because of the possibly large variations in the perimeter from layer to layer. Therefore, volumetric error per unit length was proposed as a criteria that could work. This new metric of volumetric error per unit length was merely conjectured to work but not implemented to verify its effectiveness for adaptive slicing.

Cormier et al. [110] presented an adaptive slicing approach using non uniform cusp heights. Most of the adaptive slicing algorithms presented earlier assume a maximum

allowable cusp height, which applies to the entire part. As far as the application of the part is concerned, it may not have uniform cusp height requirement everywhere. Some faces of the part are required to be smooth while other faces are relatively unimportant. This procedure uses tessellated CAD model as an input. The edges are found by edge finding algorithm and grouping of facets is carried out. The facet model of the part to be sliced is rendered. The designer has the option of specifying the maximum allowable cusp heights for different faces as per the functional requirement of the part.

Pandey et al. [2] have developed an adaptive slicing procedure by considering parabolic build edges for each layer. There have been a number of attempts to develop adaptive slicing procedures for tessellated CAD models and the slice edge profiles are implicitly assumed rectangular. Pandey et al. [2] proposed a slicing algorithm based on the realistic edge profile of fused deposition modeling (FDM). In this work, the R_a value is used as a user specified parameter instead of the cusp height. The major advantage of using the R_a value in the place of cusp height is that Ra value is most commonly used in integrated design and manufacturing. In real practice, the edge profiles of a layer manufactured part using FDM is found to be parabolic. A stochastic model is developed for surface roughness (R_a value) prediction. The R_a value of the part was reported to be dependent on the layer thickness (t) and build orientation (θ) as given Equation 14 for a 99 percent confidence limit:

$$R_a(\mu m) = (69.28 - 72.36)\frac{t(mm)}{cos\theta} \tag{14}$$

They studied the effect of radius of curvature on R_a and concluded by using a neural network model that the radius of curvature effect on surface roughness can be thought of independent as it varies within 5 percent over a wide range of curvature values. The layer thickness for a maximum specified Ra value on the part surface is calculated using the following expression:

$$t(mm) = \frac{R_a(\mu m)\ cos\theta}{70.82} \tag{15}$$

3.2.1.2 Adaptive Slicing of CAD models

Adaptive slicing of a CAD model was first implemented by Suh and Wozny [5]. First the given CAD model is inspected for some special geometric features and into distinct slicing regions. Then, for each slice plane height in these regions, a sampling of points are taken on the corresponding slice contours. Local curvature (κ) of the bounding surfaces at each of these points are then computed and the surface is approximated by a sphere with the corresponding radius (radius of curvature $\rho = 1/\kappa$). Figure 37 illustrates this process of approximation of the surface with a sphere.

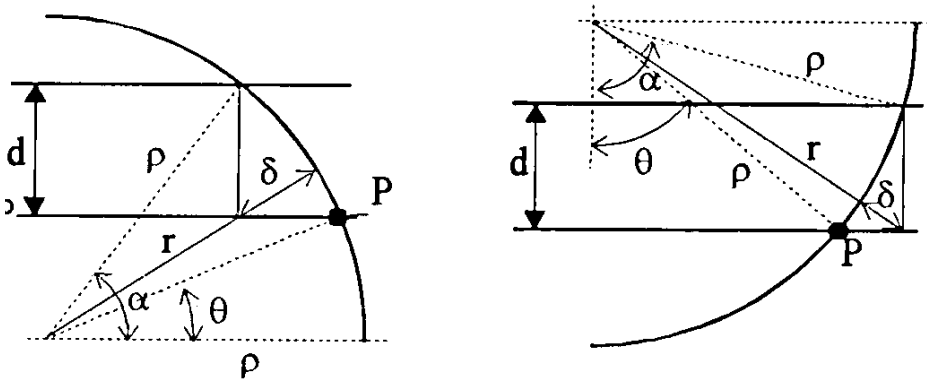


Figure 37: Approximation of the CAD part boundary with a sphere , [5].

Once the surfaces are approximated by spheres at each of the sampled points, geometric expressions that relate the cusp height (δ), radius of curvature (ρ) and surface normal angle w.r.t build direction (θ) to the layer thickness (d), at a point P were developed. Equation 16 gives the expression for computing the next layer thickness for the case when point P lies on the northern hemisphere and Equation 17 gives the expression for the case of point P lying on the southern hemisphere.

$$d = -\rho \sin\theta + \sqrt{\rho^2 \cos^2\theta - 2\delta\rho - \delta^2} \tag{16}$$

$$d = \begin{cases} -\rho\cos\theta - \sqrt{\rho^2\cos^2\theta - 2\delta\rho - \delta^2} & \text{if } \rho^2\cos^2\theta - 2\delta\rho - \delta^2 > 0 \\ \rho\cos\theta & \text{otherwise} \end{cases} \tag{17}$$

Using these expressions, in order to satisfy the cusp height criterion, the maximum allowable layer thickness at each of the sampled points is calculated. The minimum layer thickness out of all these calculated values is chosen as the new layer thickness height. This process is repeated for each slice in the part for adaptively slicing over the full height.

Kulkarni and Dutta [6] extended the approach given by Suh and Wozny [5] for adaptively slicing a model bound by parametric surface patches. They developed detailed expressions for computing the normal vertical curvature at any arbitrary point on the slice contour of the part bounded by parametric surfaces. Moreover, they extended the approach of adaptive slicing to also address the containment problem that also occurs in layered manufacturing. Figure 38 illustrates the containment issue that arises in the fabrication of a sphere where S denotes the closed curve representing the 2D profile of the sphere and S' denotes the stepped approximation obtained after layered manufacturing.

The following three containment situations can occur as illustrated in Figure 38(a), Figure 38(b) and Figure 38(c) respectively:

(a) $S \subset S'$

(b) $S' \subset S$

(c) $S \not\subset S'$ and $S' \not\subset S$

Depending on the intended application of the additive manufactured part, it would be desirable to impose the containment situations (a) or (b) but not (c). Situation (a) implies extra material has been deposited in the additive manufactured part and this enables a polishing operation to improve surface finish. This situation was referred to as 'positive tolerance'. If however, the additive manufactured part is to act as a

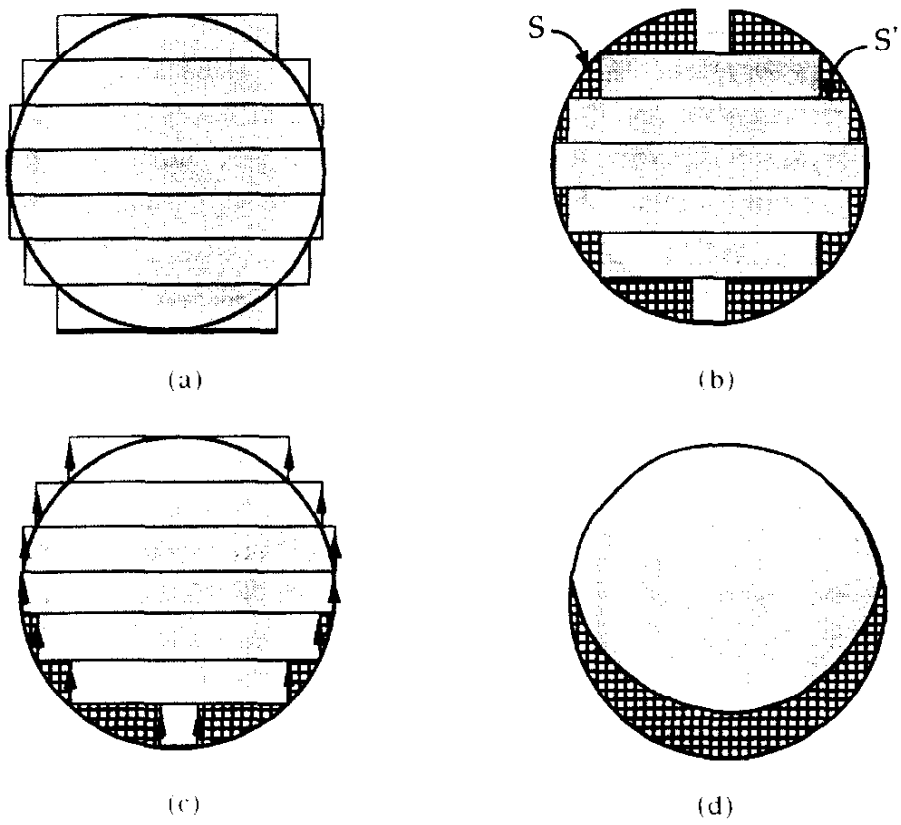


Figure 38: Illustration of the containment issue in additive manufacturing, [6].

master pattern for making cores, then situation (b) would be desirable since a smaller core would imply extra material in the cast part which would in turn enable surface finishing operations on it. This situation was referred to as 'negative tolerance'. The situation (c) is not attractive in most applications. But using a naive slicing strategy, this is the situation that results and if a surface finishing operation were to be performed on the sphere in Figure 38(c) one would end up with a distorted shaped sphere as shown in Figure 38(d). Hence, in order to address this problem, Kulkarni and Dutta [6] developed several expressions similar to the ones shown in Equations 16 and 17 to ensure that in each scenario, one would precisely obtain negative or positive tolerance as specified by the designer. In addition to this, to avoid the possibility of missing regions of high curvature as is the case in sampling points, an optimization

approach was implemented where the entire contour was examined for identifying the minimum layer thickness. The optimization problem is formulated as follows:

$$\text{maximize } \kappa(u, v) \tag{18}$$

$$\text{subject to the constraint } z(u, v) = constant$$

where u and v are parameter values of the surface patches defining the CAD model. It was noted that the layer thickness is indeed a function of both the curvature κ ($\kappa = 1/\rho$) and the angle θ (as can be seen from similar expressions in Equations 16 and 17). Hence, a better optimization approach in which the layer thickness d is minimized is formulated as shown in Equation 19 and was demonstrated to yield much better results.

$$\text{minimize } d(u, v) \tag{19}$$

$$\text{subject to the constraint } z(u, v) = constant$$

The optimization problem was solved by implementing a sequential quadratic programming algorithm in MATLAB.

Hope et al [1, 7] used a similar approach to adaptively slice CAD models but instead of assuming rectangular build edges (2.5D layers) like before, sloping build edges were considered for a better approximation of the part. The main advantage achieved is improved surface finish and decreased build time as thick layers can be used. In this work, multiple B-spline surfaces from IGES format were used to define a part. They sliced the model by tracing surface contours, and computing the cutting direction at a number of points as specified by the designer. These points and corresponding cutting vectors are used to generate computer numerical control (CNC) code to machine the slices. Figure 39 shows the error between the CAD model, cut layers and cutting vectors.

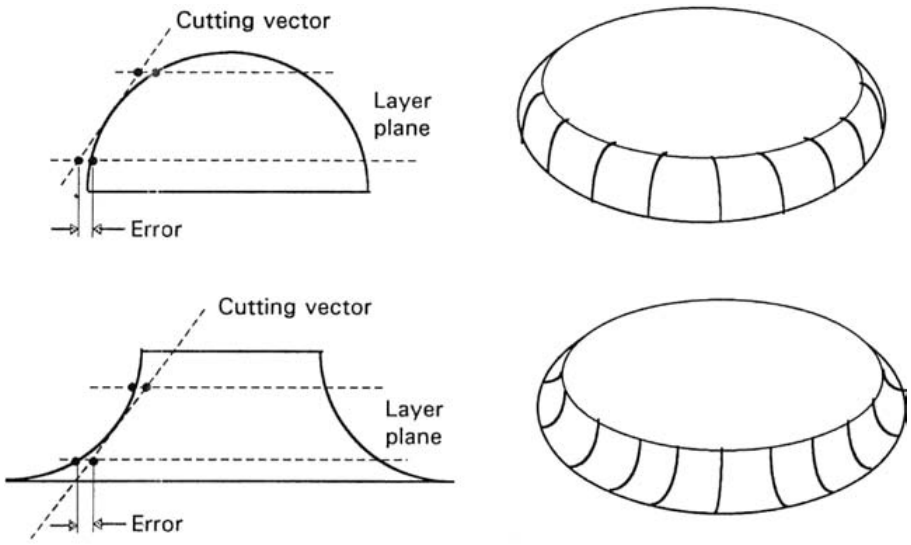


Figure 39: Error between CAD model, cut layer and cutting vector, [7].

Two types of errors namely cusp height and maximum difference in the plane layer are introduced as shown in Figure 40.

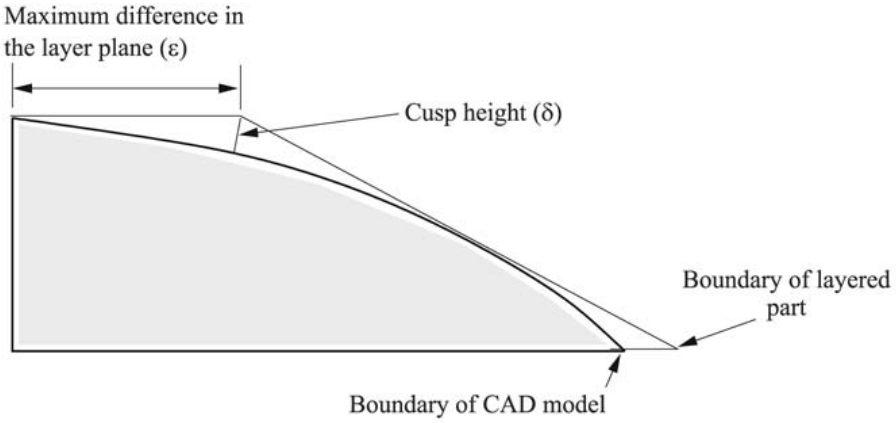


Figure 40: Errors in sloping surfaces consideration of build edges, [1].

The expression for cusp height (δ) and for maximum difference in the layer plane (ϵ) (refer to Figure 41) is shown below:

$$\delta = \left[\left(\frac{t}{2\cos\alpha} \right)^2 + R_c^2 \right]^{\frac{1}{2}} - R_c \tag{20}$$

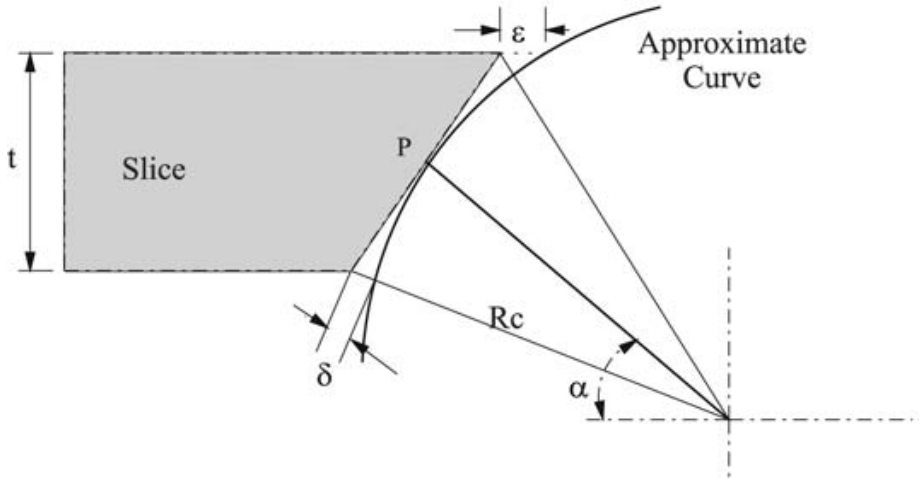


Figure 41: Geometric computation of slice height for sloping surfaces, [1].

$$\epsilon = (\delta + R_c)\cos\phi - \left[\{(\delta + R_c)\cos\phi\}^2 - \left(\frac{t}{2\cos\alpha}\right)^2\right] \tag{21}$$

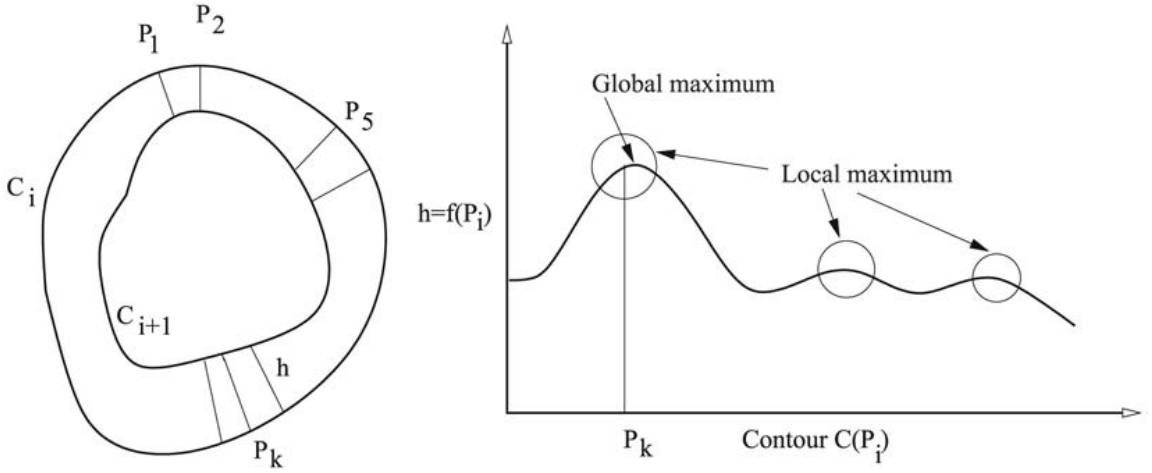
where

$$\phi = \alpha \pm \arctan\left\{\frac{t}{2R_c\cos\alpha}\right\} \tag{22}$$

here R_c is the radius of curvature, t is the layer thickness and α is the angle between the horizontal and normal. They have provided a choice between using the cusp height (δ) and maximum difference in the layer plane (ϵ) as the measure of error. In adaptive slicing, module slicing is performed and the CNC code is generated to machine them. These slices are glued together to form the prototype.

Lee and Choi [8] combined the work of Dolenc and Makela [108] for polyhedral features and Kulkarni and Dutta [6] for freeform features. The computing time is reduced by introducing the concept of sampling points at z = constant for solving the optimization problem to determine the perpendicular distance between two consecutive slices projected on the same x-y plane. In place of minimum slice thickness, they minimized the perpendicular distance between the two consecutive slices projected on the same x-y plane. They were able to reduce the computational time further by introducing the concept of vertical character line i.e. the curve that connects

the optimum sampling points, P_k, where the perpendicular distance between the two contours C_i and C_{i+1} is maximum. The character line, therefore passes through the points where the slope of the model is most moderate for each slice as shown in Figure 42. It is important that the optimum sampling point is found out rapidly and exactly (by taking thicker slabs in the beginning) in this adaptive slicing procedure and character line is established. If the optimum sampling points are located on or close to the character line, their locations can be predicted in advance.



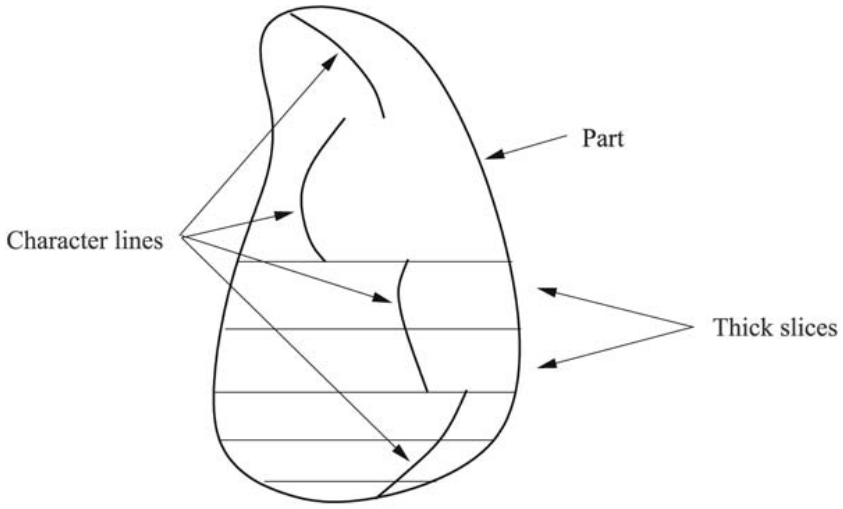


Figure 42: Sampling of points and vertical character lines, [8].

Many researchers implemented fixed cusp height restriction on the entire part.

Flexibility of imposing different cusp heights on the different surfaces of the model was implemented by Mani et al. [9]. A similar concept of specifying nonuniform cusp heights on different faces of a model is also implemented by Cormier et al. [110] for tessellated CAD models. The difference between traditional adaptive slicing and region-based adaptive slicing is shown in Figure 43.

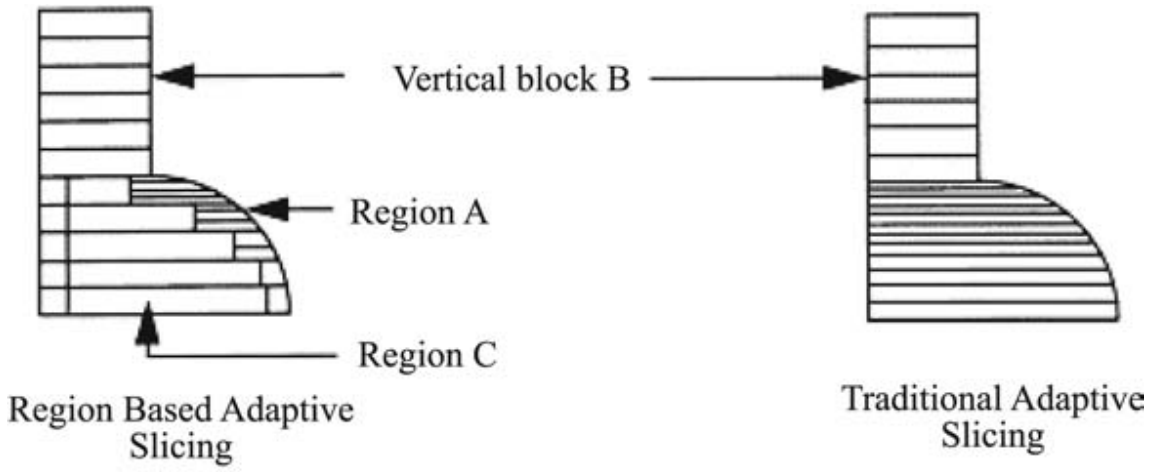


Figure 43: Difference between traditional and region-based adaptive slicing, [9].

The region-based slicing procedure was implemented in four steps. In the first step, critical surfaces are identified for cusp height requirements, spatial decomposition of the model into adaptive layer thickness and the common interface layer region is then carried out. The common layer region (region C, Figure 20) is built with maximum possible fabrication thickness on RP system and adaptive layer thickness region (region A, Figure 20) is sliced based on the geometry and cusp height requirement. The process of decomposition of the model into two regions is almost similar to fast interior and accurate exterior implemented by Sabourin et al. (1997). Finally, the sequence of layer deposition is decided.

Jamieson and Hacker [10], and Zhao and Laperriere [88] implemented an adaptive slicing approach based on the limited area deviation. The area deviation is defined as

$$\sigma = \left| \frac{A_{i+1} - A_i}{A_i} \right| \leq \delta \tag{23}$$

where A_i and A_{i+1} are areas of two consecutive slices as shown in Figure 44. In this

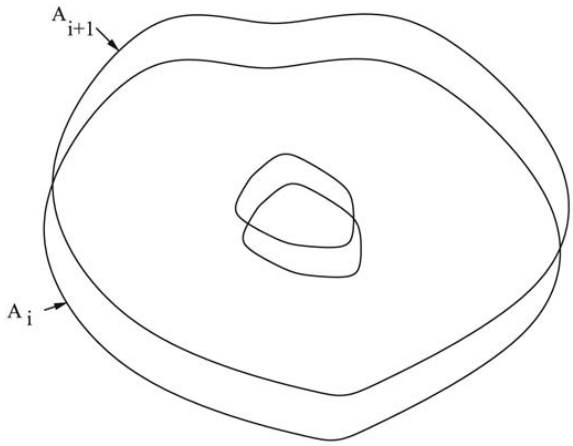


Figure 44: Areas of two consecutive slices in area deviation method, [10].

work, δ is considered as 5 percent. The thickness of a layer always remains between the maximum and minimum fabrication thickness available in the RP system. This area deviation is only a two dimensional measure and does not give a true estimate of the geometry deviation from the surface of the original CAD model. Hayasi and Aisabanpour [11] proposed an adaptive slicing approach that tries to rectify this problem. For each slice, the internal and external contours are projected onto the XZ and YZ planes and the area of the triangular region that contributes to the stair stepping effect is computed to get a true measure of the geometry deviation.

Direct adaptive slicing of axisymmetric parts is also implemented by Pandey et al. [2]. Slice thickness was calculated using expression the given in Equation 15. Their work predicts the average part surface quality in terms of standard R_a value. The direct adaptive slicing software also gives cusp height (Dolenc and Makela, 1994) and relative area deviation (Jamieson and Hacker [10], Zhao and Laperriere [88]) for two consecutive slices. They drawn the conclusion that for most RP processes, the R_a value is proportional to $t/\cos\theta$. The constant of proportionality will be different

for different RP processes. Therefore, this procedure can be used for different RP systems.

3.2.2 Volume Deviation based Adaptive Slicing

A new volume deviation based adaptive slicing method for BRep models is developed in this thesis. BRep stands for Boundary Representation, a kind of data structure widely used for storing CAD model data. The ACIS modeling kernel used for implementing the direct slicing algorithm discussed in Section 2.2 uses the BRep format and so do many commercially available CAD softwares packages like Solidworks, Pro/Engineer, CATIA, Unigraphicx etc. The motivation for this approach and the details of the algorithm are presented in this section.

3.2.2.1 Motivation

Although several adaptive slicing approaches have been presented in the literature as discussed in the previous section, there are some limitations especially in the case of adaptively slicing BRep models. The mostly widely used approach of maximum cusp height criteria for adaptive slicing works well for STL files due to the simplicity of facet data. However, it becomes extremely complicated and computationally intensive when implemented for slicing direct CAD models. The approach presented by Kulkarni and Dutta [6] for using the cusp height criterion in slicing CAD models is a good illustration of this complexity. For calculating the cusp height for each layer, a complex optimization scheme is implemented (as defined in Equation 19) which involves calculating the vertical normal curvature at each point in the slice contour (which in itself needs several complicated calculations of local tangent and normal vectors) and these computations take a long time. Moreover, the approach only works for parametric surface patches like B-spline and NURBS and are hence not readily extensible to generic BRep CAD models which are defined by a mixture of parametric splines and analytic curves.

Another approach presented in the literature for adaptively slicing CAD models uses an area deviation approach as reported by Jamieson and Hacker [10] and Zhao and Laperriere [88]. This approach, while being much more simple than the cusp height approach for CAD models, is fraught with difficulties as it does not consider the local surface geometry of the part. It leads to situations where the actual geometric deviation of the additive manufactured part with respect to the original CAD geometry is not properly estimated as shown by the example in Figure 45. As shown

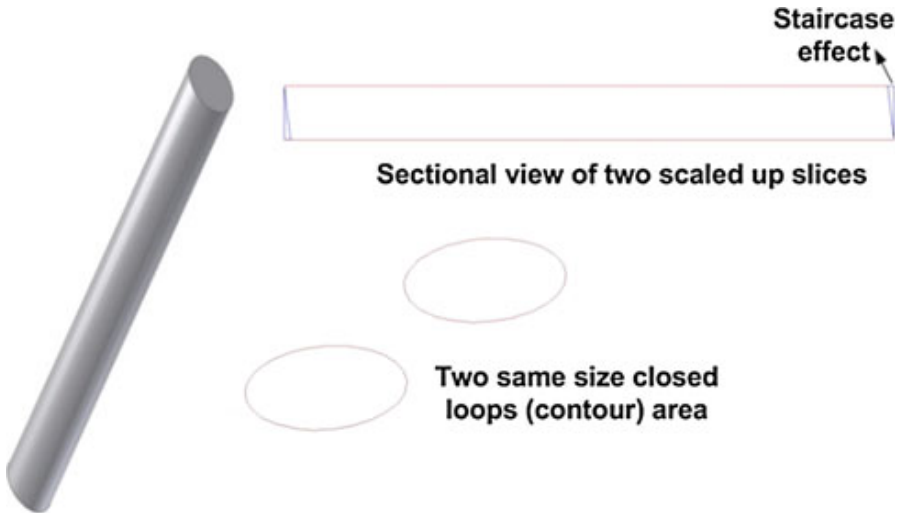


Figure 45: The staircase effect while two created contours are same size in inner area, [11].

in Figure 45, if the CAD geometry were an inclined cylinder, there would still be a staircase effect in the layered part while having slice contours of exactly the same area. In such a situation, the area deviation approach would fail to identify the geometry deviation due to staircase effect and hence layer height adaptation will not be achieved.

The other major approach presented in the literature for adaptively slicing CAD models uses a maximum bound on the surface roughness parameter R_a as reported by Pandey et al. [2]. This approach is also simple to implement as a closed form expression relating the layer thickness to the surface roughness R_a for determining

the next layer height to be used. However, a substantial amount of empirical and statistical modeling for identifying the cured layer shape and its relation to the surface roughness parameter R_a for a given additive manufacturing technique needs to be performed.

In order to overcome these limitations and to accomplish the ultimate objective of adaptively slicing a generic BRep model with a reasonable simplicity, the volume deviation-based adaptive slicing technique has been developed and implemented. Details of this technique are given next.

3.2.2.2 Approach

In the volume deviation-based approach developed in this thesis, the entire volume of the cusp is calculated in order to use as a measure for estimating the geometric deviation of the layered part. The concept of cusp volume is illustrated in Figure 46, where the geometric deviation between a hemispherical part and the corresponding

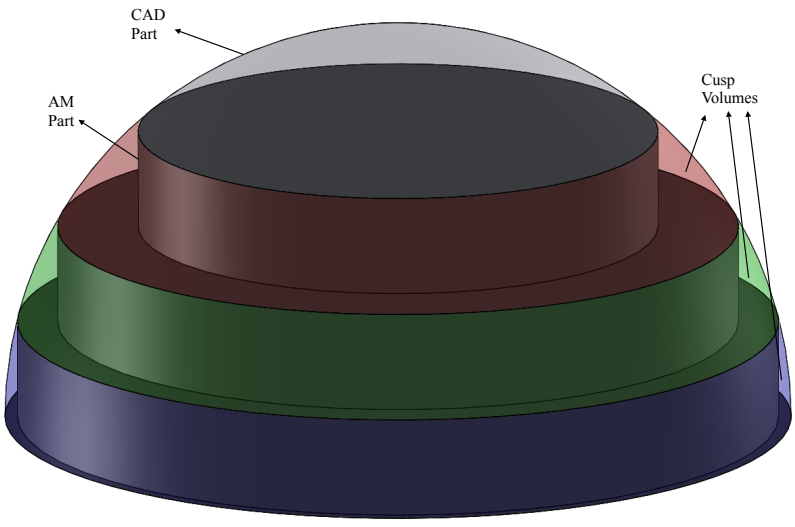


Figure 46: Illustration of Cusp Volume for a hemispherical part. Cusp volumes corresponding to each layer are colored different for clarity.

additive manufactured part is highlighted. As is clear from the illustration, the geometric deviation of each 2.5D layer with the corresponding 3D layer of the part is labeled as the cusp volume.

In order to adaptively slice the part, at each slice height, the cusp volume is computed and used as a measure to determine the height of the next layer. However, as observed by Tata et al. [106], the cusp volume can dramatically change over the height of a part as it is a function of the perimeter of the cross-section, layer thickness and the angle made by the local surface tangent vector with the build direction. Hence, it is not possible to use a constant cusp volume criterion for determining the layer thickness. In order to alleviate this difficulty, the cusp volume is normalized by the volume of the 3D slice of the part to give an estimate of the percentage volumetric error within each layer of the additive manufactured part as compared to the original geometry as shown in Equation 24.

$$\%VolumetricError = \frac{Cusp\ Volume}{3D\ Slice\ Volume} \tag{24}$$

An upper bound on the $\%VolumetricError$ is placed and the layer thickness at each slice height is determined so as to satisfy this upper bound criterion. At each slice height along the length of the part, the $\%VolumetricError$ is first estimated using the maximum possible layer thickness. If the computed error is less than the maximum bound, the maximum layer thickness is used. If not, the layer thickness is successively converged to a value that yields the specified maximum error using a 'bisection' scheme. If the resulting layer thickness is greater than the minimum layer thickness that can be built, then it is used as next layer height. Otherwise, the next layer height is set to the minimum layer thickness and the operation is repeated. Implementation details of this approach using ACIS kernel are given next.

3.2.3 Implementation Details

The pseudo code for the adaptive slicing operation using volume deviation approach is shown in Algorithm 12. The part of the given name is first loaded into the program and all the important ACIS parameters like resolution are set. The bounding box of the part is computed next to identity its min and max extents. Slicing is started at a height just slightly above $minZ$. For each slice height $minZ < z < maxZ$, the slice contour is computed and slice image created just as discussed in Section 2.2.

Algorithm 12 Adaptive Slicing Algorithm.

```
1: load the given part and store in wig
2: compute bounding box and store the bounds in min and max
3: for z:= minZ to maxZ do
4:     create horizontal plane at height z
5:     compute slice by intersecting the plane with the part
6:     create image from slice
7:     z = z + layerThickness(wig, z)
8: end for
```

Once the slice image is created, the slice height of the next layer is determined by passing the part (wig) and the current slice height (z) to a function called $layerThickness$. The pseudo code for this function is shown in Algorithm 13. In this algorithm, three height trackers denoted by low, mid and $high$ are used. For finding the layer thickness at height z, these three trackers are first set to z, 0 and $MAXTHICKNESS$ (denotes the maximum layer thickness that can be built) respectively. First the % volume deviation of the part at height z is computed at the maximum allowable layer thickness. If this deviation is either non zero or greater than the maximum allowable volume deviation denoted by $MAXDEVIATION$, the height of the next layer is adjusted using a scheme similar to the bisection method in root finding [111]

(until the volume deviation is in the vicinity of the maximum allowable volume deviation). So, if at *MAXLAYERTHICKNESS*, the volume deviation is greater than the *MAXDEVIATION*, the height marker *mid* is adjusted to its new value as shown in Equation 25:

$$mid = \frac{high + low}{2} \tag{25}$$

The volume deviation at this new height given by *mid* is then computed. If it is still higher than *MAXDEVIATION*, then *high* is set to current value of *mid* so that in the next iteration, the volume deviation is computed at a lower height. If the volume deviation at the current value of *mid* is lower than the *MAXDEVIATION*, then *low* is set to the current value of *mid* so that in the next iteration, the volume deviation is computed at a higher z height. In this manner, through successive iterations, the value of *mid* converges to a z height where the volume deviation is within a tolerance range denoted by TOL in the vicinity of *MAXDEVIATION*. Once the value of *mid* has converged to a stable value, the next layer height is computed as shown in Equation 26 and returned back.

$$thickness = mid - z \tag{26}$$

This ensures that at each slice height z the part is sliced at the maximum possible layer thickness to satisfy the volume deviation criteria in order to obtain the minimum number of slices for the part thereby reducing the total build time while also maintaining accuracy. The implementation of the function *computeDeviation* which is used for calculating the volume deviation of a given generic BRep part *wig* at a slice height z and a layer thickness (denoted by *thickness*) is fairly simple and easily scalable to parts of arbitrary complexity. The pseudo code of this function showing the various operations that need to be carried out for computing volume deviation is shown in Algorithm 14.

Algorithm 13 computing layer thickness.

```
1: function layerThickness(wig, z)
2:     low ← z
3:     high ← z + MAXTHICKNESS
4:     deviation = computeDeviation(wig, z, MAXTHICKNESS)
5:     thickness = MAXTHICKNESS
6:     mid = 0
7:     while deviation ≠ 0 and |deviation − MAXTHICKNESS| > TOL do
8:         mid = (high+low)/2
9:         thickness = mid − z
10:        if thickness > MINTHICKNESS then
11:            deviation = computeDeviation(wig, z, thickness)
12:            if deviation = 0 then
13:                break
14:            else
15:                if deviation > MAXDEVIATION then
16:                    high = mid
17:                else
18:                    low = mid
19:                end if
20:            end if
21:        else
22:            thickness = MINTHICKNESS
23:            break
24:        end if
25:    end while
26: end function
```

Algorithm 14 computing % Volume Deviation.

1: **function** *computeDeviation*(wig, z, $thickness$)
2: //**compute 3D slice:**
3: $block \leftarrow$ a cuboid of height equal to $thickness$
4: $3DSlice \leftarrow$ solid geometry obtained from intersection of wig and $block$
5: //**compute 2.5D slice:**
6: $slicePlane \leftarrow$ a plane created at height $z + thickness$
7: $contour \leftarrow$ intersection of wig and $slicePlane$
8: $2.5DSlice \leftarrow$ sweep $contour$ down by a distance equal to $thickness$
9: //**compute volume lost:**
10: $cuspVolume1 \leftarrow$ subtract $2.5DSlice$ from $3DSlice$
11: //**compute volume gained:**
12: $cuspVolume2 \leftarrow$ subtract $3DSlice$ from $2.5DSlice$
13: //**compute volume of 3D slice:**
14: $3DVolume \leftarrow$ volume of $3DSlice$
15: //**compute % volume deviation:**
16: $deviation \leftarrow \frac{cuspVolume1 + cuspVolume2}{3DVolume} * 100$
17: return $deviation$ back to the calling function
18: **end function**

First, the 3D slice geometry is computed. In order to do this, a rectangular cuboid of height equal to the given layer thickness (denoted by *thickness*) is created and stored in the variable named *block*. The 3D slice geometry can then be computed by performing a solid intersection operation in ACIS between the given part *wig* and the cuboid denoted by *block*. Next, the 2.5D slice (the geometry of each printed layer assuming rectangular walls) geometry is computed by the following steps

a) creating a slice plane at height $z + thickness$,

b) computing the intersection of the plane with the given CAD part *wig* to get the

2D slice contour

c) sweeping the 2D slice contour vertically down by a distance equal to the current layer thickness.

Once, the 3D and 2.5D slices are computed, the volume lost by the layered part (denoted by $cuspVolume1$) at the given height z and the given layer thickness $thickness$ is determined by performing a subtraction operation in ACIS using the 3D slice as the 'blank' body and 2.5D slice as the 'tool' body and computing the volume of the resulting geometry. Similarly, the volume gained by the layered part (denoted by $cuspVolume2$) is determined by swapping the blank and tool bodies from the previous step and computing the volume of the resulting geometry. Finally, the % volume deviation can be computed as shown in the expression on Line 16 in Algorithm 14, where $3DVolume$ denotes the volume of the 3D slice. Figure 47 illustrates each of these steps for calculating the volume deviation while slicing a sample CAD part.

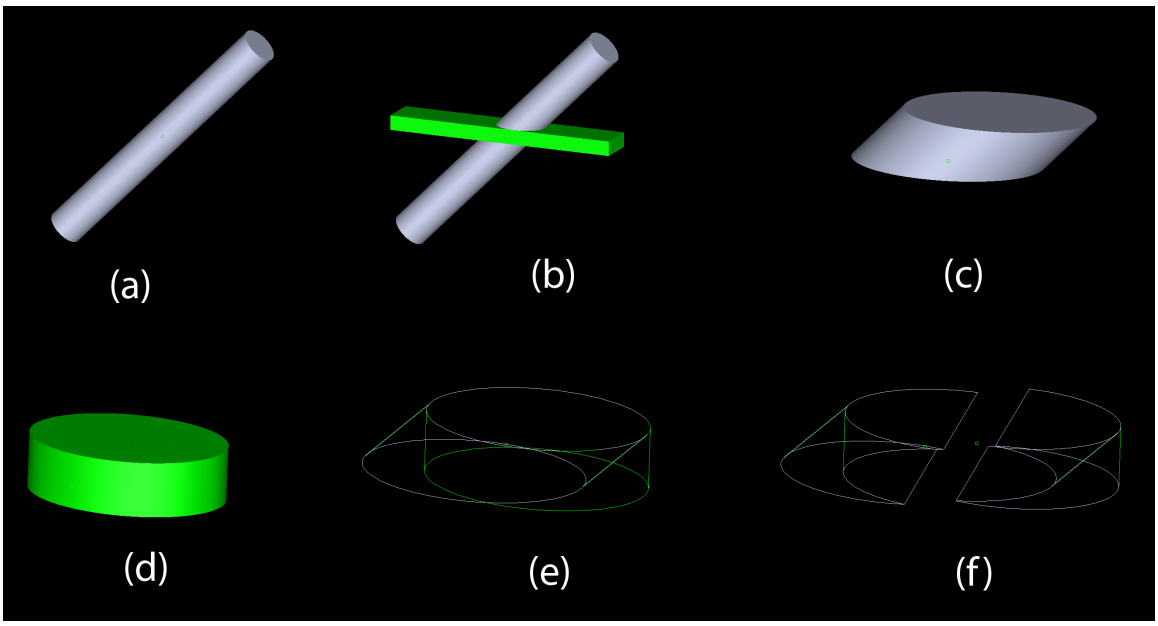


Figure 47: Sequence of steps for computing % volume deviation: (a) Sample CAD Part (b) Intersecting a cuboid with the Part (c) Resulting 3D Slice (d) 2.5D Slice obtained by sweeping the upper 2D slice contour (e) Overlap of the 3D slice and the 2.5D Slice showing the total volume deviation (f) Juxtaposition of volumes gained and lost obtained by subtracting the 2.5D slice from the 3D slice and vice versa.

Since all the operations discussed above are carried out using robust ACIS functions, the algorithm can be easily scaled to BRep parts of arbitrary complexity unlike any of the approaches previously presented. Moreover, it is assumed that each printed layer is 2.5 dimensional with constant cross-section and vertical walls but can be easily extended to a more accurate case if the actual cured layer profile is known to get a better estimate of the volumetric deviation of the printed part with respect to the original part. The results obtained while slicing a sample test part using this approach are presented next.

3.2.4 Results & Discussion

The effect of adaptive slicing on a sample CAD part composed of three different primitives (cylinder, cone and a sphere) and a free form cross-section is shown in Figure 48. The minimum and maximum layer thicknesses used were 0.001 inch and 0.1 inches respectively. A maximum volumetric deviation of 2% is used as the adaptive slicing criteria. The following observations can be made from Figure 48:

(a) The maximum layer thickness of 0.1 inch is used for the region with vertical cylindrical cross-section since the volume deviation for this region is zero.

(b) A more or less constant layer thickness which is smaller than the maximum value is used for slicing the conical section. The slight variation in the layer thickness in this region is caused due to the fact that % volume deviation is a relative measure and it changes with respect to the cross-section location along the height of the part.

(c) For the spherical section, the layer thickness is varied gradually with thickness decreasing towards the top.

(d) For the free form section, the layer thickness is varied continuously, with thickness increasing or decreasing depending on the local surface complexity.

