Optimized Mask Image Projection for Solid Freeform Fabrication

Chi Zhou
chizhou@usc.edu
Yong Chen*
yongchen@usc.edu
Richard A. Waltz
rwaltz@usc.edu

Daniel J. Epstein Department of Industrial and Systems Engineering
University of Southern California
Los Angeles, CA 90089

*Corresponding author: 213-740-7829

ABSTRACT

Solid freeform fabrication (SFF) processes based on mask image projection have the potential to be fast and inexpensive. More and more research and commercial systems have been developed based on these processes. For the SFF processes, the mask image planning is an important process planning step. In this paper, we present an optimization based method for mask image planning. It is based on a light intensity blending technique called pixel blending. By intelligently controlling pixels’ gray scale values, the SFF processes can achieve a much higher XY resolution and accordingly better part quality. We mathematically define the pixel blending problem and discuss its properties. Based on the formulation, we present several optimization models for solving the problem including a mixed integer programming model, a linear programming model, and a two-stage optimization model. Both simulated and physical experiments for various CAD models are presented to demonstrate the effectiveness and efficiency of our method.

KEYWORDS
Pixel Blending, Mask Image Projection, Optimization, Linear Programming, Solid Freeform Fabrication.

1 INTRODUCTION

Solid freeform fabrication (SFF) is a direct manufacturing process, one that can fabricate parts directly from computer-aided design (CAD) models without part-specific tooling and fixturing. Such a direct manufacturing process can significantly shorten product development cycle by rapid prototyping. Recently it has been used in direct manufacturing of various products such as Boeing’s air duct (aerospace application) and Siemens’ hearing aid shell (medical application) [1]. During the last twenty years, many novel SFF processes have been developed by using various technologies such as laser (e.g. stereolithography and selective laser sintering), nozzle (e.g. fused deposition molding and contour crafting), jetting (e.g. 3D printing and multi-jet modeling) and others [1]. A SFF process’s speed and cost largely depend on the type of technology it uses. In this paper, we are interested in mask projection based solid freeform fabrication processes. In these processes, the 3-Dimensional CAD model of an object is first sliced by a set of horizontal planes. Each slice is converted into a 2-Dimensional image which serves as a mask corresponding to the layer to be built. Using a light projection device, light defined by the mask image is then projected onto a surface to form a layer of the object. By repeating the process, 3-Dimensional objects can be formed on a layer-by-layer basis.

Digital micromirror devices (DMD) and liquid crystal displays (LCD) are the two most popular light projection devices due to their dynamic mask generation capability. Several research systems have been developed based on DMD [2-5] and LCD [6-11] by using visible-light-cured photopolymer and UV curable resin. These research systems mainly focus on fabricating micro parts (mask projection micro stereolithography). Research systems for tissue engineering and medical research were also reported [12-13]. In addition, commercial systems based on DMD have also been developed such as the Perfactory system from Envisiontec GmbH [14] and the V-Flash system from 3D Systems Inc. [15]. A SFF process based on mask projection technology can form a whole layer of multiple objects simultaneously and dynamically. Therefore it provides a potentially faster approach that is independent from part size and geometry. The mask projection technology also removes the requirement of an accurate XY motion control subsystem. Instead an off-the-shelf projector such as a projector based on DLP® technology developed by Texas Instruments can be used. Therefore the SFF system can be relatively inexpensive.

The mask image used in such a mask projection based SFF process is one of the most important process parameters. For an input 3-Dimensional CAD model, the planning of mask images is similar to the tool path planning for the CNC machining processes. However, unlike the CNC tool path planning which has been
Extensively studied, the mask image planning has not been well studied before. The problem of how to use mask image to achieve the best part quality (such as smoothness, surface finish, feature accuracy and resolution) has not been well understood for the mask projection based manufacturing processes. For example, the resolution of a DMD device is limited by the number of mirrors. Most DLP projectors currently under $1,000 have only 1024x768 mirrors. So in order to achieve a $XY$ resolution of 300 dpi (that is, 0.0033” per pixel), we would have to limit the platform size of a SFF system to an area of size 3.4”x2.6”. To achieve an even higher $XY$ resolution, it seems we would have to sacrifice the platform size more. As another example, the optical components (lens and mirrors) of a commercial light projector have inherent accuracy limitations. Therefore, each pixel has some fuzziness or image blurring. Is it better to use more expensive optical components to achieve a clearer pixel or are we better off with some fuzziness in a pixel?

In this paper, we present an optimization based method for mask image planning. For each pixel, instead of simply determining an “ON” or “OFF” state (corresponding to a white or black pixel), we calculate its gray scale value for achieving the best part quality. A commercial DLP or LCD projector usually supports at least 256 different gray scale levels. Based on an idea of pixel blending [16], we demonstrate that by intelligently manipulating pixels’ gray scale values in a projected image, we can vary exposure levels in a higher resolution within a layer. Therefore a SFF system based on our mask planning method can achieve sub-pixel $XY$ resolution. As further discussed in the paper, it is necessary to maintain some controllable “fuzziness” in a pixel for achieving desirable pixel blending. Our mask image planning framework that consists of four major steps is shown in Figure 1.

Figure 1: A mask image planning framework based on optimized pixel blending. In our method, we first use a desired sub-pixel resolution to slice an input 3D CAD model into a set of 2D images. For each image, we use geometric heuristics to set pixels’ gray scale values; we then solve a first optimization model to refine the gray scale values for minimizing the blending errors; finally, we solve a second optimization model to get the gray scale values for maximizing separation of boundary pixels with different values.

The remainder of the paper is organized as follows. In Section 2, we first discuss the principle of pixel blending for mask projection based SFF processes. In Section 3 we give a formal definition of the pixel blending problem and also review related literatures. We discuss the three major steps: geometric heuristic, optimization model for minimizing pixel errors, and optimization model for maximizing separation, in Sections 4, 5 and 6 respectively. In Sections 7 and 8, we present the simulated and physical experimental results of several test cases to further illustrate our optimization based image planning method. Finally, we conclude and discuss future work in Section 9.

