Lightfield and Image-based Rendering

Name: Jie Wu u5835210
Supervisor: Matt Adcock

The Australian National University
Canberra ACT 2601 Australia
www.anu.edu.au

CRICOS Provider No. 00120C
Abstract
In view rendering and synthesis, the maximum-likelihood method is likely to produce large regions of errors, thus priors are introduced to improve the rendering quality. While complex structures and textures can be modelled with priors, methods with large-patch priors tend to reach local optimum instead of global ones, which affects the effectiveness of these methods in practice.

In this report, we use the energy minimization framework similar to the above methods, but substitute the large patches with small cliques in prior terms. We use the local dictionary instead of the global one, so output cliques only train from the corresponding local regions in the input, which makes rendering faster and improves the output quality. We propose a simple, fast method to discretize the colors of output pixels, and demonstrate an efficient algorithm to compute the set of possible colors at each pixel.

The work above, combining together, makes it practical to solve our problems with global optimization techniques. We implement our method and compare it with other popular rendering algorithms. We prove our ideas by analyzing experiment results and discuss the underlying reasons for the different performances of these algorithms.

Keywords
Unary energy, prior energy, clique, local dictionary, global dictionary
Acknowledgement

I would like to express my thanks to my supervisor, Matt Adcock, for his generous help and the plentiful resources he has provided.
Secondly, I would like to thank course coordinator Weifa Liang, who has given me valuable suggestion on my project, presentation and report writing.
I would also like to thank my parents and friends for their support, and our college and ANU as a whole for the convenient access to powerful lab machines and other equipment.
The project is supported by CSIRO and CSIT.
Table of contents

Chapter 1: Introduction and Background........................................................................5
Chapter 2: Notation........................................................................................................7
Chapter 3: Unary Energy ................................................................................................10
  Section 3.1: Form of Unary Energy...........................................................................10
  Section 3.2: Discretization of Color and Depth.........................................................10
  Section 3.3: Calculating Likely Colors Given a Depth.............................................11
Chapter 4: Prior Energy ...............................................................................................14
  Section 4.1: Form of Prior Energy...........................................................................14
  Section 4.2: Local dictionary....................................................................................14
  Section 4.3: Summary and Minimization................................................................15
Chapter 5: Evaluation....................................................................................................16
Chapter 6: Discussion....................................................................................................19
Chapter 7: Conclusion....................................................................................................20
References....................................................................................................................21
Chapter 1: Introduction and Background

In computer vision, image-based rendering refers to the discipline of generating new views directly from multiple two-dimensional images without 3D modelling stage [1]. One of the main motivations for the development of light field and image-based rendering is that the geometry of the real world is very complex, such as hair and trees. The difficulties of creating accurate models and representing objects' texture and shines also contribute to the strong interest of researchers to work on this field. Besides, photorealism is a main measure of image rendering quality, that is, how the rendered image is indistinguishable from a photograph, which makes image-based rendering more attractive because it uses images especially photographs directly.

In this report, we will focus on the area of view synthesis. The problem is that given a set of images of a 3D scene, synthesize the new view from a given new viewpoint beyond the original viewpoint set. It is a poorly constrained problem, so carefully constructed priors are essential in order to produce a good result. The priors mentioned above refer to the prior information on the input images, taking the form of energy functions. These energy functions correspond to patches or cliques in the output view and model texture or depths of the objective scene [2]. With the use of priors, the original problem can be placed into an energy minimization framework.

The factor that has the strongest influence on the tractability of these problems is the size of the patches, which is also called the order of the prior. Depending on the form of prior terms, a global or strong optimum of the energy function could be effectively calculated with global optimization techniques such as tree-reweighted message passing algorithm [3] and graph cuts [4]. This is often the case if, for example, the assumption of piecewise smoothness holds [5]. However, if the prior order is larger than two, the global optimization might be impractical due to the complicated formulations. Historically, there has been a tradeoff in the choice of priors. While
larger patches can better represent the complex structure in the real world, the
global optimum can hardly be reached. On the contrary, while small cliques can only
model simple structures, the resulting optimization problems are often tractable [6].
The aim of this report is to demonstrate techniques and approaches that use
tractable small cliques and the local dictionary to model complex structures and
texture.
Chapter 2: Notation