Figure 49 gives a plot of how the layer thickness varies along the height of the part to give a more clear illustration of the observations presented above. It can seen that

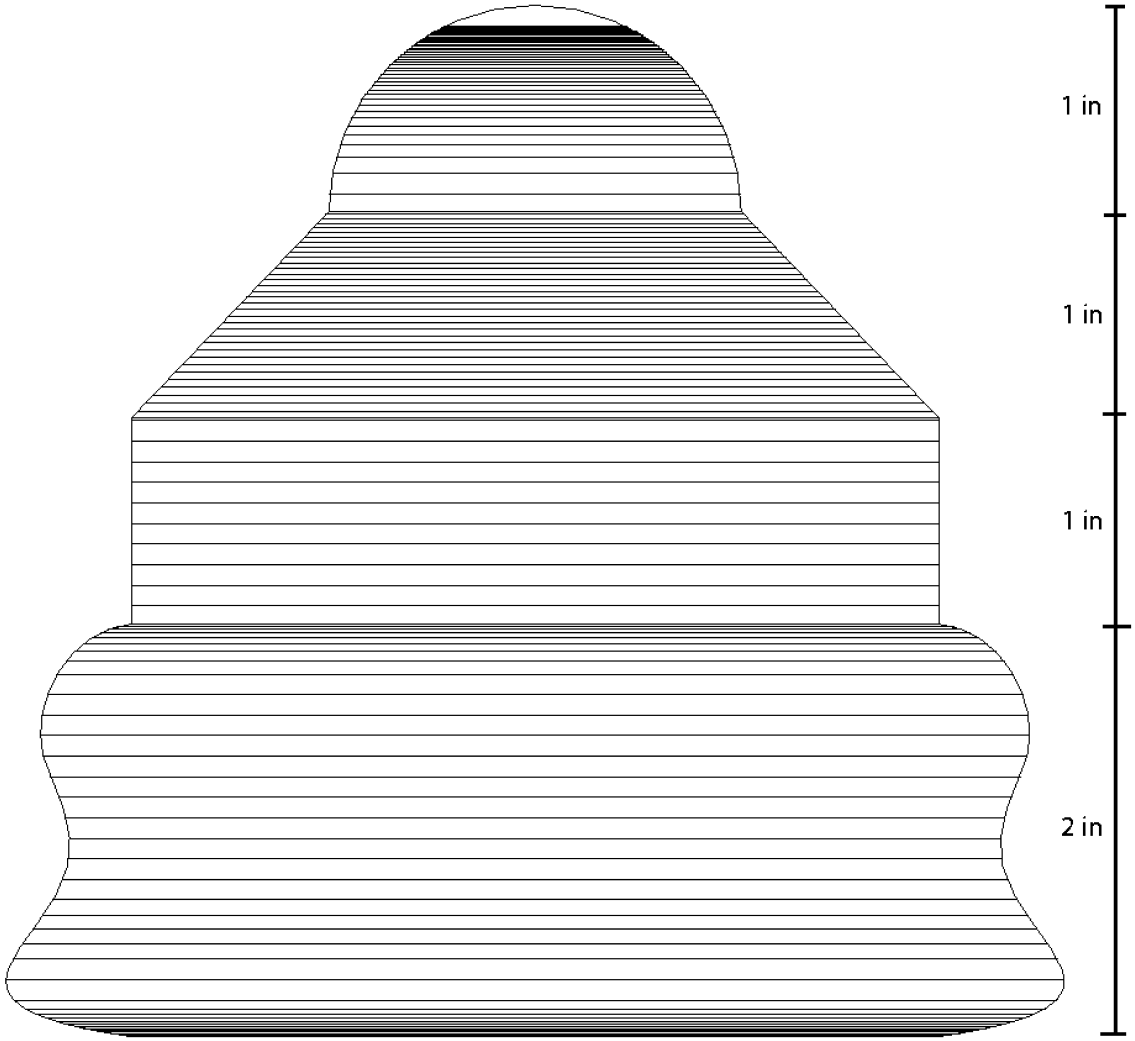


Figure 48: A sample CAD part adaptively sliced using the volume deviation approach.

the layer thickness ranges between the maximum and minimum thickness specified in the algorithm.

Figure 50 gives a plot of % volume deviation present in each layer along the height of the part and how this varies for the adaptively slicing as compared to uniform slicing at the maximum layer thickness. As can be clearly seen, the adaptively sliced part has a volumetric deviation of at most 2% as specified by the maximum bound in the algorithm whereas, for the uniformly sliced part, the volumetric deviation fluctuates

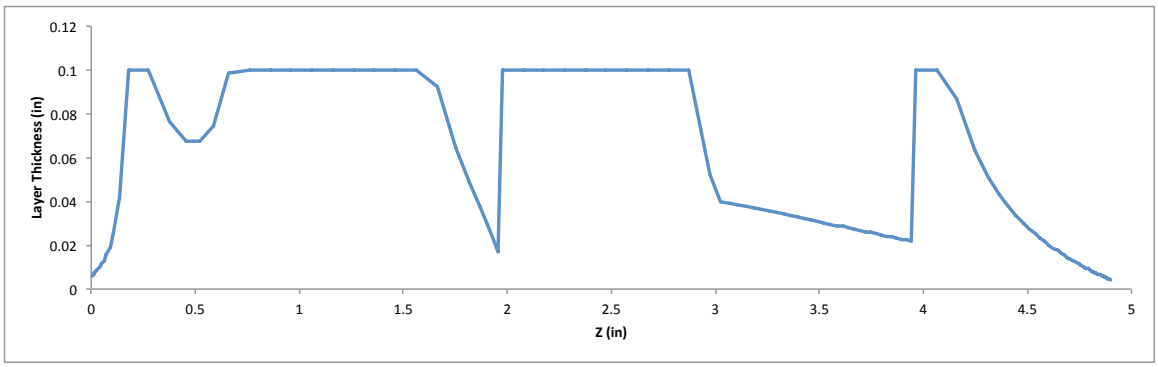


Figure 49: Variation of layer thickness vs height z for sample part.

through a wide range from 0% to nearly 35% as a function of location in the build direction. Figure 51 gives a plot showing the variation of the total absolute volume in (in^3) lost or gained in the part for both the adaptive slicing and uniform slicing.

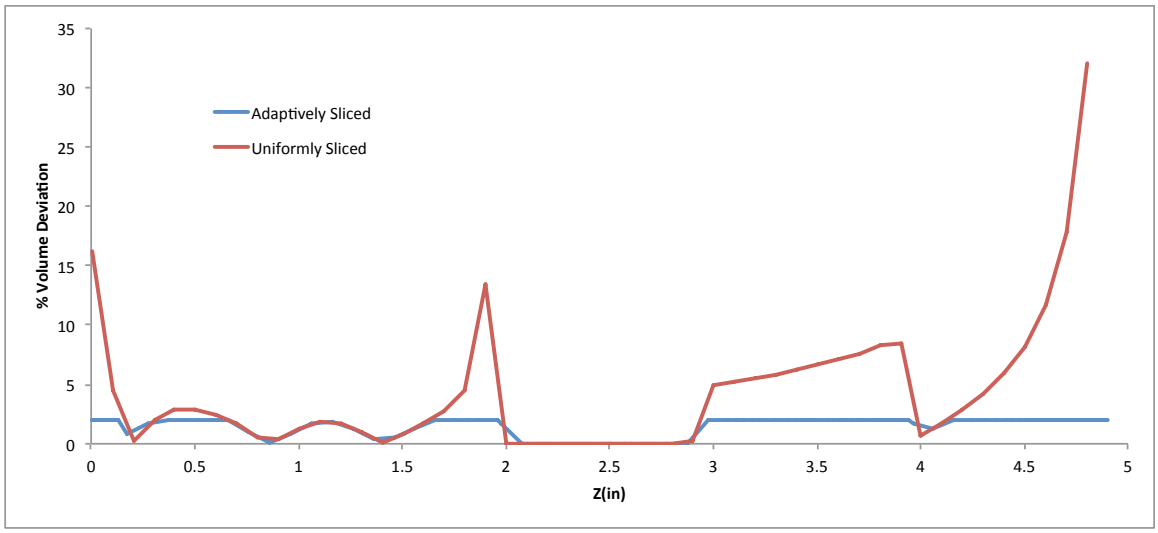


Figure 50: % Volume Deviation vs height for sample part.

From the results shown, it is evident that this approach of using % volumetric deviation as a criterion for adaptive slicing works very well. Since it computes the full three dimensional volume of the cusp, this approach is free of the limitations (discussed in the previous section) of the area deviation approach presented by Jamieson and Hacker [10] and Zhao and Laperriere [88] for slicing BRep models.

It is to be noted that the % volumetric deviation metric is a relative measure as

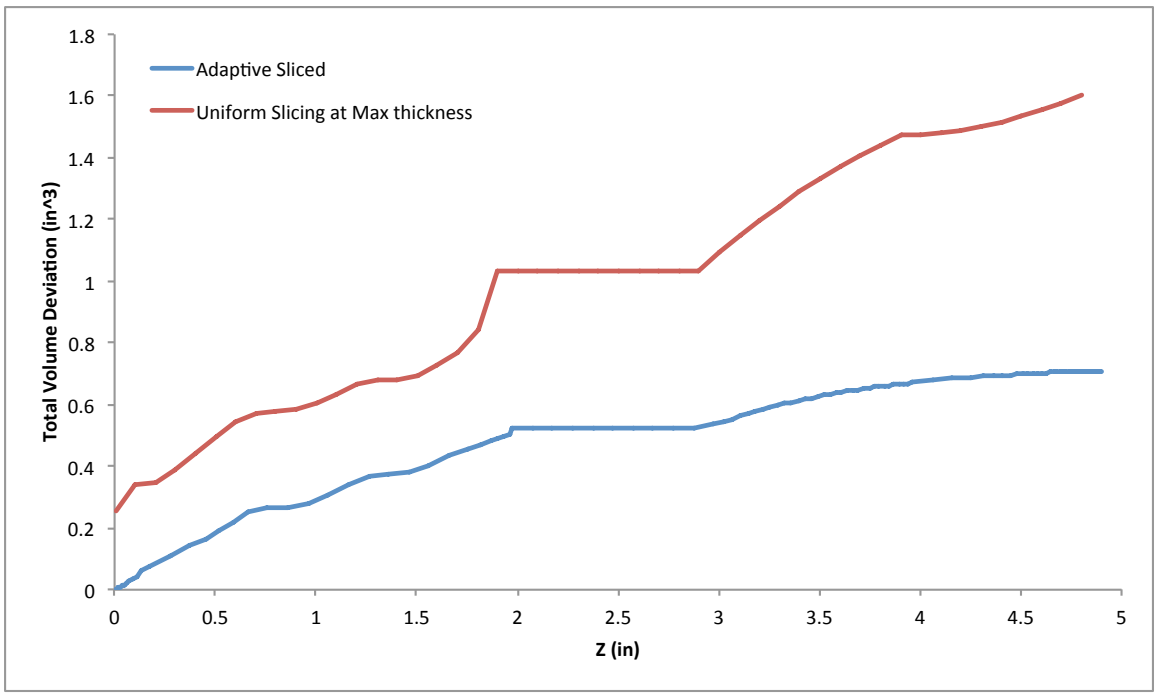


Figure 51: % Total Volumetric error vs height for sample part.

compared to the cusp height metric (which is an absolute one) implemented by Kulkarni and Dutta [6]. However, as shown by the results, this approach yields satisfactory outcomes and its evident simplicity (quicker computation time as a consequence) and scalability to handle generic BRep models with more complex geometry (as compared to only parametric surface splines handled by Kulkarni and Dutta) give it the advantage. The relativeness of the volume deviation metric can be alleviated by having more designer knowledge of the parts being built (like the minimum feature sizes, maximum curvature regions etc.) while setting the parameters (minimum and maximum layer thickness ranges and the maximum volume deviation bound) in the slicing algorithm.

As future scope of this work, for the specific HP turbine blade designs intended for fabrication via LAMP, empirical studies can be performed to relate the % volumetric deviation to the absolute surface roughness parameter R_a of the fabricated parts in order to infer a more accurate parameter range for the slicing algorithm. However,

for successfully fabricating adaptively sliced parts, some hardware changes need to be made in the LAMP machine as well. In its current configuration, it is not possible to change the exposure time dynamically in a build in an automated fashion (It can still be specified manually before the exposing each layer but becomes very tedious for large builds). This capability needs to be achieved in order to cure layers of arbitrary thickness. Controlling the wet layer thickness while re-coating each fresh layer is also crucial as it is dependent on phenomena like surface tension, suspension rheology etc. Once these changes are implemented, it is expected that a simple and scalable adaptive slicing algorithm like the one presented here would greatly improve the part quality.

3.3 Gray Scaling and Dithering Studies for Cure Depth Modulation

The other major approach pursued in this thesis to address the issue of stair stepping is through gray scaling and dithering. Details of this approach are given in this section.

3.3.1 Concept of Dithering

Dithering, also referred to as half-toning or screening (there are slight technical differences between half-toning/ screening and dithering, but for in this case they will used interchangeably as both try to accomplish the same end result), is a technique used in computer graphics and image processing to create the illusion of color depth with a limited color palette [112]. In a gray scale image, the illusion of various shades of gray is obtained from just a binary color palette (only consisting of black and white). This effect is shown in Figure 52. An original black and white image is shown in Figure 52(a). This image has to be dithered/ screened if it were to have a gray scale intensity shown by the gray scale image in Figure 52(b) using just a black and white color palette. The screened image corresponding to the gray scale image in

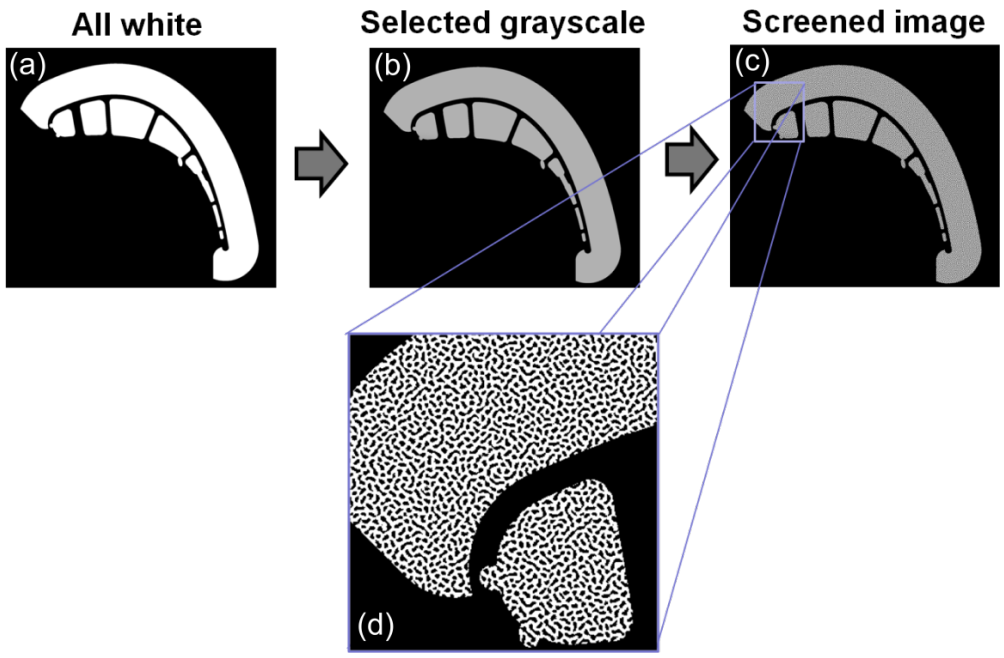


Figure 52: Illustration of dithering process: (a) the original black and white (b) the image with a selected gray scale value (c) screened image with finely distributed black and white pixels to give the optical illusion of the selected gray scale value. (d) a zoomed view of the screened image showing the finely distributed black and white pixels.

Figure 52(b) is shown in Figure 52(c). It has finely distributed black and white regions which effectively give the optical illusion of the intended gray scale value show in Figure 52(b). A close up view of the screened image is shown in Figure 52(d) to give a clearer view of these finely distributed black and white regions.

3.3.2 Basic Approach

The basic idea behind using gray-scaling and dithering in LAMP is to effectively modulate the cure depth in a single exposure by using screened gray scale regions in the build images in place of using the original all white regions for the cured regions in the slice images. The stair stepping effect observed in additively manufactured parts, as discussed previously, is a result of the fact that the cured layers have 2.5D

geometry with a constant depth across the entire region of exposure. For surfaces that are facing downward (i.e surfaces whose normal vectors make an angle greater than 90^o and less than 270^o with respect to build direction), this means that the cured layer overshoots the part geometry at the edges. This effect is illustrated in Figure 53.

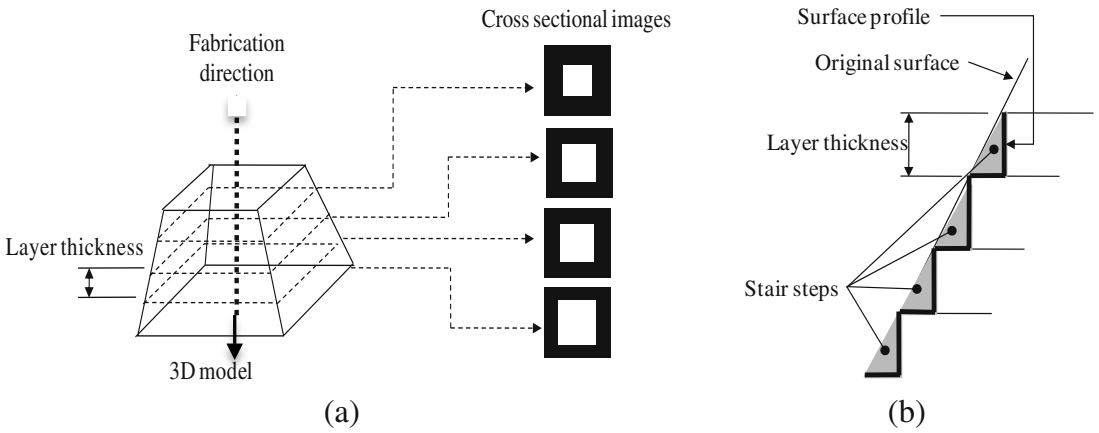


Figure 53: Illustration of stair stepping caused on downward facing surfaces while using all white build images: (a) 3D model that is to be built and some sample all white slice images (b) the layer profile as compared to the surface profile obtained using all white slice images.

For a 3D CAD model shown in Figure 53(a), if all-white slice images are used to represent the exposure dose for curing each layer of a part, the resulting cured layers overshoot the actual surface profile as shown in Figure 53(b).

In order to rectify this overshooting effect, the exposure dose that the material system receives needs to be modulated locally at the edges of each exposure image (where the full cure depth leads to overshoot) to get a cured profile that represents the surface profile more accurately. Energy dose (E) is a product of the light intensity (denoted by I and has units of W/m^2) and exposure time (t) as shown below:

$$E = I * t \tag{27}$$

Thus, one method of locally modulating the exposure energy dose E involves

manipulating exposure time t. However, in the current LAMP equipment, there is no facility to locally manipulate the exposure times within each exposure. An alternate method for manipulating the exposure dose involves by manipulating the light intensity. Since LAMP and most other projection systems use a single light source with a fixed power output, locally manipulating the light intensity would is also very challenging. As an alternative to local manipulating the actual light intensity, gray-scaling followed by dithering is used to manipulate the effective light intensity incident upon the material.

3.3.3 Relevant Literature

Recently, some work has been reported in the literature on approaches to modulate the effective light intensity using the gray scaling approach. A brief summary of this work is given in this section. Atencia et al. [113] used gray scale masks to fabricate fine point micro needles and micro channels of variable height within a single exposure. Lu and Chen [114] fabricated micro lens arrays with smooth curved surfaces from a single exposure by incorporating gray-scaling. Park et al. [115] reported the use of gray scaling and dithering in a projection micro-stereolithography set up. Pan et al. [116] also reported the use of gray scaling in achieving smooth surface finish. Although, the work of Park et al. [115] and Pan et al. [116] accomplishes the objective in a projection system similar to LAMP, some key differences exit. The material parameters E_c and D_p (will be explained in the next section) were essentially assumed to be constant in their work. The effect of gray scaling on the light intensity, E_c and D_p was not studied or presented in their work. Moreover, the material system used in LAMP fundamentally differs from the material systems used in their work due to the very high (up to 55 Volume %) ceramic loading present. Extensive studies on the effects of gray scaling and dithering on fundamental curing parameters of the ceramic loaded LAMP suspension were performed in this work. Studies of this nature have not been

reported in the literature to date. Based on these studies, a more comprehensive approach than the ones presented in [115] and [116] for utilizing gray scale exposure is presented.

3.3.4 Cure Depth Model

The cure depth model used in LAMP is same as the Jacob's model popularly used in Stereolithography [117]. It is derived from the Beer-Lambert's law as follows. The Beer-Lambert's law postulates that the energy dose attenuates exponentially with respect to the distance traveled in a medium. It is given by Equation 28.

$$E(z) = E(z = 0) * e^{-\frac{z}{D_p}} \tag{28}$$

Where E is the energy dose as a function of the distance z traveled through the medium, and D_p is known as sensitivity and is treated as the property of the specific medium. Mathematically, D_p denotes the distance z at which the energy dose at the surface attenuates by a factor of e^{-1}. In order to develop an expression for cure depth C_d, a new parameter E_c, known as the critical energy dose, is introduced and is assumed to be yet another property of the medium. It denotes the minimum energy dose required to initiate the photopolymerization chain reaction in photocurable suspensions used in Stereolithography and in LAMP. Hence, at an energy dose $E = E_c$ the material system just begins to gel and forms a soft solid and at energy does $E > E_c$ the material is assumed to be increasingly cured into a solid. Hence, the distance z below the surface of the suspension that receives an energy dose $E = E_c$ gives the cure depth C_d and can be solved from Equation 28 as follows:

$$E(z = C_d) = E_c = E(z = 0) * e^{-\frac{C_d}{D_p}} \tag{29}$$

by rearranging we get,

$$C_d = D_p * \ln\left(\frac{E}{E_c}\right) \tag{30}$$

Here, the energy dose at the surface $E(z = 0)$ is simply denoted by E. This is also referred to as the working curve equation of a given material system and is instrumental in determining the energy dose required to cure a layer of given thickness C_d. Expanding Equation 30, we get:

$$C_d = D_p * \ln\left(\frac{I * t}{E_c}\right) \tag{31}$$

Thus the cure depth C_d is a function of resin sensitivity D_p, critical energy dose E_c, lamp intensity I, and exposure time t. As discussed in Section 3.3.2, during gray scale exposure, the energy dose is only modulated by manipulating light intensity I while holding the exposure time t constant. D_p and E_c are considered material constants but it will be shown in the subsequent sections that gray scale in fact modifies these parameters and they can no longer be treated constant. The procedure for experimentally determining these parameters is discussed next.

3.3.5 Determination of D_p & E_c

The material constants D_p and E_c of the specific system are first determined experimentally by plotting the cure depth C_d versus the natural logarithm of exposure dose. This plot is a linear regression of the experimental cure depth measurements, which is known as the "working curve". Figure 54 shows the working curve for the material system used in LAMP. From the working curve plot, the resin sensitivity D_p can be determined as the slope of the linear fit and the critical energy E_c can be obtained from the x-intercept. For the material formulation studied in this thesis, the resin sensitivity was found to be $211 \pm 1.3\mu m$ and the critical energy was $131.5 \pm 5.3\ mJ/cm^2$ with a 95% confidence interval. Figure 55 shows a schematic of how cure depth samples were prepared. A UV transparent glass slide was placed on the surface of the photocurable ceramic-loaded material system and the exposure head introduced a uniform intensity of UV light onto the resin. The intensity from the UV source was determined to be $1.6W/cm^2$ and the exposure dose was adjusted

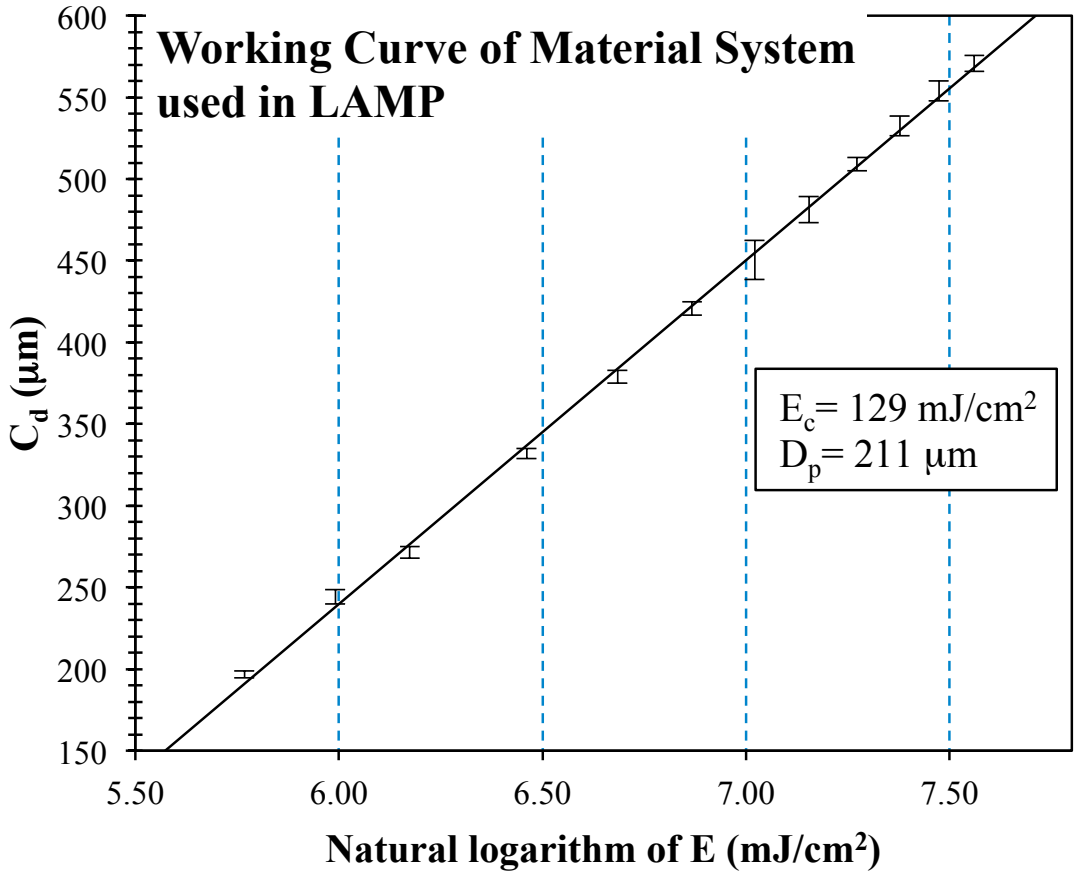


Figure 54: Working curve obtained from cure depth measurements of the material system used in LAMP.

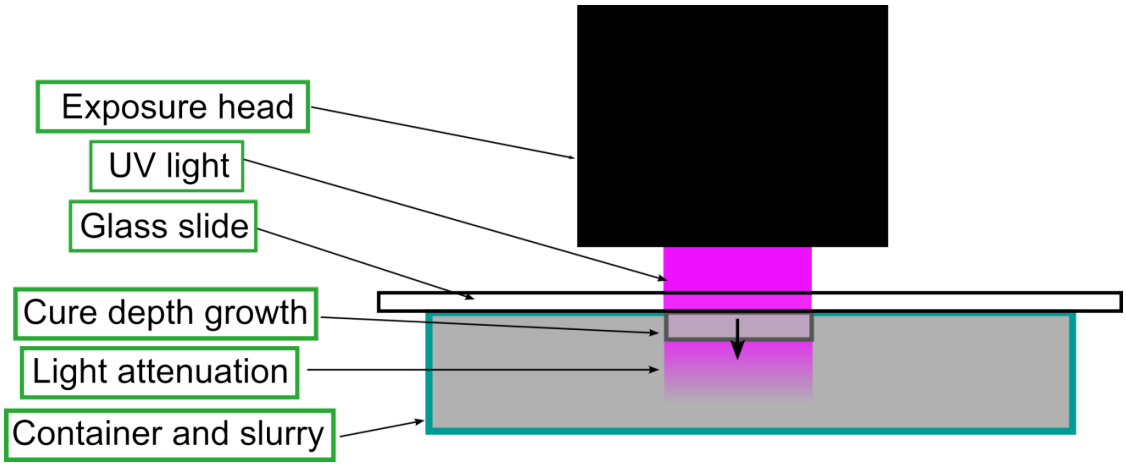


Figure 55: Schematic of the experimental setup for the determination of cure depth.

by selecting exposure times between $200ms$ and $1200ms$ to produce cured samples of different thicknesses. The solidified polymer network adhered to the glass, which enabled the removal of the cured sample from the uncured monomer. Residual monomer was rinsed from the glass slide and cured samples with isopropanol. After this "development" procedure, the isopropanol is allowed to dry and the thicknesses of the samples were measured using a micrometer. Results from this procedure reliably produced cure depth measurements within a range of $\pm 5\mu m$ from the average cure depth.

3.3.6 Homogeneous Transition

Before discussing the details of modulating cure depth using gray scaling, the concept of Homogeneous Transition is introduced. As previously explained, a gray scale value of given intensity can be effectively achieved from a binary palette by imposing finely dispersed areas black and white regions (black corresponds to zero exposure, and white corresponds full exposure) in the image. However, to achieve the same average gray scale value, multiple different patterns can be used and the relative sizes (referred to as screening resolution) of these black and white regions in the screened image can be changed. The gray scaling of images acts as an effective reduction of light intensity. It has been observed that cured layers of uniform thickness (less than the thickness achieved at an all white exposure) are achieved only if the size of these regions is below a critical threshold [12]. For any size beyond the critical threshold value, the light intensity is not averaged and the cured films have distinct regions that are either completely (corresponding to white regions of the image) or barely (corresponding to the black regions of the image) cured. Figure 56 shows the varying gray scale resolution for screening the same gray scale value using a checkerboard pattern. Figure 57 shows the films that are cured using each of these differing screening resolution images. As can be clearly seen from this image, as size of the square

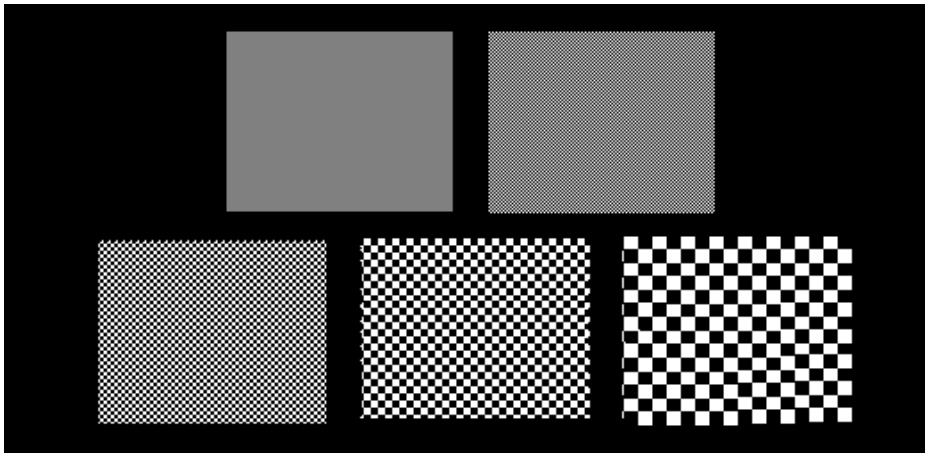


Figure 56: Illustration of Gray Scaling Resolution. The top left tile is an actual gray scale image followed by images that are screened at different resolutions (i.e, square sizes) of $17\mu m, 85\mu m, 170\mu m, 255\mu m$ [12].

increases, distinct cured and uncured regions start to materialize. At a $17\mu m$ square length, a homogeneous (uniform thickness) film is obtained as shown in Figure 57(a). All the subsequent exposures with higher square lengths show inhomogeneity.

This averaging effect of gray scale on light intensity to achieve homogeneous layers of uniform thickness is referred to as Homogeneous Transition. An extensive experimental investigation on characterizing the homogeneous transition for the LAMP suspension, its dependence on exposure times used, and some first principles based modeling using the concept of scattering length has been reported by Conrad [12]. Operating beyond the homogeneous transition threshold is beneficial in some applications like stress alleviation in the cured films as it leaves puddles of uncured of partially cured resin that allows for stress relaxation to occur. However, in the current application of modulating light intensity and cure depth, we operate well below the threshold to get an averaging effect on light intensity, thereby resulting in cured layers of uniform thickness.

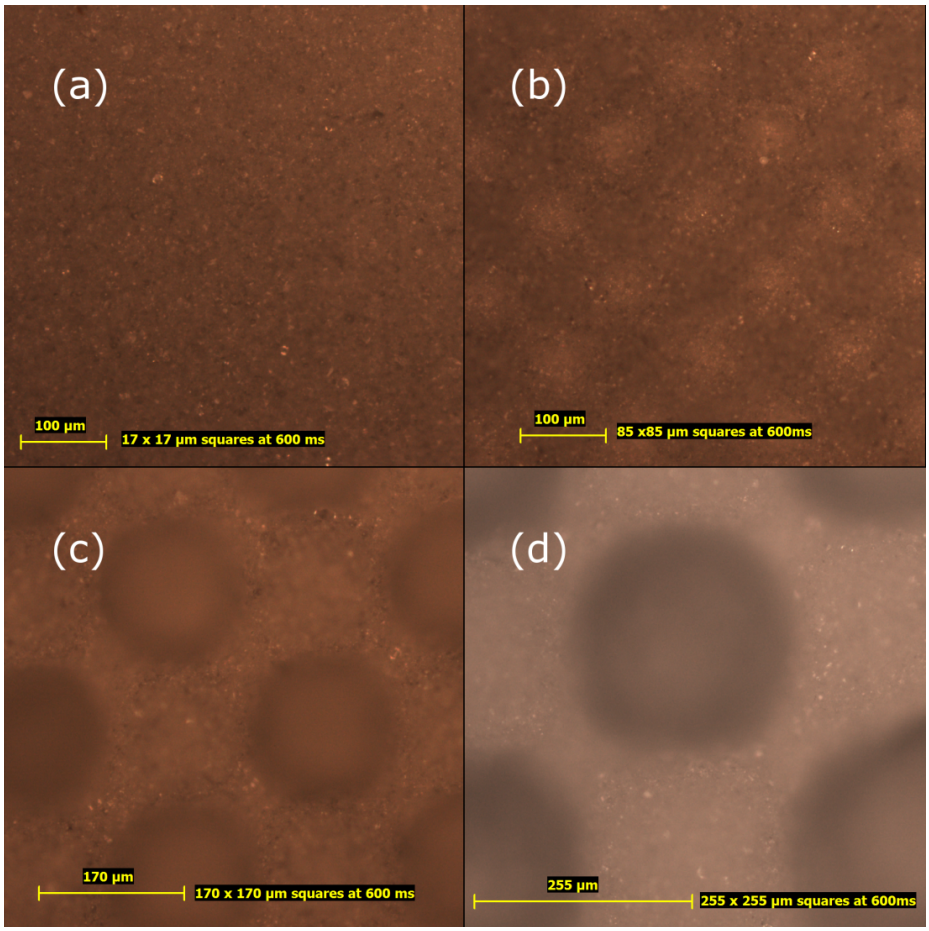


Figure 57: Images of cured films at 600ms exposure with different square sizes: (a) $17\mu m$ (b) $85\mu m$ (c) $170\mu m$ and (d) $255\mu m$ [12].

3.3.7 Effect of Gray Scaling on Light Intensity I

The effect of gray scaling and screening (well below the homogeneous threshold as explained in the previous section) on the light intensity received by the material during an exposure is discussed in this section. The factor by which the light intensity reduces is referred to as the gray scale factor (g) [12] and is given by:

$$g = \frac{I_{gr}}{I_0} \tag{32}$$

where I_{gr} is the averaged light intensity incident on material resulting from a screened gray scale image and I_o is the light intensity resulting from an full exposure without the use of gray scaling. The relation between the gray scale factor and the gray

scale value of image was determined experimentally by Conrad [12] and the results are shown in Figure 58. For each screened image with a given gray scale value, the

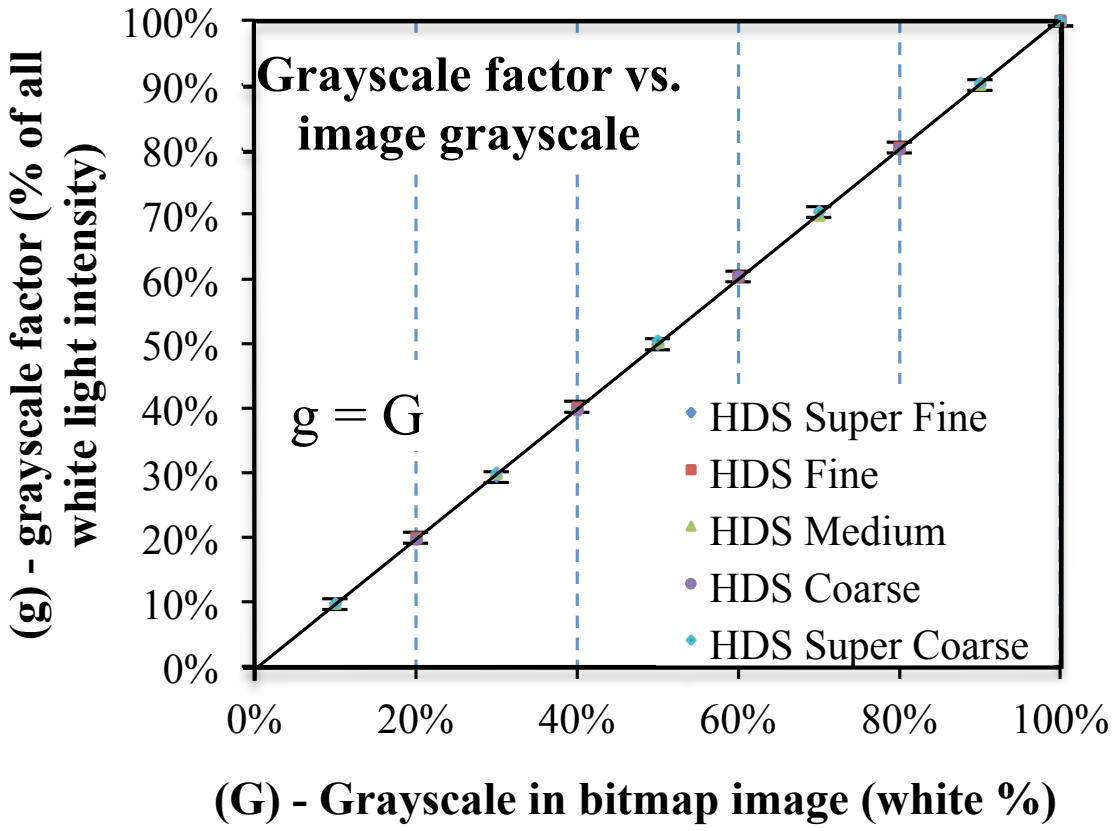


Figure 58: Plot showing the relation between gray scale factor g and gray scale value in the image G [12].

corresponding light intensity from the exposure is measured using a photodetector. The gray scale factor was computed by taking a ratio of the measured intensity to the actual light intensity measured without the use of gray scale. Six different screening patterns have been used with varying screening resolutions, all of which are below the homogeneous transition threshold. These screening techniques are proprietary and provided by Harlequin RIP Global Graphics.

The slope and intercept of the linear fit were found to be 1.004 and -0.003 respectively. Considering the error in the experimental procedure, it is reasonable to assume

a slope of 1 and intercept of 0. Hence it can be concluded from this investigation that the average gray scale light intensity incident of the surface of the LAMP suspension, I_{gr} is equal to the original (all white) light intensity multiplied by the gray scale value G of the projected image. This result is summarized by Equations 33 and 34

$$g = G \tag{33}$$

$$I_{gr} = I_0 G \tag{34}$$

3.3.8 Effect of Actual Light Intensity Modulation on D_p & E_c

Before the effect of light intensity modulation through gray scaling on the resin sensitivity D_p and critical energy E_c is studied, the effect of true light intensity modulation on these parameters needs to be understood. For this purpose, working curve experiments were conducted as discussed in Section 3.3.4 and the data plotted for several values of the light intensity and the variations in the computed E_c and D_p were observed.

For modulating the true light intensity, either the light source should have a facility to modify the power output or different light sources with varying power densities should be used. Since both options are not viable for LAMP in its current setup, neutral density filters are used as a means to modulate the intensity. The experimental set up for performing this study is shown in Figure 59.

The set up is nearly identical to the one used for the experiments required to plot working curves as discussed in Section 3.3.4. The the only difference here is the addition of a neutral density filter in the light path. Neutral density filters uniformly absorb radiation at all wavelengths and hence result in a reduction of intensity across the full emission spectrum (shown in Figure 60) of the light source which has three peaks approximately at 365 nm, 407 nm, and 438 nm.

To understand the effect of true light intensity modulation, two different light intensities were chosen apart from the full unfiltered intensity which was used as the

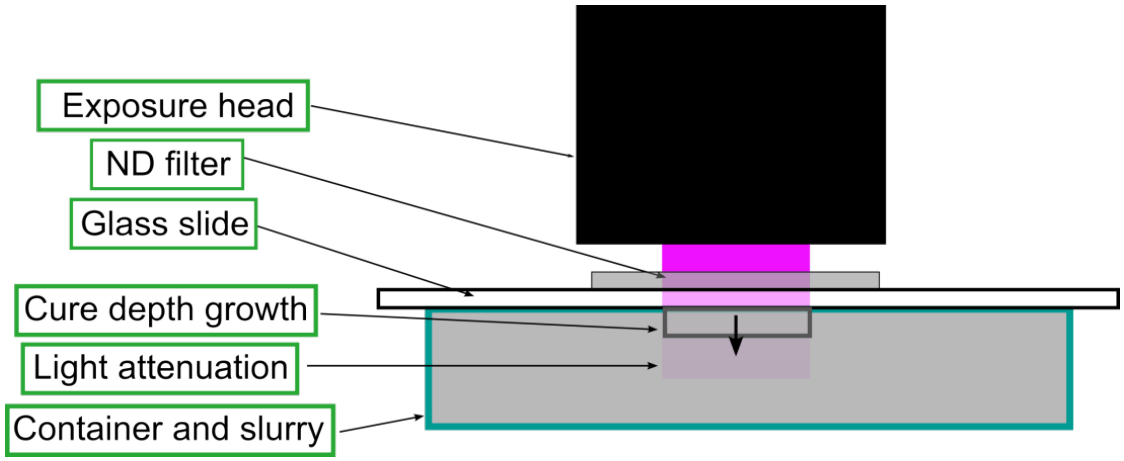


Figure 59: Schematic of the experimental setup for performing true light intensity modulation studies.

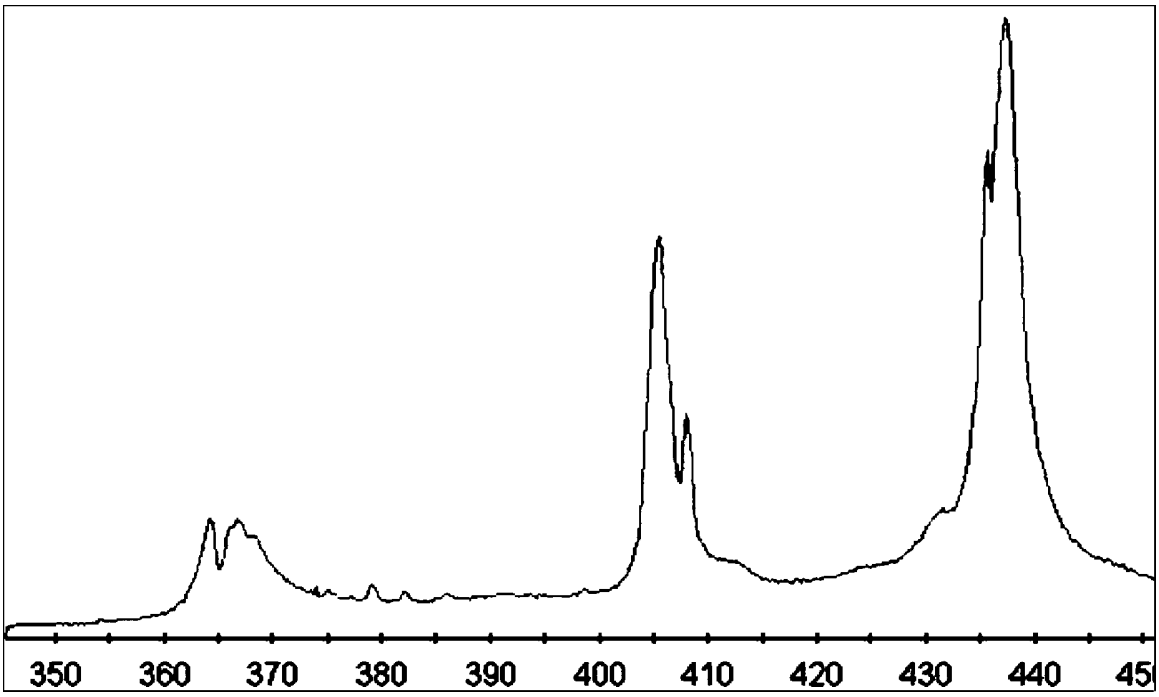


Figure 60: Emission Spectrum of light source used in the LAMP setup.

baseline and working curves were plotted. For achieving these intensities, two neutral density filters were used with optical densities of 0.3 and 0.5. Optical density (OD)

is defined as shown in Equation 35

$$OD = \log \frac{I_o}{I} \tag{35}$$

Where I_o is the intensity of light incident on the filter and I is the light intensity emanating out of the filter.

