(Draft – Comments Welcome)
On Synthesizing Large Images: Issues, Algorithms and Tools*

Chee K. Yap   Amiya Dingare   Ting-Jen Yen
Jian Tan
Department of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York University

July 14, 1998

Abstract

Many applications need images whose size can be gigabytes or terabytes in size. The large size alone raise some practical issues of manipulation and visualization. We study the basic task of synthesizing large images from a suitable set of subimages, focusing on the associated algorithmic issues. We design and implement an efficient algorithm for the basic problem of computing a "dilated rigid transformation" of an image. A suite of tools to facilitate the image synthesis task is described. These tools enable one to create large images using commodity equipment. We introduce a 'superfile' graphics format for representing large images.

1 Introduction

We are interested in the manipulation and visualization of very large-scale (raster) images. The images essentially have arbitrary sizes[1] and in many applications can easily reach gigabyte or terabyte sizes. Of course, such an image has an extent that cannot be fully shown on a computer screen, and may have considerably more resolution than the 100dpi resolution of computer monitors. Instead, viewing such an image requires "navigation" tools for zooming and panning (among other techniques). Such images can arise in a variety of ways:

*Supported by NSF Grant CCR-9619846.
[1]Naturally, limited by the file system capacity.

from computational simulation, from satellite photography or simply by scanning large analogue images. Museums, libraries, governmental agencies, educational institutions and commercial enterprises need such large images. Another obvious application area is in geographical information systems (GIS). In our research, we further wish to view such images across a thinwire connection.

There is a currently a wealth of commercial software for the visualization and manipulation of graphic images. Many of these systems are unable to deal with arbitrarily large images. Current cost to produce and view even moderate-scale images can be prohibitive. The Kodak Photo CD format, first introduced in 1990, did not receive widespread and rapid acceptance because of its need for special hardware and software. An example of synthesizing large images is the Ray Sterner of Johns Hopkins' Applied Physics Laboratory `http://fermi.jhuapl.edu/states/states.html` created the "Color Landform Atlas of the United States" by tiling together a collection of 5 degree tiles based on digital elevation model (DEM) data. See also Andrew Birrell's mosaic "Relief Map of the United States" created from these images `http://www.research.digital.com:80/SRC/personal/Andrew_Birrell/reliefMaps/` One goal of the present work is to investigate some of the algorithmic issues associated with such tasks. We also want to build software tools aimed at letting users use off-the-shelf (commodity) equipment to synthesize, visualize and manipulate arbitrarily large images. One motivation of our take on this problem is the fact that the scanning of analogue images is widely available today. However, scanning surfaces (say 17" $\times$ 14") are smaller than the size of images we are interested in. So we expect to make multiple scans. Piecing together these scans is complicated by several factors: successive scans of an image do not perfectly align, the physical conditions such as temperature change, paper warp, etc of each scan are different. But given that we have assembled a collection of partially overlapping images, we then address the general problem of synthesizing a superimage this collection. Such a tool would be extremely useful for organizations as well as individuals (see the links in Sterner's page for sample applications of his map). Another example of this "collage" approach to big images arises when we want a panaromic image of a large scene. It is preferable to take a series of camera shots using a commodity camera, rather than using some specialized panaromic (or even omnidirectional) camera. Sometimes even specialized equipment may not help: if the entire scene is always partially occluded, regardless of the camera viewpoint, then the collage approach may be the only solution. This paper does not address the 3 dimensional synthesis problem needed in this example.

What are some of the issues in the use of large images?

- Conventional images are stored in a one-per-file basis. This is unwieldy for manipulating large images. We introduce a superfile format for representing an image as a collection of files. The image need not be rectangular. The format supports multiresolution representation.

- Most software and file formats require the entire image to be read before they can be visualized and manipulated. The image size is then limited by the amount of virtual memory of the computer. This requirement will be removed in our work.

- Algorithmic issues in this area of large images needs to be systematically studied. The payoff is big since processing large images is computationally expensive.

- The synthesis of large images is a nontrivial multi-step process. We design a suite of tools to facilitate this process. These tools are publicly available from our website, `http://cs.nyu.edu/visual/tools`.