The synthesized view $V$ is a set of colors of all pixels in the output, in the form of a high-dimensional matrix. Suppose there are $M$ pixels, indexed by integers in raster-scan order, $V_i$ is the color of pixel $i$, defined in the color space $\mathbb{R}^3$, which corresponds to the RGB color model. Thus, $V_i$ is a vector of length three and rendered view matrix $V = [V_1, V_2, ..., V_M]^T$. A clique, or neighborhood is a set of pixel indexes. For a $W \times H$ image, for instance, pixel $i$, which is not in the boundary, might have a neighborhood $N_i = \{i, i + 1\}$, which is also indexed by an integer. The set of neighborhood colors can then be written as $V(N_i)$. Neighborhoods form a neighborhood system, that is, a set of all neighborhoods, $\{N_i\}_{j=1}^K$, supposing there are totally $K$ neighborhoods in the output image. There are three types of common neighborhood systems:

1. **Patch neighborhood system**: a patch neighborhood is written as $P_j$, which is a set of pixel indexes in the $5 \times 5$ patch, whose center is pixel $j$. Thus, the system, as a set of patch neighborhoods, can be written as $\{P_j\}_{j=1}^K$ and $V(P_j)$ is the $5 \times 5$ image patch in the output. If we ignore boundary bookkeeping, the number of neighborhoods equals to the number of pixels, that is, $K = M$.

2. **4-connected neighbor system**: a clique contains two pixels and can be written as $C_{4j}$ where $j$ is the clique’s integer index. The system, as a set of cliques, can then be written as $\{C_{4j}\}_{j=1}^K$. Again if we ignore boundary effects, each pixel is associated with two cliques, which might be the “east” clique $\{i, i + 1\}$ and the “north” clique $\{i, i - W\}$. Thus, the number of cliques is twice the number of pixels, that is, $K = 2M$.

3. **8-connected neighbor system**: again a clique consists of two pixels and can be written as $C_{8j}$ where $j$ is its index. The system is the set $\{C_{8j}\}_{j=1}^K$ in which each pixel is associated with four cliques if we ignore boundary cases.
For each pixel, besides the “east” and “north” cliques the same as above, there are two more cliques, the “south-east” clique \{i, i + W + 1\} and the “north-east” clique \{i, i - W + 1\}. Therefore, there are four times as many cliques as pixels, that is, \( K = 4M \).

Depth map \( Z \) should also be put into consideration, in which \( z_i \) refers to the depth of pixel \( i \) in the output view. The problem can then be placed into an energy minimization framework, and our aim is to obtain \( X \), which minimizes the energy function

\[
E(X) = E_{\text{unary}}(X) + E_{\text{prior}}(X)
\]

where \( X \) can be either \( V \) or \( Z \), or both of them.

The term \( E_{\text{unary}}(X) \) refers to unary energy which is a summation of local energy over all pixels, with the form

\[
E_{\text{unary}}(X) = \sum_{i=1}^{M} \psi_i(X_i)
\]

where the function \( \psi_i \) is a function of the input data. Note that function \( \psi_i \) varies with \( i \). An example in [6] is the measure of photo consistency, \( \psi_i: \mathbb{R}^3 \to \mathbb{R}^+ \).

Another example is the piecewise smooth regularization of depth, which takes a similar form in the unary term [6].