Hence, the light intensity for the two filters roughly corresponds to 50.1% and 31.6% of the unfiltered light intensity respectively. The full unfiltered light intensity is measured to be 1.6 mW/cm^2 and hence these two filtered light intensities are computed to be 896 mW/cm^2 and 512 mW/cm^2. Cure depth measurements were performed at each of these intensities and the resulting working curves were plotted as shown in Figure 61.

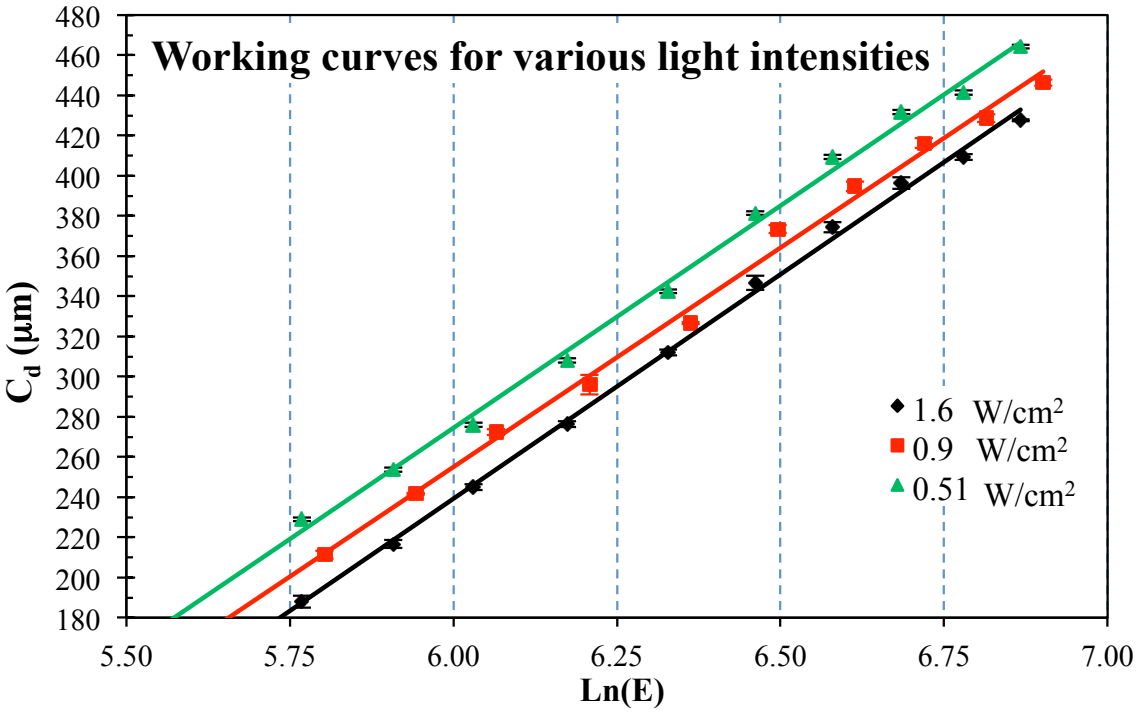


Figure 61: Working curves for varying true light intensities [12].

As can be seen from the plot, the working curves for each successive lower intensity is shifted to the left of the plot to yield a different x-intercept, i.e, different E_c while the slope (D_p) of these curves stays relatively constant. For each of these curves, E_c

and D_p values were computed and plotted as shown in Figure 62 and linear regression curves have been fitted.

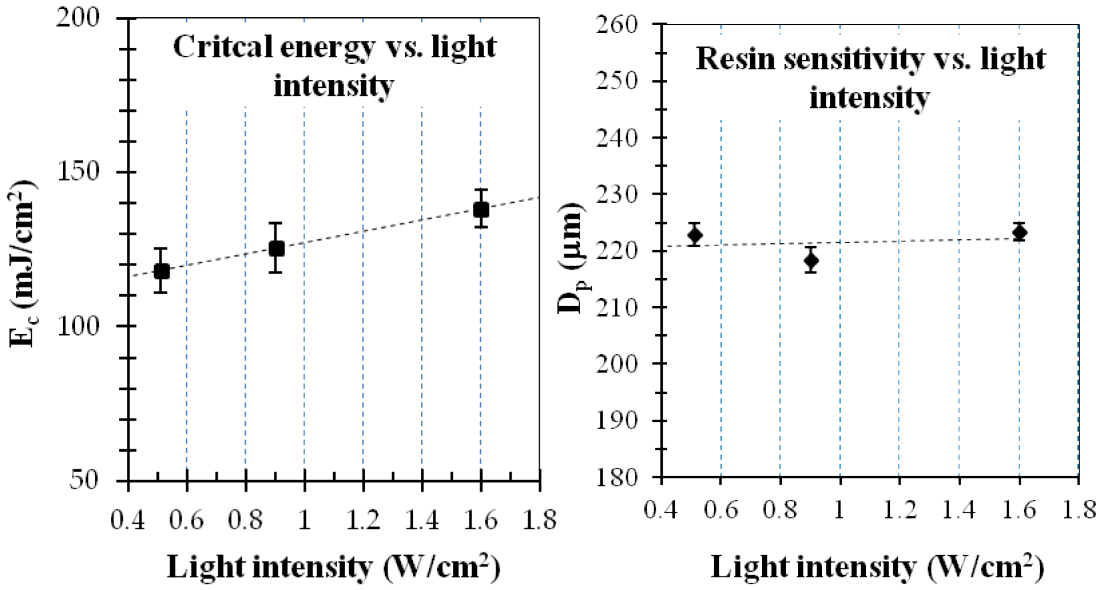


Figure 62: Variation of E_c and D_p with respect to true light intensity.

The variation of E_c w.r.t intensity is identified from the plot as follows:

$$E_c = 18.31 * I + 109 \tag{36}$$

where E_c has the units of mJ/cm^2, I has units of W/cm^2 and the slope has units of ms. The R^2 value for this fit was determined to be 0.999 indicating a linear dependence of E_c on true light intensity I. The D_p on the other hand stays relatively constant as shown in Figure 62.

It is important to note that the Jacob's equation (Equation 30) does not predict this variation of E_c and infact assumes E_c and D_p to be material constants irrespective of the incident light intensity.

3.3.9 Effect of Light Intensity Modulation through Gray Scaling on D_p & E_c

Having considered the effect of true light intensity modulation on D_p & E_c, the effect of light intensity modulation through gray scaling is presented in this section. Working curves were determined experimentally for various values of gray scale levels in the exposure images and the variation of E_c and D_p was studied by Conrad [12]. Figure 63 shows the working curves obtained by him for four different gray scale levels of 20%, 40%, 50% and 100% using HDS (Harlequin Dispersive Screening) superfine screening resolution.

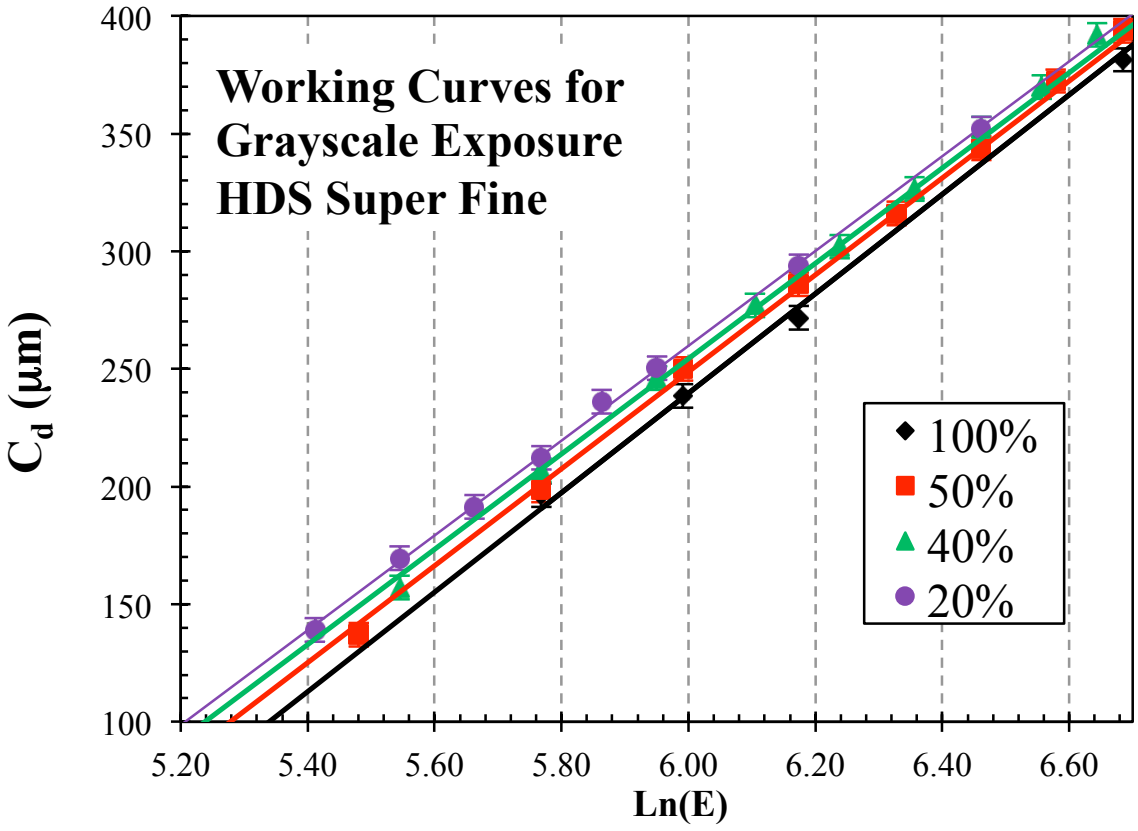


Figure 63: Working curves for different levels of gray scale using HDS superfine screening resolution [12].

As can be seen from the figure, just like in the case of true light intensity modulation, for successive reductions in the gray scale level (reduction in effective light

intensity), the working curve shifts left and hence causes a reduction in E_c. However, unlike the case of true light intensity modulation where the slope of the curves remained constant, different slopes are observed for varying levels of gray scale exposures. These results were obtained by the use of only one screening resolution (HDS Super Fine).

In order to understand the effect of different screening resolutions on the curing parameters E_c and D_p, Conrad [12] obtained similar working curves experimentally with various screening resolutions and the corresponding parameters were determined and plotted. The results from this investigation are shown in Figure 64

It can be seen from Figure 64, that the screening resolution used does not have as much of an impact on E_c and D_p. Similar cure depth values were obtained for each gray scale level while using the various resolutions thus leading to similar variation trends for E_c and D_p. It can also be concluded that D_p varies linearly with respect to the gray scale value as verified repeatedly in the many cases shown in Figure 64.

In order to compare the trends in E_c and D_p obtained using gray scaling to the trends obtained by modulating the true light intensity, the gray scale values are converted to equivalent light intensity using Equation 34 as discussed in Section 3.3.7 and the trends are replotted as shown in Figure 65. The regression obtained for E_c and D_p with respect to the equivalent gray scale light intensity is shown in Equations 37 and 38

$$E_c = 19.6 * I + 103.7 \tag{37}$$

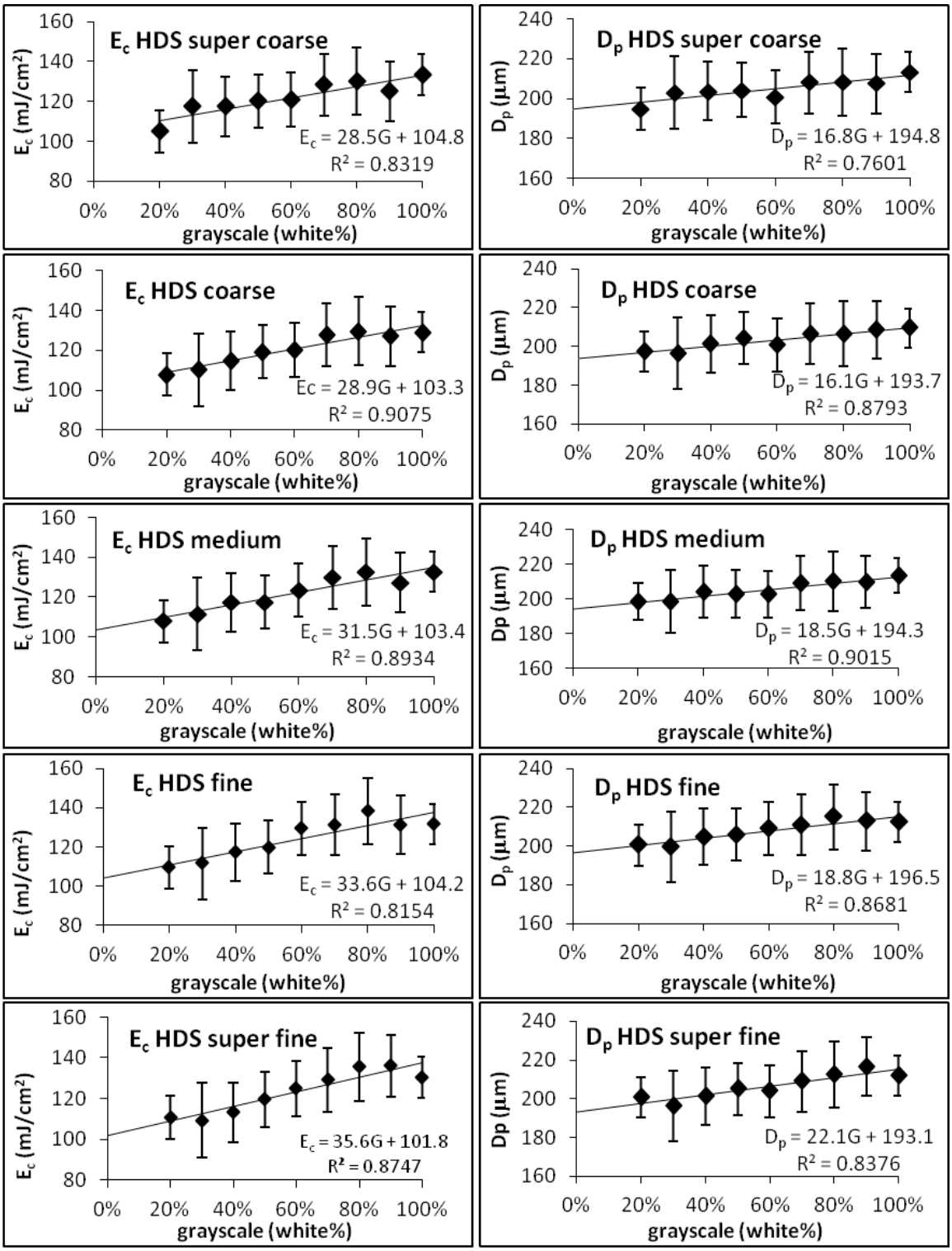


Figure 64: Variation of E_c and D_p with respect to gray scale level for different screening resolutions [12].

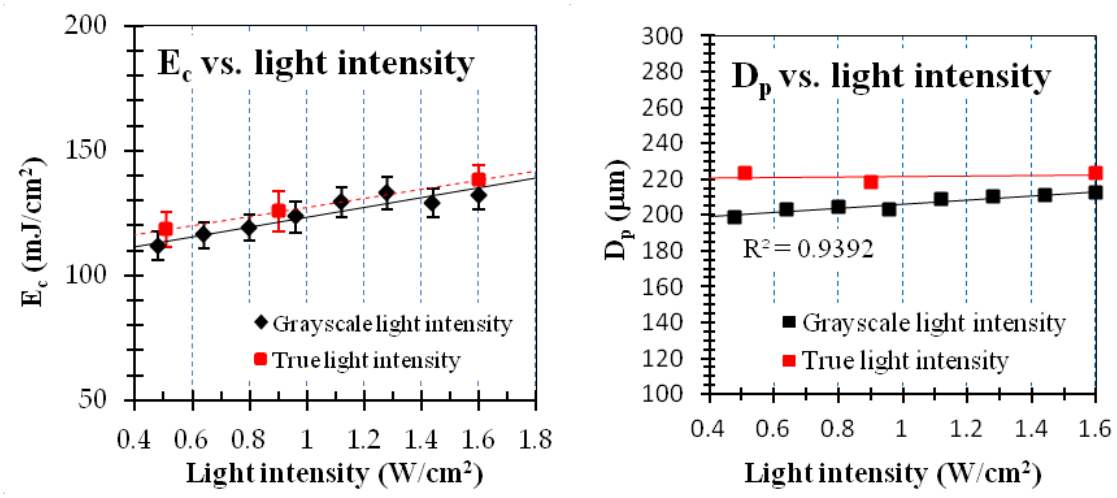


Figure 65: Comparison of trends in E_c and D_p with respect to true light intensity and gray scale intensity.

$$D_p = 11.3 * I + 194.8 \tag{38}$$

As can be seen from Figure 65, the trends followed by E_c with respect to the variation in gray scale light intensity is very similar to the trend followed with respect to true light intensity modulation. However, for the case of D_p, the trends differ. While D_p remains constant for various true light intensities, it follows a linear trend with respect to gray scale light intensity.

3.3.10 Methodology for generating gray scale images

Details of the algorithm and the methodology followed for generating gray scale images in order to modulate the cure depth within each exposure to reduce the stair-stepping effect on downward facing surfaces, are presented in this section. The cure depth C_d is a function of light intensity I, resin parameters sensitivity D_p and critical energy E_c and exposure time t as discussed in Section 3.3.4 and Equation 31. As previously discussed, the exposure time t is held constant in this approach. Through the experimental investigations presented in the previous sections, it was determined

that rest of the parameters are in turn functions of the gray scale value G. This functional dependence on gray scale value G is shown in Equation 39.

$$C_d = D_p(G)\ \ln \frac{I(G)\ t}{E_c(G)} \tag{39}$$

These functional dependencies have been explicitly identified in the previous sections. Therefore, the final expanded form of the expression relating C_d and gray scale value G for HDS super fine screening resolution can be written as follows:

$$C_d = (22.1\ G + 193.1)\ \ln \frac{G\ I_o\ t}{35.6\ G + 101.8} \tag{40}$$

where I_o is the full light source intensity from an all white exposure which was measured to be 1.6 mW/cm^2 for the light source used in LAMP. Therefore, using the expression in Equation 40, for a given exposure time t, the gray scale value G that results in the required cure depth C_d can be computed. The details of the algorithm to identify the required cure depth C_d and thereby the required gray scale value G at each pixel in the slice image are given next.

3.3.10.1 Algorithm for Generating Gray Scale Slice images

The direct slicing algorithm presented in Section 2.2 is extended to output gray scale slice images instead of the usual black and white images. The pseudo code for accomplishing this is presented in this Algorithm 15.

The given CAD model is first loaded into the program and assigned to the variable *wig*. For each slice height z along the height of the part, first a three dimensional slice, denoted by $3DSlice$, is computed in order to identify the accurate geometry that needs to be cured. This is accomplished by creating a cuboid, denoted by *block* of thickness equal to the build layer thickness denoted by $layerThickness$ and computing the intersection of it with the given part denoted by *wig*. Next, the 2D slice contour denoted by $2DSlice$ is computed at a height $z + layerThickness$ by creating a slice plane at this height and computing its intersection with *wig*. Now,

for each pixel in the slice image corresponding to the slice height $z + layerThickness$, first the corresponding coordinate values denoted by (x, y) of the pixel are identified. The required cure depth C_d that needs to be achieved is determined by computing the thickness of the 3D slice geometry at these coordinates. The exposure time t is fixed and corresponds to a cure depth equal to the full layer thickness used in the build. Using these values for C_d and t in Equation 40, the gray scale factor G can be solved for by using any of the standard root finding techniques like Newton-Raphson or Bisection method. Once the required gray scale value G at the current location is determined, the corresponding pixel value of the slice image is set to $G * 255$ (For an 8-bit gray scale image, like the one being created in this case, a value of 255 corresponds to full white and a value of 0 corresponds to full black). In this manner, the gray scale values for each of the pixels in the slice image are determined and the slice image is created. Figure 66 illustrates the steps described above for a sample part. Figure 66a shows the sample part used for computing gray scale slice images. Figure 66b shows the process of identifying the required cure depth corresponding to each pixel of the slice image. From the cured depth determined by this process of ray intersections with the 3D slice, the gray scale value required at each pixel value can in turn be computed from the cure depth model established in Equation 40. The gray scale slice image obtained from this process is shown in Figure 66c. Figure 66d shows a zoomed in view of the dithered gray scale region obtained by dithering the gray scale slice image.

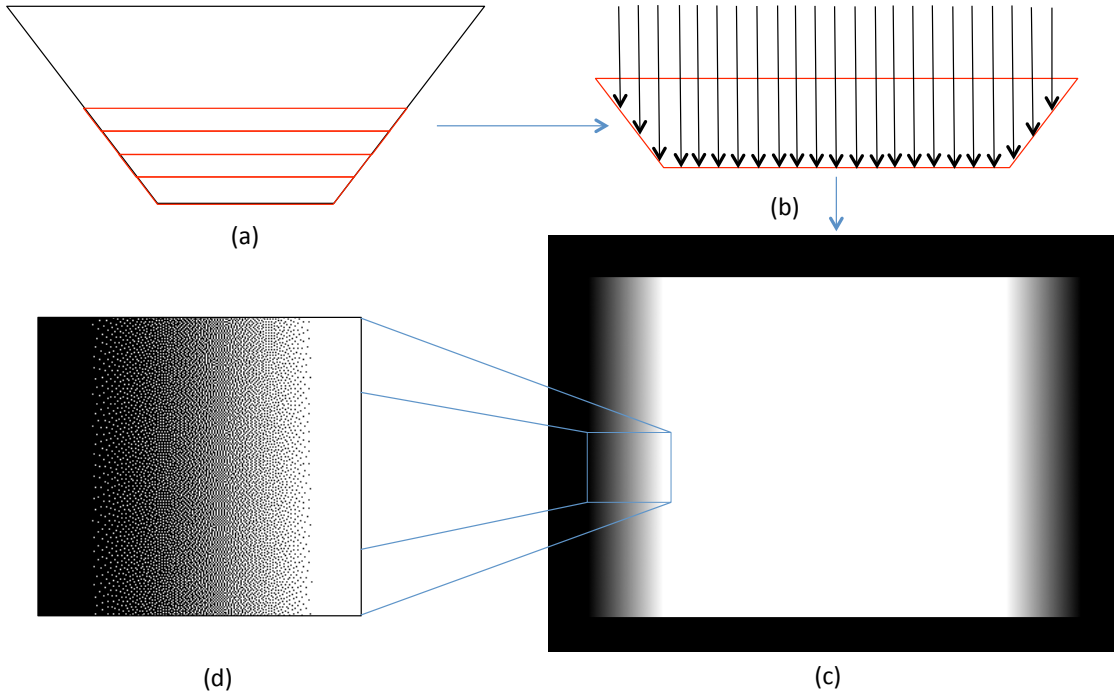


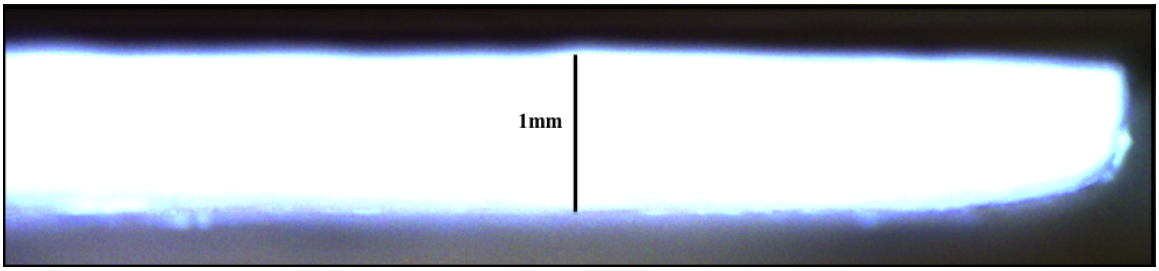
Figure 66: Illustration of the various steps involved in producing gray scale images: (a) Sample part (b) compute required cure depth at each pixel location (c) gray scale slice image (d) dithered region.

Algorithm 15 Gray Scale slice image generation.

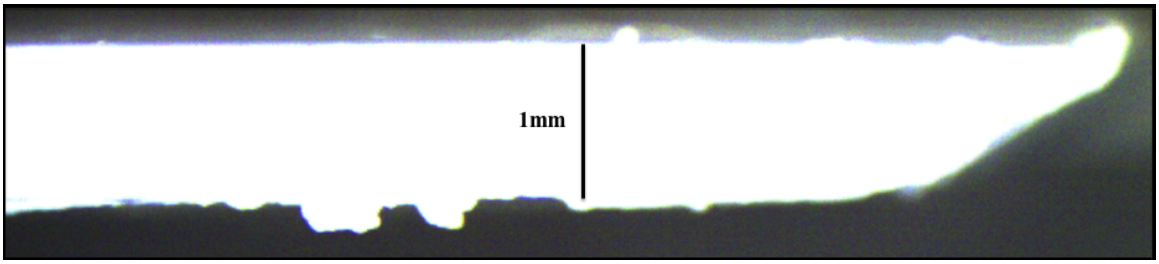
1: $wig \leftarrow$ given part
2: **for** each slice height z along the height of the part **do**
3: **//compute 3D Slice:**
4: $block \leftarrow$ a cuboid of height equal to $layerThickness$
5: $3DSlice$ solid geometry obtained from intersection of wig and $block$
6: **//compute 2D Slice:**
7: $slicePlane \leftarrow$ plane at a height of $z + layerThickness$
8: $2DSlice \leftarrow$ contour obtained from intersection of wig and $slicePlane$
9: **//compute Slice Image Pixel Values:**
10: **for** each pixel in the slice image corresponding to the 2D slice **do**
11: $(x, y) \leftarrow$ coordinates corresponding to the pixel
12: $C_d \leftarrow$ thickness of $3DSlice$ at (x, y)
13: $t \leftarrow$ exposure time corresponding to a cure depth equal to $layerThickness$
14: $G \leftarrow$ gray scale value solved from Equation 40 using C_d and t
15: $pixelValue \leftarrow G * 255$
16: **end for**
17: **end for**

3.3.11 Results

For the sample part shown in Figure 66a, one sample layer is cured for demonstration of the concept. Figure 67(a) shows the cured profile obtained with an all white exposure. As expected, the profile is more or less 2.5 D cross-sectional. The cured profile of the same layer now exposed with the gray scale slice image obtained using the process discussed in the previous section is shown in Figure 67(b). As can be seen, the cured profile obtained with the gray scale exposure is very close to the actual 3D slice geometry of each layer for the sample part shown in Figure 66a.



(a) All white exposure.



(b) Gray scale exposure.

Figure 67: Gray scale exposure results.

These single layer cured profiles serve as a proof of concept. Multilayer parts can easily be built using this exposure technique to yield smooth downward facing surfaces. Moreover, since the cure profile can now be accurately controlled, higher layer thicknesses can be used in builds without any compromise on part accuracy and surface smoothness thereby potentially reducing build times as well.

3.4 Image Compensation for Dimensional Accuracy

3.4.1 Motivation

The final component of the computational schemes discussed in this chapter addresses the issue of dimensional accuracy of the parts produced through LAMP. There are several factors in the LAMP process that have the potential to distort the part geometry produced thereby hindering their accuracy. While, identifying and understanding the effects of all of these factors on the full three-dimensional parts with complex geometries is beyond the scope of this thesis, very useful qualitative information can

be obtained by studying the dimensional accuracy of single layers cured using slice images featuring simple geometry. Such information will be very useful to provide a basis for image compensation prior to the build so that resulting parts match the CAD geometry as closely as possible. With this goal in mind, a fundamental experimental study is pursued in order to understand the effects of some important parameters on the two-dimensional accuracy (i.e, lateral expansion or shrinkage) of single layers built through LAMP .

The green parts produced by LAMP need to survive the stresses of the firing cycle to yield dense ceramic molds that can be used for investment casting. To ensure their survival, it is imperative to obtain very good layer-to-layer bonding in the green parts and hence these parts are often built with high energy doses with cure depths that are a few multiples of the layer thickness. At such high energy doses, it was observed that the resulting parts have larger cross sections when compared to their corresponding CAD geometries. Such expansion is associated to the excess lateral curing that is caused while building each layer of the part. The extent of this excess curing in the horizontal direction is termed as cure width (C_w), analogous to cure depth (C_d) which gives the extent of the curing in the vertical direction. The experimental studies presented in this section aim to characterize the first order effects of some chosen parameters (empirically identified to be most influential) on cure width C_w. Before presenting the experimental methodology followed and the results obtained, a brief discussion on the various factors that have an influence on cure width C_w is given.

3.4.2 Influencing Parameters

Several factors influence the cure width in the lateral direction. A brief discussion on each of these parameters and some relevant work reported in the literature characterizing the influence of these parameters, wherever available, is presented here.

3.4.2.1 Light Intensity Distribution

The peak intensity of the light source and the intensity distribution profile significantly impact the cure width of the process. In order to get a better appreciation of this fact, a simple analysis for the cure profile and cure width obtained for a Gaussian intensity distribution (typically mono modal laser beams, like the ones used in Stereolithography, have a Gaussian intensity distribution) is given. It will be shown later on that the intensity profile in LAMP can be approximated as a summation of Gaussian distributions.

A Gaussian intensity distribution at the resin surface can be represented by the expression shown in Equation 41

$$I(y, z = 0) = I \ \exp\left(-\frac{2y^2}{W_o^2}\right) \tag{41}$$

where W_o is the Gaussian width, I (has units of mW/cm^2)is the peak intensity of the laser beam, y is the distance from the center of the beam and z is the depth from the surface of the suspension. Therefore the energy dose at the surface can be calculated by multiplying both sides of Equation 41 by the exposure time t as shown in Equation 42.

$$E(y, z = 0) = E \ \exp\left(-\frac{2y^2}{W_o^2}\right) \tag{42}$$

Hence, the energy distribution within the suspension can be calculated by assuming the resin to be a Beer-Lambert absorber as shown in Equation 43.

$$E(y, z = 0) = E \ \exp\left(-\frac{2y^2}{W_o^2}\right) \ \exp\left(-\frac{z}{D_p}\right) \tag{43}$$

As explained in Section3.3.4, it is assumed that polymerization (curing) occurs only in the regions where the Energy dose $E(y, z)$ is greater than the critical Energy dose E_c. Hence from Equation 43, the cure profile can be calculated as the locus of all points where $E(y, z) = E_c$ as shown in Equation 44.

$$z^* = D_p \ln\left(\frac{E}{E_c}\right) - \frac{2D_p}{W_o^2} \ y^{*2} \tag{44}$$

where (y^*, z^*) gives the locus of points where $E = E_c$. Therefore, from this equation, width of the cured region at the surface of the resin ($z^* = 0$) can be computed as shown in Equation 45.

$$W_{guass} = W_o\sqrt{2}\sqrt{\ln\left(\frac{E}{E_c}\right)} \tag{45}$$

Hence, from Equation 45, it can be clearly seen that the cure width obtained is a function of the Energy dose E input into the resin and the critical energy of the resin E_c. Even for the same resin ($E_c = constant$), as the energy dose is increased (longer exposure time t), a greater cure width is obtained as shown in Figure 68.

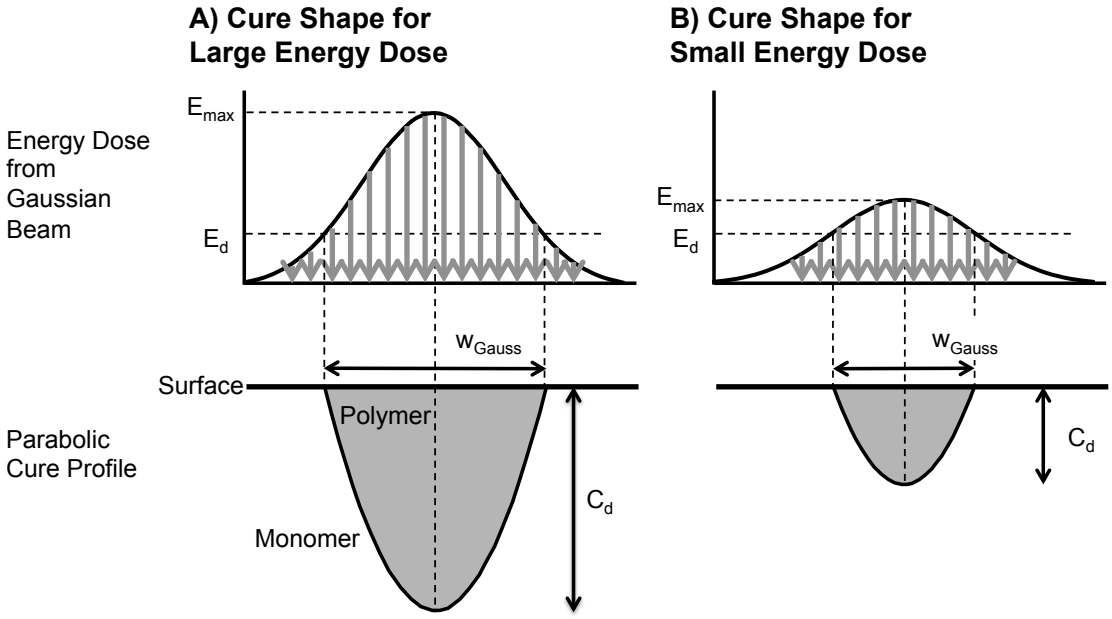


Figure 68: Comparison of cure shapes for suspensions cured with a Gaussian beam with a larger energy dose (A) and a smaller energy dose (B). The width W_{gauss} and the cure depth C_d increase with the energy dose [13].

Equation 45 gives a good predictive model for an ideal resin. However, for a ceramic-loaded suspension like the kind used in LAMP, it is no longer valid. Hinczewski et al. [76] have proposed a model for the cure width in ceramic suspensions by replacing W_o and E_c in Equation 45 with fitting parameters F_1 and E_{Hincz} as shown in

Equation 46.

$$w = F_1\sqrt{2}\sqrt{\ln\left(\frac{E}{E_{Hincz}}\right)} \tag{46}$$

F_1 and E_{Hincz} depend on the Gaussian beam width and materials parameters like refractive index and solids loading of the ceramic powder in the suspension.

The foregoing analysis is possible only for a perfect mono-modal Gaussian beam. In the exposure set up for LAMP, a DMD chip is used as described before. The light intensity distribution measured with a CCD beam measurement sensor is shown in Figure 69. The intensity profiles measured by Conrad **??** for lines of various lengths (1, 2, 3, 4, 5 and 10 pixel long lines) are shown (units presented are in counts/px-sec which is analogous to intensity). The 2D color contours in each image tile show the actual intensity measured by the sensor. A cross-sectional profile of intensities measured is also plotted below the color contours for each image. The discrete points in the plots represent the actual measured intensities while the continuous curves represent Gaussian regressions fitted to these points (For multi pixel long lines of length 'n', the fit was computed as a summation of 'n' Gaussian distributions).

As can be seen from the figure, the intensity is closely approximated by a summation of Gaussian profiles. Therefore, the analysis for the cure width obtained for a single Gaussian distribution can be extended for this case and the results will be analogous. Since the cure width obtained in the samples varies with respect to the energy dose for a single Gaussian distribution, it must vary with respect to the summation as well. Although the Gaussian regressions give a good approximation, additional effects like diffraction and interference come into play that cannot be accurately predicted, especially at longer line widths as seen in Figure 69. Moreover, the exposure system in LAMP is continuously scrolling over the build region. The intensity profiles shown in Figure 69 are only for the static exposure case. The resulting intensity distribution that a fixed point in the build region receives, as the DMD system scrolls over it, will

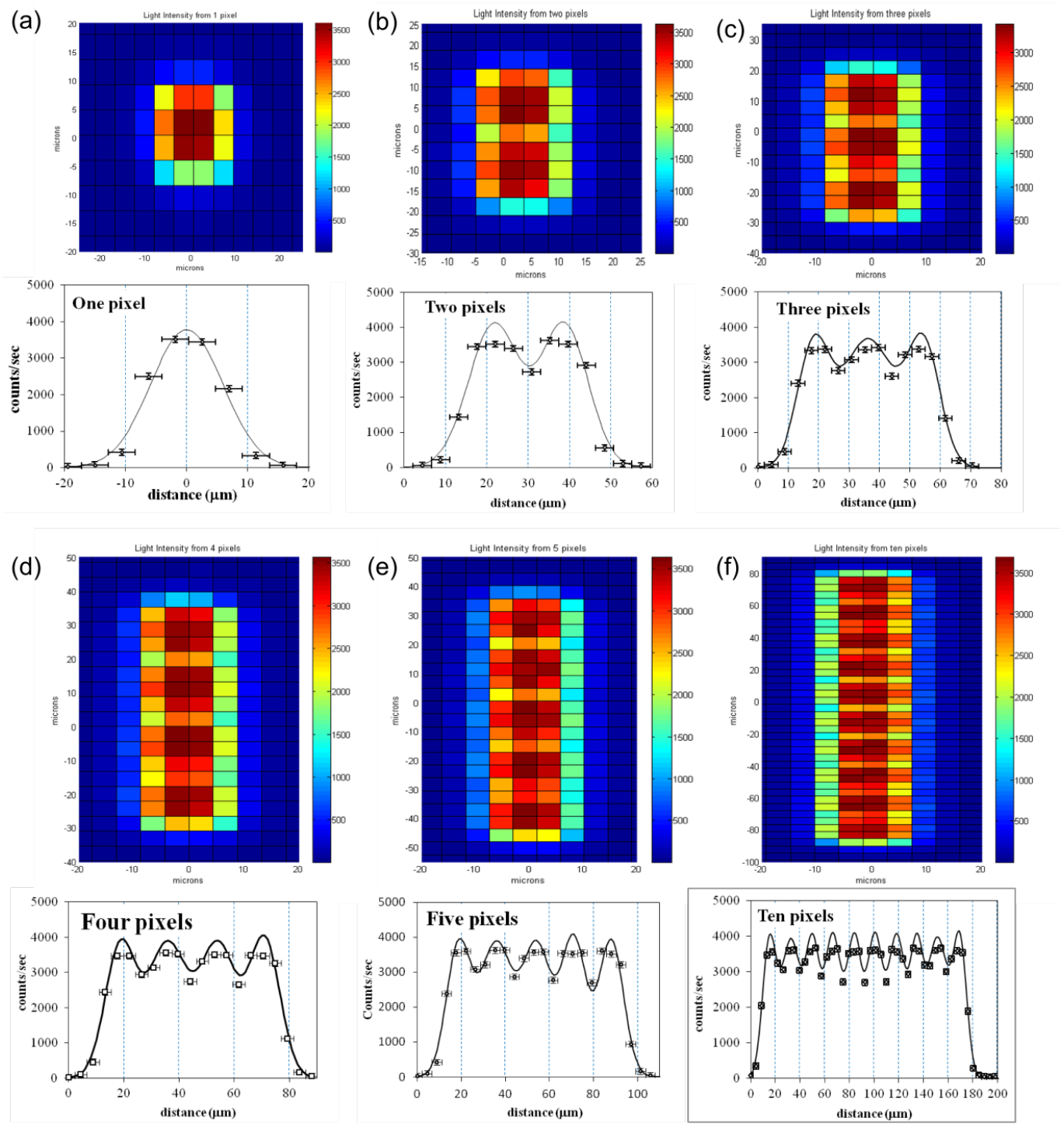


Figure 69: Intensity distributions profiles for (a) 1 pixel (b) 2 pixel (c) 3 pixel (d) 4 pixel (e) 5 pixel and (f) 10 pixel long line projections from DMD chip [12].

be arguably different. Hence, unlike the simple analysis presented for a single Gaussian beam, precisely identifying an expression for cure width in the LAMP exposure set up would be difficult. However, it is hoped that the material presented in this section gives an appreciation of the significant impact the intensity distribution has on the cure width C_w.

3.4.2.2 Shrinkage

Polymerization shrinkage strain is another factor that affects the cure width. When light is exposed, the monomer in the system is converted to radicals which then combine to form long chained networks resulting in volumetric shrinkage. This volumetric shrinkage can be characterized by measuring the degree of conversion of the cured samples. LAMP suspension uses acrylate monomers in which the polymerization reaction propagates by the conversion of carbon-carbon double bond ($-C{=}C-$). The degree of conversion (α) gives a measure of the number of double bonds converted in a sample. Experimentally, the volume change per mole of acrylate groups due to the conversion of the double bonds is reported to be $\Delta V_{C=C} = 22.5\ cm^3/mol$ [118] while the molar volume of the monomer is

$$\frac{M_m}{\rho_m} = \frac{226}{1.01} = 223.76\ cm^3/mol \tag{47}$$

where M_m is the molecular weight and ρ_m is the density of the monomer. Hence a first estimate of volumetric shrinkage strain rate of acrylate systems is:

$$\frac{\Delta V}{V}(\%) = \frac{22.5}{223.76} * 100 = 10.05\% \tag{48}$$

In the more general case of multiacrylates with filler particles, the number of functional groups present in volume (V) is

$$\left[f * \frac{V\rho_m}{M_m}\right] * \left[1 - \frac{FL}{100}\right] \tag{49}$$

where FL is the filler loading in percentage, and f is the functionality of the monomer. The number of functional groups reacted in volume (V) is given by

$$\alpha(t) * \left[f * \frac{V\rho_m}{M_m} \right] * \left[1 - \frac{FL}{100} \right] \tag{50}$$

Therefore, the percentage relative change in volume (volumetric shrinkage strain) can be computed as

$$\frac{\Delta V}{V}(\%) = 10.5 * \alpha(t) * \left[f * \frac{V\rho_m}{M_m} \right] * \left[1 - \frac{FL}{100} \right] \tag{51}$$

For a mixture of monomers of any functionality,

$$\frac{\Delta V}{V}(\%) = 10.5 * \alpha(t) * \frac{\Sigma(f_i\chi_i)}{\Sigma(M_{mi}\chi_i)}\rho_{mix} * \left[1 - \frac{FL}{100} \right] \tag{52}$$

where f_i is the functionality of monomer i, χ_i is mole fraction of monomer i, M_{mi} is molecular mass of monomer i and ρ_{mix} is density of the monomer mixture. For the case of the LAMP suspension which uses a mixture of two monomers, HDDA (Hexanediol diacrylate, di-functional) and EPETA (Ethoxylated penta erythryitol tetra acrylate, tetra-functional), the volumetric shrinkage can be expressed as follows:

$$\frac{\Delta V}{V}(\%) = 10.5 * \alpha(t) * \frac{2\chi_1 + 4\chi_2}{M_{m1}\chi_1 + M_{m2}\chi_2}\rho_{mix} * \left[1 - \frac{FL}{100} \right] \tag{53}$$

Thus, it is evident from the above relation that the volumetric shrinkage obtained is directly proportional to the degree of the conversion in the monomer systems. The degree of conversion (α) in turn is dependent on the energy dose and material composition used and can be measured using Fourier Transform Infrared Spectroscopy (FTIR) among other techniques where the absorption spectrum of a sample is measured before and after exposure. Figure 70 shows the FTIR spectra of the LAMP suspension obtained with different exposure times. The peaks at wavenumbers 1620 cm^{-1}, 1410 cm^{-1}, and 1750 cm^{-1} are characteristics of $C{=}C$ stretching, twisting and $C{=}O$ stretching respectively. As the polyermization reaction proceeds, the magnitude

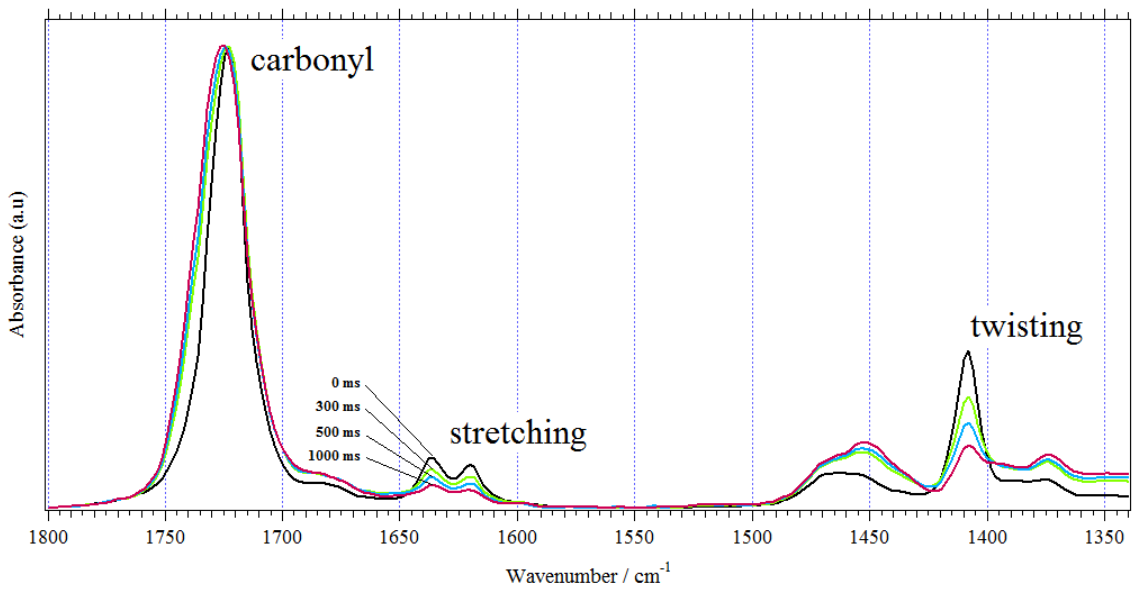


Figure 70: FTIR spectra of the LAMP suspension for different exposure times [12].

of the peak corresponding to $C{=}C$ stretching and twisting reduce while the magnitude of the peak corresponding to $C{=}O$ remains constant since it does not participate in the reaction. The degree of conversion in the sample is estimated by computing the change in area of $C{=}C$ peaks normalized with the area of $C{=}O$ peak as follows:

$$\alpha(t) = \frac{\left[\frac{Area_{1620}+Area_{1410}}{Area_{1750}}\right]_0 - \left[\frac{Area_{1620}+Area_{1410}}{Area_{1750}}\right]_t}{\left[\frac{Area_{1620}+Area_{1410}}{Area_{1750}}\right]_0} * 100 \tag{54}$$

Following this approach, extensive experimental studies have been reported by Kambly [14] to identify the shrinkage caused in the LAMP process for various compositions. Figures 71 and 72 show the variation of degree of conversion and corresponding volumetric shrinkage measured for LAMP suspension with varying percentages of photo initiator concentration and energy dose respectively.