We already mentioned some applications for large images. But our immediate motivation is the construction of a web gallery focusing on the visualization of large scale images. The visualization aspects will be the subject of a future paper. The use of foveation and multiresolution techniques are central in our approach [2, 3].

2 Basic Algorithms Problems

We identify three basic computational problems in synthesizing large images:

(P1) Registration Problem: Given two (partially) overlapping images I and J, to compute a transformation τ from the coordinate system of J to the coordinate system of I so that the overlapping parts of I and J are consistent.

(P2) Transformation Problem: Given an image I and a transformation τ, to compute the new image $\tau(I)$.

(P3) Merging Problem: Given a collection of partially overlapping images, to merge them into a unified single image.

Problem (P1) belongs to a class of "registration problems" that is extensively studied in the image processing and computer vision literature (e.g., [6, 1]), and in various disciplines such as astronomy, computer tomography, etc. The issues vary with the nature of the images. In problem (P1), we assume that geometric distortion is the main issue. As formulated in (P1), the algorithm has to compute τ from I and J alone – we may call this the **automatic** registration problem. Unfortunately, the automatic problem is ill-defined for most inputs. Hence any automatic algorithm must use heuristics whose performance depends on additional assumptions about the nature of the images I and J. The **semiautomatic version** of (P1) is one in which the algorithm is given some additional inputs, amounting to hints about τ. A human user can provide these hints, and we can focus on designing tools to facilitate this human input (see Section 5). Clearly, there is a spectrum of problems, from fully automatic, through semi-automatic to completely "explicit" registrations.

The nature of the transformation τ is of great interest. The simplest case is where τ is just a linear transformation. We can further restrict this – for instance to Euclidean transformations. Unfortunately in practice, linear transformations alone is insufficient because of local distortions. The other extreme is to assume that every point in the source images are distorted and we need a complete "point-to-point" description of τ. We shall examine other intermediate formulations of τ.

Once τ is given, we can address the transformation problem (P2) is clearly very fundamental. There are various solutions for (P2) based on sampling theory (e.g., [4, Chapter 17]) In Section 7, we address some interesting issues in this problem. We also give an implementation of an efficient, practical algorithm.

Graphics File Formats. Graphic file formats is an area that is mostly technology-driven rather than having inherent intellectual interest. There is a remarkable variety of such formats and several books survey them (e.g., [7]). As computing technology continues to evolve, there is no end in sight to the invention of such file formats. We briefly review some formats related to this work.

- FlashPix: this is a relatively new multiresolution format in which the image is stored as a series of independent "tiles" at different resolutions. This allows the image to be viewed in different output devices. Each tile is 64x64. The resolution hierarchy can be full (down to a single tile) or flat (the highest resolution only). Viewing parameters include translation, rotation, scale, sharpness, crop, color and contrast adjustment. Color space include monochrome, Photo YCC, FlashPix RGB, RGB uncalibrated, unspecified. Region of interest includes opacity channel overlay. Image description data include file source, image content, camera data, picture capture data, film data, document scan data, scanner data, unicode character representation. JPEG compresion is an option. Although the format may need 33% more storage because of resolution hierarchy, it need much less RAM for viewing (20% of `TIFF`), and faster time for modification and storing a revision. Ultimately, it requires a less powerful computer to manipulate and visualize such images. A source of information for this format is `http://www.kodak.com/US/en/digital/dlc/book2/`.

- Image Pac file or PCD format, from Kodak Photo CD System: this is a proprietary file format that uses a new "Photo YCC" color encoding scheme to support multiple display resolutions, fast access to video resolution, device independent color encoding. Thumbnail images is included. There are 6 resolutions: Base resolution is 512x768. The lowest is Base/16=128x192. The highest is Base*64=4096x6144 (professional format). The scanner needs to operate at resolution of 2048x3072 pixels for each color to achieve a near-photographic quality image – a single 35mm image would need 18 MB of storage. With a CD disc capacity of 540 MB, this allows only 30 images to be stored in a disc. But using compression, a Image Pac file ranges from 4.5 to 6.5 MB, and allows about 100 images/Photo CD.