Similarly, \( E_{\text{prior}}(X) \) is prior energy, or clique energy, which sums the energy of all cliques, with the form

\[
E_{\text{prior}}(X) = \sum_{j=1}^{K} \mu_j[X(N_j)]
\]

where \( X(N_j) \) refers to color patch \( j \), or clique \( j \), and the function \( \mu_j \) can also be calculated as a function of the input. Again, the piecewise smooth regularization of depth is an example, in which 4-connected neighbor system is used, that is, \( N_j = C4_j \). In this case, the prior term has the form \( \mu_j(C4_j) = \mu_j(z_k, z_i) = \).
\[ \rho(|z_k - z_l|) \] where \( \rho(\cdot) \) is a robust kernel, for instance, the truncated quadratic function \( \rho(x) = \min\{x^2, 1\} \). As another example, in [6], \( \mu_j : \mathbb{R}^{5\times5\times3} \to \mathbb{R}^+ \) measures the squared distance from the \( 5 \times 5 \) patch \( X(N_j) = V(P_j) \) to the nearest patch in a global dictionary \( D \). Denoting the color patch as \( Y_j, \mu_j(Y_j) = \min_{T \in D} \|T - Y_j\| \). Note that in both examples above, \( \mu_j \) is independent of index \( j \) as you can see from their definitions, so we can drop the index \( j \) from \( \mu_j \), denoting it as \( \mu \) instead.

As we are focusing on view synthesis, in the following chapters, the energy function (1) will take the form
\[
E(V, z) = \sum_{i=1}^{M} \psi_i(V_i, z_i) + \sum_{j=1}^{K} \mu_j[V(N_j)]
\]
(4)
where \( \psi_i \) is a function of \( V_i \) and \( z_i \), and \( \mu_j \) is a function of \( j \) patch, \( V(N_j) \).

As is mentioned above, the size of cliques or patches is the parameter that has the strongest influence on how difficult the optimization can be solved. However, besides clique size, there is another significant factor, the discretization of \( V \) and \( z \). In the energy functions above, both \( V \) and \( z \) are continuous variables, which makes minimization under general priors intractable without discretization. However, it would be impractical to discretize \( V_i \), that is, the pixel color, directly, for there are 256 values (colors) in total for 8-bit RGB color graphics. Therefore, we discretize \( V_i \) by obtaining a small set of probable colors at each pixel, introduced in chapter 3.1. After the discretization, the energy function can be optimized over a small set of colors, rather than the whole color space.
Chapter 3: Unary Energy

Section 3.1: Form of Unary Energy

First, we need to define $\psi_i$ in unary energy term, which is a measure of photo consistency at pixel $i$. We use the definition from [6]:

$$
\psi_i(V_i, z_i) = \sum_{k=1}^{n} \rho(||C_i(k, z_i) - V_i||)
$$

where $n$ is the number of input images, and $C_i(k, z_i)$ is the color of the pixel in image $k$ that corresponds to pixel $i$ at the depth $z_i$ in the output view. The pixel color, $C_i(k, z_i)$, can be obtained by bilinear interpolation. $\rho(\cdot)$ is the robust kernel, which will be a truncated quadratic function

$$
\rho(t) = \min\{x^2, \tau^2\}
$$

where $\tau$ is a parameter set manually. With this kernel, we assume that pixel $i$ is generated either with an inlier process, in which pixels of the input image are measurements of the actual color with normally distributed noise, or an outlier process in which occlusion might exist in the input images.

Unlike multi-view reconstruction of depth, in view synthesis, only pixel colors are what we need. In this case, $\psi_i$ should be a function only of $V_i$. Therefore, we modify the definition of $\psi_i$ to eliminate $z_i$:

$$
\psi_i(V_i) = \min_{z_m < z_i < z_M} \psi_i(V_i, z_i)
= \min_{z_m < z_i < z_M} \sum_{k=1}^{n} \rho(||C_i(k, z_i) - V_i||)
$$

where $z_m$ and $z_M$ are lower and upper bounds for $z$ in the scene, respectively.

Section 3.2: Discretization of Color and Depth

As is discussed above, both color $V_i$ and depth $z_i$ are continuous variables in nature, making minimization intractable. For efficient minimization, the depth $z_i$ will be evenly discretized into 50 to 100 values between $z_m$ and $z_M$. Discretization of $V_i$
is a bit more complex. Following [6], we calculate the local minimum points of \( \psi_i(V_i) \) for each pixel \( i \), which, as the maximum points of pseudo-likelihood \( p(V_i) = e^{-\psi_i(V_i)} \), can be regarded as the most likely colors at that pixel. The set of these colors normally has a size of four to twenty.