2 PIXEL BLENDING FOR SFF PROCESSES

In mask projection based SFF processes, 3-Dimensional objects are formed from a solidifiable photopolymerizable medium in response to the exposure by UV or visible light. The solidifiable medium can be liquids, gels, pastes, or semi-liquid materials (e.g. mixtures of liquid materials with solid materials). In order to initiate the polymerization reaction at a certain area in a thin layer, sufficient energy has to be provided on the medium surface to the area. As extensively studied in the stereolithography process [17], a critical energy exposure threshold ($E_c$), can be found for a given type of liquid resin. When energy input per unit area is less than the
minimum energy requirement of $E_c$, the material will remain as a liquid or gel. The material will then be removed during the post-processing processes due to the lack of mechanical strength. Curing models also indicate that the input energy determines the depth of penetration of resin ($D_p$) which should be bigger than the layer thickness. The values of $E_c$ and $D_p$ can be experimentally determined.

Suppose the input light energy at a pixel is $E(K, t)$, where $K$ is the light intensity and $t$ is the exposure time of the image. Ideally, the light beam should only cover the inside area of the pixel. Also, the light intensity should be uniform inside the pixel. Such an “ideal” pixel is shown in Figure 2 as pixel $\alpha$. Obviously, the energy distribution of an “ideal” pixel is a square curve $E_\alpha$. In practice, however, a commercial light projector cannot be perfectly focused. Therefore, the light beam of a pixel will spread to its neighboring pixels (refer to pixels $\beta$ and $\chi$ in Figure 2). In addition, there are different types of optical errors including spherical aberration, astigmatism, coma, distortion, etc. Ameya [18] discussed some of those errors. We assume the energy distribution of a pixel’s light beam follows Gaussian distribution. In [4], Gaussian distribution was also used as the approximation for a point-spread function. Therefore, we can draw the energy distribution functions of pixels $\beta$ and $\chi$ ($E_\beta$ and $E_\chi$) as shown in Figure 2.

A DMD or LCD projector can dynamically control the light intensity at each pixel. For example, a projector based on Texas Instruments’ DLP® technology can control a set of tiny mirrors to reflect light into related pixels (white) or on to a heat sink (black). By controlling the reflecting time, the light intensity $K$ of each pixel can be a floating point value varying from 0 to 1 (gray). A user can specify a gray scale level (such as a value from 0 to 255) in each pixel. Accordingly, the input energy at a gray pixel is smaller than a white pixel for the same exposure time $t$ (i.e. $E_\chi < E_\beta$). This is illustrated in Figure 2, where we suppose pixel $\beta$ is a white pixel while pixel $\chi$ is a gray pixel whose light intensity is only $1/5$th of $K_\beta$. Suppose the critical exposure $E_c$ for a resin is given, which is also shown in Figure 2. Based on $E_\alpha$, $E_\beta$, $E_\chi$, and $E_c$, we can know that the input light beams at pixels $\alpha$ and $\beta$ will generate the cured geometry as shown in Figure 2. Notice nothing is fully cured for pixel $\chi$ since the energy $E_\chi$ is less than $E_c$.

Figure 2: Principle of pixel blending. The light energy of three pixels $A$, $B$ and $C$ has different shapes and sizes.

Unlike an “ideal” pixel, the light beam of a pixel can overlap its neighboring pixels. Therefore, the light intensity $K$ at any position $P(x, y)$ is actually the sum of all light intensities contributed by all its neighboring pixels. That is, $K_P = K_1 + K_2 + K_3 + \ldots + K_m$. In this paper, we refer to the convolution of neighboring pixels’ light intensity as pixel blending. For a certain exposure time $t$, we can adjust the light intensities $K_1 \sim K_m$ to achieve a desired accumulated light intensity $K_P$ and related energy input $E_P$. Comparing $E_\beta$ to $E_c$, we can determine if the resin around $P$ will be solidified. Notice we can change the light intensity $K_P$ or exposure time $t$, or both of them to vary the energy input $E_P$ at a position $P(x, y)$. Since it is easier to control the light intensity and calibrate its accumulation effect, we will fix the exposure time to a certain value $t$ and only consider adjusting the light intensity $K_P$ for all pixels in this research. Although the same exposure time $t$ is used for all the pixels in a layer, the pixel blending of...
light intensity has given us a tremendous capability in selectively solidifying resin into a certain shape for building an object. We can even move an object’s boundary in increments much smaller than a pixel size. Therefore, the boundary smoothness can be significantly improved. Both dimensional accuracy and surface smoothness are verified by our physical experiments (refer to Section 8).

Therefore a SFF system can achieve a higher $XY$ resolution by intelligently blending light intensities of neighboring pixels. In order to do that, we must adjust the projector’s optical system to ensure a certain degree of image blurring between pixels. We can further calibrate a projector’s “out of focus” condition and incorporate the calibrated results in our mask image planning framework. In addition to the controlled “out of focus” condition, we also need to intelligently set the light intensity of each pixel for generating the desired pixel blending result. In this paper, we investigate the approaches for setting a pixel’s light intensity in a mask image for a given 3-Dimensional CAD model. For a typical image size 1024x768, we will have nearly 1 million design variables and several times more design constraints. Our goal in this research is to develop an effective and efficient method for intelligently blending the light intensities of different pixels to achieve a desired shape specified by an input model.

3 PROBLEM FORMULATION AND LITERATURE REVIEW

In this section, we first mathematically define the pixel blending problem based on the principle discussed in Section 2. As shown in Figure 3, suppose $S$ denotes an image region that consists of $w \times h$ pixels. We define a gray scale matrix $H_{i,j}$ with dimension $w \times h$ where each pixel value of $H_{i,j}$ is in the range of $[0, 1]$, which is normalized from 0 to 255. To achieve sub-pixel resolution, we divide each pixel $S_{i,j}$ into small sub-pixels $(n \times n)$. As shown in Figure 1, an input 3-Dimensional model can be sliced into a target 2-Dimensional monochrome bitmap $F_{pq}$ with $(n \times w) \times (n \times h)$ pixels. Each pixel is given as a 2-value (0 or 1) matrix. Therefore, the pixel blending problem is to adjust the gray scale values of $H_{i,j}$ with $w \times h$ pixels to achieve a given bitmap $F_{pq}$ with $(n \times w) \times (n \times h)$ pixels.