- Another commercial proprietory file format is called `MrSID` ("Multiresolution Seamless Image Database") from Lizardtech, Inc., `http://www.lizardtech.com/`. The file size is unlimited and is billed as a image compression and visualization technology based on wavelets. A lossy compression ratio of 22:1 is typical.

- `TIFF` (Tag Image File Format) was developed by Aldus Corporation in 1986 for scanner images, frame grabbers and paint or photo retouching programs. It is one of the most versatile and widely supported. `TIFF/EP` is an adaptation for the electronic photography.

- Chris Vickery (vickery@qcunix1.qc.edu, Professor at CUNY Queens, met at Industry Evening 1/14/99, also see email) pointed out a product that stitches photos together is called PhotoVista. You can download a trial copy from www.livepicture.com if you are interested in seeing what it can do. I think it can do only 360 degrees horizontally.

3 Preliminaries

We recall some basic facts about (linear) transformations of Euclidean space $\mathbb{R}^n$. We may restrict ourselves to $\mathbb{R}^2$. A tranformation $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ is called **rigid** if it preserves Euclidean distances and orientation (i.e., it does not do a mirror flip). Such transformations are equivalent to a rotation and a translation. To be specific, we write $\tau = [\theta, \alpha, \beta]$ if τ amounts to a counter-clockwise rotation through angle θ followed by a translation by (α, β). We need a slightly larger class of transformation, which we may call **dilated rigid transformations**. Such a transformation τ amounts to combining a rigid transformation with a dilation (scaling) by some factor $d > 0$. We then write $\tau = [d, \theta, \alpha, \beta]$ for the operation of dilation by d, followed by operation of $[\theta, \alpha, \beta]$. In this extended notation, $[1, \theta, \alpha, \beta]$ is the same as $[\theta, \alpha, \beta]$.

A **(greyscale) image** is a function $J : \mathbb{R}^2 \to \mathbb{R} \cup \{\uparrow\}$ with domain $\{p \in \mathbb{R}^2 : J(p) \neq \uparrow\}$. We say J is a **connected image** if its domain is connected. To make J a color image, we just have to change its range to $\mathbb{R}^3 \cup \{\uparrow\}$. In this paper, for simplicity, we assume our images are greyscale. Our results extend to color images in the obvious ways.

If the domain of J is a subset of $\mathbb{Z}^2$, then we call J a **discrete image**. For any discrete image J, we can define a related continuous image $\widetilde{J}$ as follows:

$$\widetilde{J}(p) := J(\lfloor p \rfloor)$$

where $\lfloor p \rfloor = (\lfloor p_x \rfloor, \lfloor p_y \rfloor)$, with $p = (p_x, p_y) \in \mathbb{R}^2$. Note that the definition of $\widetilde{J}$ is still meaningful even when J is not a discrete image, In this paper, *we assume that every image J (discrete or not) satisfies the property that $\widetilde{J}$ is connected.*

We say images I and J are **consistent** if $I(p) = J(p)$ whenever p is in the domains of both I and J. An image I is a **subimage** of J if they are consistent and further, the domain of I is contained in the domain of J. Alternatively, J is a **superimage** of I. Two continuous images I, J are said to **overlap** if the intersection of their domains contains two or more points.

4 The Idealized Image Synthesis Problem

Suppose we have a collection $K = \{I_j : j = 1, 2, \ldots, n\}$ of subimages from a single (unknown) superimage I. Our goal is to synthesize I. It is useful to first solve the problem in an idealized world where each I_j just dilated rigid transformations of some subimage of I.

The subimages I_j form a "cover" of I. For example, in a traditional atlas, we often need several images (spread over several pages) to cover a large geographical region I. Thus, if we scan each page in this atlas, we have such a collection K. Figure 1 indicates a cover K for the continent of africa using 4 subimages (numbered 58, 60, 62, 64). These subimages have a common physical dimension (covering two pages of the atlas). Hence it is clear from the figure that they are at different scales, and rigid transformations alone is insufficient for this reconstruction.