In [6], a gradient descent method is used to find the minimum points. However, this method is slow and we have found that it cannot find all the minimum points in some occasions. Thus, we managed to find a faster, deterministic approach to search the likely colors at each pixel, given the likely colors at a particular depth \( z_i' \) (introduced in Section 3.3 below). To be specific, rather than directly minimizing the redefined \( \psi_i(V_i) \), which is difficult due to the outer \( \min \) operation, we first minimize the original \( \psi_i(V_i, z_i) \) over \( V_i \) given a \( z_i' \) and then decide whether the obtained \( V_i \) is the minimum point we need using the properties of \( \psi_i(V_i) \). From equation (7), we have

\[
\psi_i(V_i) = \min_{z_m < z_i < z_M} \psi_i(V_i, z_i) \tag{8}
\]

For any given \( V_i' \), if \( \psi_i(V_i') = \psi_i(V_i', z_i') \) then \( z_i' \) must satisfy

\[
z_i' = \arg\min_{z_m < z_i < z_M} \psi_i(V_i', z_i) \tag{9}
\]

where the symbol ‘ in the upper right corner of \( V_i, z_i \) means that they are concrete values instead of variables. It is obvious that equation (9) is equivalent to

\[
\psi_i(V_i', z_i) \leq \psi_i(V_i', z_i'), \quad \forall z_i \in (z_m, z_M) \tag{10}
\]

Therefore, we have proposed our approach below:

(i) For each \( z_i' \in (z_m, z_M) \), calculate all minimum points of \( \psi_i(V_i, z_i') \).

(ii) For each obtained minimum point \( V_i' \), reject it as a likely color for pixel \( i \) if

\[
\exists z_i \in (z_m, z_M), \quad \psi_i(V_i', z_i') > \psi_i(V_i', z_i) \tag{11}
\]

**Section 3.3: Calculating Likely Colors Given a Depth**

Given a depth \( z_i' \), the minimization of \( \psi_i(V_i', z_i') \) over pixel color \( V_i \) is dependent on the kernel \( \rho \) that we use. An example is the simple quadratic kernel, which can be
proved to have a single minimum point, the mean. This convenient property means that the minimum points in step (i) in the previous section can be calculated efficiently. Indeed, the effectiveness of our algorithm to find the likely colors \( V_i \) relies only on the capability to find them at arbitrary given depth. Thus convex kernels, e.g. absolute or quadratic kernels, would be optimal, for it generally has a single minimum point which can always be obtained with a non-linear optimizer.

As mentioned at the beginning of Section 3.1, we use the truncated quadratic function as the kernel. Although it is not convex and might have multiple minimum points, it can still be proved that the minimum points are the means of different sets of inlier pixels in the input. To find these means, we use a mean shift algorithm [7] which iterates the update function below with an initialization \( V_0 \):

\[
V_{m+1} = \frac{\sum_{k=1}^{n} c_i(k,z) h(\|C_i(k,z) - V_m\|)}{\sum_{k=1}^{n} h(\|C_i(k,z) - V_m\|^2)}
\]

\[
h(x) = \begin{cases} 
1, & x^2 < \tau^2 \\
0, & \text{otherwise}
\end{cases}
\]

where index \( i \) is dropped in \( V_i \) and \( z_i \) for brevity. Iterations terminate if \( V_{m+1} = V_m \), which is guaranteed to occur in finite steps. From the above two equations, we can find that the obtained \( V_m \) must be the mean of all such inliers \( C_i(k,z) \) that \( \|C_i(k,z) - V_m\|^2 < \tau^2 \). As our goal is to decrease the number of likely colors at each pixel in discretization, we only accept likely colors with at least two inliers, that is, \( \sum_{k=1}^{n} h(\|C_i(k,z) - V_m\|^2) \geq 2 \). In this way, the number of likely colors could be largely reduced because there may be many likely colors with only one inlier, showing little consistency between input images. Rejection of the correct minimum point only happens when it is visible in only one input image, which is a rare case and is acceptable considering the benefit of significant reduction of color numbers.