Suppose the light intensity related to $H_{i,j}$ in its neighboring domain follows a Gaussian function $G_{pq}(H_{i,j}) = G_{pij}(x, y)H_{i,j}$(refer to Section 8.1 for physical verification). Each small pixel $(p, q)$ in the neighboring domain of $(i, j)$ will get a light intensity $K_{pq}(i, j) = G_{pq}(H_{i,j})$, which represents the blending effect by the big pixel $(i, j)$ at the small pixel $(p, q)$. Suppose set $S^{pq}$ is defined as a neighboring region of a small pixel $(p, q)$ such that the gray scale $H_{i,j}$ within $S^{pq}$ will be accumulated at $(p, q)$. In the calibration process, we can easily calculate $S^{pq}$ based on the $\sigma$ value of a calibrated Gaussian function for a selected projector and layer exposure time. Therefore, we can define the accumulated light intensity at the small pixel $(p, q)$ as $K_{pq} = \sum_{(i,j) \in S^{pq}} G_{pq}(H_{i,j})$. For a certain exposure time $t$, suppose the light intensity related to the critical exposure $E_c$ is a gate value $\delta_t$. That is, if $K_{pq}$ is greater than $\delta_t$, the resin will be cured within $(p, q)$ so we will open the gate; otherwise, the resin will remain liquid and the gate is closed. Correspondingly, we can denote the solidifying status of pixel $(p, q)$ as $F_{pq}'$ (1-solid or 0-liquid) based on a gate function. Finally, we get the objective value by summing all of the discrepancies between the generated bitmap $F_{pq}'$ and the target bitmap $F_{pq}$. Ideally $F_{pq}'$ should be the same as $F_{pq}$. In summary, we define the following notations ($R$ and $R$ are further discussed in Section 5.3):

$S$: Region consisting of all the big pixels with size of $w \times h$
$S_{i,j}$: A big pixel of $S$ where $(i, j) \in S$.
$S^{pq}$: Set of points which can cover $(p, q)$ during the convolution process, where $(p, q) \in S_{i,j}$.
$H_{i,j}'$: Gray scale value of the big pixel $(i, j)$.
$F_{pq}':$ Solidifying status of each small pixel $(p, q)$.
\( F_{pq} \): Desired solidifying status of each small pixel \((p, q)\).

\( K_{pq} \): Accumulative light intensity for each small pixel \((p, q)\).

\( \delta \): The threshold light intensity related to the critical exposure \( E_c \) and exposure time \( t \).

\( f \) or error: Objective function value, which equals the total number of small pixels where \( F_{pq} \neq F'_{pq} \).

\( B \): Boundary set of the image.

\( \mathcal{S} \): Boundary region of the image. It is the set of pixels around the boundary with the offset of the convolution size.

Therefore, the pixel blending problem can be defined as shown in Figure 4. In the research, we mainly use the Gaussian function as the convolution function. Notice, however, the problem definition given in Figure 4 is general to other convolution functions. Our optimization methods to be discussed in Sections 5 and 6 also do not depend on the convolution functions. It can even be applicable to a set of experimental measurement results.

In the literatures related to the mask projection based SFF processes, researchers have investigated the minimum resolution that can be achieved by micro-stereolithography, which is developed for MEMS fabrication. Research emphasis has been on characterizing the resins for better vertical (Z) resolution and on experimental determination of the lateral (XY) resolution [2~10]. Empirical models have been built based on physical experiments [6, 9]. Sun et al. [4] studied the underlying mechanisms of projection micro-stereolithography and developed a process model for it. Ameya and Rosen [5] developed an irradiance model and a layer cure model for the process planning of mask projection micro-stereolithography. However, all of this research treats each pixel as having only two possible states, “on” and “off”. Therefore, the mask image planning, which usually requires the slicing of CAD models (the first step in our framework), is much simpler. Kerekes et al. [16] first proposed to use gray scale pixels in a mask image. However, only some heuristics are proposed to determine the gray scale values. There are no theoretical analyses or modeling which are important for the mask image planning problem.

In computer graphics and the 2-Dimensional imaging industry, it is common to use pixels with gray scale values for anti-aliasing. Fundamentally, for a part as shown in Figure 18, the artifacts on the side wall are also a type of aliasing due to the limited resolution of a mask image. Therefore our method is actually an anti-aliasing technique for 3-Dimensional solid imaging processes. However, the physical principle behind the anti-aliasing in computer graphics and 2-Dimensional printing is based on human perception. We have a fundamentally different principle in our problem, which requires us to calculate both convolution and gate functions.

In the literatures on operations research, a similar research topic is the radiation treatment planning problem. It aims at quantifying optimality in radiation therapy to delivery a high dose of radiation to the tumor while sparing the healthy neighboring regions at risk. This therapy plan is a kind of conformal therapy that resembles our problem (mapping problem). Many researchers have formulated various models and proposed lots of algorithms for this problem. Lim et al. [19] presented an optimization framework for conformal radiation treatment planning problem. Lee et al. [20] offered a mixed integer programming method to radiation therapy treatment planning. Romeijn et al. [21] proposed a new model which can retain the linearity of the optimization problem as well as achieve most of the goals of the treatment plan. The main difference between our problem and the treatment planning problem is the gate function, which breaks down the continuity and makes the optimization much more difficult.

Therefore, the pixel blending problem is a challenging problem. It has both continuous and discrete variables; in addition, it has both convolution and gate functions. In the next section, we first discuss some of its properties and some geometric heuristics to simplify the problem.

### 4 GEOMETRIC HEURISTICS

In essence, the pixel blending problem aims to find a gray scale matrix \( H_{ij} \), a mask image that will be sent to a projector. Since \( H_{ij} \) will be convoluted by some transformation into a light intensity matrix \( K_{pq} \), we know \([\Phi]_{(n \times w) \times (n \times h)} \times (H)_{w \times h} = (E)_{(n \times w) \times (n \times h)}\) where \([\Phi] \) is a matrix to calculate the convoluted result \( K_{pq} \) based on \( H_{ij} \) values. Obviously, the number of design variables we can control \((H_{ij})\) is \(w \times h\); while the number of
main constraints to achieve a target bitmap $F_{pq}$ is $(n\times n)\times (n\times h) = (n^2 \times w \times h)$ by comparing $K_{pq}$ and $\delta$, along with $2 \times w \times h$ variable constraints. Therefore, we have the following theorems for the pixel blending problem.

**Theorem 4.1.** If $n=1$, for any given image $F$, we can always find a mask image $H$ and threshold $\delta$, such that the pixel blending result $F'$ is exactly the same as $F$.