Figure 1: No Figure Yet: A cover for South America, from an atlas

Let us formalize the cover notion. We say a collection of images $K = \{I_i : i = 1, 2, \ldots, n\}$ is a **cover** of the image I if:

- There is a collection of dilated rigid transformations $\{\tau_i : i = 1, \ldots, n\}$ such that $I = \cup_{i=1}^{n} \tau_i(I_i)$.
- Define the **overlap graph** G be the undirected graph with vertex set $V = \{1, \ldots, n\}$ and edge set E such that $(i, j) \in E$ iff $\tau_i(I_i)$ and $\tau_j(I_j)$ overlap. We require that G be a connected graph.

The **image synthesis problem** is therefore: *given a cover K for an unknown image I, to reconstruct I.* We may assume that K has at least one solution. Note that I is underdetermined up to some dilated rigid transformation. We may remove this ambiguity, for instance, by assuming that τ_1 is the identity transformation. Even with this restriction, the problem may still have multiple solutions. For example, imagine that each subimage is uniformly white! However, for "realistic" input instances, there is almost surely a unique solution.

We will solve the image synthesis problem in the following three steps:

(S1) For each i, compute the transformation τ_i.

(S2) For each i, compute the transformed image $\tau_i(I_i)$.

(S3) Merge the images $\tau_1(I_1), \ldots, \tau_n(I_n)$ to form I.

So far we have described the mathematical setting. In practice, many things can mar this ideal setting: (a) the images are discrete, (b) the images are at different orientation – which affects our ability to carry out steps (S2), (c) the subimages are not really dilated rigid transformation of some ideal superimage (so strictly speaking there are no solutions). Thus, step (S3) is a triviality in the ideal setting but in practice it is less clear how to get around inconsistent data. The less than ideal situation also justifies our preference for the semiautomatic approach to step (S1).

5 A Tool for Generating Registration Data

We address step (S1) in the image synthesis problem. It is intuitively clear that the registration problem (P1) is the key to (S1).

Let us define a **correspondence** to be pair of points (p, q) where $p, q \in \mathbb{R}^2$. We usually write $p \to q$ instead of (p, q). Suppose

$$T = (p_1 \to q_1, p_2 \to q_2)$$

is a pair of correspondences such that $p_1 \neq p_2$ and $q_1 \neq q_2$. Then, as we will show below, there is a unique a dilated rigid transformation τ such that $\tau(p_i) = q_i$ for $i = 1, 2$.

Theoretically, only two correspondences are sufficient to establish the transformation. In practice, because of various errors, it is more robust to specify more than two correspondences and to compute some transformation that is some kind of average of the transformations implied by these correspondenses. We feel 3 or 4 correspondences $p_i \to q_i$ with the p_i's well distributed is reasonable. This issue should be a subject for further study.

We provide a GUI interface tool called `TIE` which lets a user specify a sequence of (at least two) correspondences. Given two images I and J, `TIE` will display the two images in two separate windows. (The software assumes that the images I, J can fit into the virtual memory of the computer system.) Furthermore, we assume I and J overlap when viewed as transformed subimages of a superimage. The user can next **select** (by a mouse click at the desired position) any number of correspondences (but at least two) between the two images. That is, the user selects a point p in I and another point q in J to specify the correspondence $p- > q$. Usually, p and q are positions of interesting features in the overlap areas of the subimages (say, the corner of a building or the left eye in a face). The `TIE` software outputs the chosen set of correspondences in a file.

To solve (S1), recall that the overlap graph G is connected. Then we require that the human user provide us with a spanning tree S of G. In the Africa example figure 1, the overlap graph G and its spanning tree S is illustrated in figure 2.

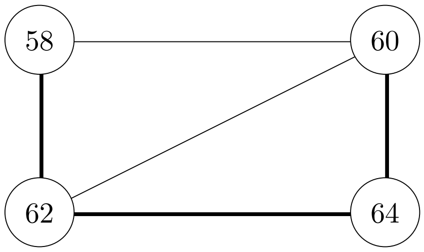

Figure 2: Overlap graph for the Africa cover, with a spanning tree (in thick lines).