In order to find likely colors with at least two inliers, we start with the following initial estimates:
\[ V_0 = \frac{C_i(j, z) + C_i(k, z)}{2} \]  
\[ \text{for all } j, k \in \{1, 2, \ldots, n\} \text{ such that } j \neq k \text{ and} \]
\[ ||C_i(j, z) - C_i(k, z)|| < 2\tau \]

The initial estimates above do not guarantee to avoid results with only one inlier, but it is easy to calculate and performs well in practice.

Combining the algorithm above with the approach in Section 3.2, we can discretize \( V_i \) efficiently, and the color at each pixel is restricted to a small set of likely colors, denoted as \( L_i \). Note that each likely color \( V_i^l \) is associated with a depth \( z_i^l \). We can now turn to define the prior term arising from clique energy.
Chapter 4: Prior Energy

Section 4.1: Form of Prior Energy

Prior energy, also called clique energy, arises from texture prior. We use a non-parametric regularizer that looks up patch in nearest neighbors. It is similar to [6] while the clique size is changed from $5 \times 5$ to $2 \times 1$ and it uses a local dictionary similar to [8]. Suppose clique $j$ consists of two pixels, $s$ and $t$, function $\mu_j$ in prior energy term can be defined as

$$
\mu_j[V(N_j)] = \mu_j(V_s, V_t) = \min_{T \in D_j} ||T - (V_s, V_t)||^2
$$

(16)

where $(V_s, V_t)$ refers to clique $j$ and $D_j$ is the local dictionary for clique $j$, which is introduced in Section 4.2.

Section 4.2: Local dictionary

The local dictionary makes use of the epipolar lines. Each pixel in the output corresponds to an epipolar line segment in every input image. If clique $j$ consists of pixels $s$ and $t$, the local dictionary will be made of all $2 \times 1$ or $2 \times 1$ cliques in the input that are within a threshold distance $d$ of any of the epipolar lines associated with $s$ and $t$. Threshold $d$ is generally set between 0 to 3 pixels. Note that when $d$ is set to zero, only cliques that intersect an epipolar line are incorporated. In this case, $D_j$ can be written as

$$
D_j = \{(C_s(k, z), C_t(k, z))|\forall k, z\}
$$

(17)

where the definition of $C_s(k, z)$ and $C_t(k, z)$ is the same as Section 3.1, i.e. the color of the pixel in image $k$ that corresponds to the output pixel $s$ or $t$ at the depth $z$. Note that the correspondence between clique $j$ and $(C_s(k, z), C_t(k, z))$ only holds when the clique is fronto-parallel in the output view. Figure 3 below shows the creation of a local dictionary.
With the local dictionary above, the single pixel colors at occluded boundaries are not corrupted by training from either side of the boundary, producing correct reconstruction with the same texture as input. In textureless cases, the dictionary will promote piecewise smoothness, which shows consistency with the simple case of order 2 priors.

**Section 4.3: Summary and Minimization**

In chapter 4, we determine the form of $E_{\text{unary}}$, and for each pixel we discretize $V_i$ into a small set of likely colors, $L_i$. In Chapter 5, we determine and simplify the form of $E_{\text{prior}}$. Combining these two terms, we have

$$E(L) = \sum_{i=1}^{M} \psi_i(L_i) + \lambda \sum_{j=1}^{K} \mu[L(N_j)]$$

(18)

where $\lambda$ is a tuning parameter that controls the effect from the prior term. By minimizing equation (18) over $L = [L_1, ..., L_M]$, we can generate the new view.
Chapter 5: Evaluation

In the experiments, the new algorithm is implemented and tested on three publicly available image sequences: “Edmontosaurus”, “Plant & Toy” and “Monkey”. Besides visual assessment, a leave-one-out test is also conducted, in which one image is chosen for synthesis using the rest 8 input images in the sequence. Comparing the output view with the original image, we can evaluate the quality of rendering, measured by the difference between them.