From these results, it is evident that the degree of conversion and hence volumetric shrinkage vary greatly with the energy dose and the composition of the Suspension.

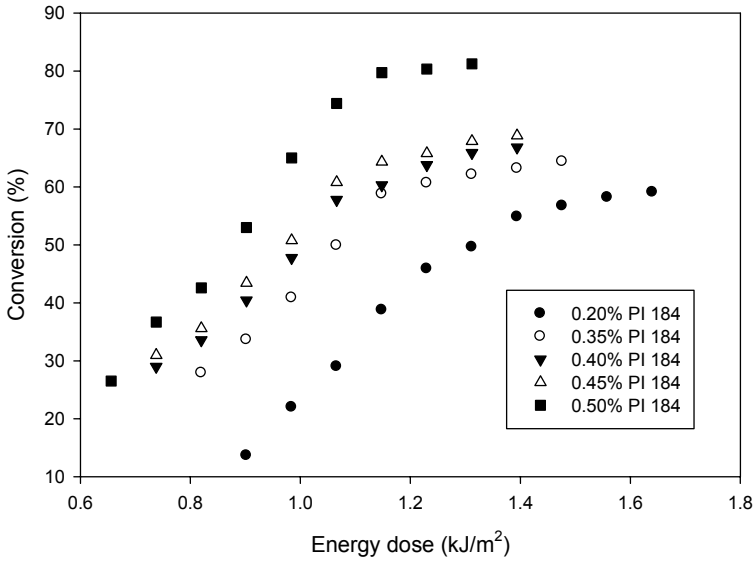


Figure 71: Degree of conversion with respect to energy dose for LAMP suspension with varying Photoinitiator concentration [14].

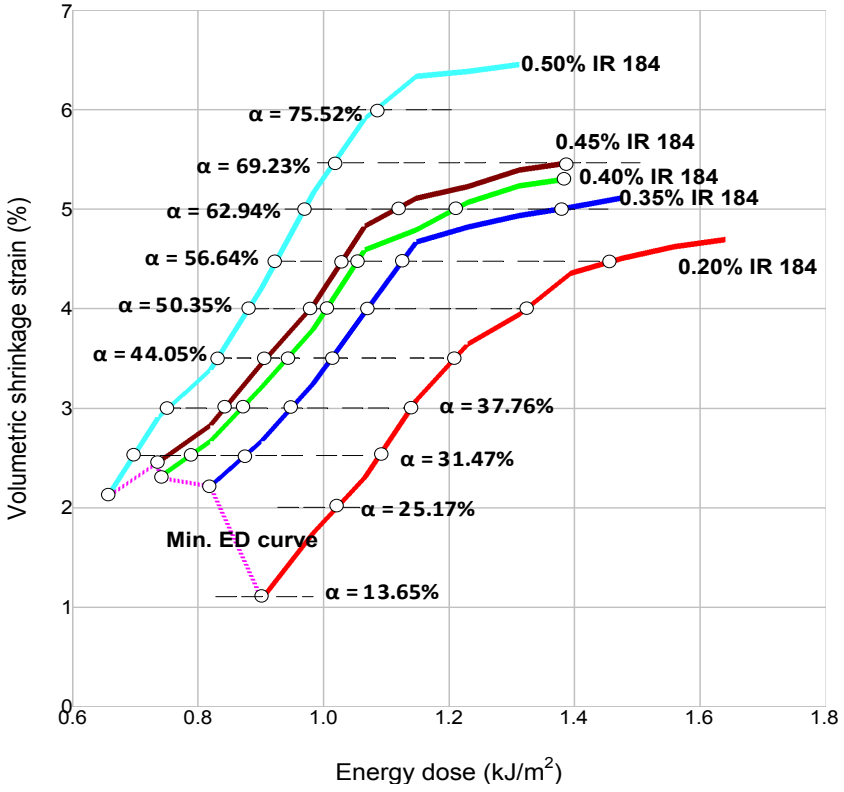


Figure 72: Volumetric shrinkage with respect to energy dose for LAMP suspension with varying Photoinitiator concentration [14].

The volumetric shrinkage presented here is microscopic shrinkage and unless there is significant porosity generated in the samples, it will directly translate to macroscopic shrinkage thereby affecting the cure profile and cure width C_w.

3.4.2.3 Ceramic Powder

The ceramic powder used in LAMP suspension has an effect on the cure width C_w as well. It causes light scattering and thereby significantly modifies the polymerization behavior of the suspension. These scattering effects are related the refractive index contrast of the ceramic particles defined as follows:

$$\frac{\Delta n}{n} = \frac{n_{cer} - n_o}{n_o} \tag{55}$$

where n_{cer} is the refractive index of the ceramic particles and n_o is the refractive index of the liquid medium. For suspensions that have a small contrast, the majority of the energy propagates in the forward direction and only a small fraction of it is scattered to the side. At higher refractive index contrasts, a much larger portion of the energy is scattered to the sides there by increasing the cure width. The scattering behavior can be quantified by the scattering length, l_{sc}, which is the distance over which a photon must travel before its propagation direction becomes randomized. A large scattering length corresponds to minimal scattering, while a small scattering length indicates a large degree of scattering. In the LAMP suspension, the refractive index of the monomer ($n = 1.4560$) is closely matched to that of silica ($n = 1.4603$), and so the refractive index contrast is quite low (0.3%). However, even at such a low contrast, it was reported that the curing characteristics of the LAMP suspension significantly deviate from commercially available resins that are devoid of any ceramic loading [13]. Several other works presented in the literature have studied the effect of particle concentration, size, refractive index and volume fraction of the ceramic loading on the curing characteristics of ceramic loaded stereolithography suspensions [74, 119–124].

3.4.2.4 Suspension Composition

Apart from the ceramic powder, LAMP suspension has many other components like photoinitiator, monomer, dispersant, absorber etc. The relative concentrations of each of these components has a significant impact on the curing characteristics of the mixture. More specifically, they effect the E_c and D_p of the suspension which change the cure profile and thereby affect the cure width C_w. A change in E_c and D_p also has a direct effect on each of the previous parameters discussed. It was shown that cure width dependence on light intensity for a simple Gaussian beam (and by extension for LAMP intensity which is a summation of Gaussians) is a function of E_c. The volumetric shrinkage also changes with respect to E_c and D_p because the degree of conversion obtained is dependent on E_c. Hence, a change in E_c and D_p due to a change in suspension composition not only directly effects the cure profile (and therefore cure width) but also significantly changes the sensitivity of the suspension to each of the parameters effecting cure width. Therefore, an understanding of the dependence of E_c and D_p on at least some of the major components of the suspension is vital.

Some prior work has been reported by Tomeckova and Halloran [125–127] that is aimed at developing a predictive model for E_c and D_p of the LAMP suspension as function of the ceramic loading, photo initiator and absorber concentrations. The total attenuation coefficient, α (which is simply the reciprocal of sensitivity, $\alpha = \frac{1}{D_p}$) is treated as the sum of the attenuation coefficients each of the following:

(a) scattering by ceramic particles,

(b) absorption by ceramic particles,

(c) absorption by photo initiator, and

(d) absorption of the UV-absorbing dye

.

The resin sensitivity D_p was modeled as follows:

$$\frac{1}{D_p} = \alpha_{sc} + \alpha_{cer} + \alpha_P + \alpha_D \tag{56}$$

where α_{sc}, α_{cer}, α_P and α_D are the attenuation coefficients due to scattering of the ceramic particles, absorbance of the ceramic particles, absorption due to photo initiator and absorption due to the dye. Therefore assuming the ceramic is UV transparent, the sensitivity can be predicted in terms of the concentrations of the components and their extinction coefficients, ϵ, by

$$\frac{1}{D_p} = \frac{1}{l_{sc}} + (1 - \Phi)(c_P\epsilon_P + c_D\epsilon_D) \tag{57}$$

where l_{sc} is the scattering length of the suspension as discussed in the previous section,Φ is the volume fraction of ceramic powder in the suspension, c_P is the concentration of photo initiator, ϵ_P is the extinction coefficient of the photoinitiator, c_D is the dye concentration, and ϵ_D is the extinction coefficient of the dye. The extinction coefficient is wavelength dependent, and for non-laser sources like the one used in LAMP, the extinction coefficient must be convoluted with the intensity distribution. Hence, the sensitivity of the suspension D_p can be predicted from this expression, provided that the scattering length, concentrations and extinction coefficients of the components with respect to each of the light intensity peaks of the illumination source used in LAMP (Refer to Figure 60) are known. Wu et al. [122] further characterized variation of D_p in absorption dominated systems (systems with high scattering length l_{sc}, i.e, systems with low refractive index contrast $\Delta n/n$ like the LAMP suspension) and scattering dominated systems (systems with high l_{sc} and high contrast $\Delta n/n$).

Similar to the behavior of the resin sensitivity, the critical energy (E_c) can be predicted from the individual components in the suspension, using the Tomeckova inhibitor exhaustion model [125–127]. Photons are either absorbed by inhibitors or dye or react with the photoinitiator to release free radicals. These radicals can either be annihilated by the inhibitors in the system or can contribute to free radical

polymerization. Inhibitors included native oxygen in the suspension and inhibitors such as quinones which are added to make the monomer stable during storage. In order for polymerization to take place, all of the inhibitor must be consumed by free radicals so that there are excess radicals to propagate the reaction. The inhibitor is exhausted at the critical energy dose, E_c, which is dependent on the composition of the suspension as

$$E_c = (\gamma_{INH} c_{INH} + \gamma_D c_D) \frac{h\nu}{\Omega} \left[\frac{\frac{1}{l_{sc}} + (1-\Phi)(c_P \epsilon_P + c_D \epsilon_D)}{c_P^2 \epsilon_P^2} \right] \tag{58}$$

where γ_{INH} is the number of radicals removed per inhibitor (such as oxygen or added quinone), c_{INH} is the the concentration of the inhibitor, γ_D is the number of radicals that were not generated due to the presence of the dye, h is Plancks constant, ν is the frequency of the light, and Ω is the number of free radicals given off per photon absorbed [125–127]. Neglecting the higher order terms, this can be written as

$$E_c = (1-\Phi) \frac{h\nu}{\Omega} (\gamma_{INH} c_{INH} + \gamma_D c_D) \frac{1}{c_P \epsilon_P} \tag{59}$$

Below this energy, the inhibitors and dye absorb free radicals and no curing occurs. Above this energy, free radicals are available to allow the propagation of polymerization reaction and curing occurs.

Hence, from these models, a qualitative understanding of the dependence of E_c and D_p, which effect the cure width C_w, on the suspension composition (does not include effects of dispersant that modifies the rheology of the suspension) can be gained. However, an explicit expression for C_w in terms of the suspension composition would be very difficult to achieve as the light scattering effects need to be modeled accurately which is challenging.

3.4.2.5 Suspension Preparation

Apart from suspension composition, it is determined from experience that the suspension preparation method has an effect on the curing parameters D_p and E_c as

well. The procedure followed for preparing the LAMP Suspension is described here. The monomers (HDDA and EPETA), dispersant and UV absorber are combined and mixed together as all of these are liquid ingredients. Next, the photo initiator is added, which is in powder form, mixing for several minutes until it has fully dissolved in the liquid solution. Finally, the ceramic powder is added. Since LAMP suspension uses a very high ceramic loading (55 vol%), the powder is added incrementally in two or three batches to allow for full incorporation. This mixture is then balled milled in a ceramic jar filled with milling media.

From prior experimental trials, it was observed that the several Suspension preparation parameters like speed of milling, total duration of milling, amount of milling media in the ceramic jar with respect to the Suspension quantity, blending time allowed for photo initiator and UV absorber (if they are added separately prior to a build) each have an impact on the curing characteristics of the suspension and hence the cure widths obtained.

In order to minimize these effects, the Suspension preparation is standardized as far as possible. It is milled at a speed of (15 rpm) for 3 days. The milling media in the jar is also held constant for a constant volume of the mixture (4 kg of milling media for 3Ltrs of suspension). When the photo initiator and UV absorbers are added separately before a build, it is ensured that the suspension is milled for at least 24 hours before it is used.

3.4.3 Methodology

As discussed in the previous section, the cure width C_w obtained is a result of the complex interaction of several parameters which is very difficult to model accurately. Hence, in order to understand the cure depth behavior of LAMP suspensions, a simple experimental study is proposed. From experience, four important parameters are singled out for studying their specific effects on cure width C_w. They are:

(a) Feature size,

(b) Peak light intensity,

(c) UV absorber concentration, and

(d) Photoinitiator (PI) concentration.

Discrete values for each of the parameters were identified and the cure width characteristics at each of these parameters are determined experimentally. Cure widths were determined by exposing squares of known length over a glass slide (the experimental setup is exactly same as the cure depth measurement set up shown in Figure 55) and by measuring the deviation of the cured square lengths obtained. Figure 73 shows a sample image with known squares that is used for exposure and Figure 74 shows an image of the corresponding cured layers obtained.

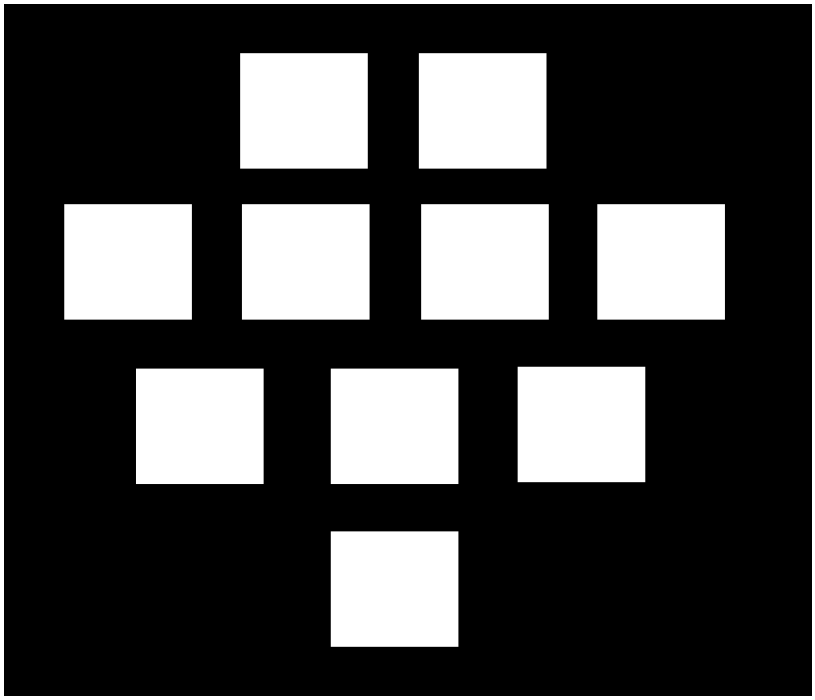


Figure 73: A sample exposure image with a known constant square length with 10 different tiles. Each tile is exposed at a different energy dose.

It is to be noted that each tile in the exposure image in Figure 73 is exposed at a different exposure dose and hence the resulting square lengths obtained in the cured square tiles shown in Figure 74 are different. The corresponding cure widths C_w at

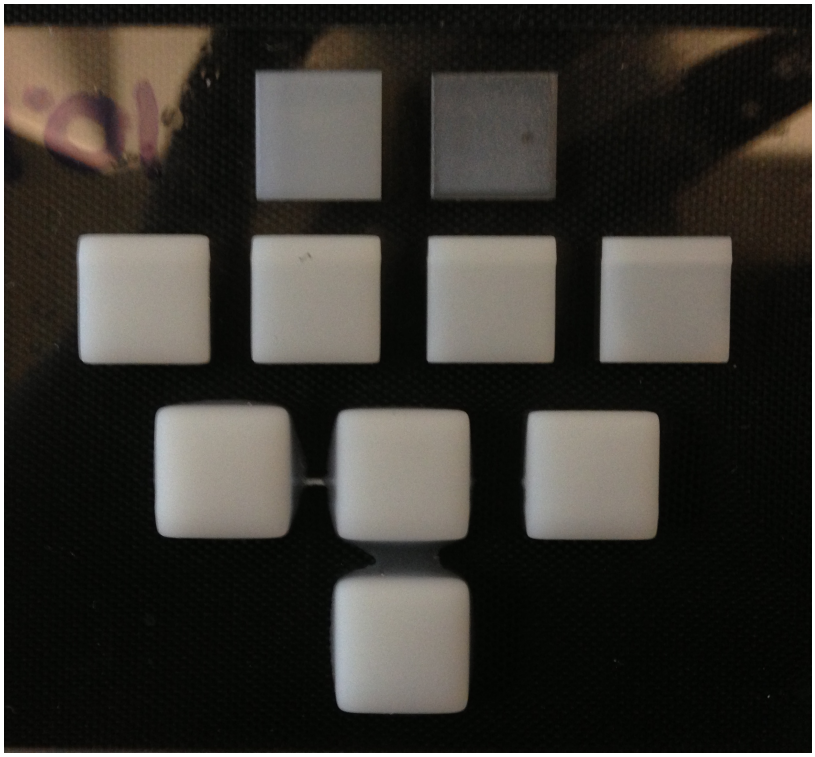


Figure 74: Cured squares obtained by exposing the image in Figure 73. Each tile is exposed at a different energy dose, and hence its deviation from the nominal square length in the exposure image is different.

each of the exposure doses is computed as follows:

$$C_w = \frac{l_{cured} - l_o}{2} \tag{60}$$

where l_{cured} is the square length obtained after curing each tile, and l_o is the nominal square length in the exposure Image. In this manner, at discrete values of each of the parameters (a),(b), (c), and (d), the cure width trends with respect to the energy dose are identified. Critical energy dose and sensitivities for cure width C_w analogous to E_c and D_p for the case of cure depth C_d are introduced. For the sake of clarity, from here on, a different notation is used for identifying the critical energy doses and sensitivities corresponding to cure depth and cure width respectively. The critical energy dose corresponding to cure depth C_d is denoted from here on by E_c^d and the sensitivity for cure depth is denoted by D_p^d. Similarly, the critical energy dose and sensitivity for cure width C_w are represented by E_c^w and D_p^w respectively. The

extra superscripts 'd' and 'w' are added to the usual parameters E_c and D_p where 'd' denotes depth and 'w' denotes width.

A new parameter known as broadening depth B_d is introduced, which gives the maximum cure depth that can be achieved before the layers begin to cure in the width direction. It is determined by computing the cure depth obtained at an energy dose equal to the cure width critical energy dose E_c^w at which lateral curing just begins to occur as shown in Equation 61.

$$B_d = D_p^d \ln \frac{E_c^w}{E_c^d} \tag{61}$$

This is a good measure for characterizing the side-scatter induced cure width broadening of each composition. Ideally, the composition should be optimized for maximum broadening depth in order to get deep cured parts with good layer-to-layer bonding and minimal excess side scattering.

The parameters introduced here to analyze the cure width characteristics of the material system for LAMP are analogous to the ones introduced by Gentry [13] for characterizing the line widths of the LAMP suspension. However, it will be shown that the results obtained here deviate from the 'quasi-Beer-Lambert' law introduced in her thesis. These differences can be attributed to following:

1) differences in the exposure set up used,

2) difference in the curing times used,

3) difference in the method by which cure widths C_w are calculated.

The results obtained from the experimental investigations using the methodology described in this section are given next.

3.4.4 Dependence of broadening parameters on feature size

In this section, the dependence of cure width C_w on feature size is investigated. As mentioned in Chapter 2, the HP turbine blades that LAMP intends to fabricate have feature sizes across a range of length scales (all the way from a few 100 microns to

a few centimeters). Hence, it is very crucial to determine whether and how various feature sizes scale differently in a build. In order to investigate this, squares of several different sizes are considered. For each square length, several multiple tiles are exposed at different energy doses and their corresponding cure widths are computed, as discussed in the previous section, in order to identify the width critical energy E_c^w and the width sensitivity D_p^w at that feature size. Square lengths ranging from 1 pixel to 900 pixels in width were considered (Each pixel is of length 1/1500 inches, i.e, $\approx 16.9\mu m$. Specifically, following square lengths (in pixels) were tested: 1-5, 10, 25, 50, 75, 100, 200, 300, 450, 600, 750, 900. However, during the development process, features corresponding to tiles less than 25 pixels wide were washed out as they are so minuscule. Hence, the data for 25 pixels wide or higher are presented. The suspension composition used for this experimental set is as shown in Table 6

Table 6: Suspension composition used for characterizing the dependence of cure width C_w on Feature Size.

Ceramic Powder	55 Vol%
Dispersant	8.32 g w.r.t 100g powder
Monomer	9:1 ratio of HDDA and EPETA
Photo Initiator	5 g w.r.t 100g monomer
UV Absorber	0 g w.r.t 100g monomer

The variation of cure width C_w with respect to the energy dose computed for a few square sizes is shown in Figure 75. The following observations can be made from the figure:

1) Cure Width C_w varies linearly with respect to energy dose, unlike in the case of a 'quasi-Beer-Lambert' law where it varies linearly with the logarithm of energy dose.

2) The side curing characteristics do in fact change with feature size as evidenced by the changes in slope (and x-intercept).

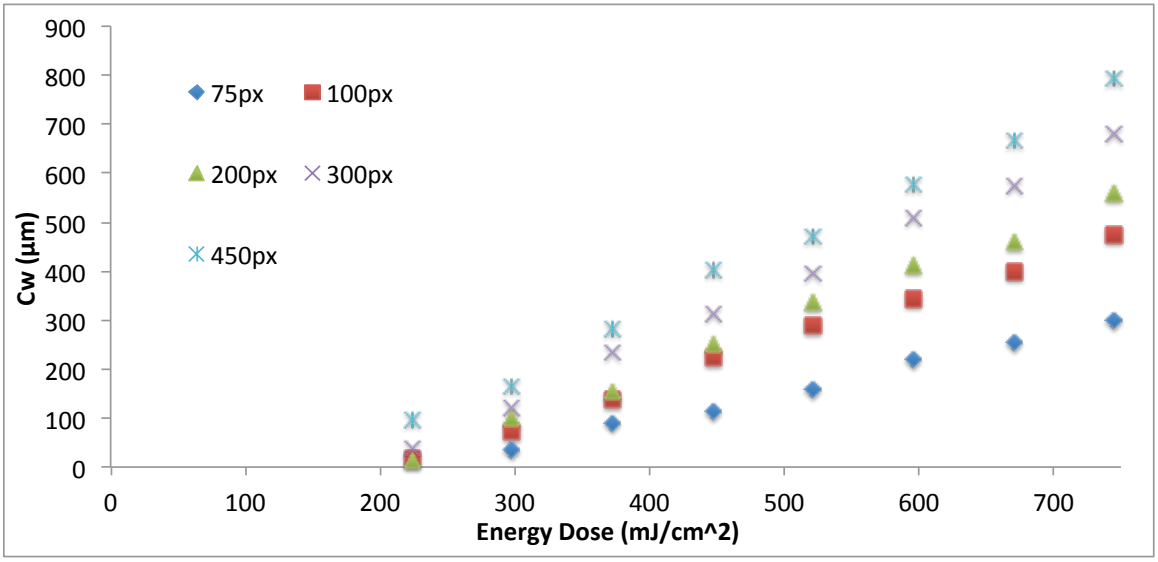


Figure 75: Variation of cure width C_w with respect to Energy Dose for various feature sizes.

Only a sample of the set of square lengths investigated are shown in Figure 75 to avoid clutter. As the square lengths were increased, it was observed that the cure width curves approached a steady slope and intercept resulting in almost a constant width sensitivity D_p^w and width critical energy E_c^w beyond a critical feature size. This phenomenon is depicted in Figures 77 and 76 which show the variation of D_p^w and E_c^w with respect to square size, respectively. As can be observed, beyond a feature size of $450\mu m$, both D_p^w and E_c^w taper off to constant values.

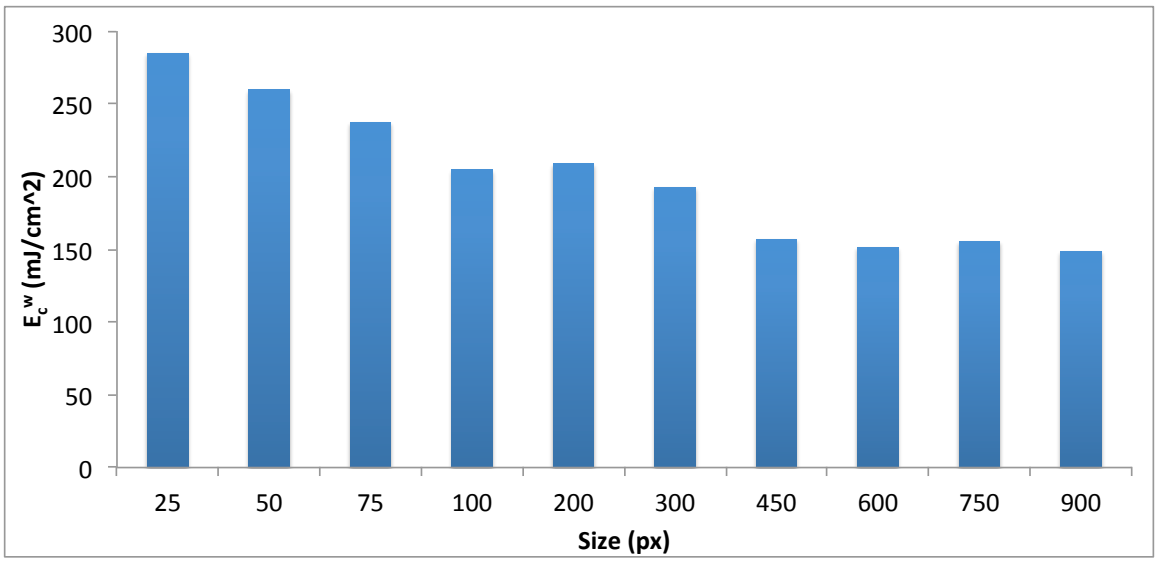


Figure 76: Variation of E_c^w with respect to various feature sizes.

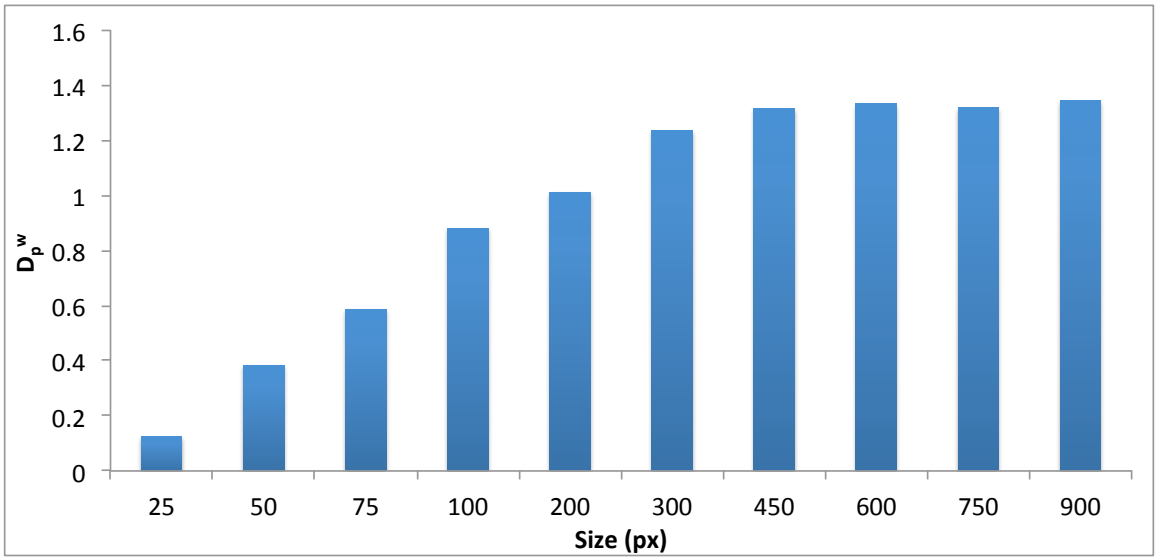


Figure 77: Variation of D_p^w with respect to various feature sizes.

The cure depth C_d values were observed to be constant and independent of feature size. The cure depth curve for this composition is shown in Figure 78 which yields a E_c^d and D_p^d of 39.4 mJ/cm^2 and 638.51 μm. Once the cure depth curve is determined, the broadening depth at each of these feature sizes can be computed using Equation 61. Figure 79 shows the variation of B_d observed as a function of feature size. It can be

observed that B_d follows a similar pattern to E_c^w where in it gradually decreases as the feature size is increased and reaches a constant value beyond the critical size of 450 px.

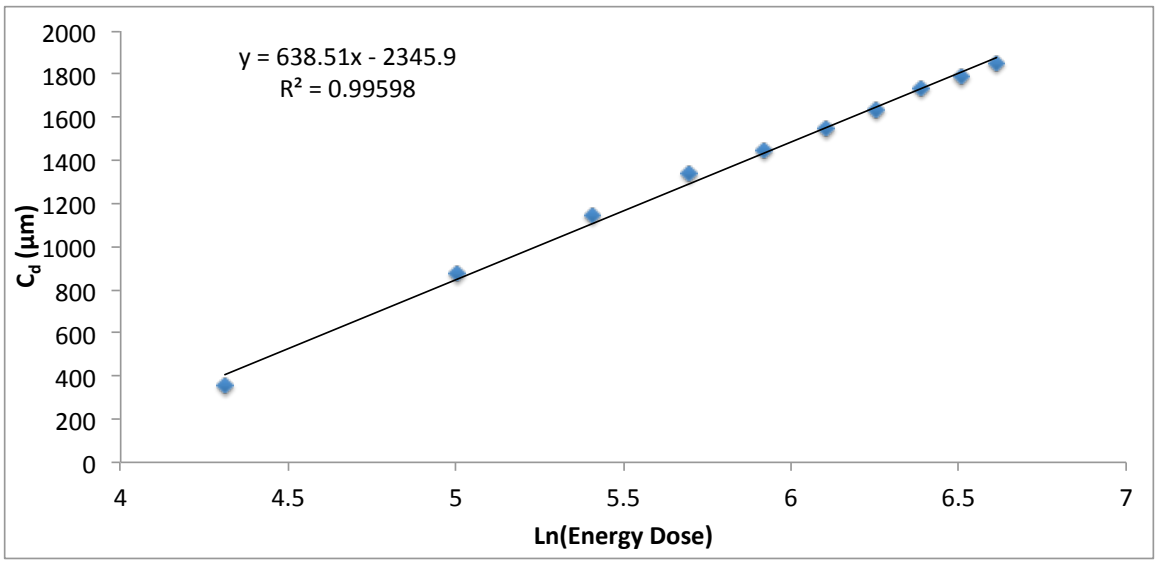


Figure 78: Cure depth curve for this material composition; Does not change significantly with feature sizes.

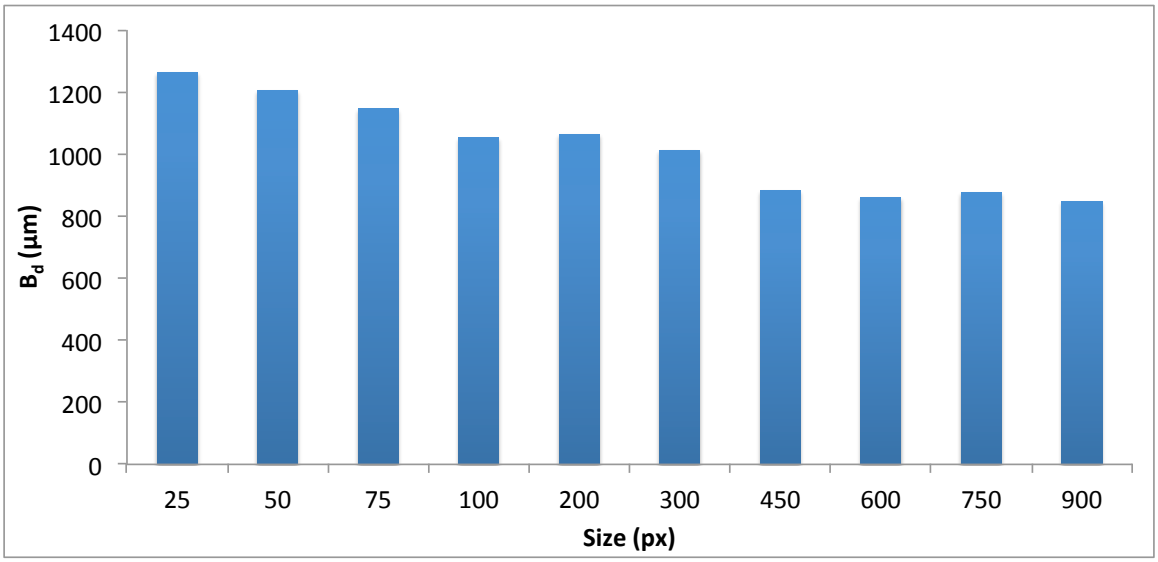


Figure 79: Variation of B_d with respect to various feature sizes.

3.4.5 Dependence of broadening parameters on light intensity

In this section, the dependence of the cure width on light intensity is presented. The light intensity is another important parameter that can influence the cure width. The power of the uv light source used in the LAMP machine drops significantly over time. It is therefore crucial to understand how this change in light source power effects the depth and width curing characteristics of the LAMP suspension.

To investigate this aspect, the width and depth curing characteristics were measured at various light source intensities. Since, the power of the light source in the LAMP machine cannot be intentionally modulated, neutral density filters were used to modify the light intensity incident on the suspension. In addition to the full intensity, two different neutral density filters were inserted in the optical path to yield approximately 50% and 30% of the full intensity. As was observed in the previous section, the width curing characteristics stabilize beyond a square size of 450 pixels. Therefore, in this as well as in the remainder of the experimental studies presented, a square size of 500 pixels is used. The experimental setup used for exposing the tiles using neutral density filters in the light path is exactly the same as presented in Figure 59. The LAMP suspension composition used for this study is shown in Table 7.

Table 7: Suspension composition used for characterizing the dependence of cure width C_w on Light Intensity.

Ceramic Powder	55 Vol%
Dispersant	8.32 g w.r.t 100g powder
Monomer	9:1 ratio of HDDA and EPETA
Photo Initiator	6 g w.r.t 100g monomer
UV Absorber	0 g w.r.t 100g monomer

The variation of the cure width C_w with respect to various energy doses at different light intensities is shown in Figure 80. At lower light intensities, the exposure

time is increased so as to achieve the same energy doses at the points of cure width measurement. As can be observed, both the width critical energy dose E_c^w and width sensitivity D_p^w are effected by light intensity.

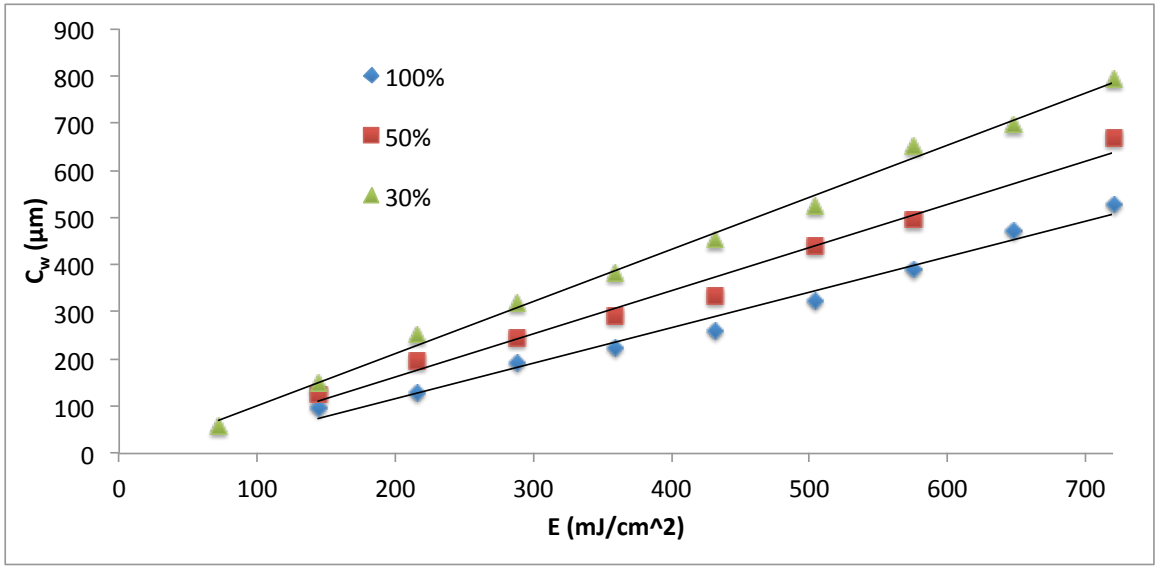


Figure 80: Variation of cure width C_w with respect to energy dose for various light intensities.

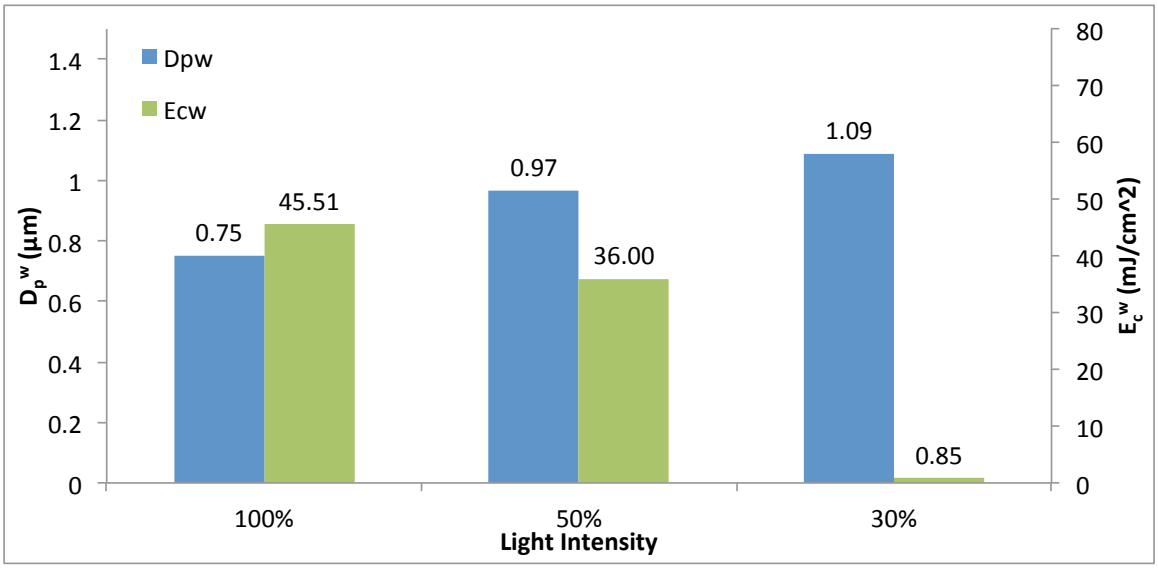


Figure 81: Variation of D_p^w and E_c^w with respect to light intensity.

The variation of E_c^w and D_p^w with respect to light intensity is shown in Figure 81. As can be seen, D_p^w increases and E_c^w decreases as the light intensity decreases. This is a little counterintuitive at the outset but does seem valid when one considers the fact that at lower energy doses, the exposure times are much higher and hence higher side curing is caused resulting in lower width critical energies and sensitivities.

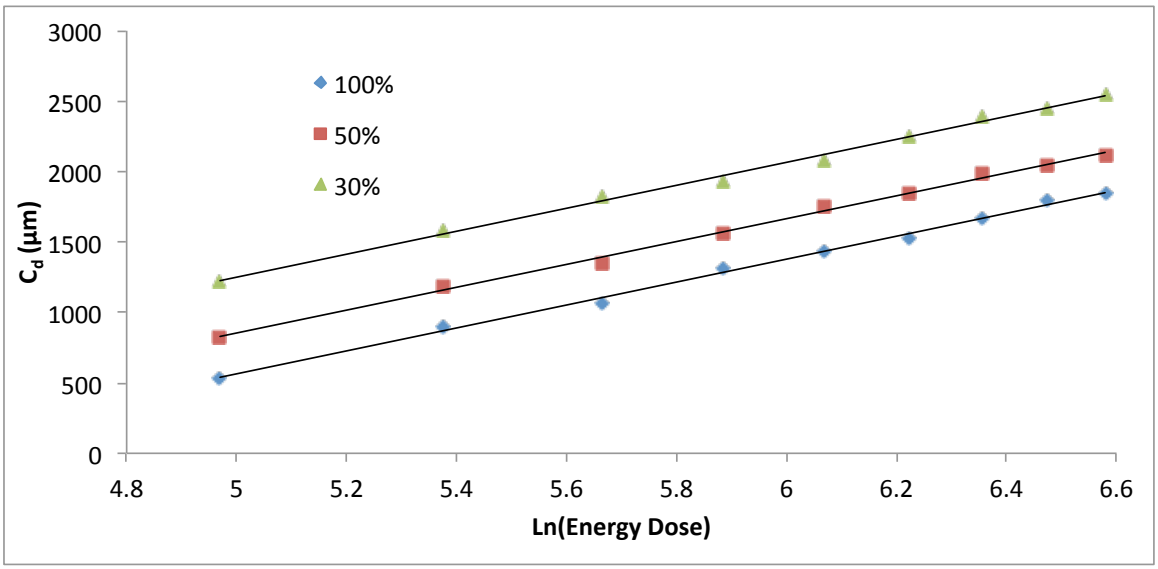


Figure 82: Cure depth curves for various light intensities.