**Proof.** When $n=1$, $F_{pq}$ is actually the same as $F$ since there is no distinction between small and big pixels. For the sake of simplicity, we can consider the 1-Dimensional problem. The same conclusion can be extended to the 2-Dimensional problem without loss of generality. (a) Suppose the convolution transformation has an effect on zero neighboring pixels (that is, an ideal pixel as shown in Figure 2). The coefficient matrix $\Phi$ is a matrix with only values at the diagonal elements. Therefore, it is invertible and we can easily get a mask image $H_w \times h = \Phi^{-1}_{(w \times h) \times (w \times h)} K_w \times h$. (b) Suppose the convolution transformation has an effect on $\lambda$ neighboring pixels ($\lambda \ll w$ or $h$). Without loss of generality, suppose $\lambda=1$. So in any row of the coefficient matrix $\Phi$, we will only find non-zero values in three neighboring elements. For example, for a Gaussian function, in row $i$, $\phi(i-1) = 0.1054$, $\phi(i) = 1$, $\phi(i+1) = 0.1054$. Therefore, matrix $K$ is a large sparse and strictly diagonally dominant symmetric matrix. For such a matrix, it is non-singular (invertible) [22]. Consequently we can always get one solution to satisfy all the constraints. In addition, its inverse $\Phi^{-1}$ is also a diagonally dominant symmetric matrix. Therefore we can always select a threshold $\delta > 0$ such that $\min \{H\}$ and $\max \{H\}$ are in the region of $[0, 1]$. \[\square\]

**Theorem 4.2.** If $n>1$, it is geometrically dependent whether we can find a mask image $H$ and threshold $\delta$, such that the pixel blending result $F'$ is exactly the same as $F$.

**Proof.** (1) For $n-1$, there are many more design constraints than design variables ($n^2 \times w \times h$ versus $w \times h$). Obviously if a given image $F_{pq}$ has more independent constraints than available design variables, we will not be able to solve the equations. Therefore, there must be some errors in the calculated blending results. For example, no image $H_0$ and threshold $\delta$ can be found to generate a blending result $F'_{pq}$ as a checker board as shown in Figure 5 (a) if $n=1$. (2) If a given image $F_{pq}$ has redundant design constraints such that the number of independent constraints is the same as the number of design variables, we may be able to find an image $H_0$ and threshold $\delta$ to generate a blending image $F'_{pq}$ without errors. For example, in the image as shown in Figure 5 (b), most equations for neighboring pixels in the regions (i) or (ii) have the same values. Therefore, they are redundant constraints according to matrix analysis theory. Only the boundary pixels (i.e. having both dark and light gray pixels in its neighbors) provide independent constraints. Therefore, the number of design constraints and variables can be the same for an image $F_{pq}$.

\[\square\]

![Figure 5](image-url) Figure 5: The existence of a solution for a pixel blending problem depends on a given image. (a) Pixels in a checker board are all independent constraints. Therefore no solution exists. (b) Pixels in a layer of a hearing aid shell have plenty of redundancy. A solution exists such that the pixel blending result is the same as the target image.

Fortunately, we are more interested in a given image such as the one shown in Figure 5 (b), rather than the one shown in Figure 5 (a). As discussed in Theorem 4.2, the boundary pixels (with both 1 and 0) are much more critical than the interior pixels (with only 1 or 0). Intuitively, for the interior pixels (with only 1 or 0), we can set the light intensities to 1 or 0 accordingly. In [16], the following heuristics are proposed for setting $H_{ij}$: (1) set $H_{ij} = 0$ if all its corresponding $n \times n$ pixels $F_{pq} = 0$; (2) set $H_{ij} = 1$ if all its corresponding $n \times n$ pixels $F_{pq} = 1$; (3) set $H_{ij}$ values between [0, 1] proportional to the ratio of the pixel number of $F_{pq} = 1$ to total pixel number ($n \times n$). This geometric heuristic can give us a good estimate. Using this heuristic, we can easily obtain all the gray scale $H_{ij}$ according to the status of its corresponding pixels $F_{pq}$. Then we can invoke the function for computing the objective value and use the trial and error method to get the best threshold $\delta$. The main steps of this geometric heuristic are shown in Figure 6.
for each big pixel $(i, j)$
set the number of black small pixels $n_h = 0$
for each small pixel $(p, q) \in S_{ij}$
    if $F_{pq} = 1$
        $n_h = n_h + 1$
    end
end
$H_{ij} = \frac{n_h}{n \times n}$
end
get the best threshold $\delta$ corresponding to the best objective function value.

Figure 6: Steps of geometry heuristic.

The geometric heuristic is an effective way to decide the mask image $H_{ij}$ for an input bitmap. However, when we deal with complex images which have lots of disconnected regions, the heuristic method may make wrong judgments since it determines the gray scale value of a pixel by only considering its local information (i.e. without checking the information of its neighboring pixels). A simple example is given in Figure 7 to illustrate the limitation. For an input square pattern as shown in Figure 7.(a), the mask image based on the geometric heuristic and the related light intensity blending result are shown in Figure 7.(d) and (b) respectively. There are some errors in the corners. In comparison, the mask image based on the optimization method is shown in Figure 7.(c) and the corresponding light intensity blending result is the same as the input image in Figure 7.(a). From the results, we can see that a mask image obtained by the optimization method may be non-intuitive and difficult to be characterized into heuristics. However, by playing with the gray scale values in the mask image, we can achieve higher accuracy in the pixel blending result. As further shown in Section 6.1, for problems with a big gap $\delta$ and large dimension of $S_{ij}$, the geometric heuristic often introduces a lot of errors, which will in turn result in defects to the built object. Therefore, we only use the geometric heuristic to calculate the initial solution. Based on the initial solution, we further use an optimization method to reduce the errors in the pixel blending result. A two-stage optimization method is presented in the following sections. In the first stage, we first try to minimize the discrepancy between $F$ and $F'$. After the pixel errors are minimized, in the second stage, we maximize the separation between boundary pixels with different values for the best part quality.

Figure 7: (a) A square pattern is given as the target image, which is also the blending result by the optimization method (in this case, we set $n=5$). (b) The blending result generated by the geometric heuristic. We can see errors around the corners. (c) The mask image obtained by the optimization method. (d) The mask image obtained by the geometric heuristic method.