Discretization of $V_i$, that is, calculating the likely colors, are enumerated beforehand, requiring 3 minutes for a 640×480 image, and figuring out all probable cliques takes another 3 minutes. By default, the tuning parameter $\lambda$ for the prior term is set to 1, and threshold $\tau$ on the kernel $\rho$ is set to 50, unless otherwise stated. We optimize the objective energy function with a publicly available implementation of the tree-reweighted message-passing algorithm [3]. Because each clique is associated with a local dictionary, as large as about 1GB of memory space is required, with an approximation of 1000 bytes per pixel. Although the optimizer we use does not guarantee to reach the global optimum, a lower bound is provided on the function. We found that our outputs were within 3% of the lower bounds, meaning that our results are very close to the global minimum. In our experiments, the following algorithms are compared:

1. **Maximum Likelihood:**
   
   It is optimized without the prior term, equivalent to $\lambda = 0$

2. **$2 \times 1$ cliques, local dictionary, 4-connected neighborhood system:**

   \[
   \{C_{4j}^K\}_{j=1}^K \text{ system is used with a local dictionary } D_1 \text{ defined in Section 4.2. It is minimized by means of TRW.}
   \]

3. **$5 \times 5$ patches, local dictionary [8]:**

   Patch neighborhood system $\{P_j\}_{j=1}^K$ is used and the local dictionary is determined based on Section 4.2. It is optimized with an approximation of
iterated conditional modes.

<table>
<thead>
<tr>
<th>Plant &amp; Toy: Leave-one-out test on</th>
<th>Maximum likelihood</th>
<th>Local 5 × 5</th>
<th>4 connected local 2 × 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rendered views</td>
<td>2 seconds</td>
<td>9 seconds</td>
<td>8 seconds</td>
</tr>
<tr>
<td>Difference from the original</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pixel error</td>
<td>15.67%</td>
<td>14.75%</td>
<td>14.42%</td>
</tr>
</tbody>
</table>

| Edmontosaurus: Synthesized view    |                    |             |                        |
| Rendered views                    | 2 seconds          | 10 seconds  | 7 seconds              |

Figure 4: Comparison of priors

The above is a table of images and relevant information about the results of experiments. The images are generated with algorithms described above in this chapter. The first row of images are the results of the leave-one-out test on image set “Plant & Toy”, with rendering time below each image. Differences from the original image are showed in the second row with a total percentage of pixel errors below each image. Rendered views of “Edmontosaurus” are shown in the third row with rendering time below each image, too.

The Figure 4 above shows the results of our experiments in terms of rendering time and quality. The fastest algorithm is the Maximum Likelihood algorithm, about 4 times faster than the one using 2 × 1 cliques with the local dictionary and a 4-
connected neighborhood system, which is the algorithm we’re discussing in previous chapters. The slowest algorithm is the one using $5 \times 5$ patches and the local dictionary, roughly about 10%-40% slower than our algorithm “4 connected local 2×1”. This result is consistent with our analysis, as the number of operations in the maximum likelihood method is the smallest due to the absence of prior terms. Also, larger patches require more time to compare, that’s why the algorithm using $5 \times 5$ patches is the slowest.

Besides, the accuracy, or quality, is also an important measurement, covered in Figure 4. Visual evaluation and the total percentage of pixel errors are two metrics we use in the evaluation. We can see that our algorithm “4 connected local 2×1” has the smallest percentage of pixel errors on “Plant & Toy”, and the best rendering quality on “Edmontosaurus” judged by visual assessment. Although pixel error percentages of the three algorithms are relatively close to each other, the differences in the rendered view of “Edmontosaurus” are visible. There is a crack in the middle and the lines are not so smooth when maximum likelihood or $5 \times 5$ patch method is used. When using our algorithm “4 connected local 2×1”, these errors do not occur. To summarize, both of the two measures above tend to favor our algorithm using small cliques and the local dictionary. The reason is that a stronger optimum close to the global minimum can be reached and the prior term is powerful enough to correct potential errors.
Chapter 6: Discussion

We have analyzed, implemented and tested the usage of small cliques for the energy priors in view rendering and synthesis. Our goal is to find effective, simplified formulation for both unary energy and prior energy which makes global minimization tractable. One of the main benefits is that obvious errors in images rendered by maximum likelihood can be corrected with global minimization. Rendering quality can be improved significantly if we reach a global minimum, rather than a local one. Another advantages is the fast speed due to precomputation of discretization and the local dictionary. With probable pixel colors and potential cliques ready, a strong optimum can be reached quickly.