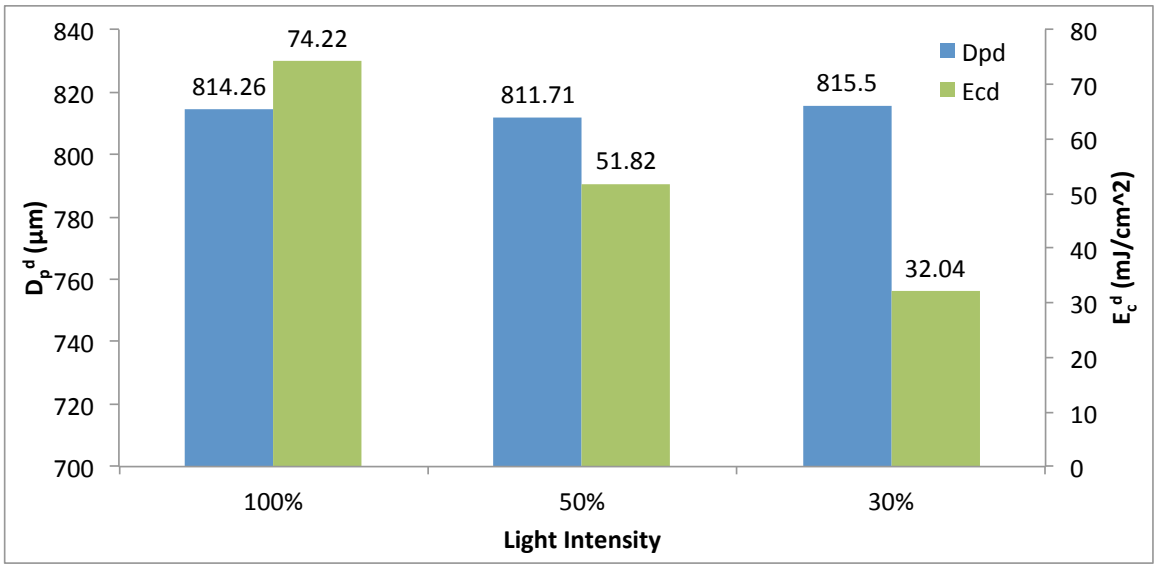


Figure 83: Variation of D_p^d and E_c^d with respect to light intensity.

However, a trend line could not be obtained for these variations as only three intensity levels are considered. This is because of the lack of availability of suitable neutral density filters (The other filters available attenuate the light by 90% or more at which point the exposure times become so large that they are not representative of the process).

The cure depth curves for these intensitites are shown in Figure 82. Analogous to the results presended on the gray scaling studies previously by Conrad [12], the depth critical energy E_c^d decreases with decreasing light intensity while the depth sensititviy D_p^d remains more or less constant as shown in Figure 83.

3.4.6 Dependence of broadening parameters on UV absorber concentration

The UV absorber is an important component in the LAMP suspension and has a high impact on the curing characteristics of the suspension due to its high extinction coefficients. Small changes in the UV Absorber concentration can lead to substantial changes in the curing parameters and hence its effect needs to be investigated. Models proposed by Tomeckova [125–127] predict the variation of depth curing parameters E_c^d and D_p^d with respect to UV absorber concentration. The depth curing results obtained here are compared against these models. However, no known models exist for width curing characteristics. The suspension compositions used for this experiment set are given in the Table 8 (only the UV absorber concentration is changed while keeping all the other parameters constant).

The cure depth curves obtained for various values of the UV absorber concentration are shown in Figure 84. As can be seen, both D_c^d and E_c^d are functions of the UV absorber concentration. However, they qualitatively vary in a fashion as predicted by Tomeckova's models [125–127] (Equations 57 and 58).

Table 8: Suspension compositions used for characterizing the dependence of cure width C_w on UV Absorber concentration.

Ceramic Powder	55 Vol%
Dispersant	8.32 g w.r.t 100g powder
Monomer	9:1 ratio of HDDA and EPETA
Photo Initiator	6 g w.r.t 100g monomer
UV Absorber	(0, 0625, 0.125, 0.1875, 0.25) g w.r.t 100g monomer

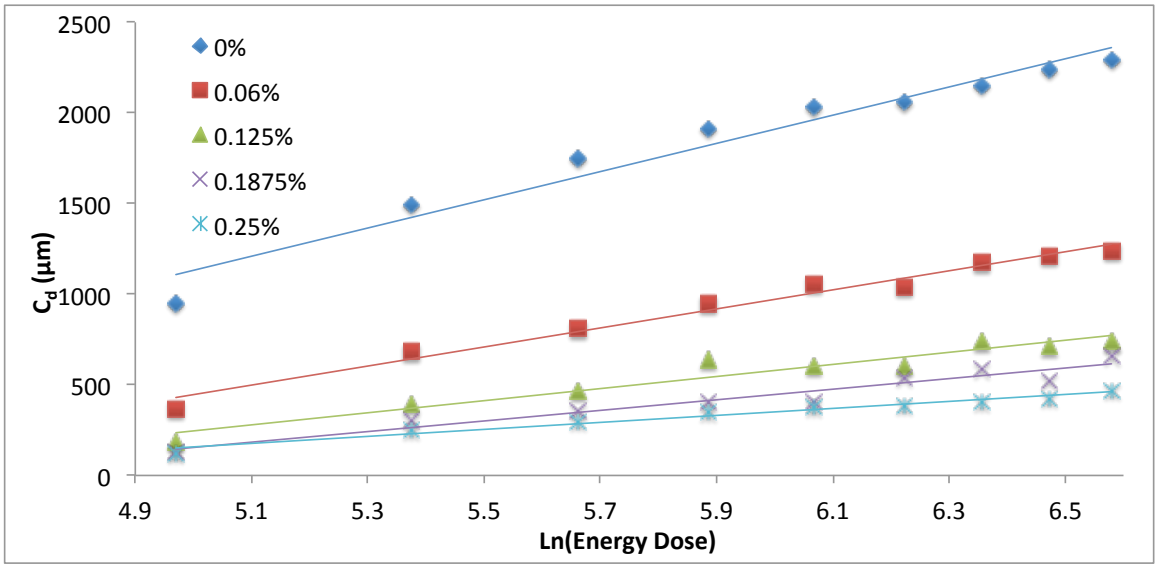


Figure 84: Cure depth curves for various UV absorber concentrations.

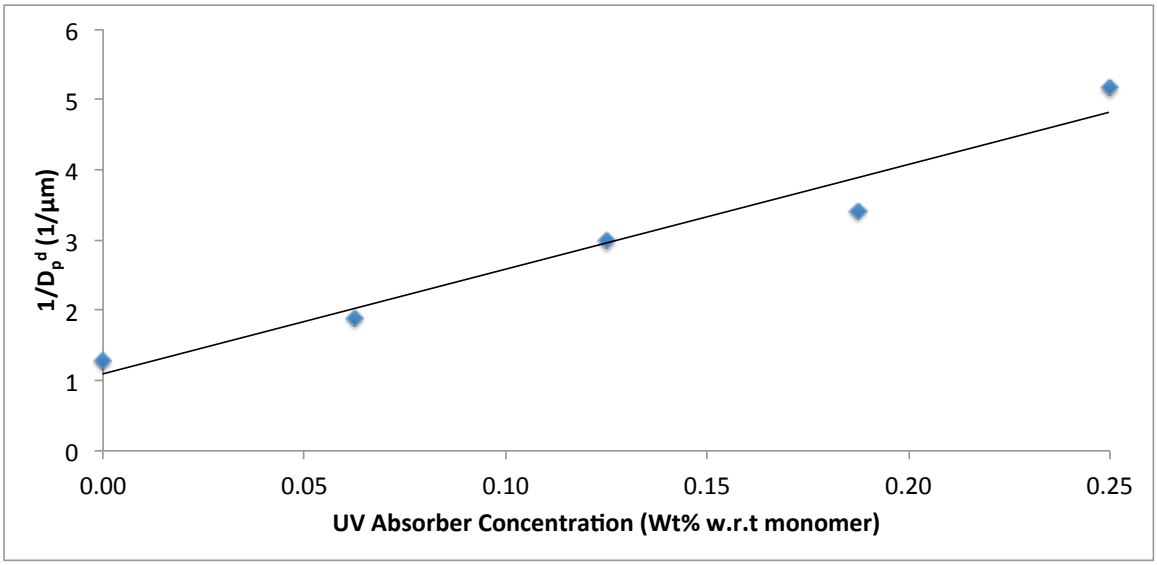


Figure 85: Variation of E_c^d with respect to UV absorber concentration.

From Equations 57 and 58, it is evident that when all the other parameters such as the wavelength of the incident light, the photoinitiator concentration, the particle size distribution and loading of the ceramic powder etc. are kept constant, the reciprocal of depth sensitivity and depth critical energy vary linearly with respect to UV absorber concentration ($1/D_p^d \propto C_D$ and $E_c^d \propto C_D$; C_D is the die concentration in Tomeckova's terminology).

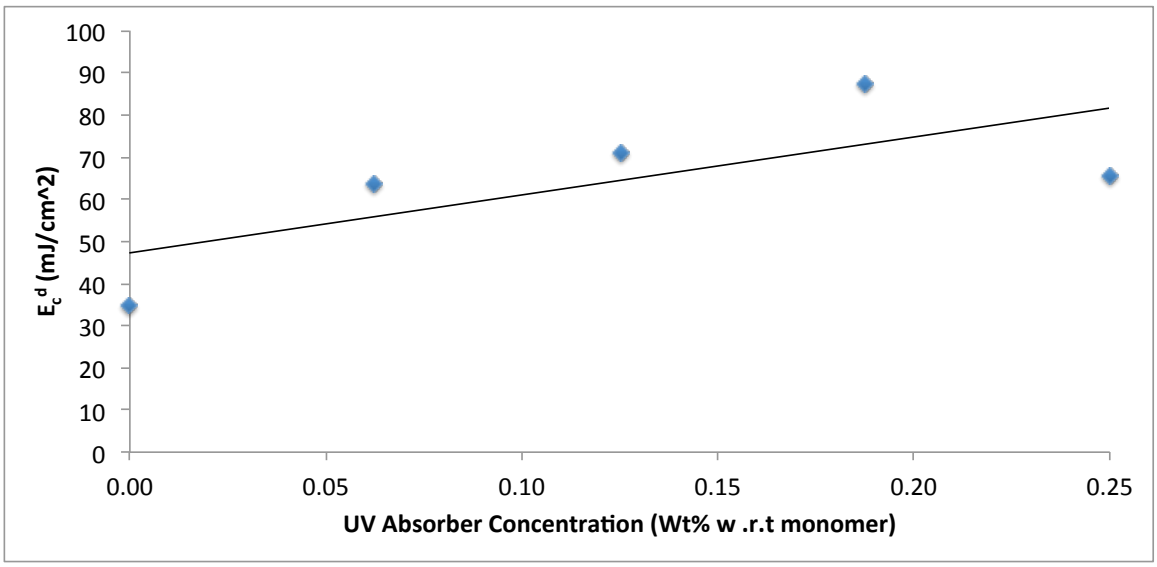


Figure 86: Variation of D_p^d with respect to UV absorber concentration.

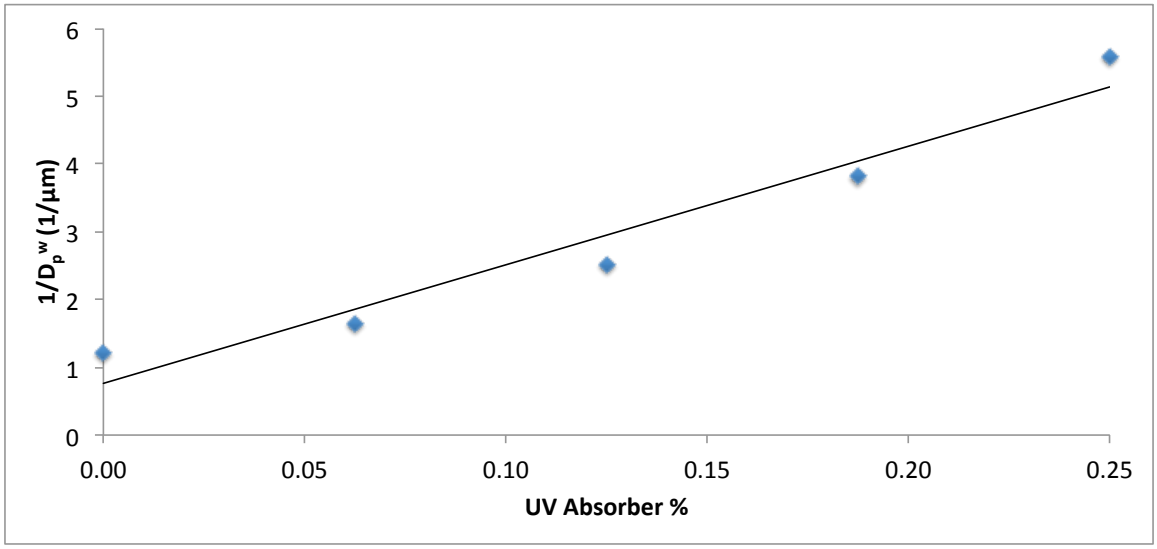


Figure 87: Variation of D_p^w with respect to UV absorber concentration.

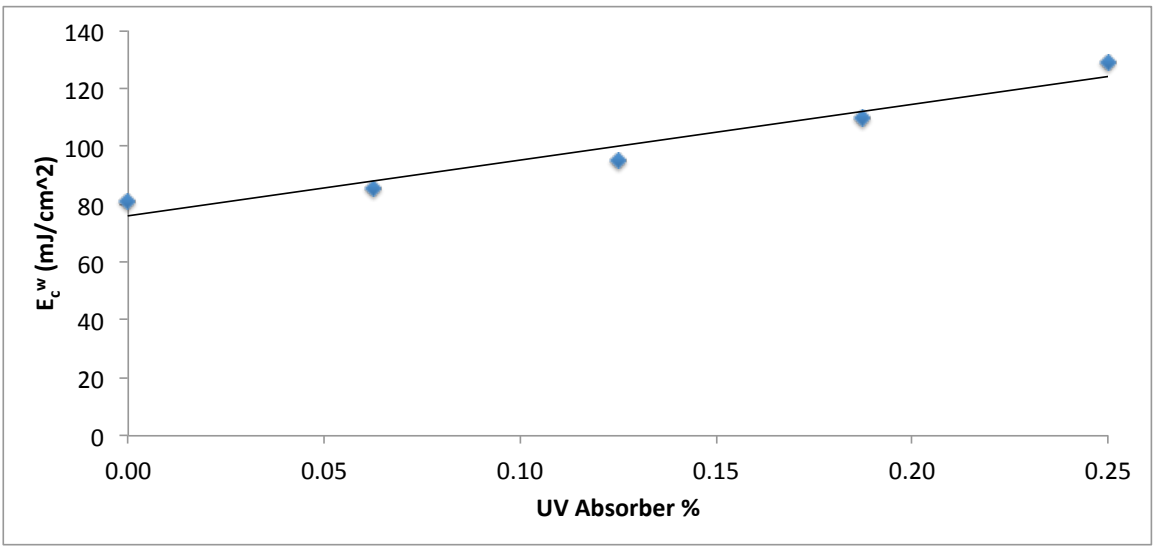


Figure 88: Variation of E_c^w with respect to UV absorber concentration.

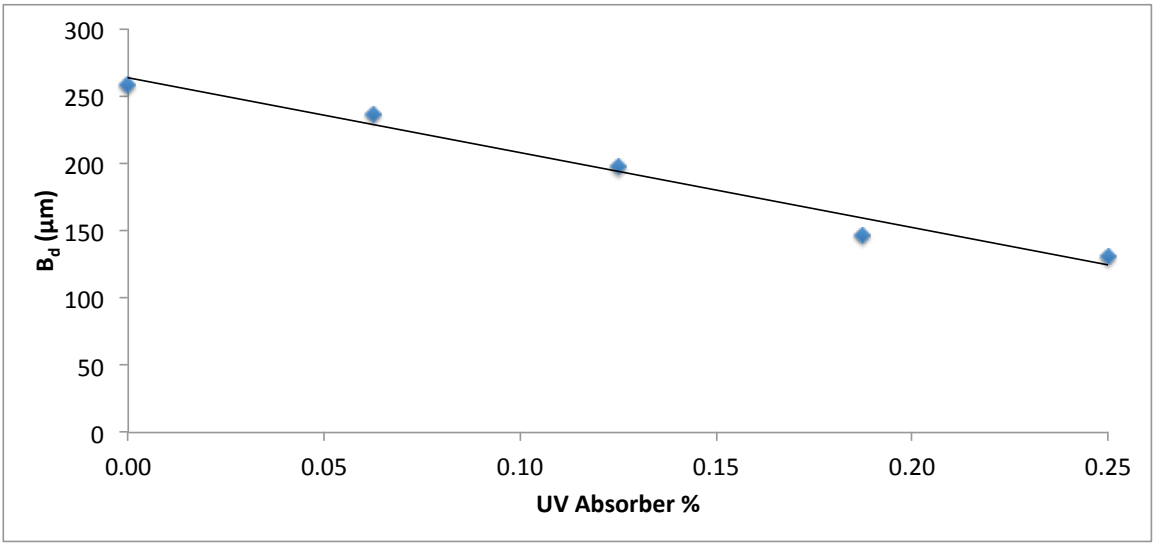


Figure 89: Variation of B_d with respect to UV absorber concentration.

These trends predicted by Tomeckova's models are observed in the data obtained as shown in Figures 85 and 86 which present the experimentally observed variation of D_p^d and E_c^d, respectively. Although, Tomeckova's models predict the variation of only the depth curing parameters, interestingly enough, the width curing parameters

also qualitatively see to follow a similar trend, i.e., $\frac{1}{D_p^w} \propto C_D$ and $E_c^w \propto C_D$. Figures 87 and 88 confirm this observation. From the depth and width curing data, the broadening depths for various UV absorbers percentages are computed and observed to be inversely proportional to the absorber concentration as shown in Figure 89.

3.4.7 Dependence of broadening parameters on photoinitiator concentration

Photoinitiator concentration in the LAMP suspension is yet another parameter that has a significant impact on its curing characteristics. Its effect is characterized in this section. The various suspension compositions used in this experimental set are shown in Table 9 (only the PI concentration is varied while keeping the rest of the parameters constant).

Table 9: Suspension compositions used for characterizing the dependence of cure width C_w on PI concentration.

Ceramic Powder	55 Vol%
Dispersant	8.32 g w.r.t 100g powder
Monomer	9:1 ratio of HDDA and EPETA
Photo Initiator	(2, 4, 6, 8, 10) g w.r.t 100g monomer
UV Absorber	0 g w.r.t 100g monomer

The Tomeckova models in Equations 57 and 58, while keeping all other parameters constant, predict that the inverse of the depth sensitivity varies linearly with respect to the PI concentration while the depth critical energy dose varies linearly with the inverse of the PI concentration, i.e., $\frac{1}{D_p^d} \propto C_P$ and $E_c^d \propto \frac{1}{C_P}$ (C_P is the PI concentration in their terminology). Similar qualitative trends are observed in the experimental data for depth curing parameters as presented in Figures 90 and 91.

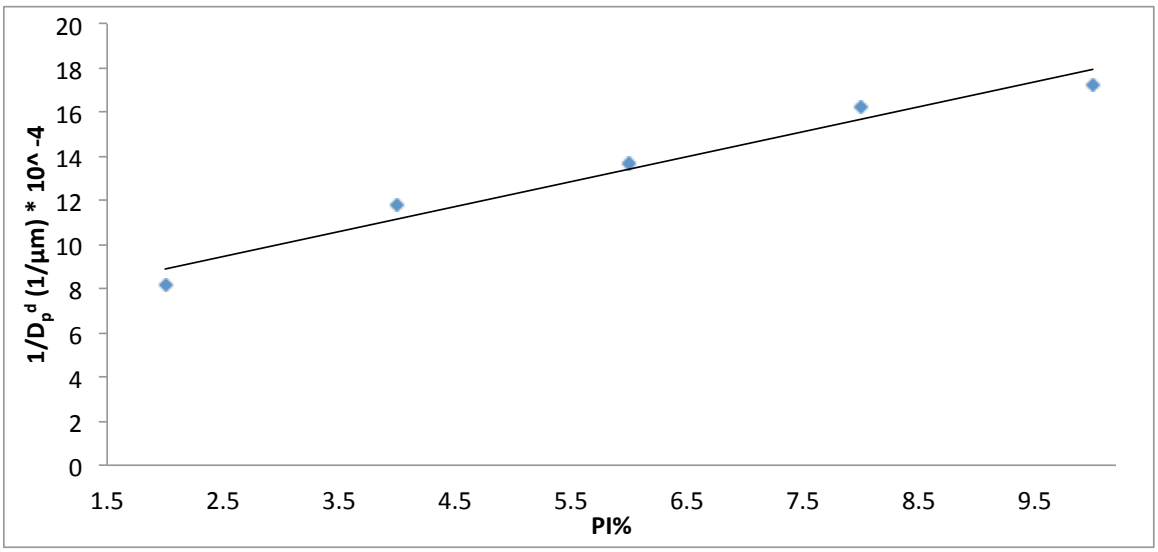


Figure 90: Variation of E_c^d with respect to photoinitiator concentration.

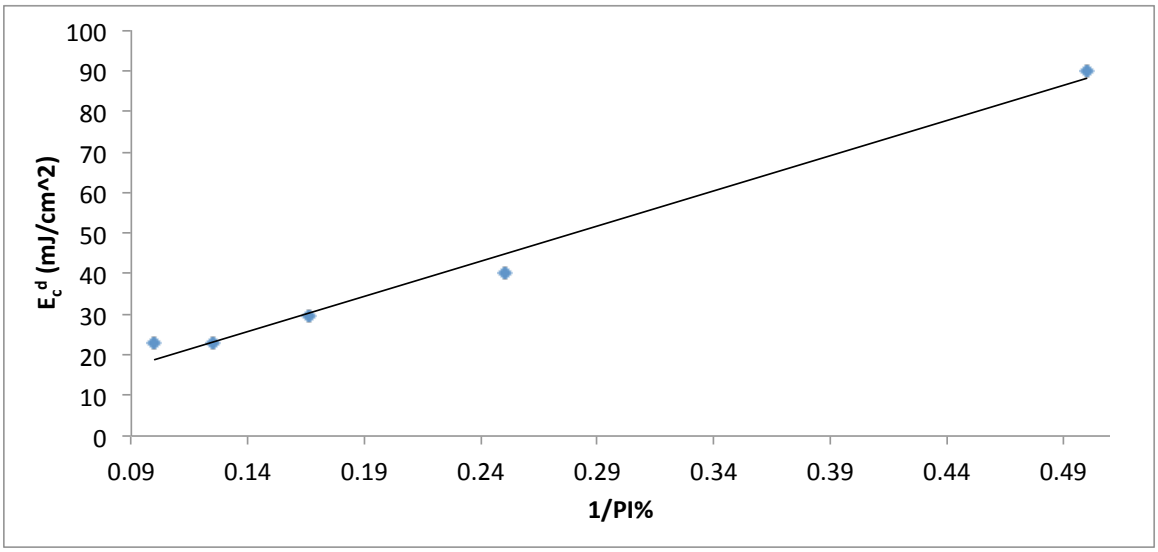


Figure 91: Variation of D_p^d with respect to photoinitiator concentration.

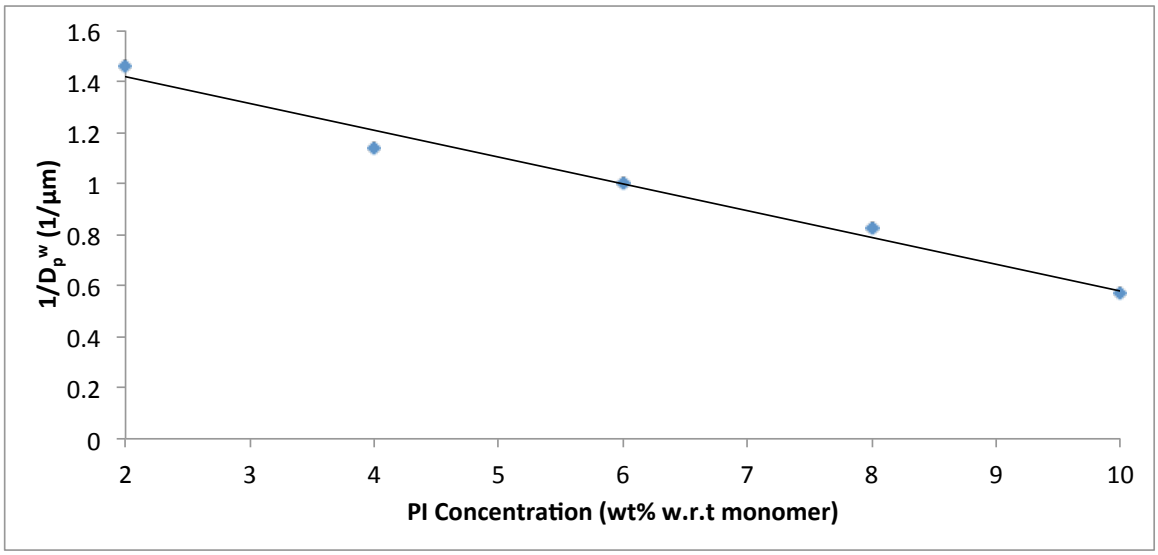


Figure 92: Variation of D_p^w with respect to photoinitiator concentration.

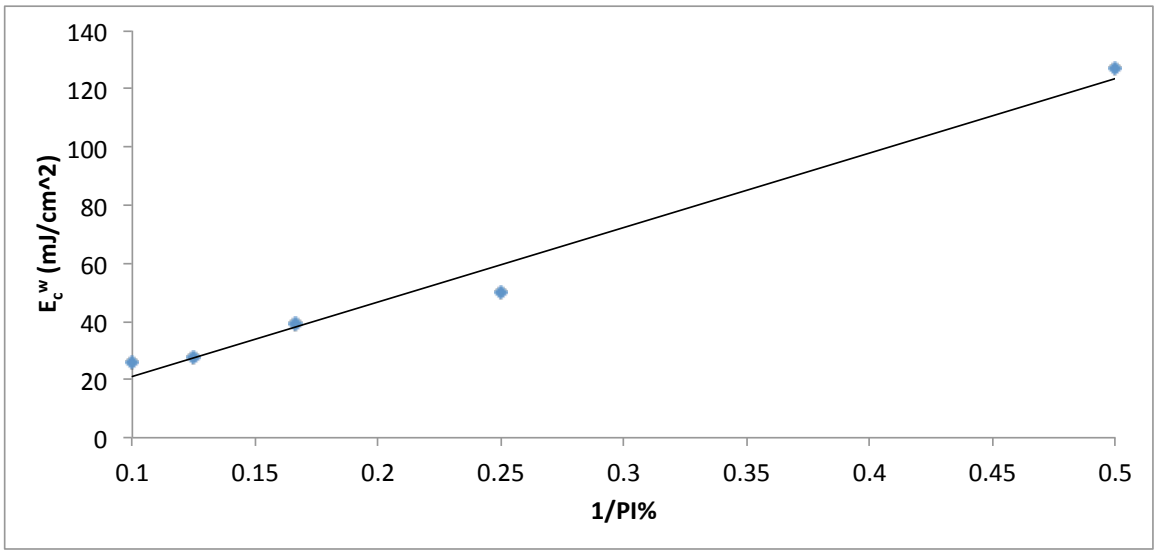


Figure 93: Variation of E_c^w with respect to photoinitiator concentration.

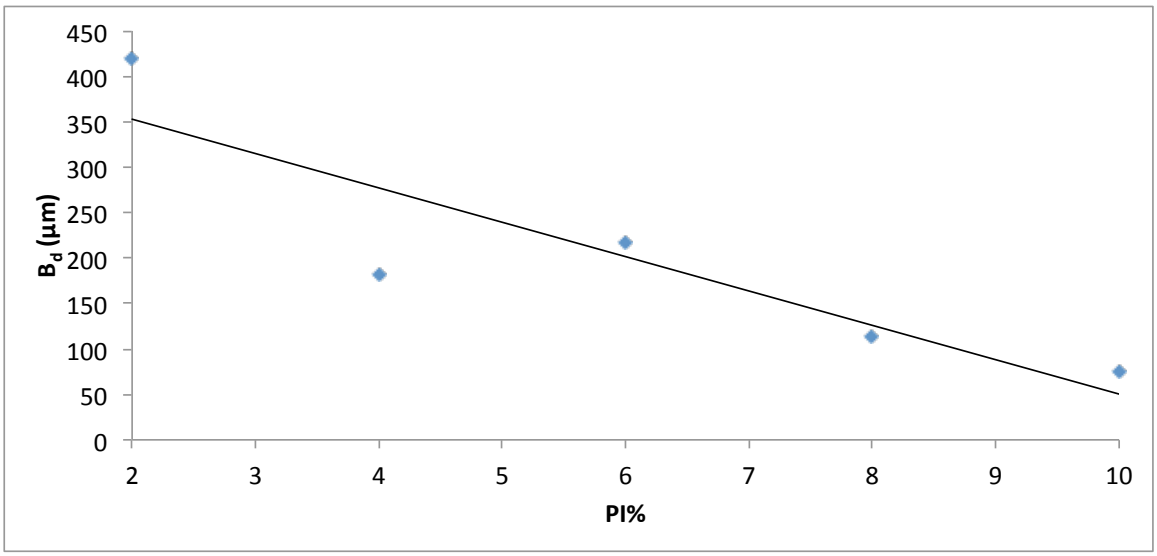


Figure 94: Variation of B_d with respect to photoinitiator concentration.

Similar to the case of UV absorber concentration, while the Tomeckova models predict only depth curing parameters, similar qualitative trends are obtained for width curing parameters D_p^w and E_c^w as well. These results are shown in Figures 92 and 93. From the depth and width curing parameters, the broadening depths at various PI concentrations are calculated and plotted in Figure 94. It can be seen that B_d decreases with increasing PI concentration which is intuitive as higher PI leads to more polymerization initiation sites which in turn leads to higher lateral curing.

3.4.8 Future Scope

The experimental studies presented here shed many qualitative insights into the cure width behavior of the LAMP suspension with respect to some important parameters and is a first step towards building a model that will help in compensating the build images to get the most accurate parts. However, much work needs to be done before these compensation factors can be robustly determined. A few of them are mentioned here as scope for potential future investigations:

(a) The experimental studies thus far have been conducted on single layers. It was observed that multilayer parts have slightly higher lateral curing than single layers due to the print-through caused by multiple exposures. Experimental investigations need to be conducted to characterize how the width curing factors scale for multi-layer builds.

(b) Slight variations between the side curing amounts in the different directions were observed, thereby indicating the possibility of anisotropic side scatter or shrinkage. Investigations to study the anisotropic nature of shrinkage or side curing need to be conducted.

(c) The cure widths measured in this study only correspond to the top surface of the cured layer. However, it was observed that this cure width is not constant across the thickness of the cured layer suggesting the possibility that layers obtained are not strictly 2.5-dimensional. This fact calls for investigations of 3D cure profiles obtained for various parameters.

(d) Lastly, the results presented here only consider the first order effects of each of the parameters in consideration, i.e, the effect of a parameter as it is varied while keeping all others constant. The second and higher order effects of these parameters, i.e, the effects on side curing behavior as two or more factors are changed simultaneously are unknown and require further study.

3.5 Summary

Several computational schemes implemented for improving the quality of parts produced through LAMP have been presented. An novel volume-deviation based method

for adaptively slicing CAD models has been implemented in order to alleviate the problem of stair-stepping in LAMP. Prior approaches reported in the literature for adaptively slicing CAD parts are either too computationally intensive or too simplistic to be useful. The approach presented here is computationally not too taxing while still producing reasonably good results.

A gray scaling and dithering approach was implemented to alleviate the stair-stepping effect manifested in surfaces with downward pointing (w.r.t build direction) normal vectors. The approach implemented here, takes into consideration the effect of gray scale on the curing characteristics of LAMP suspension. Prior gray scaling approaches presented in the literature either ignored or have not discussed these effects.

Finally, an experimental study to characterize the side curing characteristics of LAMP suspension with the ultimate objective of pre-build image compensation to achieve dimensionally accurate parts was presented. The side curing behavior with respect for parameters namely feature size, light intensity, photoinitiator and uv absorber concentrations was characterized.

CHAPTER IV

INTERNAL SUPPORT STRUCTURES

4.1 Introduction

Various data processing schemes required to process the CAD data for successful part builds are proposed in Chapter 2. Computational schemes that improve the part quality over conventional layered manufacturing techniques are proposed in the Chapter 3. This chapter discusses some of the pre-processing operations, that need to be performed on the CAD source data before it is subjected through the data processing flow discussed in the previous chapters in order to ensure a successful build. More specifically, the pre-processing operations required to tackle the problem of providing support to unsupported (also colloquially referred to as "floating" islands) features in layered manufacturing is presented. A basic overview of the need for support structures and the concept of unsupported features is given next.

4.1.1 Need for Support Structures

Supports structures are needed in additive manufacturing for a variety of reasons. Figure 95 shows one scenario in which supports are needed. Surfaces that slope away at a greater angle to build direction produce more overhangs with each successive layer than surfaces that slope away at a smaller angle. This phenomenon is illustrated in Figure 96 for surfaces that slope away beyond a certain critical tilt angle, support structures are required. Hence, in the scenario illustrated in Figure 95, Face B requires supports whereas Face A can be built successfully without any.

Yet another scenario that requires supports is illustrated in Figure 97 wherein a coffee mug is built in a upright orientation. When building parts in a layer-by-layer manner, depending on the part geometry and build orientation, new features start

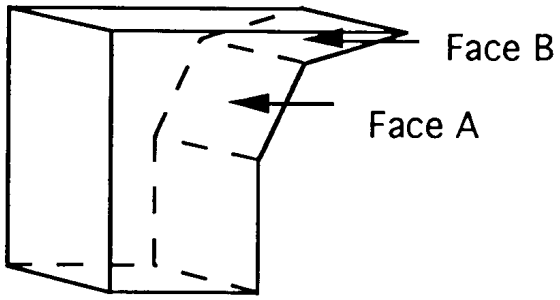


Figure 95: Scenario needing support structures for steep surfaces. Face B needs support but not Face A [15].

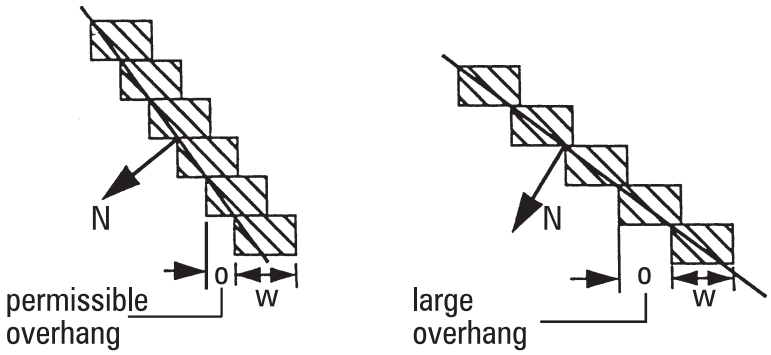


Figure 96: Overhangs for shallow and steep surfaces [16].

growing from an arbitrary height above the base of the build platform. In the coffee mug example illustrated in Figure 97, the base of the mug is connected to the build platform while the handle starts growing from some arbitrary height. Features such as these result in unsupported islands in the build layers (will be illustrated with slice images later) that need to be supported or else they will be swept away during re-coating of subsequent layers, causing the build to fail.

The final scenario that typically occurs in layered manufacturing necessitating the use of support structures is illustrated in Figure 98. If the part geometry being built in a specific direction is such that the center of gravity shifts outside of the base as more layers are built, there is a possibility for a moment to develop that ultimately

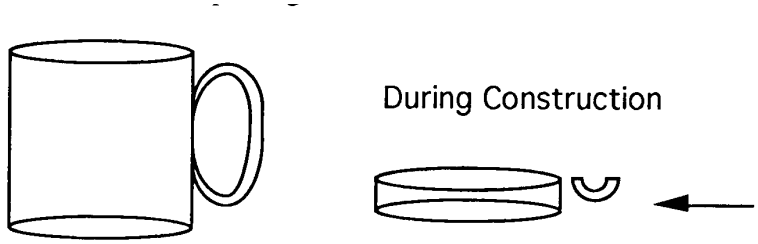


Figure 97: Scenario illustrating features growing from an arbitrary height leading to floating Islands [15].

causes the part to topple over. The part illustrated in Figure 98 poses such a problem. Although, the need for such supports is slightly weaker than the others as the base of the part is attached to the build platform, such scenarios do present themselves quite frequently.

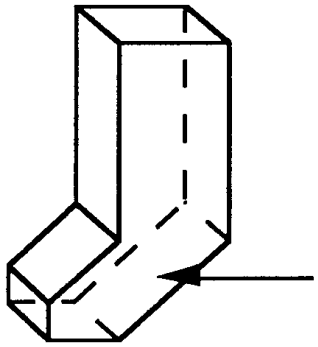


Figure 98: Scenario needing support structures to prevent parts from toppling over [15].

These scenarios pose varying degrees of problems in the different layered manufacturing techniques. In selective laser sintering (SLS) for example, the part being built is always surrounded by a bed of unsintered powder and hence all these previous scenarios are easily handled without the need for any additional support structure geometry. In fused deposition modeling (FDM) on the other extreme, the material is only deposited line-by-line (called "roads") only in the regions corresponding the part geometry and the rest of the build volume is empty. Hence, for FDM, additional

support geometry is imperative for each of the scenarios illustrated above. Stereolithography (SLA) lies somewhere in the middle of the spectrum between the two extremes of SLS and FDM where the build volume is filled with viscous photocurable resin. This offers a reasonable amount of inherent support in some of these scenarios. For example, in the third scenario, even if the center of gravity of the part is slightly outside the area of the base, the buoyancy forces offered by the viscous resin might be enough to support the part without any additional support geometry. The other two scenarios of overhang and floating islands still require supports for successfully building parts using SLA.

In addition to the scenarios requiring supports presented thus far, supports structures are also required to support the base of the part. If supports are not added to the base, the first few layer end up sticking to the build platform, which makes it difficult to retrieve the parts after the completion of the build without damaging them. In most layered manufacturing techniques, the support geometry required for successful part builds are grown from the base of the build platform along with the actual part. An illustration of the supports built in this manner for each of the scenarios presented in Figures 95, 97 and 98 is shown in Figure 99

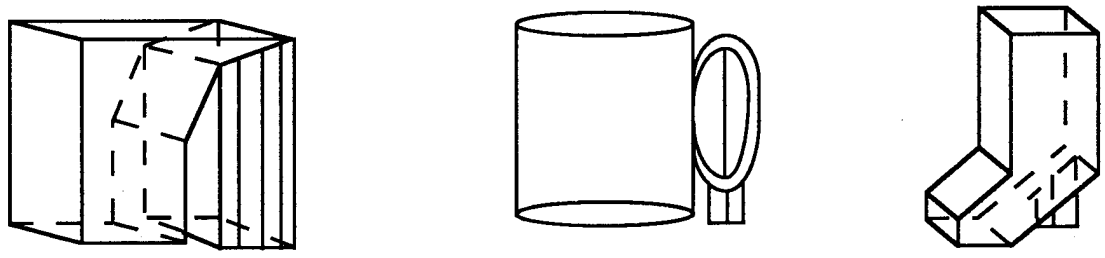


Figure 99: Supports grown from the base for various scenarios [15] illustrated in Figures 95, 97 and 98.

Supports built in this manner are removed from the part after the build is completed. However, in scenarios where the geometry requiring supports is internal to the part, removing these support structures is extremely challenging. Figure 100

illustrates the differences in internal vs external supports.

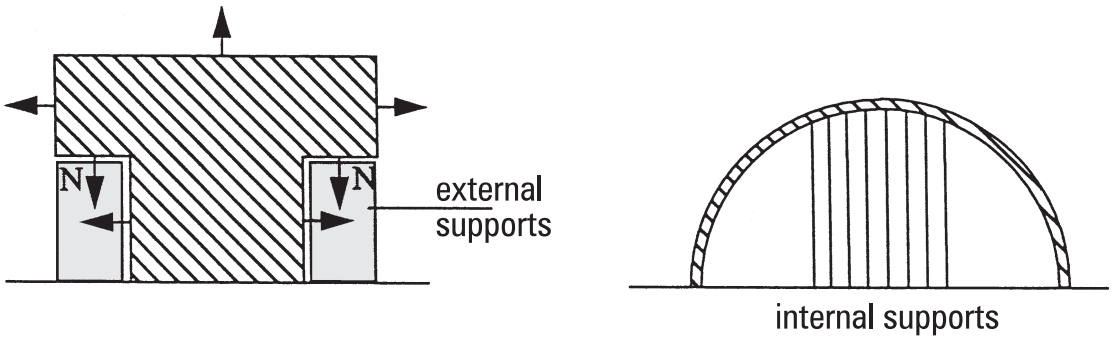


Figure 100: External supports lie exterior to the part and can be easily removed post build while internal supports are trapped inside making it very challenging to remove them [16].

In addition to posing challenges in terms of removal from internal geometries, when removed, supports also adversely affect the surface finish of the parts in the regions where they were attached. Moreover, supports lead to longer processing times (due to the need for processing additional geometry), longer build times (due to the need for depositing or curing extra material in each layer to build these supports) and excess material usage. Hence, prior to processing the CAD geometry for support structures, the build orientation of parts needs to be selected carefully to minimize the number of features requiring supports. A review of the work previously reported in the literature that is aimed at identifying and adding these support structures in CAD models is presented next.

4.1.2 Literature Review

A few different approaches for generating and identifying unsupported geometry and generating support structures were reported in the literature. These methods can be broadly categorized based on the input data they work with, i.e, STL files, CAD models or slice contours. A brief review of some of these approaches is presented in this section.