5 OPTIMIZATION MODELS FOR MINIMIZING PIXEL ERRORS

5.1 Mixed Integer Programming Model

The mathematical model as shown in Figure 4 is ideal for describing the problem; however, it is not a standard optimization model and is difficult to feed into a general-purpose optimization solver. The main challenge is the absolute value function and the gate function. We can introduce some binary variables to reformulate it as a standard mixed-integer programming (MIP) model as follows:
Here we have transformed the “if-then” gate function into two sets of linear inequality constraints using the “big M” approach. Solving this model is a NP-hard problem. In practice, solving it to optimality can be very difficult when there are many binary variables as the solution space grows exponentially. State-of-the-art algorithms like Branch and Bound and Cutting Plane Methods [23] may take several hours on a modern workstation to get solutions even for small-scale problems such as the patterns with 50×70 pixels. Large-scale instances of these models may not be solvable in a reasonable time frame even with the most advanced MIP software. In a mask projection based SFF process, each slicing image can have 1000×1400 pixels. Our experiments showed that, unfortunately, even an advanced MIP solver such as CPLEX [25] was unable to solve a problem of this size in an acceptable time.

5.2 Linear Programming Model

The most widely used mathematical programming formulation is undoubtedly the linear program (LP) [24]. Advanced linear programming solvers (e.g. CPLEX [25], Xpress-MP [26], MINOS [27]) routinely solve large-scale optimization problems with hundreds of thousands, or even millions of decision variables, using either the simplex method [28] or an interior-point method [29, 30]. In order to utilize these linear programming solvers, we change our approach to the pixel blending problem. That is, instead of a single threshold $\delta$, we consider two thresholds $\delta_1$ and $\delta_2$ such that: if $E_{pq} \geq \delta_1$, $F_{pq} = 1$ if $E_{pq} \leq \delta_2$, $F_{pq} = 0$; and $\delta_1 > \delta_2 > 0$. If a positive value exists for the gap $\Delta = \delta_1 - \delta_2$, the MIP problem has a feasible solution. If $\Delta$ is bigger, it is easier to find a solution. Physically, if $\Delta$ is bigger, it is also easier to separate boundary pixels which are either 1 or 0. Therefore, instead of finding the minimum pixel errors, we instead try to find the largest gap separating our two sets of linear inequality constraints. We can formulate a LP model for the problem as follows:

\[
\begin{align*}
\text{maximize} & \quad \delta_1 - \delta_2 \\
\text{subject to} & \quad \sum_{(i,j) \in S_{pq}} G_{ij}(p,q)H_{ij} \geq \delta_1 \quad \forall \{(p,q) \mid F_{pq} = 1\} \\
& \quad \sum_{(i,j) \in S_{pq}} G_{ij}(p,q)H_{ij} \leq \delta_2 \quad \forall \{(p,q) \mid F_{pq} = 0\} \\
& \quad \delta_1 \geq \delta_2 \\
& \quad \delta_1 \geq 0, \quad \delta_2 \geq 0 \\
& \quad H \in [0, 1]
\end{align*}
\]

In this model, there are two types of variables: $n \times h$ light intensity variables and two threshold variables $\delta_1, \delta_2$. In addition, we have three types of constraints: $(n \times w) \times (n \times h)$ judgment constraints for all small pixels, three constraints for the thresholds and $2 \times n \times h$ range constraints for $H_{ij}$. This linear program can be solved using a standard linear programming solver such as ILOG’s CPLEX. For a given image $F$, there are two cases we need to consider.

(1) If solutions exist in the original problem such that $error = 0$, i.e., $F_{pq} = E'_{pq}$ $\forall (p,q)$, we can solve the linear program to find a positive value $\Delta$. This can provide a best gap while guaranteeing the global minimizer for the MIP problem. As discussed in Section 3, we can always find a solution for any given image $F$ if $n=1$. To test the robustness of the LP model, we designed some difficult cases as follows. We generate an image of 100×140 by randomly setting white and black pixels. The probability of setting black pixels is 0.3, 0.5, and 0.7 respectively. The
generated images are , , and . The experimental results show that the LP solver can find solutions with error = 0 even for these cases.

(2) If $error \neq 0$ i.e., $\exists (p, q) \in F_{pq} \neq F'_{pq}$ in the MIP problem, some of the constraints in $F$ are irreducibly inconsistent, which implies that the LP problem above is infeasible and, thus, has no solution. However, if we can find a small number of constraints that are inconsistent with others and somehow relax those constraints, then we can get a solution with small errors compared to the original problem. Fortunately, CPLEX has an important feature called Diagnosing LP Infeasibility (IIS) that can report valuable information about the problem even if no feasible solution has been found [25]. When a LP solution has been proven to be infeasible for a given image $F$, CPLEX can assist us to detect which variables and constraints caused the infeasibility. Based on the diagnostic results, we can then alter the model to achieve a satisfactory solution. With this important feature, we can detect the hard constraints as well as get a good solution with small errors (refer to an example shown in Figure 11.).

In many cases, we can find a positive value $\Lambda$ such that $error = 0$. However, if the calculated $\Lambda$ is too small, the solutions are not robust in practice and we may not be able to achieve such a small difference in the accumulative effect. For those cases, we can specify the two thresholds $\delta_1$ and $\delta_0$ in the LP model and use IIS to find a robust solution (that is, $\Lambda$ is bigger than a certain minimum value). This is shown in the examples given in Sections 7.1 and 7.2.

5.3 LP Model with Prior Knowledge

As discussed in Section 4, we only need to consider the variables and constraints that correspond to the boundary region. An illustration of the boundary and boundary region is shown in Figure 8. For a convolution function given as a Gaussian function, the boundary region is within $\pm 3\sigma$ distance to the boundary pixel. For internal or external pixels that are outside the boundary region, we can use the geometric heuristic to calculate their light intensities. If a big pixel $(i, j)$ is in the boundary region, the variable associated with $(i, j)$ should be considered and all the coefficients associated with this variable should be recorded; otherwise, if $(i, j)$ is not in the boundary region, we can set its light intensity to 0 or 1. In addition, we judge all the small pixels $(p, q)$ covered by the convolution range of the big pixel $(i, j)$. If $(p, q)$ is in the boundary region, we calculate a constant based on the light intensity (0 or 1) and the distance between $(p, q)$ and $(i, j)$; otherwise, the constraints associated with $(p, q)$ are redundant. The algorithm for feeding the problem into a LP solver based on the prior knowledge is given in Figure 9. This approach allows us to significantly reduce the number of variables and constraints in the LP model, which enables the LP solver to solve the problem more efficiently.