It has also been shown that prior terms become more discriminative and efficient when using small cliques with the local dictionary. In this case, each clique in the output learns from the corresponding local regions instead of the whole image sets, which again increases the speed and improves the rendering quality.

We’ve mainly focused on texture priors, that is, priors over output pixel colors. It would be interesting to consider the depth priors, or even combine texture and depth priors to further improve the quality of rendering. However, troubles may occur when more values and variables are incorporated in the energy function. Rendering time will increase, more memory space is required, and a global optimum might not be reached. Simply, the depth priors are currently beyond our scope of work.

As for the experiments and evaluation, more assessment metrics might be proposed. Our current measurements are relatively limited and percentages of pixel errors cannot effectively represent the similarity between the rendered view and the original one. For example, we have no ideas how distinct or close an error pixel is from the corresponding correct pixel in the test image.
Chapter 7: Conclusion

In this report, we have explained the importance of optimization techniques and approaches to improve the quality of rendered image. The intractability of minimization over continuous variables led us to find effective ways to discretize pixel colors, and the difficulty of optimization with large patch priors prompts us to consider small cliques with the local dictionary. Although 2-pixel cliques are generally not powerful enough to correct rendering errors and model texture, we have managed to find a way of constructing effective local dictionary with small patches that largely overcomes the disadvantages. Moreover, we have demonstrated a simple, fast approach for calculating the likely pixel colors in the output, even if the kernel is non-convex.
References


INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

UnitID: uS835210

SURNAME: Wu

FIRST NAMES: Jie

PROJECT SUPERVISOR (may be external): Matt Adcock

COURSE SUPERVISOR (as RSCS academic):

COURSE CODE, TITLE AND UNIT: COMP8715, Computing Project, 12 units

SEMESTER [ ] S1 [x] S2 YEAR: 2016

PROJECT TITLE:

Light Field and Image-based Rendering

LEARNING OBJECTIVES:

1. Apply the student’s knowledge and implementation skills in the computer science to light field and image-based rendering.
2. Deepen his knowledge of computing principles and practice through undertaking the project.
3. Earn any specific knowledge and technical skills required by the topic, and apply them to project work.
4. Apply and deepen skills in oral and written communication, and apply these in a project context.
5. Learn relevant project-related skills, including project management, ethics in research, knowledge of relevant research, evaluation and production of project artefacts.

PROJECT DESCRIPTION:

Implement light field and image-based rendering algorithms for view synthesis

* Learn about and implement image-based rendering algorithms without priors.
* Implement image-based rendering algorithms with priors of order 2.
* Implement image-based rendering algorithms with texture priors.
* Implement image-based rendering algorithms with small cliques and local dictionary
* Test and compare different rendering algorithms
ASSESSMENT (as per course's project rules web page, with the differences noted below):

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: name style: software description, project background/motivation, design framework, architecture, code implementation and evaluations (e.g. research report, software description...)</td>
<td>60</td>
<td>October 28</td>
</tr>
<tr>
<td>Artefact: name kind: artefact/software, source code and datasets (e.g. software, user interface, robot...)</td>
<td>30</td>
<td>October 28</td>
</tr>
<tr>
<td>Presentation:</td>
<td>10</td>
<td>October 20-27</td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):

STUDENT DECLARATION: I agree to fulfil the above defined contract:

................................................................. ............................
Signature                                             Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.

................................................................. ............................
Signature                                             Date

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Course coordinator approval)

................................................................. ............................
Signature                                             Date

SECTION D (Projects coordinator approval)

................................................................. ............................
Signature                                             Date

Research School of Computer Science

Form updated Jun-12