Kirschman et. al. [17] were one of the first to address the issue of generating supports for additive manufacturing of parts. They presented an approach to algorithmically generate these support structures by analyzing the facets in an input STL file. Experimental work was presented in order to identify the maximum sloping angle of surfaces and the maximum overhang length that can be built without supports in their SLA system. The hatch spacing required to modulate the strength of these supports was also reported. Using these parameters as inputs to the algorithm, facets in the given STL were analyzed for their need for supports. A choice of several different support structure geometries was presented, as shown in Figure 101. As can be seen, various geometries such as outline supports (that only support the

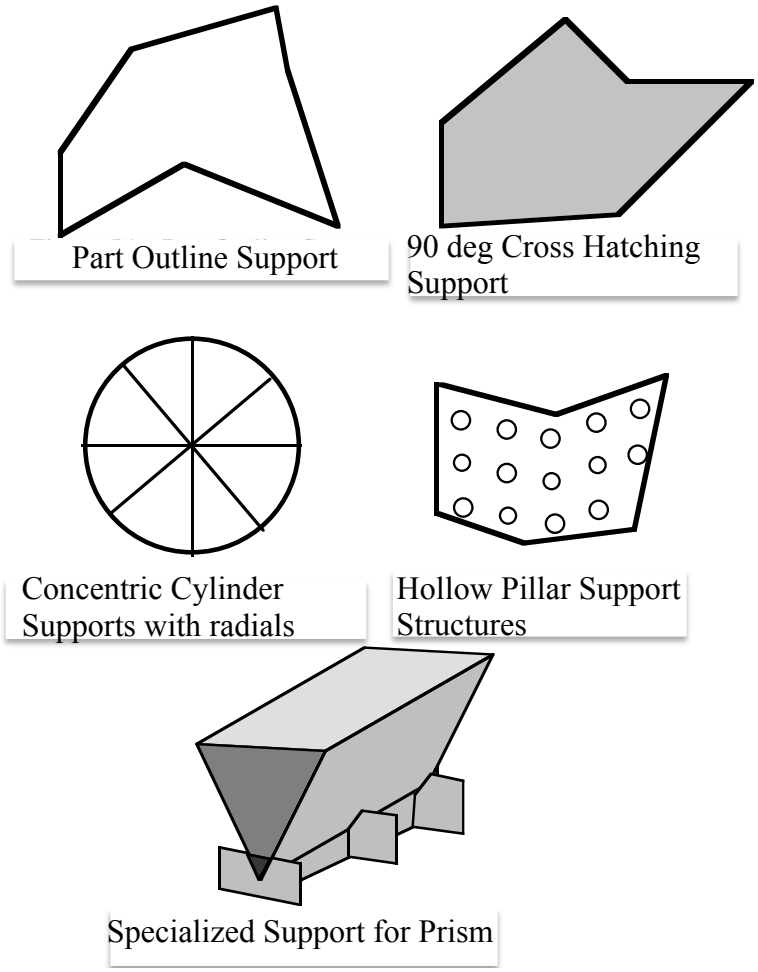


Figure 101: Types of supports to choose from as proposed by Kirschman et al. [17].

outline of the unsupported region), solid supports (that support the entire region) with 90^o cross hatching, concentric cylinders (either hollow or with internal radials) and specialized prism supports are presented. For every facet in the STL file that needs supports, surrounding facet data was examined to understand geometry of the surface defined by the facets. Based on the identified surface topology, the algorithm choses a support geometry that is best suited.

Webb et. al. [128] proposed yet another approach where they first read the STL file and identified its topology information (edge connectivity and adjacency information of the mesh) to generate a wire-frame view of the part thereby, allowing users the flexibility of selecting individual surfaces that require supports. Once the surfaces are identified, they implemented algorithms to determine the borders of these surfaces, offset them, calculate intersections between lines and then project these lines in the XY plane onto the surface to generate supports. By using this routine, every facet that is either identified by the user or by their general rules-based algorithm is supported with the right type of support structure. Once all the facets' supports have been generated, the file is sent for the data processing steps like slicing and path generation etc.

Swaelens et. al. [18] proposed yet another strategy for identifying and adding supports to unsupported features in STL files. Similar to Kirschman et al. [17], they proposed several different support geometries that provided better supports to the part surfaces while facilitating easier removal (and consequently better surface finish) and faster build times. Figures 102 - 106 illustrate these support geometries.

The block support, which is a combination of orthogonal walls, is used when the strongest support is needed over large overhangs. For narrower surfaces, a block support will be too strong leading to some damage when removed and hence a line support is used which has a center line and some perpendicular walls. By the same analogy, for even smaller surfaces, the point support is used which are a series of radial

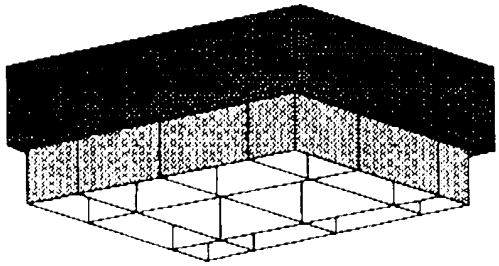


Figure 102: Block support under a part [18].

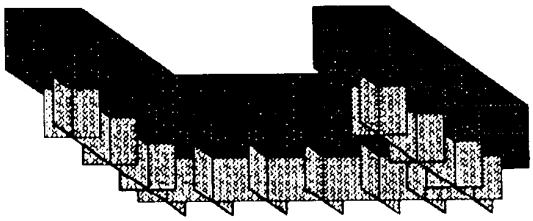


Figure 103: Line support under a narrow surface [18].

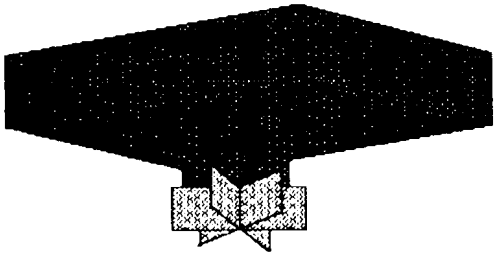


Figure 104: Point support under a narrow surface [18].

walls emanating from the point to be supported. A gusset support was introduced for supporting small and medium sized overhangs which do not need to be supported all the way to the bottom of the build platform or to the underlying structure. For easy removal of supports, tooth profiles as shown in Figure 106 are introduced at the point of contact of these supports with the part. Apart from these, some novel "perforated" support structures similar to the scaffolding used in the construction industry were discussed. It was claimed that these reduce the material usage and

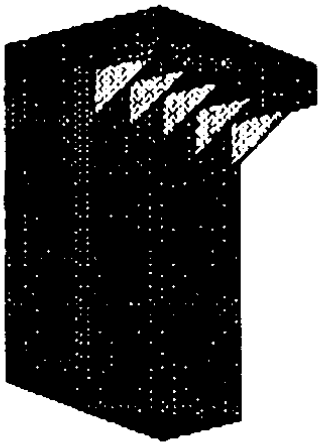


Figure 105: Gusset support under a small overhang [18].

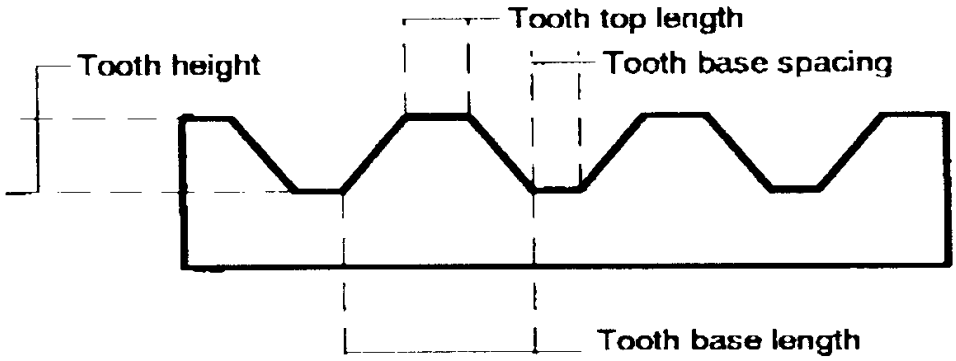


Figure 106: Tooth profile at the point of contact of support structures and the part for easier removal [18].

building time much further. Having introduced these novel geometries, an automatic support algorithm was presented. The facets from the STL file were read and grouped into supported and unsupported categories based on a critical slope criterion. For each identified unsupported facet, the neighboring facets were studied in order to determine the surface topology. Two parameters known as *overhang* and *slenderness* were then defined and used to identify the suitability of each of these various support geometries for the specific surface topology identified. Moreover, an algorithm that incorporated support features from the slice data (not from an STL file as before) was

presented. This algorithm compared successive layers to calculate the local steepness of the surface, and wherever the surface was not steep enough, supports were added.

Chalasani et. al. [19] have proposed an algorithm (tailored to the FDM process) to identify and build supports for unsupported features based on the slice data. It was proposed that this was advantageous in reducing the processing time as the slicing operation now does not have to process all the additional support geometry. Three different strategies for generating supports were presented which are:

(a) Containment - enclosing the entire part in a shape-following container.

(b) Region - supporting a specified region.

(c) Direct - automatic generation of optimal supports.

In the containment approach, a sub-optimal (i.e, generates more supports than required) but extremely quick strategy is used where first the large composite curve that encloses all of the slice data is identified. This all-enclosing composite curve is then offset and stacked repeatedly over the entire length of the part. All the regions in this composite structure that do not correspond to the part are identified as supports. The support structures computed for the slice data of a bone using such a containment approach is shown in Figure 107.

The region-based approach is a semi-automatic one where the user is allowed to choose the curves from the slice data requiring support. These user-selected curves are then copied down to each layer, constantly being trimmed if they intersect with the part area until they either reach the bottom of the build or get totally eliminated by recurring trimming due to part geometry.

The automatic support generation approach computes the difference between successive layers n and $n-1$ which was referred to as a *shadow*. If this *shadow* is less than the road width parameter used in FDM, then layer n is said to be self-supporting with respect to layer $n-1$. Starting with the top layer, the shadow of every layer on the adjacent layer below it is computed in order to compute the *stale shadow* at every

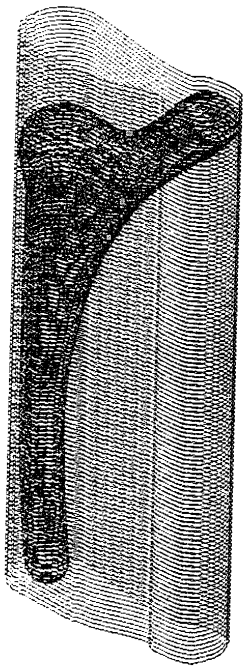


Figure 107: Creating support structures for the slice data of a bone using containment approach [19].

layer. This resulted in a *composite shadow* that is all the support required for the part. *Stale shadow* is defined as a shadow that is unchanged for at least two layers while a *fresh shadow* denotes the new shadow generated on the current level by the previous layer above it. The stale shadow forms the bulk of the support, and can be built in either the primary material or the alternate material used for supports in FDM. By accumulating only the stale shadow, and keeping the fresh shadow unmerged at the interface of the part and the composite stale shadow, a fresh shadow was built in the alternate material if desired. Figures 108a and 108b show supports built using this algorithm for two different road length parameters of 0.01 inches and 0.08 inches respectively. Otto [129] proposed a similar slice data-based approach to generate supports.

Allen and Dutta [15] proposed an approach to generate supports from input CAD model unlike the previous approaches where either STL files or slice data were used. The CAD model was first tessellated to obtain a sample of points representing the

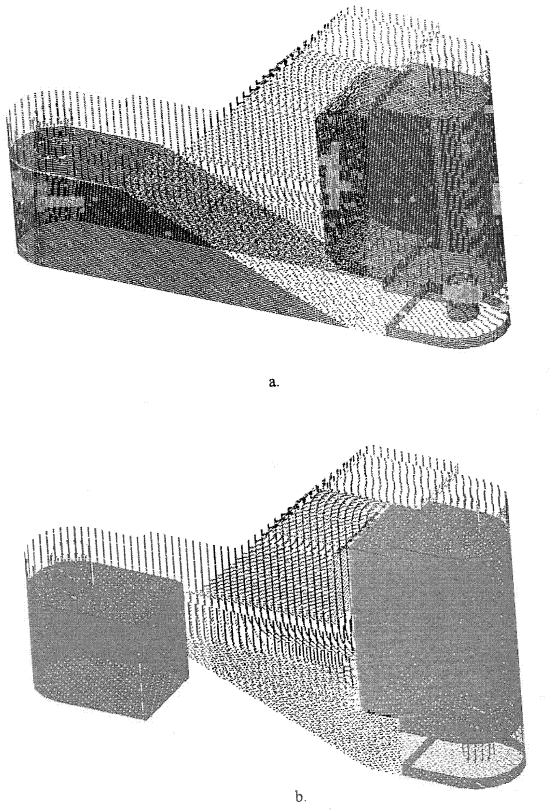


Figure 108: Supports built using automatic support generation algorithm [19] for two different road length parameters: (a) 0.01 inches (b) 0.08 inches.

geometry. The growth of the part in a layered manufacturing process was then simulated. As each layer of the part growth was simulated, the center of gravity of the partially built part was computed to identify if any points from the sample set need supports. Also, if any new structures start growing from an arbitrary height (which result in floating islands), the sample points corresponding to those structures are also identified as needing supports. Additionally, the slope of the facets corresponding to these sample points were also observed. If the slope is not steeper than a user specified threshold, the surfaces corresponding to these facets produce large overhangs. Therefore, the sample points associated with these surfaces are also marked for the need for supports. In this manner, from the sample space of points collected on the surface of the CAD model, all the points that require supports are identified.

Once these points are identified, approximate support structures were constructed by shooting rays originating from a projection of these points on the build platform until they intersect the part at the part surface corresponding to the points. Once these approximate support structures are constructed, the total contact area between these supports and the part is computed and used as criterion to be minimized in order to identify the best part orientation for build. Figure 109 shows a few stable orientations found for the CAD model of a coffee cup and the corresponding support structures computed for each of these orientations using this approach.

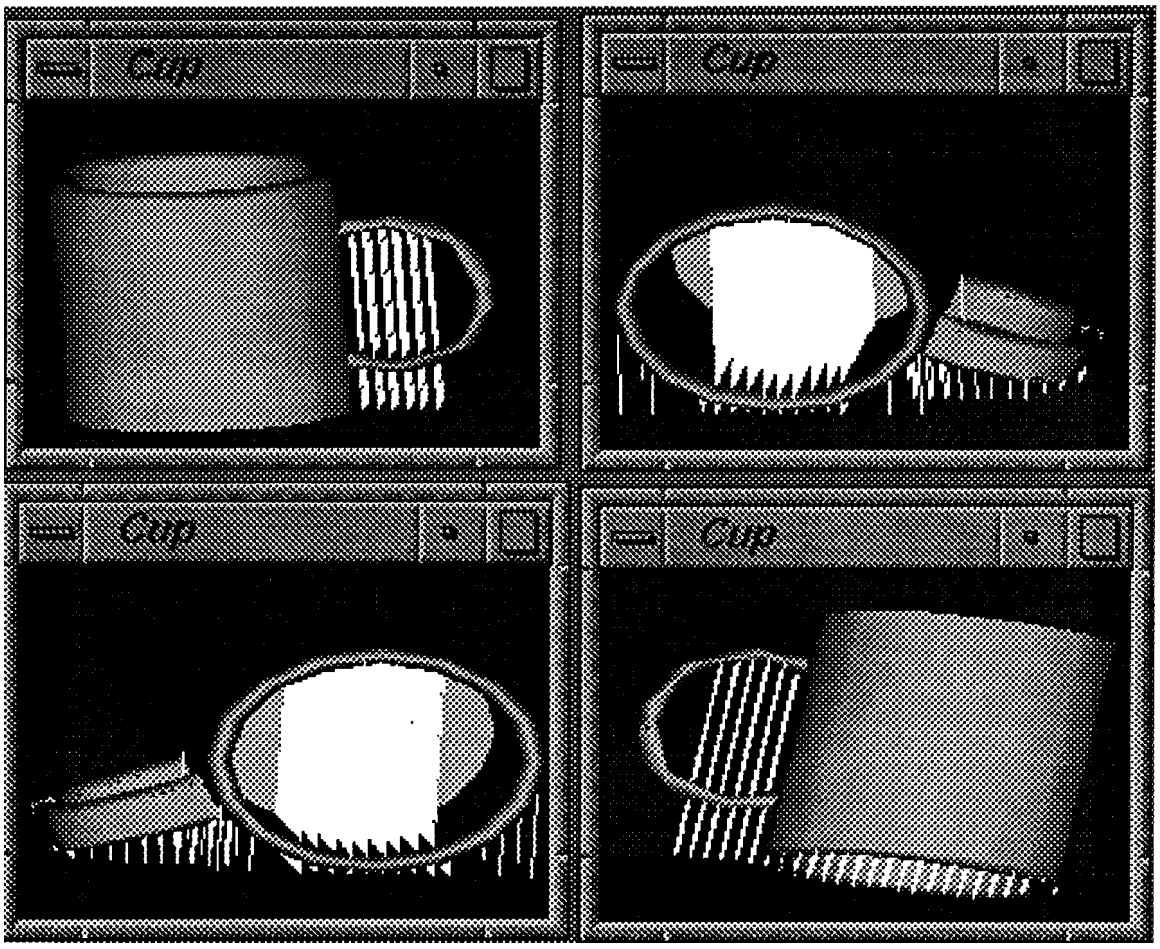


Figure 109: Four stable orientations for a coffee cup and the support structures computed for each according to the algorithm proposed by Allen and Dutta [15].

4.2 Support Structure needs specific to LAMP

Three scenarios were discussed in Section 4.1.1 that present the need for support structures in typical additive manufacturing processes. These are re-listed here for the sake of clarity:

(a) Surfaces with large overhang.

(b) Surfaces or geometries that result in floating Islands.

(c) Geometries with the potential to topple over.

As previously mentioned, the degree to which each of these scenarios necessitate supports changes with each additive manufacturing process. For example, selective laser sintering (SLS) does not need any special supports for either of the scenarios since there is always a bed of unsintered powder acting as support. Fused deposition modeling (FDM) on the other hand, requires supports for all of these scenarios since material is only deposited in the region enclosed by the part geometry and the rest of the build volume is empty unlike in the case of SLS. For the stereolithography (SLA) process, the need for supports lies somewhere in between the spectrum of these two extremes. Since the build volume in SLA consists of a viscous resin, in some instances (based on the part geometry), the buoyancy force offered by the viscous medium suffices to support the parts from toppling over, thereby eliminating the need for supports in this scenario.

From the previous discussion it is clear that, although some common scenarios that necessitate support structures exist, the degree to which they impose the need for supports varies with respect to the additive manufacturing process in consideration. The need for supports with respect to each of these scenarios specific to the LAMP process is discussed in this section. The LAMP process, as previously discussed, aims to build ceramic molds for the casting of high-pressure turbine blades. Hence the need for supports in the LAMP process specific to the needs of the geometries that arise in HP turbine blade molds is considered. Figure 110 illustrates the features

found in a representative HP turbine blade mold.

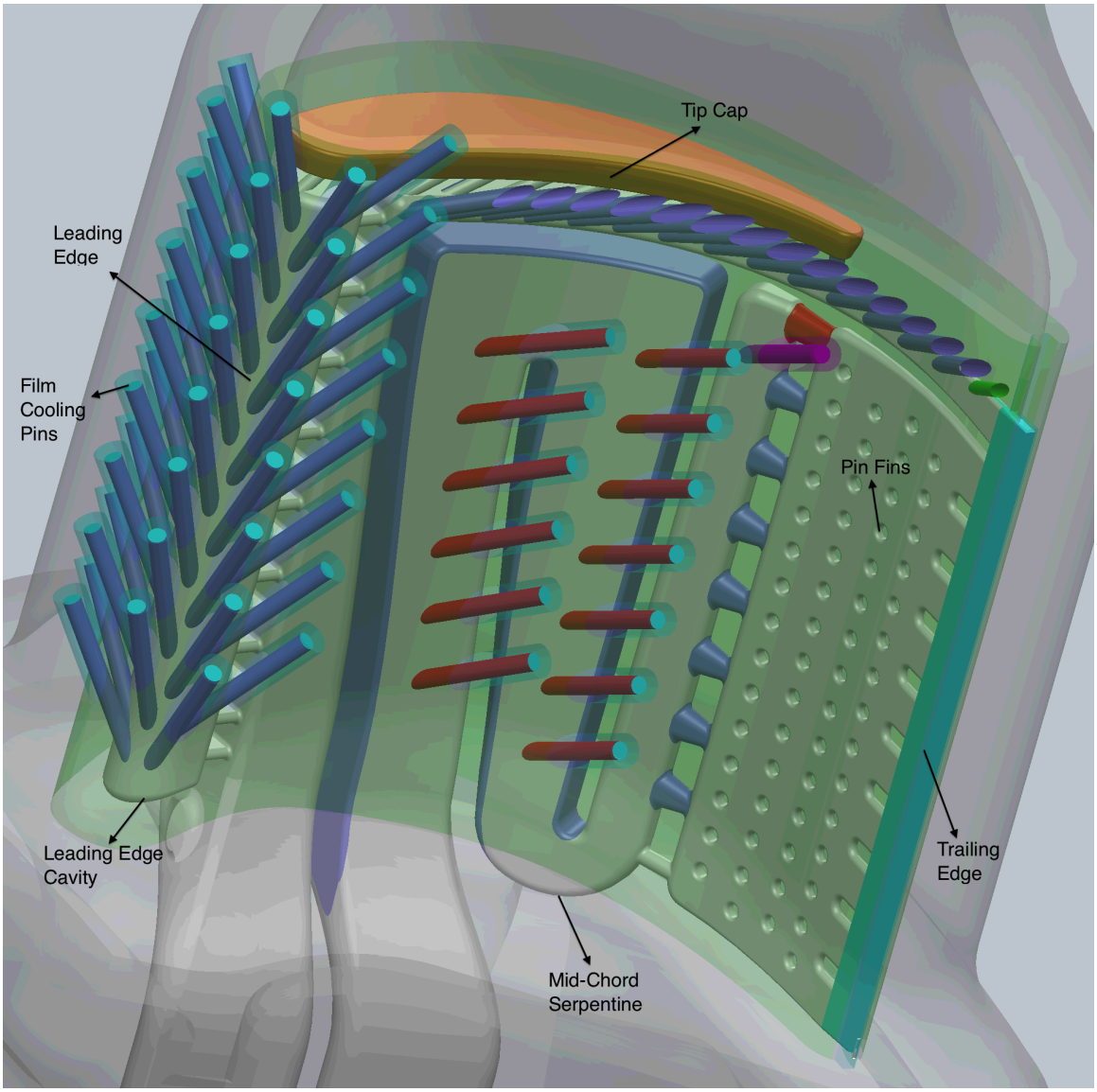


Figure 110: Features of a typical high pressure turbine blade mold.

All the geometric features observed in the figure are intended for the cooling of internal and external surfaces of the blade. Some prominent features like the leading edge, trailing edge, film cooling pins (cool the leading edge and mid chord portion of the external surface of the blade), mid-chord serpentine (creates a serpentine passage for internal air flow in the mid chord region), leading edge cavity (supplies cooling air to the leading edge film cooling pins), tip cap (creates an air cavity for cooling

the top edge of the blade), pin fins (cool the narrow cross-section of the trailing edge) are annotated. While building such a complex geometry, each of the three scenarios demanding support structures are encountered and the means by which they can be handled in LAMP is presented next.

4.2.1 Overhang

While building the complex geometry of the HP turbine blade molds, several overhangs do occur. Figure 111 shows the native orientation of the blade mold. When built in this orientation, features that define the tip cap of the blade cause very large overhangs leading to a failure of the build. However, from previous experience, for the typical HP blade geometries shown in Figure 110, an orientation can be found which minimizes these overhangs thereby resulting in successful builds. Such an orientation is shown in Figure 112 which enables the tip cap features which were previously causing build failure to grow more gradually from their root at the trailing edge.

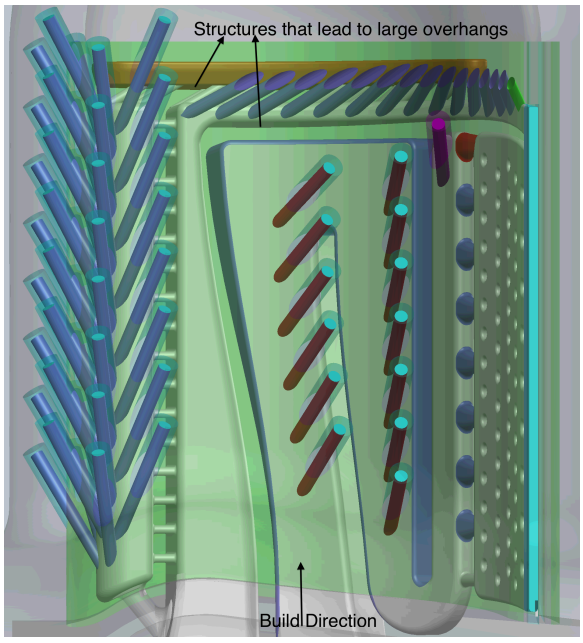


Figure 111: Building the mold in the original orientation standing up will lead to large overhangs.

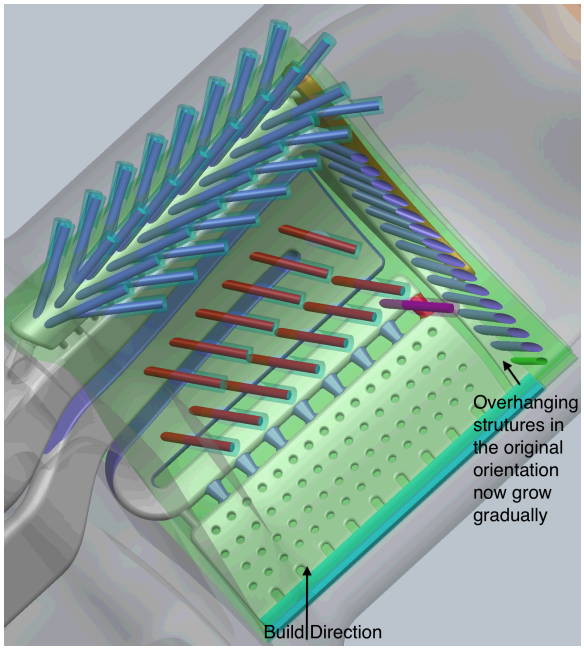


Figure 112: A build orientation that minimizes the overhangs observed in the original orientation.

Thus, for the kind of geometries encountered in the LAMP process, the problem of adding supports to overhanging structures is not so crucial and a build orientation can be found that will result in tolerable overhangs that will lead to successful part builds. This is ideal because any additional support structure added would be totally internal to the built part and impossible to remove. This leads to additional unintended features in the cast blades which will alter the cooling characteristics of the intended design. However, if in the future, a blade geometry is encountered which does not have any viable orientation that yields tolerable overhangs, then this support scenario needs to be addressed.

4.2.2 Floating Islands

The issue of floating islands while building complex geometries like turbine molds using the LAMP process is a significant one. In most simple geometries, an orientation can be found which does not result in any floating islands. However, for the parts with the kind of complexity shown in Figure 110, it is not typically possible to find

any occurrence that will eliminate the formation of floating islands. Even in the orientation that was shown in Figure 112 which minimizes overhanging structures, floating islands form when the bases of the mid-chord serpentine and the leading edge cavity begin to form. Figure 113 shows the cross-section of the part as the base of the leading edge cavity is being built. The feature highlighted by the box clearly does not have any previously built feature supporting it.

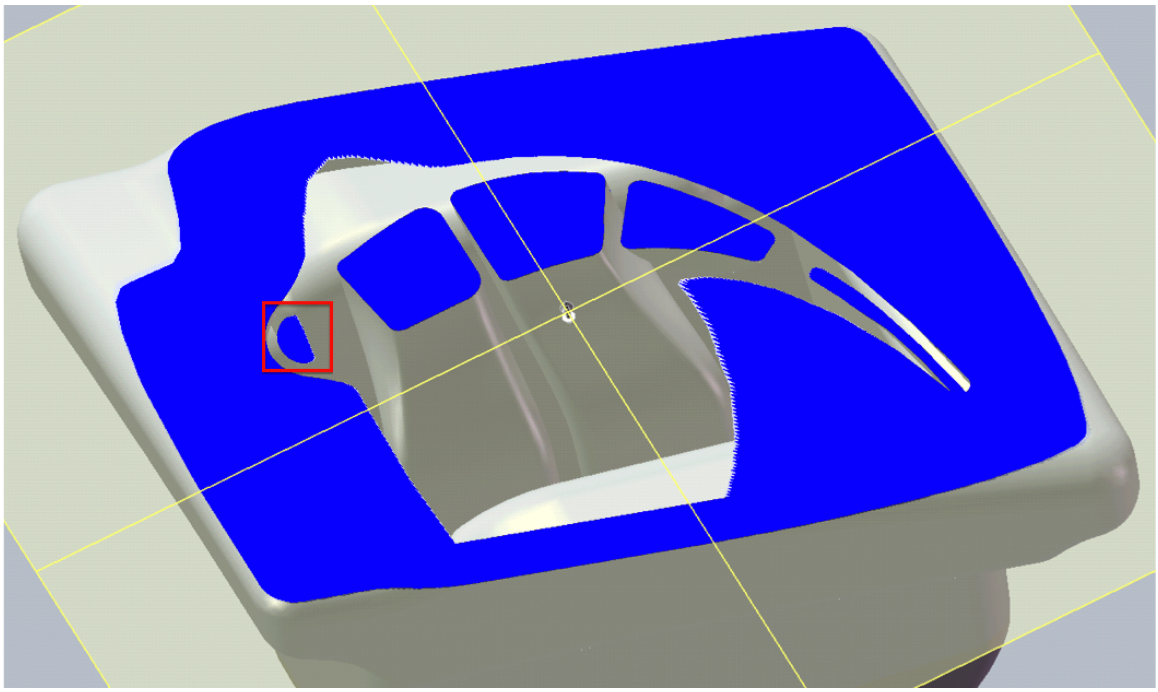


Figure 113: Floating Island formed during part builds.

When such unsupported features are cured, since there is no feature beneath it to adhere to, the shrinkage stresses have a greater effect on the feature and it curls up. Also, the re-coating process imparts significant shear stresses on the layers as it sweeps through the build area. As the blade sweeps over with a layer of viscous ceramic loaded suspension underneath it, a boundary layer is formed which imparts drag forces on the platform. The suspension was observed to be non-Newtonian but ignoring this fact, a rough estimate of the shear forces can be made. The viscosity of the slurry in the velocity ranges of the re-coating process was measured to be

in the 400-450 centi poise range. The re-coating blade travels a distance of 26 cm in 6 seconds and is at height of 200 μm above the build platform. At this speed and viscosity, assuming Newtonian behavior and a linear velocity profile, the shear stresses imparted on the part will be of the order of $\sim$100 Pa. Due to such high shear forces and the curling up effect of unsupported features due to shrinkage stresses, any unsupported features formed will be swept away by the re-coating arm causing the build to fail.

Hence support structures are necessary for any geometries that produce floating islands during a build, in order to obtain successful parts. However, as discussed previously, all of the geometric features, supports or otherwise are enclosed within the outer shell of an integrally-cored mold and removing these supports post-build is impossible. This results in additional unintended features in the cast blades which might adversely impact the designed cooling performance of the molds. This issue of floating islands is probably the only limitation potentially preventing the LAMP process from building blade designs of any arbitrary complexity.

However, it must be noted that the current blades are designed for manufacture through conventional investment casting process capabilities and constraints. As the LAMP process enters into full production and the castability of the parts produced through the process is successfully demonstrated, there is immense scope for advanced mold design specific to this technology. The work presented in this thesis is a first step towards this goal of design for manufacturing specific to the needs of the LAMP process. Details on how these floating islands are identified and supported in CAD parts and a simple performance analysis of a representative support structure in order to satisfy the dual objectives of successful part builds as well as good cooling performance is given in Section 4.5 of this chapter.

4.2.3 Supports to Prevent Toppling

The final scenario that requires supports is the case of part toppling over due to its own weight as the part is being built. In the case of the LAMP process, requirements of supports of this kind is quite weak. The parts are attached to the build platform quite rigidly with the help of a mesh structure. Moreover, as discussed in the tiling and part placement section in Chapter 2 (Section 2.4.2), the parts are built with a conformal scaffold surrounding them. Therefore, as the parts are built, there is scaffolding all around in the build volume thus eliminating the possibility of the parts toppling over due to gravitational moments.

Thus, in retrospect, of all the scenarios requiring support structures, only the scenario resulting in floating islands poses a serious threat for part failures in the case of LAMP process and therefore needs to be addressed further. Details on the methodology followed for algorithmically identifying the geometries that result in floating islands from input CAD models and the methodology for creating support structures are given next.

4.3 Methodology for Identifying Floating Islands

In order to optimally position support structures, first there needs to be a method of algorithmically identifying the geometries which result in these floating islands during a part build. The details of such an algorithm developed for the purpose of identifying floating islands for a part being built in a given direction is given in this section. ACIS kernel was again used for implementing the algorithm and hence it works directly on CAD models. The pseudo code for this procedure is shown in Algorithm 16.

Algorithm 16 Identifying Floating Islands.

1: $wig \leftarrow$ the given CAD part
2: $layerThickness \leftarrow$ layer thickness used in the build
3: $(X_{min}, Y_{min}, Z_{min}) \leftarrow$ compute minimum extents of the part
4: $(X_{max}, Y_{max}, Z_{max}) \leftarrow$ compute maximum extents of the part
5: **for** each Z location along the height of the part **do**
6: **//compute slice at height** Z
7: $(X_{bmin}, Y_{bmin}, Z_{bmin})_1 \leftarrow (X_{min}, Y_{min}, Z - layerThickness)$
8: $(X_{bmax}, Y_{bmax}, Z_{bmax})_1 \leftarrow (X_{max}, Y_{max}, Z)$
9: $block_1 \leftarrow$ a cuboid created with extents specified by $(X_{bmin}, Y_{bmin}, Z_{bmin})_1$ & $(X_{bmax}, Y_{bmax}, Z_{bmax})_1$
10: $3DSlice_1 \leftarrow$ geometry obtained form the intersection of wig and $block1$
11: **//compute slice at height** $Z + layerThickness$
12: $(X_{bmin}, Y_{bmin}, Z_{bmin})_2 \leftarrow (X_{min}, Y_{min}, Z)$
13: $(X_{bmax}, Y_{bmax}, Z_{bmax})_2 \leftarrow (X_{max}, Y_{max}, Z + layerThickness)$
14: $block_2 \leftarrow$ a cuboid created with extents specified by $(X_{bmin}, Y_{bmin}, Z_{bmin})_2$ & $(X_{bmax}, Y_{bmax}, Z_{bmax})_2$
15: $3DSlice_2 \leftarrow$ geometry obtained form the intersection of wig and $block1$
16: **//check for floating islands in** $3DSlice_2$
17: **for** each disconnected $lump\ i$ in $3DSlice_2$ **do**
18: **for** each disconnected $lump\ j$ in $3Dslice_1$ **do**
19: check for intersection between $lumps\ i$ & j
20: **end for**
21: **if** no intersections found **then**
22: lump i in $3DSlice2$ is a floating island
23: **end if**
24: **end for**
25: **end for**

The given part is first loaded into the algorithm and its minimum and maximum extents are computed. At each Z location along the height of the part, a cuboid denoted by $block_1$ with a cross-sectional area equal to the cross-sectional area of the bounding box of the part and a thickness equal to layer thickness is created. A solid body intersection is computed between the part denoted by wig and $block_1$ to yield a three dimensional slice denoted by $3DSlice_1$ at height Z of the part. Similarly, a three dimensional slice denoted by $3DSlice_2$ at height $Z + layerThickness$ is computed. At the height corresponding to the floating island illustrated in Figure 113, the two slices computed using the previous steps are illustrated in Figure 114 (For illustrative purposes, slices separated by a few layers are shown and hence the apparent large overhangs. For successive slices, the overhang will be very small but so is the floating feature generated and hence it will be difficult to perceive).

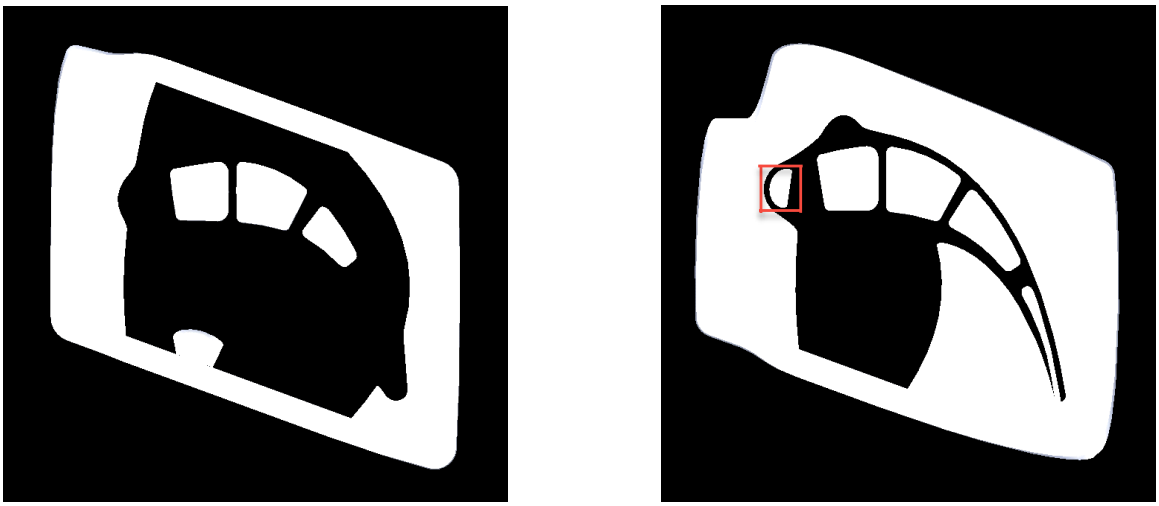


Figure 114: 3D slices of successive layers at the location corresponding to the floating island shown in Figure 113: (a) Slice at height Z (b) Slice at height $Z + layerThickness$ (floating island is marked by the red box).

Although, in the Figure, the slices look two dimensional because of very thin layer thickness (100 μm), they are in fact three dimensional because of the way they were

created. Each of the disconnected solid regions in these slices are stored as a lump in the ACIS data structure. In order to determine if there are any floating islands in the second slice, each of the lumps of this slice is checked for intersection with each of the lumps of the first slice. If there exists a lump in the second slice, which does not intersect with any of the lumps in the first slice, it is classified as a floating island and support support structures need to be created. By repeating this sequence of operations for every successive pair of slices along the length of the part, all the features that lead to floating islands during a part build can be identified. This operation can be repeated for different orientations of the part, and the orientation that yields the minimum number of floating islands can be selected. The details of the algorithm to generate supports for the floating islands identified using this approach is presented next.

4.4 Methodology for Support Structure Generation

In most of the previous work presented, supports were generated only along the build direction which either extend all the way down to the base of build platform or until the next immediate geometric feature. This approach does not work for LAMP as all of the supports will be completely encapsulated by the shell and thus cannot be removed. Hence, creating long slender supports all the way to the base will completely disrupt the intended cooling designs. Instead, the approach presented here, tries to connect the identified floating island to the immediate surrounding geometry with minimal support length to have minimal impact on the intended design. Note that, the geometries that these support structures connect to need not be directly underneath the floating island along the build direction unlike the previous approaches presented. The pseudo code for such an approach is shown in Algorithm 17.

Algorithm 17 Support Structure Generation.

1: $origin \leftarrow$ centroid of the floating island that needs to be supported
2: $maxTiltAngle \leftarrow$ maximum tilt angle with which a support can be built
3: $maxCurvature \leftarrow$ maximum curvature allowed at the point of contact
4: $n \leftarrow$ the number of orientations user wants to output
5: $r \leftarrow$ nominal radius of the support
6: $candidateOrientationList \leftarrow$ discrete orientations within the "cone" underneath $origin$
7: **for** each orientation i in $candidateOrientationsList$ **do**
8: shoot a ray from $origin$ to the surrounding geometry of the part
9: check for smoothness of the surface at the point of intersection with the ray
10: $curvature \leftarrow$ local surface curvature at intersection point
11: **if** surface is smooth and $curvature < maxCurvature$ **then**
12: $support.direction \leftarrow$ orientation i
13: $support.length \leftarrow$ length of the ray in this direction
14: $list \leftarrow support$
15: **end if**
16: **end for**
17: sort the identified support orientations in $list$ w.r.t length in ascending order
18: **for** i:= 1 to n **do**
19: retrieve i^{th} orientation from the sorted orientations in $list$
20: sweep a circle of radius r from $origin$ along the ray in orientation i
21: output the part
22: **end for**

The algorithm takes in four parameters as input as follows:

(1) The centroid of the floating island at which supports need to be generated denoted by $origin$.

(2) The maximum tilt at which supports can be built denoted by $maxTiltAngle$.

(3) The number of potential orientations that the user wants to output denoted by n.

(4) The nominal radius of the supports as specified by the user denoted by r.

First, a list of possible candidate orientations along which a support can be built is created by discretizing the "cone" underneath *origin* with a vertex angle equal to $maxTiltAngle$. For each of these candidate orientations, rays originating from *origin* are generated and their intersections with the part geometry is computed. At each of these points of intersection, the smoothness of the surface is measured. If local surface curvature is high, there is a good possibility that it corresponds to a cooling feature and this orientation is abandoned. Likewise, if the intersection point is near to the boundary of two or more surfaces and the adjoining surfaces do not maintain continuity, this orientation is abandoned as well as it was observed from experience that shrinkage stresses accumulate at such corners and cause the supports or other slender structures to fail. After eliminating orientations that connect to non smooth surfaces in this manner, the rest of orientations are stored as potential directions for support propagation. The lengths of the rays originating from *origin* in each of these potential directions are computed and the list of potential orientations is then sorted in the ascending order w.r.t their lengths. For the first n orientations in the sorted list, cylindrical supports with radius r are created by sweeping a circular cross-section along the ray until it connects to the part.

Figure 115 shows the various supports generated on the sample HP blade shown in Figure 110 using this approach. As can be seen, of the various candidate orientations, many of the orientations are discarded because they either intersect the surface at regions of high curvature or are very long. The support structure highlighted in green is the one that is preferred as it is the shortest and also interesects the part at a low curvature region.

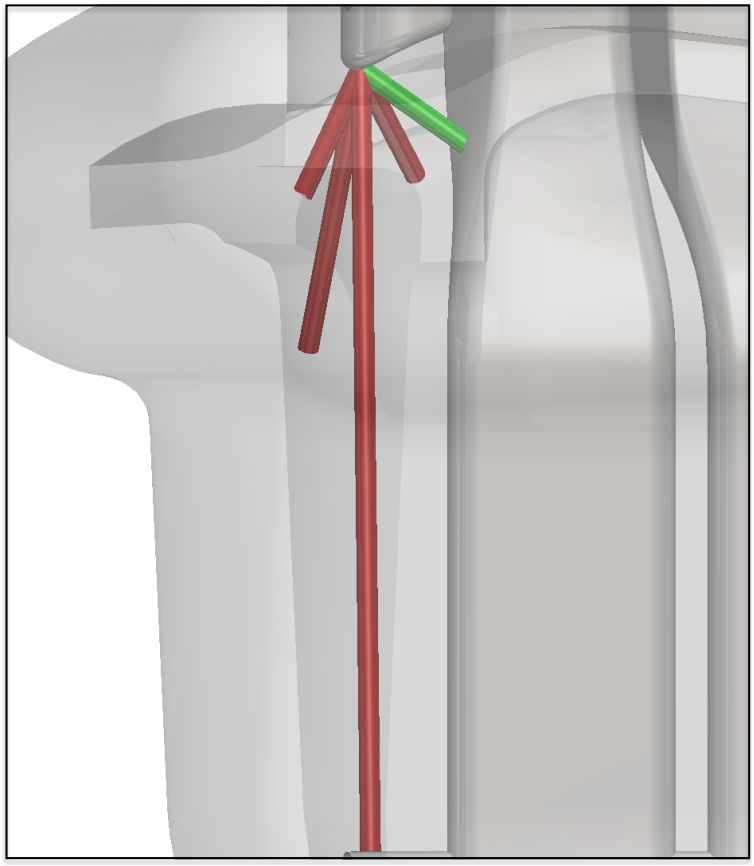


Figure 115: Supports generated by the algorithm.

In many of the instances experienced in parts of the form shown in Figure 110, the features that lead to floating islands grow continuously (i.e, grow from a small area floating region to a larger feature) and hence from past experience it is known that only one support of the type highlighted in green in Figure 115 is enough. However, when the features abruptly result in large floating islands, this algorithm can easily be extended for producing multiple supports. For supporting large floating features, an additional parameter that indicates the maximum amount of overhang a particular support feature of a given size can support needs to be incorporated in to the algorithm. Based on this parameter, the area of the large floating island can be sub divided into smaller regions and the procedure can be applied for each of the smaller regions with an additional constraint to produce non intersecting supports.