![Figure 8: Boundary pixels $R$ have both 0 and 1 in its neighbors. Boundary region $B$ are all the pixels that are within $3\sigma$ distance from $R$. The red and blue circle regions represent the convolution region for the big pixels with value 1 and 0 respectively.](image)

for each big pixel $(i, j)$
if $(i, j) \in B$, then
  for each small pixel $(p, q) \in S_{i, j}$
    add coefficient associated with $(i, j)$ to the constraint
6 OPTIMIZATION MODELS FOR MAXIMIZING SEPARATION

In the first optimization stage, the LP model aims to find the biggest upper threshold \( \delta_1 \) and smallest lower threshold \( \delta_2 \) subject to the constraints associated with each pixel and threshold. Obviously the LP problem has multiple optimal solutions. However, the LP solver will only provide us one optimal solution. Once finding the best thresholds, the LP solver will terminate immediately. Therefore, it is necessary to impose another objective function to drive the solver to find the most desirable solution among the above multiple solutions.

In Section 3, we defined \( t \) as the threshold light intensity related to the critical exposure \( E_c \) and a certain exposure time \( t \). Liquid resin at a point \( P \) will solidify if \( K_P > t \); otherwise it will remain liquid. In practice, when \( K_P \) is close to \( t \), resin will be partially cured (i.e. converting from liquid to gel instead). Therefore we want to control the light intensity for clearly separating the boundary pixels with 0 and 1. That is, the light intensity at a black pixel (0) should be as small as possible while the light intensity at a white pixel (1) should be as big as possible. Thus, we can conduct a two-stage optimization. In the first stage, we aim at finding the minimum errors between images \( F \) and \( F' \). In the second stage, we fix the thresholds based on the results of the first stage and find a set of light intensities to separate all the small pixels \( (p, q) \) with different values (0 or 1) to the largest extent. The second stage optimization model is given as follows:

\[
\begin{align*}
\text{maximize} & \quad \sum_{p=1}^{n} \sum_{q=1}^{m} \tilde{F}_{pq} \\
\text{subject to} & \quad \tilde{F}_{pq} = \sum_{(i,j) \in \mathcal{S}_{pq}} G_{ij}(p,q)H_{ij} - \delta_1 \geq 0 \quad \forall((p,q)|F_{pq} = 1) \\
& \quad \tilde{F}_{pq} = \sum_{(i,j) \in \mathcal{S}_{pq}} \delta_2 - G_{ij}(p,q)H_{ij} \geq 0 \quad \forall((p,q)|F_{pq} = 0) \\
& \quad H \in [0,1]
\end{align*}
\]

In this model, \( \delta_1 \) and \( \delta_2 \) are two given values provided by the first stage optimization. The second stage optimization will not change the optimality of the first stage. Instead it can provide us a better solution for separating the boundary pixels easily. Ideally, from the external region to the internal region, the accumulative effects will gradually change from 0 to 1. The single first stage optimization cannot guarantee this. A comparison example of the pixel blending results generated by stages 1 and 2 are given in Figure 12.

7 SIMULATION RESULTS AND ANALYSIS

A mask planning testbed has been implemented using the C++ programming language with the Microsoft Visual C++ compiler. It can automatically slice an input 3-Dimensional STL model into a set of 2-Dimensional images. For each sliced image, it then calculates the optimal gray scale values for each pixel. A network version of CPLEX is integrated in our testbed. We also use CAD models with various complexities to test the effectiveness and efficiency of our mask image planning method. The sliced images of three different CAD models are shown in Figure 10. The first one is a slice layer of a simple square model; the second one is a slice layer of a mechanical part; and the third one is a slice layer of a dragon model. The test results based on our method are provided in this section. The test platform we used is a PC with an Intel P4 2.2GHz CPU and 960 MB RAM, running on the Windows XP operating system. The size of each image is \( 1000 \times 1400 \). If we consider all the pixels, there are in total \( \frac{1400 \times 1400}{n \times n} \) variables and \( 1000 \times 1400 \) constraints (excluding the range constraints and threshold constraints). Even if we only consider the pixels in the boundary region, they are still quite large-scale optimization problems. We tested various \( n \)
values and gaps $\Delta$ in the three examples. The convolution function we used is a Gaussian function, in which we set $A = 1$ (light intensity scale factor) and $\sigma_x = \sigma_y = 1$. The convolution size is $cw = ch = 3\sigma$. We denote the single first stage optimization method as LP1 and the two-stage optimization method as LP2.

![Image of three test images: Square, Round, Dragon](a) Square (b) Round (c) Dragon

Figure 10: Three test images.

### 7.1 Results of Geometric Heuristic and LP1

The experimental results of the geometric heuristic and LP1 are shown in Table 1 and Table 2 respectively. Table 1 lists the errors for different gaps with different images and $n$ values. Table 2 lists the optimized gaps which can guarantee the global optimal solution. From Table 1, we can see the problems become more difficult as gap and $n$ become larger. The geometric heuristic cannot get the global optimal solution for most of the cases. From Table 2, we can also see the problems are more difficult when $n$ is larger. However, the optimization method can still solve the problem to global optimum with a reasonable gap for most cases. The number of reduced variables and constraints from using the prior knowledge of the problem is also shown in the table. The CPU time for solving the optimization problems demonstrates the efficiency of the LP model. As shown in Table 1, for the cases above the double lines, LP1 can always find the global optimum with $\text{error} = 0$, while the geometric heuristic cannot get $\text{error} = 0$ in most cases. Comparing the results of the geometric heuristic and LP1, we can see that the optimization method can find much better solutions. For the case as shown in Figure 9, the geometric heuristics generate lots of errors, while the result generated by LP1 is exactly the same as the original image. For the cases below the double lines, LP1 cannot find a feasible solution. Therefore we can turn to $IIS$.

<table>
<thead>
<tr>
<th>Table 1: Results of Geometry Method.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Square</strong></td>
</tr>
<tr>
<td>$n=1$</td>
</tr>
<tr>
<td>gap (1.06)</td>
</tr>
<tr>
<td>err (0)</td>
</tr>
<tr>
<td>#variables</td>
</tr>
<tr>
<td>#constraints</td>
</tr>
<tr>
<td>CPU time (s)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2: Results of LP1.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Square</strong></td>
</tr>
<tr>
<td>$n=1$</td>
</tr>
<tr>
<td>gap (1.06)</td>
</tr>
<tr>
<td>err (0)</td>
</tr>
<tr>
<td>#variables</td>
</tr>
<tr>
<td>#constraints</td>
</tr>
<tr>
<td>CPU time (s)</td>
</tr>
</tbody>
</table>
7.2 Results of LP1 with IIS

The results generated by LP1 with IIS are shown in Table 3. Comparing the results of Geometry and LP1 with IIS, we can see that the optimization method can almost always get much better solutions than the conventional Geometry method. With the specific thresholds, CPLEX’s IIS can not only find the solution with small errors, it can also provide us the upper bound or the lower bound of the thresholds which can help us detect the hard constraints.