4.5 Potential for Multi-functional Design

As was previously noted, any support structures incorporated into the mold to yield successful LAMP builds are completely encapsulated within a conformal shell. Hence there is no way to remove these structures from the mold post build and this will in turn result in additional unintended features in the cast blades. Although this is a limitation of the LAMP process, it must be noted that the current blades were designed with the conventional manufacturing techniques in view and hence there is immense scope for design for manufacturing specific to the needs of the LAMP process. In order to demonstrate this, the performance analysis of the design resulted by the incorporation of a sample support structure like the one highlighted in Figure 115 is given in this section. Intuitively, the addition of a support feature like that will reroute some of the cooling air from the mid chord serpentine channel to the leading edge cavity. In doing so, it might not only deprive some of the upper impingement cooling holes but also stymie the lower most impingement jets at the leading edge. Also, because of the cooling air rerouting, there might not be enough mass flow left in the mid chord serpentine to cool all the other features like the tip cap etc. it supplies cooling air to. Hence in order to investigate this, the flow and thermal analysis of this support feature was performed. The details of these simulations and the results obtained will be discussed below.

In order to simply the meshing process and reduce computational times, the geometry was simplified to a representative but more amenable one. A constant representative heat flux boundary condition of $25 MW/m^2$ was applied to the leading edge wall. A constant velocity of 20 m/s was applied as the input flow condition at the base of the serpentine channel with a turbulence intensity of 5%. The simulation was carried out to solve the reynolds averaged navier stokes equations and the thermal equation for the each of the cases corresponding to the native and modified (with support added) geometry. Figure 116 shows the temperature profile obtained

on the internal wall of the leading edge with and without the support structure. It

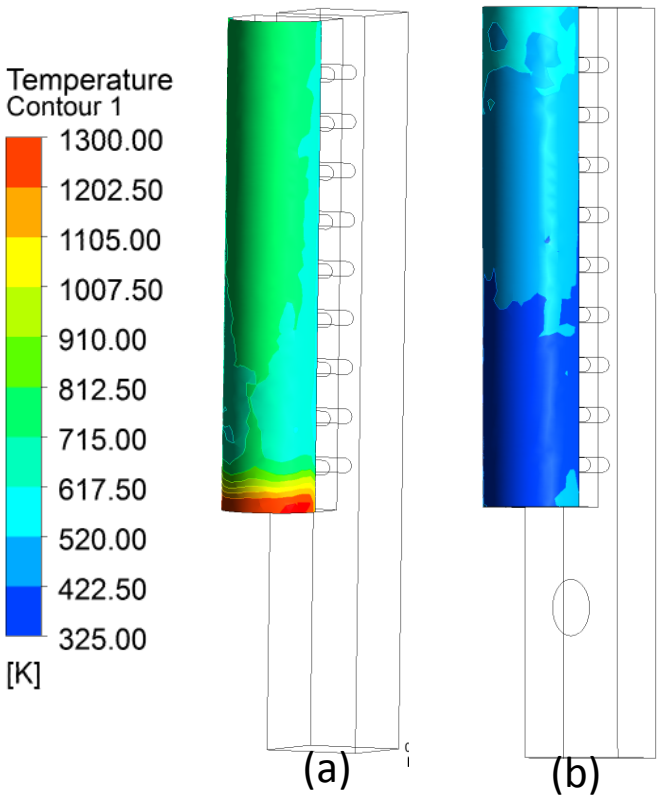


Figure 116: Temperature map of the internal leading edge wall: (a) without support and (b) with support.

can be clearly seen that, unlike what was expected, the temperature profile in the case of the support added is much lower than the case without a support. There is a hot spot at the lower region of the leading edge for the case without a support and the addition of a support reduces the peak temperature occurring on the leading edge wall significantly. Figure 117 shows the velocity streamlines of the flow to give a better understanding of temperature results obtained. It can be seen that in the case without the support, the flow streams from the lower most impingement jets are unable to reach bottom most region of the leading edge cavity wall which is not the case for the supported geometry. It can also be seen that, unlike what was intuitively expected, the support structure does not stymie the flow of the lower impingement

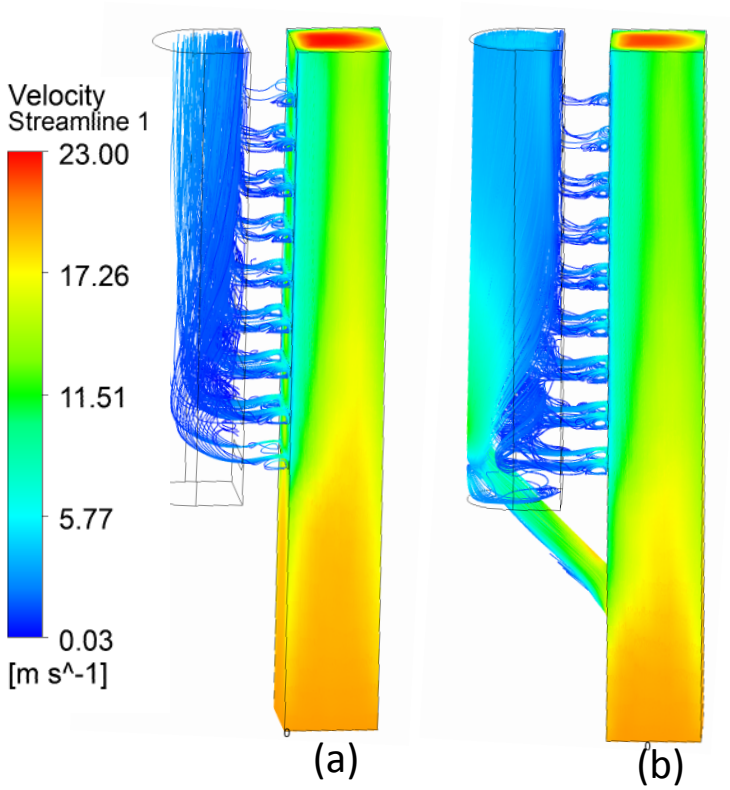


Figure 117: Velocity streamlines in the internal cavities: (a) without support and (b) with support.

jets and neither does it starve any of the upper impingement jets from cooling air. Instead it increases the mass flow from the very bottom of the leading edge cavity thereby providing better cooling overall. Also, unlike what was initially expected, the addition of the support structure does not significantly reduce the mass flow in the serpentine passage which in turn might adversely affect the cooling of other features like the tip cap. The mass flow rate difference at the exit of the serpentine channel for the two cases was found to be very small (0.145 gm/s without support versus 0.137 gm/s with support) and is not expected to lead to any adverse heating of blade regions elsewhere.

Thus, in this instance it is alright to incorporate this support feature. Of course, in a real world situation, much further analysis will be done to completely understand the effects of the addition of a new feature like this. The main intent for this analysis

was to give a better appreciation of the immense scope that exists for multi functional support design (multi functional in the sense that it not only aids in successful LAMP builds but also improves blade cooling) specific to the needs of the LAMP process.

4.6 Summary

Several common scenarios requiring the need for supports in additive manufacturing were presented. The need for supports for each of these scenarios specific to the LAMP process was evaluated. The problem of floating islands was shown to be the only factor limiting the LAMP process from building parts of any arbitrary complexity. Most of the prior approaches presented in the literature work mostly on STL meshes and typically construct supports originating from the base of the build platform or some feature directly underneath the floating island. This methodology is not applicable to the LAMP process due to the inability of removing these support structures from the part post-build. Consequently, a new approach for identifying and generating support structures for geometries that result in floating islands was presented.

CHAPTER V

NOVEL COOLING SCHEMES

As discussed in Chapter 1, the operating temperature of gas turbines has been steadily rising over the years to meet the ever increasing demands on higher efficiency. This fact combined with the extremely high fatigue loading conditions inside a turbine engine makes it very challenging to design blades that perform at acceptable levels over long periods of time. The key to achieving robust blade designs that have long life spans in such demanding conditions lies in the implementation of innovative cooling schemes. As a result, an enormous focus on developing new cooling designs has been placed over the past decade. Equipped with ever improving numerical and experimental techniques, designers and researchers have proposed several innovative designs. However, the manufacturing technology has not caught up with the pace of design innovation. As a consequence several of these designs have not been implemented in actual turbine blades in real engines. LAMP technology, with its layer-by-layer build style can alleviate this manufacturing hurdle, thereby enabling the possibility of turbine blades with advanced cooling schemes heretofore considered non-manufacturable. The work presented in this chapter aims to take a first step into this domain in order to present a better appreciation of the disruptive impact of LAMP. The major categories of cooling schemes in typical turbine blades and some novel cooling schemes in some of these categories previously reported in literature are presented next.

5.1 Types of Cooling Schemes

As illustrated in Figure 7 in Chapter 1, the cooling schemes in a typical turbine blade can be broadly categorized into internal and external schemes. Internal schemes

provide cooling to the internal surfaces of the blade while external schemes provide cooling to the outer surface of the blade which comes into contact with the hot combustion products inside the engine. Internal cooling schemes can be further sub divided based on the blade region to which they provide cooling, while external cooling consists of several different techniques like film cooling, transpiration cooling and barrier coatings to protect the outer surface of the blade from hot gases. These major categories and subcategories are summarized below (not exhaustive):

Internal:

(a) Leading edge cooling

(b) Mid-chord cooling

(c) Trailing edge cooling

(d) Tip cooling

External:

(a) Film cooling

(b) Transpiration cooling

(c) Protective barrier coatings

A brief overview of some novel cooling schemes proposed in the past is given next.

5.2 Some Novel Schemes Proposed in Literature

Several researchers in the past have focused on developing innovative schemes for each of the various cooling scheme categories discussed in the previous section. A few of them are depicted here to illustrate the kinds of challenging geometries proposed. Internal cooling is achieved by circulating cool air (borrowed from the compressor output) through internal passages inside the blades. For the mid-chord region, typically a

serpentine passage is created with trip strips on its walls in order to trip the boundary layer to a turbulent state thereby increasing the cooling achieved. Several mid-chord cooling schemes with varying cross-sections for the passages (like square, triangles and channels twisting in 3-D space etc.) and several different trip strip configurations were investigated [20–22]. A few such schemes are shown in Figure 118.

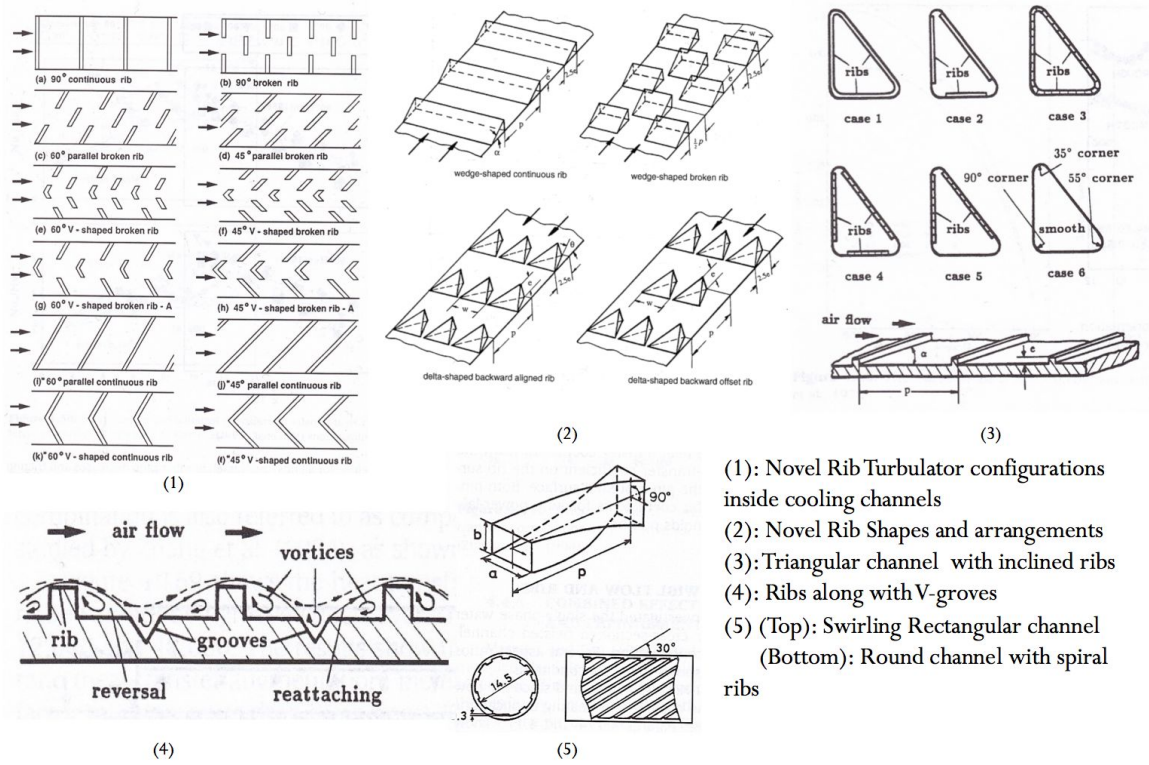


Figure 118: Some novel impingement cooling configurations [20–22].

Leading edge cooling is typically accomplished by impinging air drawn from these internal channels on to the interior wall of the leading edge. Several different leading edge cavity shapes that promote turbulence and various internal wall configurations with dimpled and ribbed surfaces that act as pins to further enhance the heat transfer have been proposed [23–25]. A few such schemes are shown in Figure 119.

Trailing edges are typically too narrow to contain internal cooling passages and therefore are cooled by thin pin-fins protruding from the internal walls to enhance the heat transfer. Several different pin fin shapes and spacial distributions have been

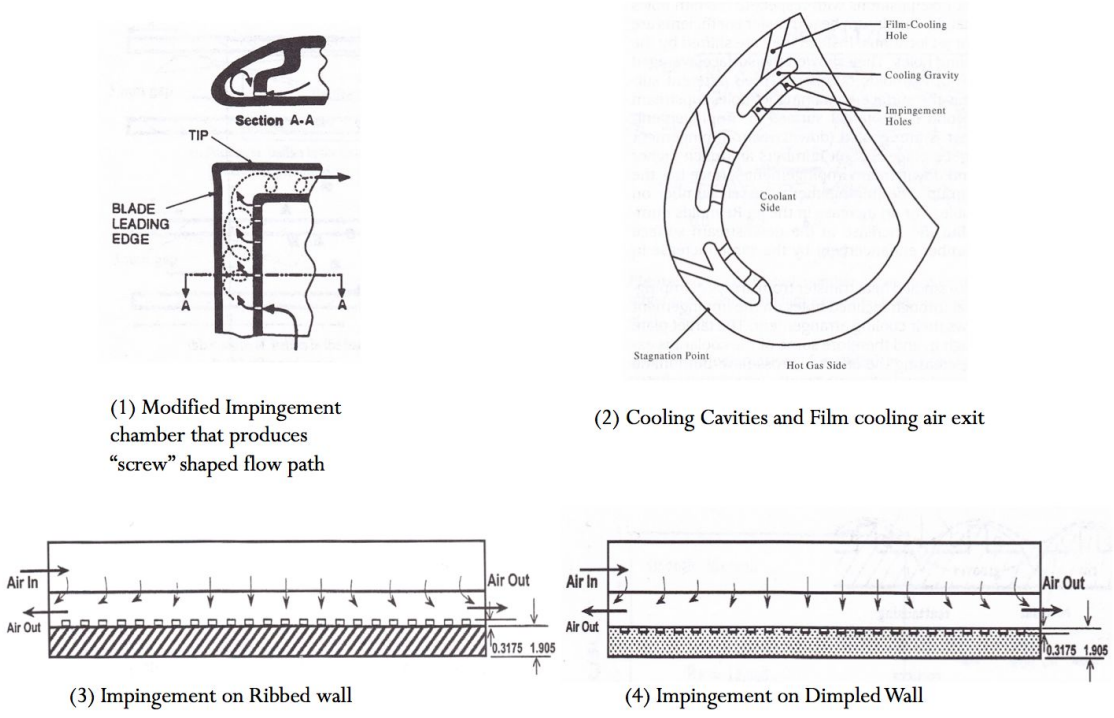


Figure 119: Some novel impingement cooling configurations [23–25].

investigated[26–28]. A Few such schemes are shown in Figure 120.

Several other novel configurations have been proposed in each of the other categories of cooling schemes. It can be observed that most of these schemes proposed have very complex geometries and will clearly be a challenge to manufacture. For the purposes of illustrating the impact of LAMP technology, the application of film cooling has been chosen. Internal cooling air from the serpentine passages is bled out though film cooling holes on the surface of the blade thereby engulfing it with a blanket of cold air to protect it from coming in contact with the hot combustion products. It has been shown that small changes in the film cooling hole placement, exit geometry and exit angle can lead to significant changes in the blade surface temperatures and consequently its life [83]. Film cooling holes are typically laser drilled after the investment casting process. Hence there is limited scope for producing novel exit geometries and shallow hole exit angles (which in turn lead to better

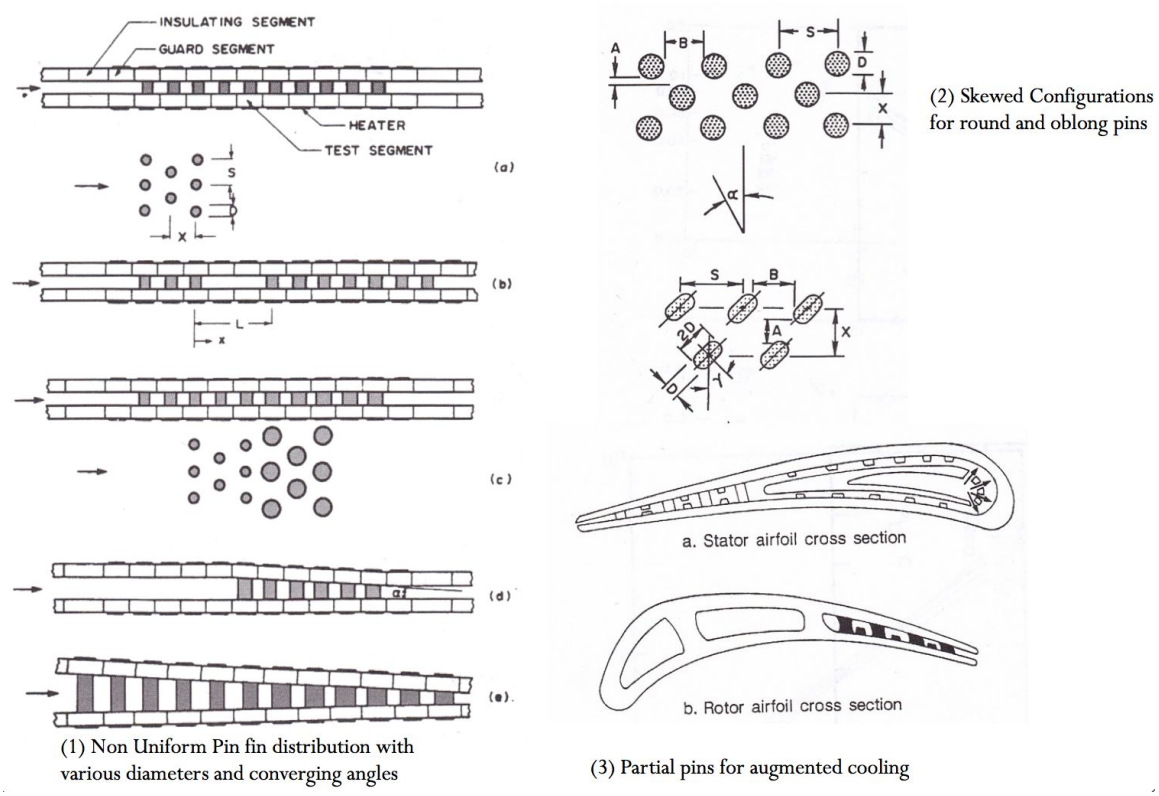


Figure 120: Some novel impingement cooling configurations [26–28].

film effectiveness). The LAMP process, on the other hand, with its layer-by-layer manufacturing process can build these film cooling hole geometries into the integrally cored molds. These cooling geometries can thus be directly cast into the blade. This can significantly alleviate cooling problems in the midst of ever increasing operating engine temperatures and hence the focus on film cooling schemes in this thesis. In this chapter, a few different film cooling hole geometries have been investigated for their cooling performance using computational fluid dynamics techniques. However, it must be noted that accurate prediction of the performance of these cooling schemes using numerical techniques is not straightforward due to the complex nature of flow physics involved. A brief discussion on this complexity and the methodology used for CFD analysis to evaluate the performance of these geometries is given next.

5.3 Flow Physics Complexity

Accurately predicting the flow physics in film cooling or any other cooling geometries in a turbine blade is extremely challenging. A few of the factors that make these kinds of studies challenging are presented in this section.

First, a turbine blade is a rotating frame of reference, rotating at speeds of the order 10,000 rpm inside an engine. This results in strong Coriolis effects on the blade which in turn significantly modify the flow physics from a static case.

Second, there are several uncertainties with respect to the initial and boundary conditions that need to be used for the simulations. Since the flow physics are very complex, only small portions of the turbine sections can be simulated to keep the computations amenable. In such a partial model setup, boundary conditions like the inflow and outflow velocity distributions of the air, input and output temperature and pressure distributions etc are very challenging to specify accurately.

Third, there is uncertainty with respect to the material properties that need to be used in these simulations. The fluids whose flow characteristics need to be evaluated in these simulations are products of combustion. The relative concentrations of the various by-products of combustion vary greatly and hence accurately specifying the material properties in the simulations is a challenge.

Fourth, the flow regimes inside a gas turbine are compressible. Due the high gas speeds and the interaction between multiple stages of turbine blades and guiding vanes, several compressibility effects like shock waves are observed. These phenomena significantly effect the performance of the designed cooling schemes but are very challenging to predict accurately.

Lastly, the flow inside gas turbines is highly turbulent. Turbulence is still an open unsolved problem in physics and the only way to predict the behavior of turbulent flows is to use approximate models. These models in turn introduce uncertainties in the solutions of the flow variables of interest. Coming up with accurate turbulence

models is an ongoing area of research with no end in sight and accurately pin pointing the inaccuracies caused by these models is very challenging.

Owing to all these complexities, simulating the performance of film cooling geometries in real engine conditions is out of the scope of the work presented in this thesis. A more simpler approach presented in the literature that is known to give a good qualitative understanding of the performance of these schemes is used. The details of this approach are presented next.

5.4 Methodology

The simulation methodology used in this thesis for comparing the performance of various film cooling hole geometries is presented in this section. To simplify the evaluation of various cooling hole geometries, it is an established practice in this domain to perform the numerical and experimental studies on flat plates rather than the complex curved geometries of turbine blades. Several key non-dimensional parameters that dictate the flow physics of these cooling geometries have been identified in the literature. A few important non-dimensional parameters are defined as follows:

(a) coolant to mainstream density ratio, $D = \dfrac{\rho_c}{\rho_m}$

(b) coolant to mainstream velocity ratio, $V = \dfrac{v_c}{v_m}$

(c) coolant to mainstream max flux or blowing ratio, $B = \dfrac{\rho_c v_c}{\rho_m v_m}$

(d) coolant to mainstream momentum flux ratio, $I = \dfrac{\rho_c v_c^2}{\rho_m v_m^2}$

(d) coolant to mainstream pressure ratio, $\dfrac{P_c}{P_m}$

(e) coolant to mainstream temperature ratio, $\dfrac{T_c}{T_m}$

When performing experimental or numerical studies on simplified geometries like flat plates instead of the complex curved geometries of turbine blades, reasonable qualitative results can still be obtained if the values of these non-dimensional parameters can be matched to the engine conditions. Most studies for evaluating film

cooling hole geometries are therefore conducted on simple flat plates to make the simulations amenable with the correct values of these non-dimensional parameters to yield reasonable qualitative insights about their performance.

Even in simulations on simpler flat plate geometries, apart from maintaining the right values for important non-dimensional parameters, care must be taken in the way the various initial and boundary conditions are specified. Many initial attempts at modeling film cooling effects on flat plates involved specifying some arbitrary arbitrary boundary conditions at the hole exit without modeling the flow physics inside the hole. However, Garg and Gaugler [130] showed that small variations of the velocity and temperature distributions at the hole exit have a large influence on the heat transfer at the blade surface near the hole, so it is important that these are accurately modeled. An alternative approach is to model not only the cooling jet-mainstream gas interaction but to also include the flows within the plenum and cooling holes. Along these lines, Leylek and Zerkle [29] developed a systematic approach to model these flows more accurately which showed good agreement with published experimental data. This approach is used for setting up simulations in this thesis. The details of the model setup using this approach are given next.

5.5 Model Setup

The model setup for accurate film cooling simulation results on flat plates as proposed by Leylek and Zerkle [29] is presented in this section. Since the flow physics inside the film cooling hole is shown to significantly impact the results [130], both the plenum that supplies film cooling air as well the hole geometry are included in the simulation domain. The geometric extents used in these simulations are matched to the extents used in experimental studies published by Pietrzyk et. al. [131–133] and Sinha et. al. [134] in order to validate the results obtained. The simulation domain used is shown in Figure 121 where D stands for the diameter of the circular hole.

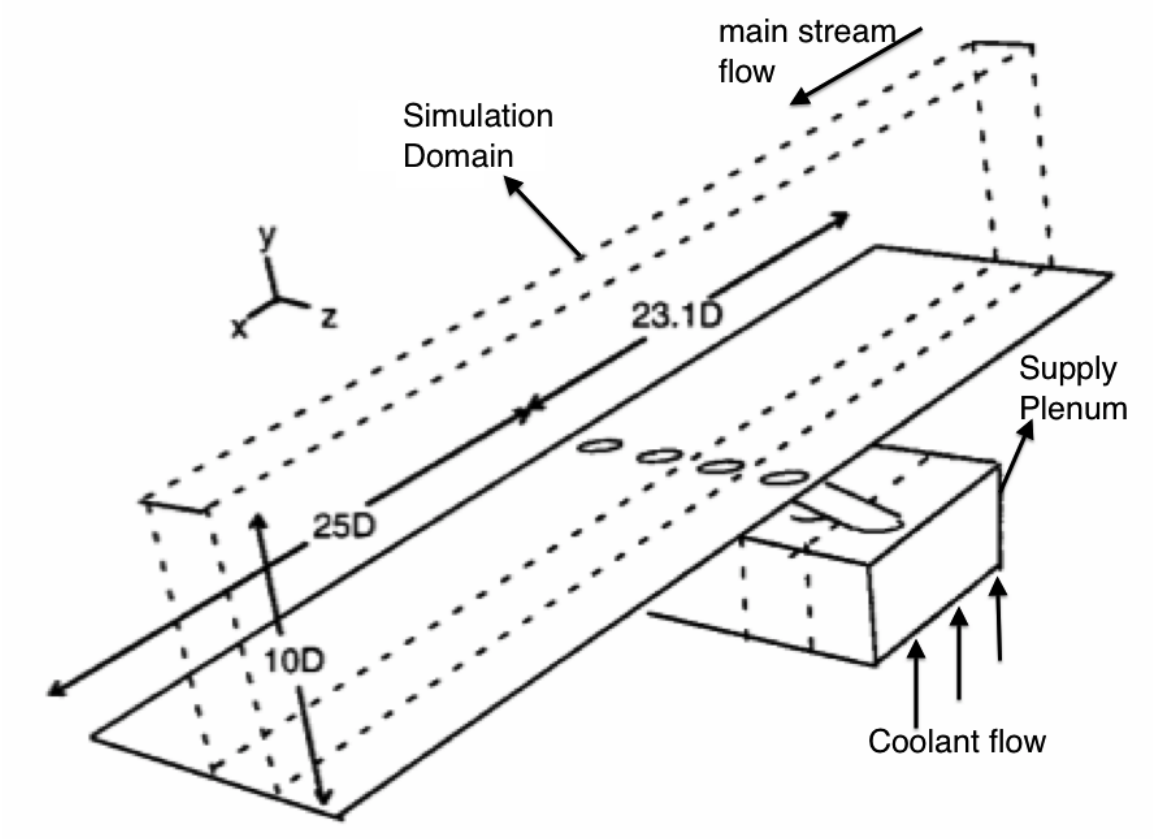


Figure 121: Film cooling simulation domain [29].

The simulations are first set up for a circular film cooling hole geometry (which is the mostly commonly used geometry in turbine blades) to serve as a base case against which the more novel geometries can be compared against. The simulation domain was meshed with a conformal structured mesh close the boundary layer (with y^+ of 30). Fully elliptic Reynolds-averaged Navier-Stokes equations and energy equation were solved over the mesh. The standard $k-\epsilon$ model with generalized wall functions for near wall treatment was used to model turbulence. A turbulence intensity of 2 percent was used. The boundary conditions were set to the conditions reported in the experimental study [131–134]. Inlet mainstream gas velocity and temperature were set to 20 m/s and 302K respectively and the outlet coolant temperature was set to 153K (which yields a density ratio of approximately 2.0). All walls were set to be adiabatic and a constant pressure condition was applied at the outlet. Simulations

were performed using the CFX software package from ANSYS with a second-order discretization scheme for all flow variables. Convergence was determined when the normalized residuals of the flow variables dropped by at least four orders of magnitude. The solution was verified to be mesh independent by first solving at a mesh size of 125,000 elements and later refining upto 250,000 elements with similar results. The results obtained for a circular film cooling hole geometry as shown in Figure 121 are presented next.

5.6 Results

Using the model setup described in the previous section, a circular film cooling hole with an aspect ratio L/D of 3.5, pitch to diameter ratio p/D of 3, and a hole inclination angle of 34^o along the streamwise direction was simulated with a blowing ratio of 1.0. Only one half of the film cooling hole geometry was simulated taking advantage of symmetry. The results obtained for this base case geometry are discussed below.

Figure 122 shows the velocity vector plot on the plane of symmetry of the circular film cooling hole domain (the mainstream gases flowing from left to right in this figure). As can be observed, a clear jetting effect occurs towards the upstream wall inside the film cooling hole leaving a large separation bubble at the hole inlet (towards the downstream wall). This is caused by the inability of the flow streamlines sharp turn around the corner at the downstream wall of the film cooling hole inlet. This separation bubble and jetting effect vorticity cause a secondary flow to form within the film cooling hole. This secondary flow is manifested as a pair of counter-rotating vortices (also referred to as a"kidney" vortex) when the coolant exits the hole and turns along the streamwise direction of the mainstream gases. This effect is evident in a velocity vector map on a plane normal to the streamwise direction as shown in Figure 123. These counter rotating vortices act against each other and lift up the coolant flow lines thereby reducing the effectiveness of the coolant film. A plot

of the coolant and mainstream gas streamlines is shown in Figure 124. It can be clearly observed that the coolant streamlines (shown in red) lift up above the surface of the plate. Moreover, the lifting up of coolant flow creates a low pressure region underneath it immediately downstream of the hole exit which results in the suction of hot gases onto the plate surface thereby causing a large heating effect. Thus, even at a modest blowing ratio of 1.0, a circular cross-section film cooling hole doesn't perform well. At higher blowing ratios, the separation and jetting effect inside the film cooling hole which was responsible for the "kidney" vortex effect resulting in the film lifting up becomes stronger thereby resulting in even worse cooling performance.

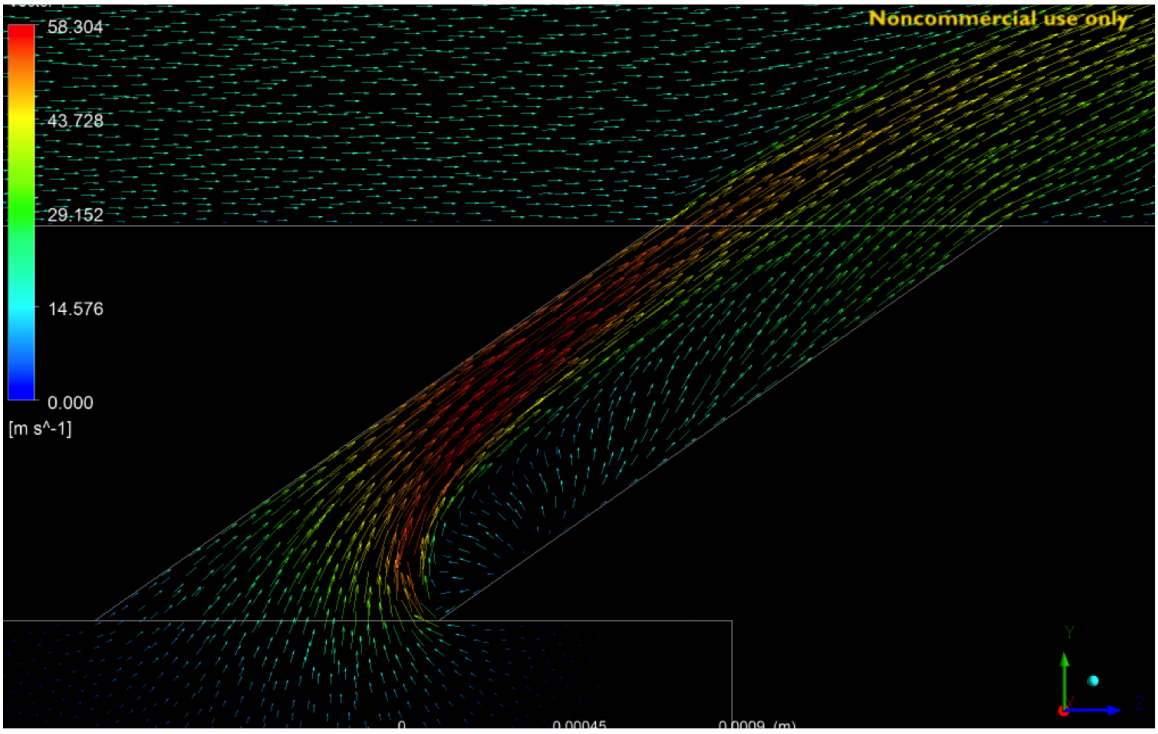


Figure 122: Velocity vector plot for a circular hole along the streamwise direction (mainstream gases flowing from left to right). Note the jetting effect inside the film cooling hole towards the upstream wall.

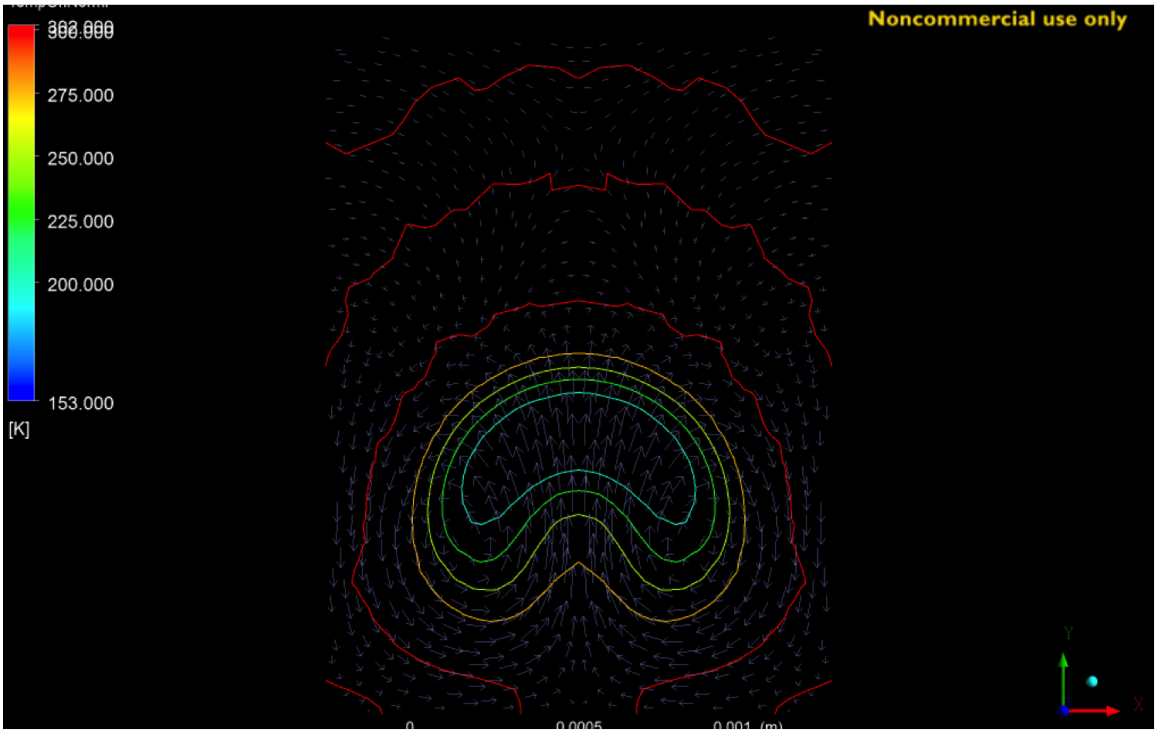


Figure 123: Velocity vector plot normal to the streamwise direction. Note the "kidney" vortex structure formed.

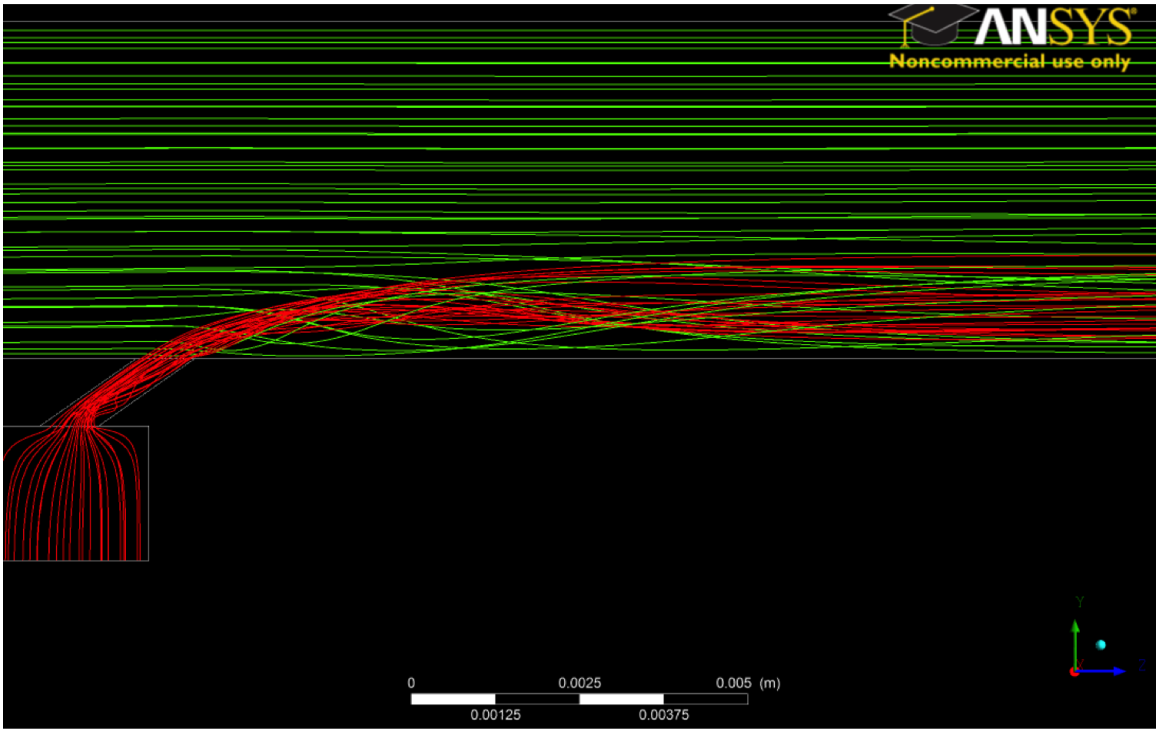


Figure 124: Streamlines of coolant and mainstream gases for circular film cooling hole geometry. Color red denotes coolant and color green denotes mainstream gases.

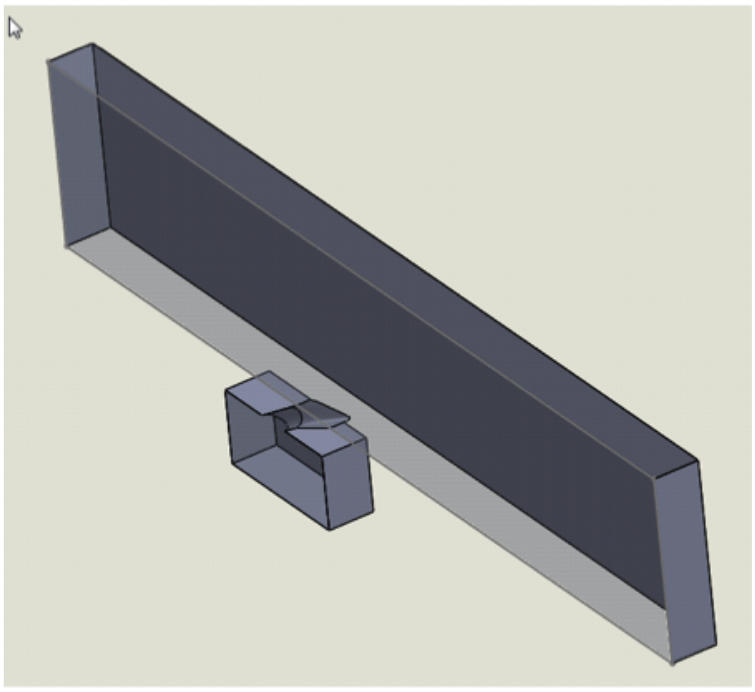


Figure 125: Diffuser hole geometry.

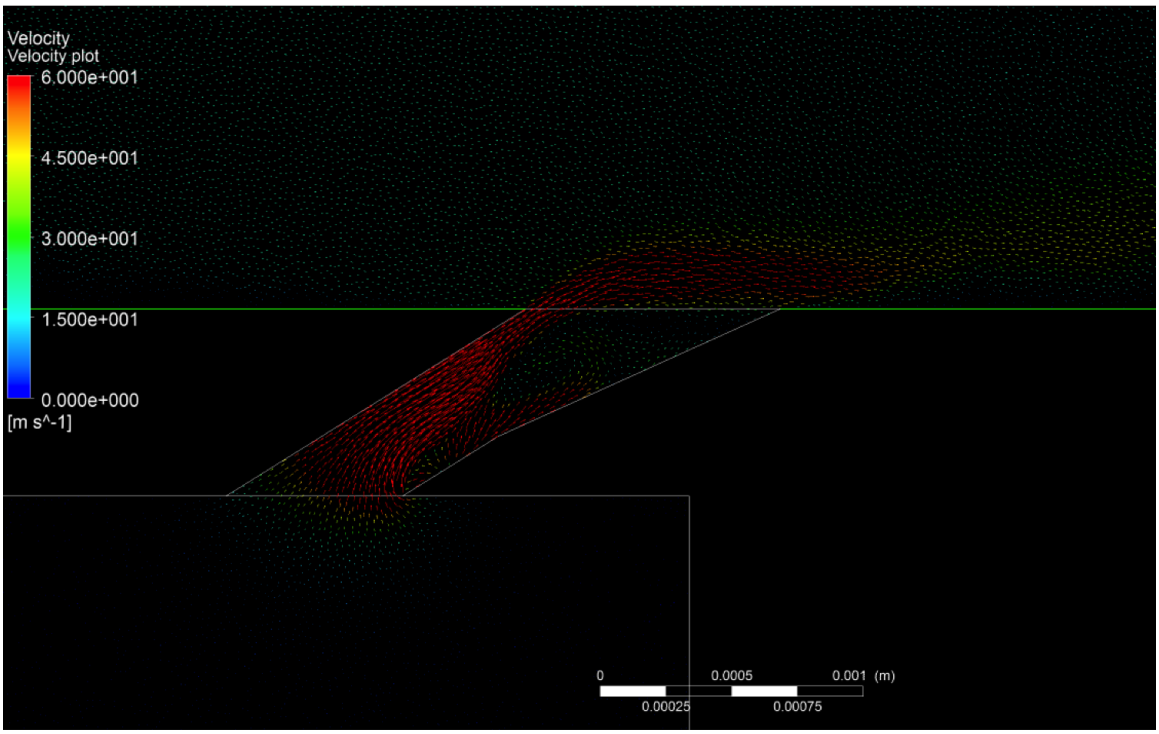


Figure 126: Velocity vector plot for a diffuser hole outlet geometry along the stream-wise direction (mainstream gases flowing from left to right).