A smaller dragon image (50x70) is shown in Figure 11(a) as an example, where we specify δ1=2.0 and δ2=1.4 for the image. An optimized result given by CPLEX’s IIS is shown in Figure 11(b). All constraints are satisfied except two constraints that are marked in red (1.9) and blue (1.5). If we can relax δ1 to less than 1.9 and relax δ2 to greater than 1.5, then we can get a solution that satisfies all the constraints. Compared with the original picture, the two hard constraints associated with two pixels resemble the checker board pattern.

Table 3: Results of LP1 with IIS.

<table>
<thead>
<tr>
<th>n=1</th>
<th>n=2</th>
<th>n=3</th>
<th>n=4</th>
<th>n=5</th>
<th>n=1</th>
<th>n=2</th>
<th>n=3</th>
<th>n=1</th>
<th>n=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>gap</td>
<td>err</td>
<td>gap</td>
<td>err</td>
<td>gap</td>
<td>err</td>
<td>gap</td>
<td>err</td>
<td>gap</td>
<td>err</td>
</tr>
<tr>
<td>1.1</td>
<td>4</td>
<td>0.6</td>
<td>8</td>
<td>0.3</td>
<td>4</td>
<td>0.18</td>
<td>4</td>
<td>0.09</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>4</td>
<td>0.7</td>
<td>20</td>
<td>0.4</td>
<td>8</td>
<td>0.24</td>
<td>8</td>
<td>0.12</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>1596</td>
<td>0.8</td>
<td>1998</td>
<td>0.5</td>
<td>32</td>
<td>0.3</td>
<td>26</td>
<td>0.15</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 4: Results of LP2.

<table>
<thead>
<tr>
<th>square</th>
<th>round</th>
<th>dragon</th>
</tr>
</thead>
<tbody>
<tr>
<td>σ²_ε</td>
<td>n=1</td>
<td>n=2</td>
</tr>
<tr>
<td>σ²_ε^*</td>
<td>0.67</td>
<td>0.96</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>49</td>
<td>31</td>
</tr>
</tbody>
</table>

Figure 11: An example of LP1 with IIS. In this case, we set threshold δ1=2.0 and δ2=1.4. The black number (2.0) and green numbers (1.4) in (b) are corresponding to the black pixels and the white pixels in (a) respectively. The red number (1.9) and blue number (1.5) are corresponding to the pixels which cannot be satisfied by LP1.
Figure 12: The accumulative effects of LP1 and LP2 for the dragon model. (a) The accumulative effects of the whole image for LP1. (b) The accumulative effects of a local profile for LP1. The color bar indicates the values of the accumulative effects. The dotted line indicates the target boundary of the profile. (c) The accumulative effects of the same profile for LP2.

Figure 13: The accumulative effects in the boundary region of the dragon model for LP1 and LP2. In each figure, the horizontal line is the index number and the vertical line is the accumulative values. Two thresholds, $\delta_1$ and $\delta_2$, are also marked in the figure.

7.3 Comparison between LP1 and LP2

The two-stage optimization results are shown in Table 4. In the table, we list the average deviation

$$\sigma = \frac{\sum_{p=1}^{N_p} \sum_{q=1}^{N_q} \tilde{F}_{pq}}{N_{pq}}, \quad (p, q) \in \mathcal{B},$$

where the deviation $\tilde{F}_{pq}$ is defined in the second stage optimization model as shown in Section 6. The $\sigma_p$ is the deviation before the second stage optimization and $\sigma_p^*$ is the deviation after the second stage optimization. As can be seen from the table, there is a large improvement ($\sigma_p^* < \sigma_p$). CPU time in the table is the total time of the two stages. Figure 12 illustrates the pixel blending results of LP1 and LP2. We can see that although LP1 can solve the main objective function (pixel error) to the global optimum, the accumulative effects around the boundary are irregularly distributed. In comparison, the accumulative effects generated by LP2 are much more regularly distributed. It changes from a small value to a big value gradually, which is what we want to achieve in the pixel blending. Figure 13 plots the accumulative effects for all the boundary pixels of the dragon model as
shown in Figure 12. To generate the figure, we assign a unique index number for each pixel \((p, q)\) and compute its \(K_{pq}\). Correspondingly, we can use the value of \(K_{pq}\) for the Y-axis and the assigned pixel index number for the X-axis. The two threshold light intensity values \(\delta_1\) and \(\delta_2\) are discussed in Section 6. Comparing the LP2 result with the LP1 one as shown in Figure 13, we can clearly see that the accumulative light intensities are separated further away from the two thresholds. That is, the light intensity for a black pixel is much bigger than \(\delta_1\); while the light intensity for a white pixel is much smaller than \(\delta_2\).

8 EXPERIMENTAL RESULTS AND ANALYSIS

We have also developed a prototype machine in order to verify the presented pixel blending technique. An off-the-shelf DLP projector (CASIO XJ-S36) was used. A picture of our prototype machine is shown in Figure 14.(a). Motion control hardware and software (not shown) were developed for feeding material during the layer building process. The mask planning software system as discussed in Section 7 was integrated in the machine which can convert an input 3-D CAD model into a set of 2-D images for the projector (refer to Figure 14.(c)). All the three image planning methods as discussed in Figure 1 have been implemented in the system. An experimental test part built by our system is shown in Figure 14-(d).