For comparison's sake, the simulation was performed using the same model set up for a diffuser hole geometry as shown in Figure 125. Figure 126 shows the velocity vector plot on the plane of symmetry. Compared to the velocity vector on the symmetry plane for the circular hole shown in Figure 122, a much smaller separation and jetting effect is observed in the diffuser hole. Therefore, the diffuser hole exit geometry is expected to give a better cooling performance than the standard circular cross-section hole. In order to objectively compare the film performance along the streamwise direction for different hole geometries, a dimensionless parameter known as adiabatic film effectiveness η is introduced. It is defined as shown below:

$$\eta = \frac{T_{aw} - T_m}{T_c - T_m} \tag{62}$$

where T_{aw} is known as the adiabatic wall temperature which is the effective temperature experienced at the plate surface as a result of the mixing of hot mainstream gases and the coolant. T_m is the mainstream hot gas temperature and T_c is the coolant temperature.

Therefore, from the expression in Equation 62, it can be observed that when the effective temperature due to film cooling experienced at the wall is equal to the coolant temperature, the adiabatic film effectiveness will be 100% (η), i.e, when $T_{am} = T_c$, $\eta = 1$. This is the ideal scenario where the film forms a perfect blanket over the surface it is intended to protect. The adiabatic film effectiveness will be zero at the other extreme when the effective temperature experienced at the wall is equal to the hot gas temperature, i.e, $\eta = 0$ when $T_{aw} = T_m$. Thus intuitively, the non dimensional value given by adiabatic film effectiveness gives an indication of the cooling performance of the film. The higher the effectiveness, the better the cooling performance.

Film cooling simulations similar to the ones peformed for a circular and diffuser hole geometries were performed for a couple of other geometries as well. For comparisions sake, the adiabatic film effectiveness obtained along the downstream direction

for each of the hole geometries analyzed is shown in Figure 127. The steep circular hole is the conventionally used geometry due to its relative ease of fabrication and can be seen to provide the least effective film. The adiabatic effectiveness drops drastically to nearly zero immediately downstream of the hole (due to the lifting up of the film) and slowly recovers to about 18% further downstream. A shallow circular hole can be seen to provide a better peformance when compared to a steeper hole and is much more challenging to fabricate. The diffuse exit hole, which is very challenging to fabricate, gives a significantly better performance than either of the circular hole geometries with an effectiveness of above 40% for the most of the downstream length. The curved diffuser hole, which is nearly impossible to fabricate using current manufacturing techniques, gives the best film cooling performance with an effectiveness of over 50% for the majority of the downstream length.

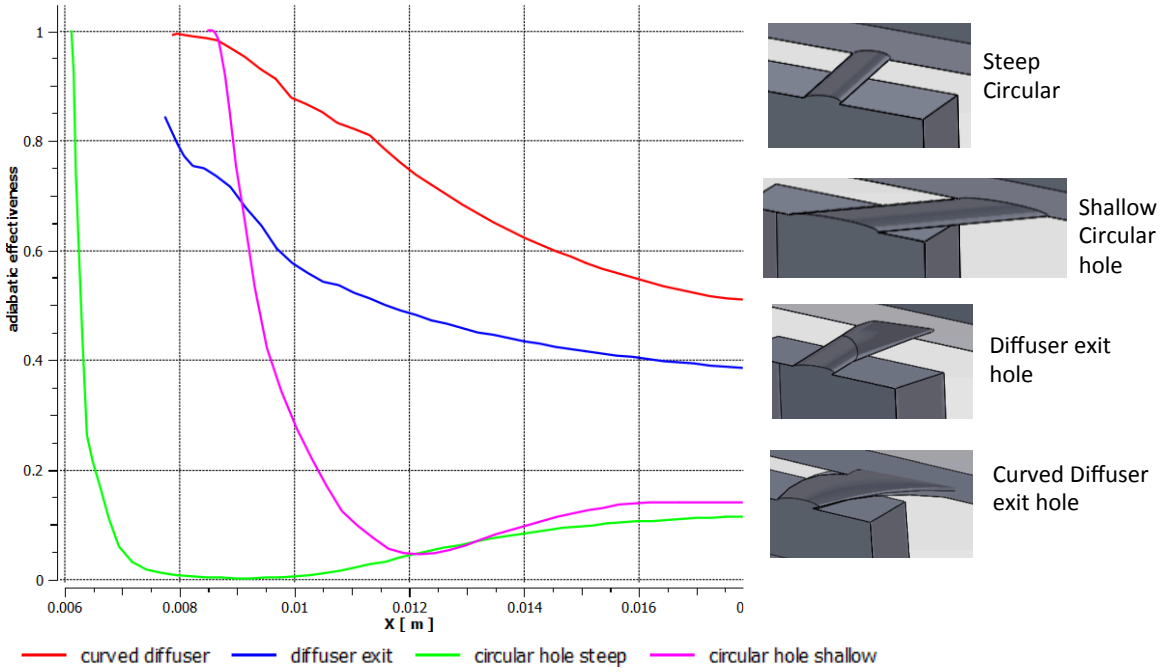


Figure 127: Adiabatic effectiveness plots along the streamwise direction for various hole geometries.

From these results, it is clear that there is immense potential to greatly improve

the film cooling performance in order to better protect turbine blades from hot operating gases by using more novel hole geometries rather the widely used circular hole geometry. Conventional manufacturing processes are unable to fabricate these geometries while the LAMP process with its layer-by-layer approach can incorporate these features on integrally cored molds to be directly cast into the blades.

5.7 Summary

A discussion on the limitations of current manufacturing methods for building novel cooling schemes is given. An outline of the varies categories of cooling schemes was presented and some challenging designs previously proposed in the literature for a few of these categories were depicted. For the purposes of demonstrating the impact LAMP technology can have in enabling the fabrication of these challenging schemes, the application of film cooling, which has received a great deal of attention due to its significant impact on blade cooling, was chosen. Film cooling holes were previously laser machined after the casting process thereby severely limiting the kinds of hole exit geometries and exit angles that can be fabricated. LAMP process with its layer-by-layer build approach can build mold designs incorporated with these hole geometries thereby directly casting them into the blades. This opens up new possibilities for film cooling hole designs. A few novel hole geometries were picked and analyzed for their cooling performance. The simulations demonstrated superior performance for these designs over the conventional circular hole geometry. A few of these designs were then fabricated on the LAMP machine to demonstrate its manufacturing capability.

CHAPTER VI

CONCLUSION

Large Area Maskless Photopolymerization (LAMP) is a disruptive additive manufacturing technology that has been developed for fabricating ceramic molds for investment casting of high pressure turbine blades. The work presented in this thesis addressed the digital data processing and computational needs of the LAMP process. Several data processing algorithms like direct slicing, STL slicing, post-processing algorithms like error checking, part placement and tiling etc. that enable the LAMP process were presented. Several computational schemes to improve the part quality like adaptive slicing, gray scaling, and cure width studies for image compensation were also discussed. Moreover, CAD data pre-processing algorithms, especially the identification of unsupported features and the generation of internal support structures suitable for the fabrication of integrally-cored molds using LAMP process were also presented. Finally, some novel cooling schemes that are not currently manufacturable but provide improved performance over the conventional schemes are presented. Such schemes can be fabricated using the LAMP process thus providing a glimpse of the potential for LAMP technology to disrupt the state-of-the-art in the investment casting of HP turbine blades. A summary of the unique contributions made in this thesis and the scope for future research along these lines is given in this chapter.

6.1 Summary of Unique Contributions

A summary of the unique contributions made in this thesis are listed here:

(a) An error-tolerant direct slicing algorithm was presented using the ACIS kernel. While previous direct slicing approaches using the ACIS kernel were reported in literature, they all propose direct slicing as a cure for all the ills inherent in STL

slicing. They claim that direct slicing will be free of errors unlike in the case of STL slicing. It was observed in this thesis that this not necessarily true. Direct slicing is free of errors only when the given geometry is simple. When the geometries are complex like the ones typically found in HP turbine blades, the CAD parts are prone to errors from two sources: 1) modeling errors on the part of the designer and 2) due to CAD translations required for direct slicing. The direct slicing algorithm presented in this thesis is tolerant to such errors and is able to produce error-free slices.

(b) For STL slicing, two novel approaches were presented: 1) A new way for reconstructing the topology information of STL meshes by extending the corner table data structure is presented. Using this topology information, STL meshes were converted to CAD files thus aiding in error correction, geometric property (volume, center of mass etc.) evaluation and modification of these meshes. 2) An extremely fast STL file slicing algorithm was implemented. Previous STL slicing algorithms reported in the literature use some sort of facet grouping strategy before they can be sliced. It was shown that this only works as long as the input mesh sizes are small. When the mesh sizes get very large (of the order of 5-6 Million facets for turbine blade molds), these approaches take exceedingly long (of the order of 4 days) to process and slice the meshes. The approach presented in this thesis bypasses the facet grouping strategy completely thereby leading to several orders of magnitude improvement in processing time (from days to just minutes).

(c) A complete post-processing work flow including error checking, part layout and tiling, image level geometry modification, data compression etc. that is based on image data was presented. Most of the previously reported approaches work with vector data obtained from the slices.

(d) A new volume deviation based adaptive slicing approach for CAD files was presented. Previous approaches presented for adaptive slicing of CAD files include determining the layer thickness based on cusp height or area deviation approach. It was pointed out in this thesis that both of these approaches have their limitations. While cusp height is an absolute criterion and a good approach for STL files, implementing this on CAD parts involves large numbers of complex calculations at each slice location, making it difficult to scale for complex parts. Area deviation on the other hand does not take the surface geometry into consideration thus seriously limiting its capability.

(e) A novel gray-scaling approach was presented to reduce the stair stepping effect on surfaces whose normal vectors point downward towards the base of the build. Previous gray scaling approaches reported in the literature either have not investigated or reported the effects of gray scaling on the curing characteristics of the material. Moreover, the material system used in the LAMP process is loaded with a high volume percentage of ceramic particles thereby making it radically different from the material systems reported in the literature for gray scale studies. The thesis work done by a previous member of the Direct Digital Manufacturing lab involving the characterization of the effects of gray scale on the curing characteristics was incorporated into the data processing algorithms to generate gray scale images that reduce the stair stepping effects on surfaces with downward pointing normal vectors.

(f) Studies to understand the side curing behavior of the LAMP suspension were conducted and some surprising conclusions were presented. It was observed that the side curing was a function of feature size in the LAMP process. Any feature smaller than about 500 pixels exhibits a different side curing behavior that is dependent on its size. It was also observed that the cure width varies linearly with respect to energy

dose in the LAMP process unlike the "quasi-Beer-Lambert" hypothesis reported in a previous study. These differences were attributed to the differences in the exposure set up and the means by which the cure width data was computed. Studies showing the variation of cure width with respect to light intensity, photoinitiator, and uv absorber concentrations were also presented.

(g) An algorithm for identifying floating islands in CAD files and generating support structures specific to the needs of the LAMP process was presented. Previous approaches mostly worked on STL meshes and they all produced straight supports aligned with the build direction that grow either from the base or from the geometry directly underneath the floating island. These approaches cannot be applied in LAMP for the fabrication of integrally-cored turbine blade molds as it is impossible to remove them post build. Hence a new approach wherein the floating islands were connected to immediate surrounding geometries is presented. The potential for design for manufacturing specific to LAMP was also discussed by means of evaluating the cooling performance of an illustrative support structure.

(h) Finally, novel cooling schemes that are currently impossible to fabricate using conventional manufacturing methods were presented. A few novel film cooling schemes have been analyzed using CFD and thermal analysis techniques and their improvements over conventional simple schemes were presented. This work gives a better appreciation of the potential LAMP offers in opening up new doors of design opportunities for building next-generation turbine blade designs.

6.2 Scope for Future Research

There is a tremendous scope for pursuing further research in several of the areas addressed in this thesis. A few potential future research directions are presented here:

(a) The post-processing algorithms presented in this thesis like image tiling etc. do scale up to much larger image sizes for larger build areas. However, there is still immense scope for optimization in order reduce processing times and data file sizes for extremely large build areas requiring very large image sizes (LAMP beta machine requires a 900 mega pixel image for each layer!). Since numerous floating point computations are involved, highly parallel algorithms running on GPUs can be developed to reduce processing times. Data sizes can be reduced by exploring the possibility of compressing the slice images as if they were individual frames of a video using key frame compression techniques. The reduction in sizes would be even more prominent at smaller layer thicknesses as the difference in frames would be smaller.

(b) The adaptive slicing approach presented in this thesis utilizes a volume deviation based approach to reduce the stair-stepping effect and hence the surface roughness in the built parts. However, since volume deviation is a relative parameter and does not directly relate to surface roughness, precise control on surface roughness cannot be obtained. Moreover, the layer profiles obtained in the process were assumed to be strictly rectangular which is not the case. Experimental studies can be pursued in order to characterize the cure profiles obtained for each layer in LAMP which in turn can be used to empirically relate layer thickness to surface roughness parameter R_a. Such a relation will be much more valuable to adaptively slice the parts while precisely controlling the surface roughness. Moreover, in order to actually build adaptively sliced parts, hardware changes need to be made to LAMP setup. Work in this direction will also play a crucial role in further improving part quality.

(c) The gray scaling approach discussed in this thesis is applicable only to downward facing surfaces. It will not work for surfaces whose normal vectors point up. Novel methods and strategies can be developed for reducing stair-stepping on such surfaces. For instance, one approach reported in the literature takes advantage of the meniscus formed by liquid resin on the stair-stepped areas to improve part smoothness. Further research can be done on such methods to produce better parts using LAMP.

(d) The cure width studies were only a first step in understanding the side curing characteristics of LAMP slurry. As discussed in that section, several factors influence this behavior and a detailed model incorporating all these effects would be ultimately beneficial. Even if this lofty goal is not possible, more immediately attainable goals can be pursued. For instance, the scaling of single layer cure width measurements to multi layer part builds can be studied. Furthermore, studies to understand non isotropic behavior of side curing, second order effects of the influencing parameters etc can be pursued.

(e) The multi-functional support structure strategy presented in the thesis can be extended into a full scale study to identify a library of suitable support geometries for typical turbine blade geometries. Also, currently all the designs and designs rules for molds are based on the conventional investment casting process. Studies to identify and create new design rules for designing molds specifically for the LAMP process can be pursued, thus enhancing the effectiveness of LAMP in successfully building them.

(f) Lastly, a plethora of design opportunities for building next generation cooling

schemes have opened up because of the LAMP process. Studies to explore more novel schemes and building these and validating their effectiveness experimentally can be pursued. These will really go a long way in truly taking the blade designs to next level of performance.

Bibliography

[1] RL Hope, RN Roth, and PA Jacobs. Adaptive slicing with sloping layer surfaces. *Rapid Prototyping Journal*, 3(3):89–98, 1997.

[2] PM Pandey, NV Reddy, and SG Dhande. Real time adaptive slicing for fused deposition modelling. *International Journal of Machine Tools and Manufacture*, 43(1):61–71, 2003.

[3] Justin Tyberg and Jan Helge Bøhn. Local adaptive slicing. *Rapid Prototyping Journal*, 4(3):118–127, 1998.

[4] Emmanuel Sabourin, Scott A Houser, and Jan Helge Bøhn. Accurate exterior, fast interior layered manufacturing. *Rapid Prototyping Journal*, 3(2):44–52, 1997.

[5] Yong Seok Suh, Michael J Wozny, et al. Adaptive slicing of solid freeform fabrication processes. In *Proc., Solid Freeform Fabrication Symposium*, pages 404–411. DTIC Document, 1994.

[6] Prashant Kulkarni and Debasish Dutta. An accurate slicing procedure for layered manufacturing. *Computer-Aided Design*, 28(9):683–697, 1996.

[7] RL Hope, PA Jacobs, and RN Roth. Rapid prototyping with sloping surfaces. *Rapid prototyping journal*, 3(1):12–19, 1997.

[8] KH Lee and K Choi. Generating optimal slice data for layered manufacturing. *The International Journal of Advanced Manufacturing Technology*, 16(4):277–284, 2000.

[9] Ka Mani, Pa Kulkarni, and Da Dutta. Region-based adaptive slicing. *Computer-Aided Design*, 31(5):317–333, 1999.

[10] Ron Jamieson and Herbert Hacker. Direct slicing of cad models for rapid prototyping. *Rapid Prototyping Journal*, 1(2):4–12, 1995.

[11] Mohammad T Hayasi and Bahram Asiabanpour. A new adaptive slicing approach for the fully dense freeform fabrication (fdff) process. *Journal of Intelligent Manufacturing*, pages 1–12, 2011.

[12] Matthew Conrad. Experimental investigations and theoretical modeling of large area maskless photopolymerization with grayscale exposure. *Masters Thesis, Georgia Institute of Technology*, 2011.

[13] Susan Gentry. Improving the spatial resolution of photopolymerizable ceramic suspensions. *PhD Thesis, University of Michigan*, 2012.

[14] Kiran Kambly. Characterization of curing kinetics and polymerization shrinkage in ceramic-loaded photocurable resins for large area maskless photopolymerization (lamp). *Masters Thesis, Georgia Institute of Technology*, 2009.

[15] Seth Allen and Deba Dutta. On the computation of part orientation using support structures in layered manufacturing. In *Solid Freeform Fabrication Symposium 1994*, pages 259–269. DTIC Document, 1994.

[16] Prashant Kulkarni, Anne Marsan, and Debasish Dutta. A review of process planning techniques in layered manufacturing. *Rapid Prototyping Journal*, 6(1):18–35, 2000.

[17] CF Kirschman, CC Jara-Almonte, A Bagchi, RL Dooley, and AA Ogale. Computer aided design of support structures for stereolithographic components. In *Proceedings of the 1991 ASME Computers in Engineering Conference*, pages 443–448. Santa Clara, 1991.

[18] Bart Swaelens, Johan Pauwels, and Wilfried Vancraen. Support generation for rapid prototyping. In *Proceedings of the Sixth International Conference on Rapid Prototyping*, pages 115–121. University of Dayton, 1995.

[19] Kumar Chalasani, Larry Jones, and Larry Roscoe. Support generation for fused deposition modeling. In *Solid Freeform Fabrication Symposium*, pages 229–241. University of Texas, Austin, August, 1995.

[20] JC Han and YM Zhang. High performance heat transfer ducts with parallel broken and v-shaped broken ribs. *International Journal of heat and mass transfer*, 35(2):513–523, 1992.

[21] DE Metzger and RP Vedula. Heat transfer in triangular channels with angled roughness ribs on two walls. *Experimental Heat Transfer An International Journal*, 1(1):31–44, 1987.

[22] Tong-Miin Liou and Jenn-Jiang Hwang. Effect of ridge shapes on turbulent heat transfer and friction in a rectangular channel. *International Journal of Heat and Mass Transfer*, 36(4):931–940, 1993.

[23] DRH Gillespie, Z Wang, PT Ireland, and ST Kohler. Full surface local heat transfer coefficient measurements in a model of an integrally cast impingement cooling geometry. *Journal of turbomachinery*, 120(1):92–99, 1998.

[24] Je-Chin Han, Srinath V Ekkad, and Yizhe Huang. Detailed heat transfer distributions under an array of orthogonal impinging jets. *Journal of thermophysics and heat transfer*, 12(1), 2012.

[25] RS Bunker and DE Metzger. Local heat transfer in internally cooled turbine airfoil leading edge regions. i-impingement cooling without film coolant extraction. ii-impingement cooling with film coolant extraction. In *Heat Transfer in*

Gas Turbine Engines and Three-Dimensional Flows, volume 1, pages 53–63, 1988.

[26] PM Ligrani, JM Wigle, S Ciriello, and SM Jackson. Film-cooling from holes with compound angle orientations. part 1: Results downstream of two staggered rows of holes with 3d spanwise spacing. *ASME Transactions Journal of Heat Transfer*, 116:341–352, 1994.

[27] Michael Gritsch, Achmed Schulz, and Sigmar Wittig. Film-cooling holes with expanded exits: near-hole heat transfer coefficients. *International journal of heat and fluid flow*, 21(2):146–155, 2000.

[28] Donald L Schmidt, Basav Sen, and David G Bogard. Film cooling with compound angle holes: adiabatic effectiveness. *Journal of turbomachinery*, 118(4):807–813, 1996.

[29] JH Leylek and RD Zerkle. Discrete-jet film cooling: a comparison of computational results with experiments. *Journal of turbomachinery*, 116(3):358–368, 1994.

[30] A Bandyopadhyay, RK Panda, TF McNulty, F Mohammadi, SC Danforth, and A Safari. Piezoelectric ceramics and composites via rapid prototyping techniques. *Rapid Prototyping Journal*, 4(1):37–49, 1998.

[31] MK Agarwala, R van Weeren, R Vaidyanathan, A Bandyopadhyay, G Carrasquillo, V Jamalabad, N Langrana, A Safari, SH Garofalini, SC Danforth, et al. Structural ceramics by fused deposition of ceramics. In *Solid Freeform Fabr. Symp. Proc*, volume 6, pages 1–8, 1995.

[32] MK Agarwala, R Van Weeren, A Bandyopadhyay, PJ Whalen, A Safari, and SC Danforth. Fused deposition of ceramics and metals: an overview. In *Proceedings of Solid Freeform Fabrication Symposium, The University of Texas, Austin*, volume 38592, 1996.

[33] E Griffin and S McMillin. Selective laser sintering and fused deposition modeling processes for functional ceramic parts. In *Proceedings of the Solid Freeform Fabrication Symposium*, volume 6, pages 25–30, 1995.

[34] A Bandyopadhyay, R K Panda, V F Janas, K Agarwala, S C Danforth, and A Safari. Processing of piezocomposites by fused deposition technique. *Journal of the American Ceramic Society*, 80(6):1366–1372, 1997.

[35] Gwenaëlle M Lous, Iván A Cornejo, Thomas F McNulty, Ahmad Safari, and Stephen C Danforth. Fabrication of piezoelectric ceramic/polymer composite transducers using fused deposition of ceramics. *Journal of the American Ceramic Society*, 83(1):124–28, 2008.

[36] Susmita Bose, Shinichi Suguira, and Amit Bandyopadhyay. Processing of controlled porosity ceramic structures via fused deposition. *Scripta Materialia(USA)*, 41(9):1009–1014, 1999.

[37] Raj Atisivan, Susmita Bose, and Amit Bandyopadhyay. Porous mullite preforms via fused deposition. *Journal of the American Ceramic Society*, 84(1):221–223, 2004.

[38] Seyi Onagoruwa, Susmita Bose, and Amit Bandyopadhyay. Fused deposition of ceramics (fdc) and composites. *School of Mechanical and Materials Engineering Washington State University Pullman, WA*, pages 99164–2920, 2001.

[39] Imen Grida and Julian RG Evans. Extrusion freeforming of ceramics through fine nozzles. *Journal of the European Ceramic Society*, 23(5):629–635, 2003.

[40] JM Taboas, RD Maddox, PH Krebsbach, and SJ Hollister. Indirect solid free form fabrication of local and global porous, biomimetic and composite 3d polymer-ceramic scaffolds. *Biomaterials*, 24(1):181–194, 2003.

[41] Jan-Thorsten Schantz, Arthur Brandwood, Dietmar Werner Hutmacher, Hwei Ling Khor, and Katharina Bittner. Osteogenic differentiation of mesenchymal progenitor cells in computer designed fibrin-polymer-ceramic scaffolds manufactured by fused deposition modeling. *Journal of materials science: Materials in medicine*, 16(9):807–819, 2005.

[42] M Allahverdi, SC Danforth, M Jafari, and A Safari. Processing of advanced electroceramic components by fused deposition technique. *Journal of the European Ceramic Society*, 21(10):1485–1490, 2001.

[43] MA Jafari, W Han, F Mohammadi, A Safari, SC Danforth, and N Langrana. A novel system for fused deposition of advanced multiple ceramics. *Rapid Prototyping Journal*, 6(3):161–175, 2000.

[44] PF Blazdell, JRG Evans, MJ Edirisinghe, PBAB Shaw, and MJ Binstead. The computer aided manufacture of ceramics using multilayer jet printing. *Journal of materials science letters*, 14(22):1562–1565, 1995.

[45] Matthew Mott, Jin-Hua Song, and Julian RG Evans. Microengineering of ceramics by direct ink-jet printing. *Journal of the American Ceramic Society*, 82(7):1653–1658, 2004.

[46] M Mott and JRG Evans. Zirconia/alumina functionally graded material made by ceramic ink jet printing. *Materials Science and Engineering: A*, 271(1):344–352, 1999.

[47] QF Xiang, J RG Evans, MJ Edirisinghe, and PF Blazdell. Solid freeforming of ceramics using a drop-on-demand jet printer. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 211(3):211–214, 1997.

[48] AR Bhatti, M Mott, JRG Evans, and MJ Edirisinghe. Pzt pillars for 1-3 composites prepared by ink-jet printing. *Journal of materials science letters*, 20(13):1245–1248, 2001.

[49] X Zhao, JRG Evans, MJ Edirisinghe, and JH Song. Ink-jet printing of ceramic pillar arrays. *Journal of materials science*, 37(10):1987–1992, 2002.

[50] WD Teng and MJ Edirisinghe. Development of continuous direct ink jet printing of ceramics. *British ceramic transactions*, 97(4):169–173, 1998.

[51] Remi Noguera, Martine Lejeune, and Thierry Chartier. 3d fine scale ceramic components formed by ink-jet prototyping process. *Journal of the European Ceramic Society*, 25(12):2055–2059, 2005.

[52] Seerden K A M Derby B Halloran J W Reis, N and J R G Evans. Direct ink-jet deposition of ceramic green bodies. i-ii. In *Solid Freeform and Additive Fabrication: a Materials Research Society Symposium*, pages 141–151, 1998.

[53] Kitty AM Seerden, Nuno Reis, Julian RG Evans, Patrick S Grant, John W Halloran, and Brian Derby. Ink-jet printing of wax-based alumina suspensions. *Journal of the American Ceramic Society*, 84(11):2514–2520, 2004.

[54] C Ainsley, N Reis, and B Derby. Freeform fabrication by controlled droplet deposition of powder filled melts. *Journal of materials science*, 37(15):3155–3161, 2002.

[55] Dichen Li Xiaoyong Tian and Jürgen G. Heinrich. *Net-Shaping of Ceramic Components by Using Rapid Prototyping Technologies, Advances in Ceramics - Synthesis and Characterization, Processing and Specific Applications*. InTech, China, 2011.

[56] Kamatchi Subramanian, Neal Vail, Joel Barlow, and Harris Marcus. Selective laser sintering of alumina with polymer binders. *Rapid Prototyping Journal*, 1(2):24–35, 1995.

[57] JG Heinrich. Lsd-basiertes selektives lasersintern. *cfi/Ber. DKG*, 86(10):D26–27, 2009.

[58] P Regenfuss, A Streek, F Ullmann, C Kühn, L Hartwig, M Horn, R Ebert, and H Exner. Laser micro sintering of ceramic materials, part 1. *Interceram*, 56(6):420–422, 2007.

[59] P Regenfuß, A Streek, F Ullmann, C Kuehn, L Hartwig, M Horn, R Ebert, and H Exner. Laser micro sintering of ceramic materials, part 2. *Interceram*, 57(1):6–9, 2008.

[60] N Coulon and P Aubry. Results on laser sintering system for direct manufacturing of metallic or ceramic components. In *Proceedings of 23rd International Congress on Applications of Lasers and Electro-Optics*, 2004.

[61] Hwa-Hsing Tang. Direct laser fusing to form ceramic parts. *Rapid Prototyping Journal*, 8(5):284–289, 2002.

[62] Ph Bertrand, F Bayle, C Combe, Patrice Gœuriot, and I Smurov. Ceramic components manufacturing by selective laser sintering. *Applied Surface Science*, 254(4):989–992, 2007.

[63] I Shishkovsky, I Yadroitsev, Ph Bertrand, and I Smurov. Alumina–zirconium ceramics synthesis by selective laser sintering/melting. *Applied Surface Science*, 254(4):966–970, 2007.

[64] Jan Wilkes, Yves-Christian Hagedorn, Wilhelm Meiners, and Konrad Wissenbach. Additive manufacturing of zro2-al2o3 ceramic components by selective laser melting. *Rapid Prototyping Journal*, 19(1):7–7, 2012.

[65] Nahum Travitzky, Hans Windsheimer, Tobias Fey, and Peter Greil. Preceramic paper-derived ceramics. *Journal of the American Ceramic Society*, 91(11):3477–3492, 2008.

[66] Curtis Griffin JoDee Daufenbach and Scott McMillin. Solid freeform fabrication of functional ceramic components using a laminated object manufacturing technique. *Solid Freeform Fabrication*, page 17, 1994.

[67] Curtis Griffin, JoDee Daufenbach, and Scott McMillin. Desktop manufacturing: Lom vs. pressing. *Am. Ceram. Soc. Bull.*, 73(8):109–113, 1994.

[68] E Alair Griffin, Daniel R Mumm, and David B Marshall. Rapid prototyping of functional ceramic composites. *American Ceramic Society Bulletin*, 75(7):65–70, 1996.

[69] Donald A Klosterman, Richard P Chartoff, Nora R Osborne, George A Graves, Allan Lightman, Gyoowan Han, Akos Bezeredi, and Stan Rodrigues. Development of a curved layer lom process for monolithic ceramics and ceramic matrix composites. *Rapid Prototyping Journal*, 5(2):61–71, 1999.

[70] Don Klosterman, Richard Chartoff, Nora Osborne, and George Graves. Laminated object manufacturing, a new process for the direct manufacture of monolithic ceramics and continuous fiber cmcs. In *Proceedings of the 21st Annual Conference on Composites, Advanced Ceramics, Materials, and Structures-B: Ceramic Engineering and Science Proceedings, Volume 18, Issue 4*, pages 112–120. Wiley Online Library, 2008.

[71] Lars Weisensel, Nahum Travitzky, Heino Sieber, and Peter Greil. Laminated object manufacturing (lom) of sisic composites. *Advanced Engineering Materials*, 6(11):899–903, 2004.

[72] Hans Windsheimer, Nahum Travitzky, Andreas Hofenauer, and Peter Greil. Laminated object manufacturing of preceramic-paper-derived si? sic composites. *Advanced Materials*, 19(24):4515–4519, 2007.

[73] Karin Schindler and Andreas Roosen. Manufacture of 3d structures by cold low pressure lamination of ceramic green tapes. *Journal of the European Ceramic Society*, 29(5):899–904, 2009.

[74] Michelle L Griffith and John W Halloran. Freeform fabrication of ceramics via stereolithography. *Journal of the American Ceramic Society*, 79(10):2601–2608, 1996.

[75] T Chartier, C Chaput, F Doreau, and M Loiseau. Stereolithography of structural complex ceramic parts. *Journal of materials science*, 37(15):3141–3147, 2002.

[76] C Hinczewski, S Corbel, and T Chartier. Ceramic suspensions suitable for stereolithography. *Journal of the European Ceramic Society*, 18(6):583–590, 1998.

[77] Jae Hyuk Jang, Shuhai Wang, Steven M Pilgrim, and Walter A Schulze. Preparation and characterization of barium titanate suspensions for stereolithography. *Journal of the American Ceramic Society*, 83(7):1804–1806, 2000.

[78] Nicolas Delhote, Dominique Baillargeat, Serge Verdeyme, Cyrille Delage, and Christophe Chaput. Ceramic layer-by-layer stereolithography for the manufacturing of 3-d millimeter-wave filters. *Microwave Theory and Techniques, IEEE Transactions on*, 55(3):548–554, 2007.

[79] X Zhang, XN Jiang, and C Sun. Micro-stereolithography of polymeric and ceramic microstructures. *Sensors and Actuators A: Physical*, 77(2):149–156, 1999.

[80] Arnaud Bertsch, Sébastien Jiguet, and Philippe Renaud. Microfabrication of ceramic components by microstereolithography. *Journal of micromechanics and microengineering*, 14(2):197, 2003.

[81] JC Han, Jong Soo Park, and CK Lei. Heat transfer and pressure drop in blade cooling channels with turbulence promoters. *Final Report Texas A&M Univ., College Station. Dept. of Mechanical Engineering.*, 1, 1984.

[82] A. J Clegg. *Precision casting processes / A.J. Clegg.* Oxford [England] ; New York : Pergamon Press, 1st ed edition, 1991. Includes bibliographical references and index.

[83] Ronald S Bunker. The effects of manufacturing tolerances on gas turbine cooling. *Journal of turbomachinery*, 131(4), 2009.

[84] Sashidhar Guduri, Richard H Crawford, and Joseph J Beaman. A method to generate exact contour files for solid freeform fabrication. In *Proceedings of the 3 rd Solid Freeform Fabrication Symposium*, pages 95–101. DTIC Document, 1992.

[85] P Vuyyuru, CF Kirschman, G Fadel, A Bagchi, and CC Jara-Almonte. A nurbs-based approach for rapid product realization. In *Proceedings of the 5th International Conference on Rapid Prototyping, The University of Dayton*, pages 229–239, 1994.

[86] Mukund Rajagopalan, Nadim M Aziz, and Cecil O Huey Jr. A model for interfacing geometric modeling data with rapid prototyping systems. *Advances in Engineering Software*, 23(2):89–96, 1995.

[87] B Starly, A Lau, W Sun, W Lau, and T Bradbury. Direct slicing of step based nurbs models for layered manufacturing. *Computer-Aided Design*, 37(4):387–397, 2005.

[88] Zhiwen Zhao and Zhiwen Luc. Adaptive direct slicing of the solid model for rapid prototyping. *International Journal of Production Research*, 38(1):69–83, 2000.

[89] W Cao and Y Miyamoto. Direct slicing from autocad solid models for rapid prototyping. *The International Journal of Advanced Manufacturing Technology*, 21(10-11):739–742, 2003.

[90] Ren C Luo and Yawei Ma. A slicing algorithm for rapid prototyping and manufacturing. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 3, pages 2841–2846. IEEE, 1995.

[91] X Chen, C Wang, X Ye, Y Xiao, and S Huang. Direct slicing from powershape models for rapid prototyping. *The International Journal of Advanced Manufacturing Technology*, 17(7):543–547, 2001.

[92] CC Chang. Direct slicing and g-code contour for rapid prototyping machine of uv resin spray using powersolution macro commands. *The International Journal of Advanced Manufacturing Technology*, 23(5-6):358–365, 2004.

[93] WM Zhu and KM Yu. Dexel-based direct slicing of multi-material assemblies. *The International Journal of Advanced Manufacturing Technology*, 18(4):285–302, 2001.

[94] Aaron P West, Shiva Prasad Sambu, and DW Rosen. A process planning method for improving build performance in stereolithography. *Computer-Aided Design*, 33(1):65–79, 2001.

[95] A Marsan, Seth Allen, Prashant Kulkarni, V Kumar, and D Dutta. An integrated software system for process planning for layered manufacturing. In *Proceedings from the 1997 Solid Freeform Fabrication Symposium, Austin, TX*. Kluwer Academic Publishers, 1997.

[96] O Reilly. http://www.fileformat.info/mirror/egff/ch09_05.htm.

[97] ACIS Documentation Portal. http://doc.spatial.com/index2.php.

[98] Robert B. Tilove. Set membership classification: A unified approach to geometric intersection problems. *Computers, IEEE Transactions on*, 100(10):874–883, 1980.

[99] Jarek Rossignac. *Solid and Physical Modeling, Chapter in the Wiley Encyclopedia of Electrical and Electronics Engineering Education, Year = 2007*. Webster.

[100] ACIS Topology. http://doc.spatial.com/index.php/tutorial:acis_tutorials_(topology).

[101] Stephen J Rock and Michael J Wozny. Generating topological information from a bucket of facets. In *Solid freeform fabrication symposium proceedings*, pages 251–259. Citeseer, 1992.

[102] Hélio Lopes and Geovan Tavares. Structural operators for modeling 3-manifolds. In *Proceedings of the fourth ACM symposium on Solid modeling and applications*, pages 10–18. ACM, 1997.

[103] Bruce G Baumgart. Winged edge polyhedron representation. Technical report, DTIC Document, 1972.

[104] Marcelo Kallmann and Daniel Thalmann. Star-vertices: a compact representation for planar meshes with adjacency information. *Journal of Graphics Tools*, 6(1):7–18, 2001.

[105] Jarek Rossignac, Alla Safonova, and Andrzej Szymczak. Edgebreaker on a corner table: A simple technique for representing and compressing triangulated surfaces. 2002.

[106] Kamesh Tata, Georges Fadel, Amit Bagchi, and Nadim Aziz. Efficient slicing for layered manufacturing. *Rapid Prototyping Journal*, 4(4):151–167, 1998.

[107] Ren C Luo, Pao-Ta Yu, Yih-Fang Lin, and Hou-Tin Leong. Efficient 3d cad model slicing for rapid prototyping manufacturing systems. In *Industrial Electronics Society, 1999. IECON'99 Proceedings. The 25th Annual Conference of the IEEE*, volume 3, pages 1504–1509. IEEE, 1999.

[108] A Dolenc and I Mäkelä. Slicing procedures for layered manufacturing techniques. *Computer-Aided Design*, 26(2):119–126, 1994.

[109] Emmanuel Sabourin, Scott A Houser, and Jan Helge Bøhn. Adaptive slicing using stepwise uniform refinement. *Rapid Prototyping Journal*, 2(4):20–26, 1996.

[110] Denis Cormier, Kittinan Unnanon, and Ezat Sanii. Specifying non-uniform cusp heights as a potential aid for adaptive slicing. *Rapid Prototyping Journal*, 6(3):204–212, 2000.

[111] Wikipedia. http://en.wikipedia.org/w/index.php?title=bisection_method&oldid=556514866.

[112] Wikipedia. http://en.wikipedia.org/wiki/dither.

[113] Javier Atencia, Susan Barnes, Jack Douglas, Mark Meacham, and Laurie E Locascio. Using pattern homogenization of binary grayscale masks to fabricate microfluidic structures with 3d topography. *Lab on a Chip*, 7(11):1567–1573, 2007.

[114] Yi Lu and Shaochen Chen. Direct write of microlens array using digital projection photopolymerization. *Applied Physics Letters*, 92(4):041109–041109, 2008.

[115] In Baek Park, Young Myoung Ha, and Seok Hee Lee. Dithering method for improving the surface quality of a microstructure in projection microstereolithography. *The International Journal of Advanced Manufacturing Technology*, 52(5-8):545–553, 2011.

[116] Chen Yong Pan, Yayue and Chi Zhou. Fabrication of smooth surfaces based on mask projection stereolithography. In *Solid freeform fabrication symposium proceedings*, pages 263–278. Citeseer, 2011.

[117] Paul F Jacobs. *Rapid prototyping & manufacturing: fundamentals of stereolithography*. Sme, 1992.

[118] Nick Silikas, Abdulaziz Al-Kheraif, and David C Watts. Influence of p/l ratio and peroxide/amine concentrations on shrinkage-strain kinetics during setting of pmma/mma biomaterial formulations. *Biomaterials*, 26(2):197–204, 2005.

[119] Cheng Sun and Xiang Zhang. The influences of the material properties on ceramic micro-stereolithography. *Sensors and Actuators A: Physical*, 101(3):364–370, 2002.

[120] Weizhao Zhou, Dichen Li, and Zhangwei Chen. The influence of ingredients of silica suspensions and laser exposure on uv curing behavior of aqueous ceramic suspensions in stereolithography. *The International Journal of Advanced Manufacturing Technology*, 52(5-8):575–582, 2011.

[121] T Chartier, A Badev, Y Abouliatim, P Lebaudy, and L Lecamp. Stereolithography process: Influence of the rheology of silica suspensions and of the medium on polymerization kinetics–cured depth and width. *Journal of the European Ceramic Society*, 32(8):1625–1634, 2012.

[122] KC Wu, KF Seefeldt, MJ Solomon, and JW Halloran. Prediction of ceramic stereolithography resin sensitivity from theory and measurement of diffusive photon transport. *Journal of applied physics*, 98(2):024902–024902, 2005.

[123] Michelle L Griffith and John W Halloran. Scattering of ultraviolet radiation in turbid suspensions. *Journal of applied physics*, 81(6):2538–2546, 1997.

[124] Younes Abouliatim, Thierry Chartier, Pierre Abelard, C Chaput, and C Delage. Optical characterization of stereolithography alumina suspensions using the kubelka–munk model. *Journal of the European Ceramic Society*, 29(5):919–924, 2009.

[125] Vladislava Tomeckova and John W Halloran. Critical energy for photopolymerization of ceramic suspensions in acrylate monomers. *Journal of the European Ceramic Society*, 30(16):3273–3282, 2010.

[126] Vladislava Tomeckova and John W Halloran. Cure depth for photopolymerization of ceramic suspensions. *Journal of the European Ceramic Society*, 30(15):3023–3033, 2010.

[127] Vladislava Tomeckova and John W Halloran. Predictive models for the photopolymerization of ceramic suspensions. *Journal of the European Ceramic Society*, 30(14):2833–2840, 2010.

[128] Douglas Webb, Victor Verdes, and Constantin Cassapis. Computer-aided support-structure design for stereolithography models. In *Proceedings of the Fifth International Conference on Rapid Prototyping*, pages 221–228. University of Dayton, 1994.

[129] Harald E Otto, Fumihiko Kimura, Ferruccio Mandorli, and Umberto Cugini. Extension of feature-based cad systems using tae structures to support integrated rapid prototyping. *Computers In Engineering*, pages 779–794, 1995.

[130] Vijay Kumar Garg and Raymond E Gaugler. Effect of velocity and temperature distribution at the hole exit on film cooling of turbine blades. *Journal of turbomachinery*, 119(2):343–351, 1997.

[131] JR Pietrzyk, DG Bogard, and ME Crawford. Hydrodynamic measurements of jets in crossflow for gas turbine film cooling application. In *ASME, Gas Turbine and Aeroengine Congress and Exposition*, volume 1, 1988.

[132] Joe Riley Pietrzyk. Experimental study of the interaction of dense jets with a crossflow for gas turbine applications. Technical report, Texas Univ., Austin, TX (USA), 1989.

[133] JR Pietrzyk, DG Bogard, and ME Crawford. Effects of density ratio on the hydrodynamics of film cooling. *American Society of Mechanical Engineers*, 1, 1989.

[134] AK Sinha, DG Bogard, and ME Crawford. Film cooling effectiveness downstream of a single row of holes with variable density ratio. In *ASME, 35th International Gas Turbine and Aeroengine Congress and Exposition*, volume 1, 1990.

[135] Chandan Kumar and A Roy Choudhury. Volume deviation in direct slicing. *Rapid Prototyping Journal*, 11(3):174–184, 2005.

[136] Seth Allen and Deba Dutta. Determination and evaluation of support structures in layered manufacturing. *Journal of Design and Manufacturing*, 5:153–162, 1995.

VITA

Anirudh Rudraraju was born in Hyderabad, a city in the southern Indian state of Andhra Pradesh, in 1986. He was raised in Visakhapatnam, a nearby city where he finished his primary and secondary education in 2003. Having graduated from high school, he attended the Birla Institute of Technology and Science at Pilani (BITS-Pilani), one of the premiere higher education institutions in India in order to pursue his undergraduate study. During his undergraduate studies he pursued two internships: one at the Advanced Systems Laboratories (ASL), an entity of the Defense Research and Development Organization (DRDO) of India, in the summer of 2005 and the other at Altair Engineering Pvt. Ltd. in the spring of 2007. He graduated from BITS with a Bachelor of Engineering with Honors [BE(Hons)] degree in Mechanical Engineering in 2007. Both of the internships he pursued as well as the primary focus of his undergraduate studies were in the areas of geometric modeling and finite element analysis. This experience has helped him secure a position as a Research Assistant in the Direct Digital Manufacturing (DDM) group at Georgia Tech under the advisement of Dr. Suman Das. He joined the graduate program at Georgia Tech in the Fall of 2007. As part of the research done in the DDM group, he was involved in the development of the Large Area Maskless Photopolymerization (LAMP) technology, a novel ceramic based additive manufacturing process. He completed his Master of Science (MS) degree in Mechanical Engineering from Georgia Tech in 2010 and received his Doctor of Philosophy(PhD) degree, also in Mechanical Engineering, in 2013 for his dissertation entitled *Digital Data Processing and Computational Design for Large Area Maskless Photopolymerization.* Following graduate studies, he intends to take up an industry job.