8.1 Pixel Shape Verification

A fundamental assumption we made in the optimized pixel blending is that a DLP projector is not perfectly focused on a single pixel (an ideal pixel). Instead the light beam of a pixel will spread to its neighboring pixels (refer to Figure 2). Consequently it is possible for us to intelligently control pixels’ light intensity to improve the surface quality and dimension accuracy of built parts. We further assume the light intensity of a pixel can be modeled as a Gaussian function. Correspondingly, a problem formulation based on the convolution function is presented in Section 3. To verify the assumptions, we projected a single pixel onto the resin and used a high-quality digital camera to take a set of pictures under different gray scale levels. A picture for one pixel with the gray scale level at 255 (pure white) is shown in Figure 15.(a). Notice the light of the pixel spreads out around a radius of approximately three “ideal” pixels. In addition, the center of the pixel is the brightest. We plotted the brightness of the taken picture for a gray scale distribution, which is shown in Figure 15.(b). If we get rid of the noise and the errors introduced by the camera, we can reasonably assume such a distribution is a Gaussian distribution. We can further approximate the light intensity of the pixel by a Gaussian function as \(G(x, y) = 0.049 + 0.959e^{-\frac{1}{2}[(x-\mu)^2+(y-\nu)^2]}.\) The fitting Gaussian function is shown in Figure 15.(b). By changing the light intensity of a pixel within the range of 0–255, we can repeat the analysis process to get a set of functions for different gray scale levels. A comparison
of such light intensity distributions related to different gray scale levels (raw data) is shown in Figure 15.(c). The generated Gaussian functions can be used in solving the optimization problems as discussed in Section 5.

Based on the calibration result, our experimental verification consists of two parts: dimension accuracy and surface quality.

8.2  Dimensional Accuracy Study

Dimensional accuracy refers to the differences of dimensions between the experimental results and the original geometry. As discussed in Section 2, the traditional method based on binary images does not consider the energy blending between neighboring pixels. Therefore, the built part may lose some thin portions and shrink around the boundary. In contrast, the optimized pixel blending method can retrieve those portions and guarantee high accuracy by manipulating the light intensities or exposure time of neighboring pixels. In this study, we fixed the exposure time for the whole layer and only change the light intensities for each pixel. The exposure time we set is around 20 seconds. This is based on a study by using the design of experiment (DOE) method to find the best penetration and binding property of built part.

Figure 16: A simple test case for dimensional accuracy study.
We designed several test parts for a comparison of the building results by different methods. A simple test part with thin walls is shown in Figure 16.(a) with a 2-Dimensional cross-section of the input geometry. The experimental results generated by the optimization and traditional methods are shown in Figure 16.(b) and (c) respectively. Based on the results, it can be seen that the traditional method will lose two thin walls at left and bottom, while the pixel blending method can get the shape quite close to the original one. To get a better understanding of the results, we simulated the energy distribution based on the convolution of pixels’ light intensity. The generated energy distributions are also given in Figure 16.(b) and (c). The study demonstrates the dimension accuracy difference between the traditional and optimization methods.

We also tested the dragon case as discussed in Figure 12. To compare the dimensional accuracy, we built the same geometry side by side with different image planning methods. The experimental results obtained by the traditional method (i.e. using binary images), geometric heuristic and optimized pixel blending method are shown in Figure 17.(b)–(d) respectively. From the result, we can see the traditional method will completely lose the isolated thin potion; the geometric heuristic method can only save portion of it; while the optimized pixel blending can get a shape that is close to the original geometry. Therefore, the optimized pixel blending method can significantly improve the dimension accuracy of a SFF process that is based on the mask image projection.

8.3 Surface Quality Study

The DLP projector has a finite resolution. In our machine, the projector has a resolution of 1024×768. The platform size based on our design is 8.5×6.4 inches. Thus the physical resolution of our system is about 120 dpi. The traditional method based on binary images will not be able to guarantee a high surface quality due to the limited resolution. As studied in 2-Dimensional image printing, anti-aliasing techniques can improve the quality of curve rendering. Our pixel blending method uses a similar idea to get a higher surface quality for 3-Dimensional geometries.
The main surface quality problem occurs when the normal of a surface is close to horizontal or vertical. We designed several test parts with beveled surfaces to test the building results by different methods. The cross-section of a test part is shown in Figure 18.(a). The results obtained by the traditional and optimized pixel blending methods are shown in Figure 18.(b) and (c) respectively. We marked the edges to get a better visual effect. It is obvious that the building result based on the pixel blending method has a much better surface quality. In Figure 18.(b) and (c), the simulated energy distributions by convoluting the light intensity functions of all the pixels are also given.

A quantitative difference on surface quality is shown in Figure 19. This is obtained by mounting a micrometer (Brown & Sharpe AGD Indicator) on a lead-screw linear system. The part is fixed on the table. Initially the probe touches a starting point of the part. Each time a number is recorded from the micrometer when the probe advances one step. A surface quality comparison by plotting the recorded data is shown in Figure 19. It is obvious from the figure that the traditional method will get a ragged surface, while the optimized pixel blending method can achieve a much smoother surface quality.

![Figure 19: Surface quality measurement setup and surface quality comparisons between the traditional and optimized pixel blending methods.](image)

Notice in all the tests on dimensional accuracy and surface quality, we used the same building process except different mask images for each layer. The differences in the built parts illustrate the importance of the mask image planning method.

9 CONCLUSION AND FUTURE WORK

Mask image planning is an important process planning step for the mask projection based solid freeform fabrication processes. Pixel blending is a technique that can significantly increase the XY resolution of these SFF processes. For any given 3-Dimensional CAD model, an effective method is necessary for intelligently setting pixel’s light intensity in a mask image to achieve the desired blending result. We formulate a general mathematical model for the pixel blending problem. Due to the large number of variables and constraints, an efficient method is required for solving the problem. Our solution strategy is based on solving a two-stage optimization model. We propose to use Linear Programming (LP) models and simplify the problem based on geometric heuristics. Experimental results show that a LP solver (CPLEX) along with an advanced infeasibility analysis feature (IIS) for the LP model can effectively and efficiently solve most practical problems. The second stage optimization can
further improve the part quality without losing the optimality of the first stage optimization. Both simulation and physical experiments demonstrate the effectiveness and significance of our method.

Our future work includes the following. (1) Different images have different complexities (e.g. checker board vs. dragon vs. square). We would like to establish a quantitative measure of the complexity of a given image. Such a measure could guide us to formulate and solve these problems more effectively. (2) We would like to further incorporate the projector’s calibration data into our framework. The light intensity of a commercial projector is generally non-uniform. We are developing a testbed to characterize the non-uniformity. Accordingly we plan to improve our optimization solver to consider the additional requirement.

ACKNOWLEDGMENTS

This work is partially supported by a research initiation award provided by the Society of Manufacturing Engineering (SME) Education Foundation. The third author is also supported by National Science Foundation grants CMMI-0728036 and CMMI-0728334. We acknowledge Dr. Fernando Ordonez for his support and for making the resources of optimISE lab at the USC available to us.

REFERENCES