We further direct the edges of the spanning tree so that there is a unique node with outdegree 0, the **root** of S. Now S can specified by listing all its directed edges:

$$S = (58- > 62, 62- > 64, 60- > 64)$$

The root of S is here 64. The user can specify this information in the catalog file as follows:

```
Source Files
#-----------
map_58.tif
map_60.tif
map_62.tif
map_64.tif

Spanning Tree   # list of edges with correspondences
#------------
58->60: (145,112)-->(24,264),(191,23)-->(70,176)
...etc...
```

For each edge of the spaning tree, we list an associated set of correspondences obtained using the `TIE` software.

We call the collection $\{I_i\}$, together with the registration data, or **correspondes**.

6 Computing Transformations from Correspondences

A dilated rigid $\tau = [d, \theta, \alpha, \beta]$, can be represented by the following 3×3 matrix:

$$M = M_\tau = \begin{pmatrix} c & -s & \alpha \\ s & c & \beta \\ 0 & 0 & 1 \end{pmatrix}$$

where $c = d\cos\theta$ and $s = d\sin\theta$. To apply transformation τ to a point (x, y) to get a point (x', y'), we can perform the matrix-vector multiplication

$$\begin{pmatrix} c & -s & \alpha \\ s & c & \beta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

From this it is clear that transformations can be composed by multiplying together their corresponding matrices.

Given a pair of correspondences, $p \to p'$ and $q \to q'$, let us now determine the transformation τ such that $\tau(p) = p'$ and $\tau(q) = q'$. Suppose the matrix of τ is given by M above. Then we have

$$\begin{pmatrix} c & -s & \alpha \\ s & c & \beta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \begin{pmatrix} p'_x \\ p'_y \\ 1 \end{pmatrix}$$

and

$$\begin{pmatrix} c & -s & \alpha \\ s & c & \beta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} q_x \\ q_y \\ 1 \end{pmatrix} = \begin{pmatrix} q'_x \\ q'_y \\ 1 \end{pmatrix}.$$

These two equations are succinctly written as

$$\begin{pmatrix} p_x & -p_y & 1 & 0 \\ p_y & p_x & 0 & 1 \\ q_x & -q_y & 1 & 0 \\ q_y & q_x & 0 & 1 \end{pmatrix} \begin{pmatrix} c \\ s \\ \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} p'_x \\ p'_y \\ q'_x \\ q'_y \end{pmatrix}.$$

From this equation, we may verify that if $p' = q'$ then

$$\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} p_x - q_x \\ p_y - q_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

This implies $p = q$. Hence, if we assume $p \neq q$ (see below) then we also have $p' \neq q'$. Let C denote the matrix

$$C := \begin{pmatrix} p_x & -p_y & 1 & 0 \\ p_y & p_x & 0 & 1 \\ q_x & -q_y & 1 & 0 \\ q_y & q_x & 0 & 1 \end{pmatrix}.$$

The determinant of C is

$$\Delta := p_x^2 - 2p_xq_x + p_y^2 - 2p_yq_y + q_x^2 + q_y^2.$$

Note that $\Delta = 0$ iff $p = q$. Hence, assuming $p \neq q$, we can solve for c, s, α, β by inverting the matrix C:

$$\begin{pmatrix} c \\ s \\ \alpha \\ \beta \end{pmatrix} = C^{-1} \begin{pmatrix} p'_x \\ p'_y \\ q'_x \\ q'_y \end{pmatrix}$$

where

$$C^{-1} =$$

$$\Delta^{-1} \cdot \begin{pmatrix} (p_x - q_x) & (p_y - q_y) & (-p_x + q_x) & (-p_y + q_y) \\ (-p_y + q_y) & (p_x - q_x) & (p_y - q_y) & (-p_x + q_x) \\ \begin{matrix}(-p_y \cdot q_y - p_x \cdot q_x \\ +q_x^2 + q_y^2)\end{matrix} & \begin{matrix}(-p_y \cdot q_x \\ +q_y \cdot p_x)\end{matrix} & \begin{matrix}(p_x^2 - p_x \cdot q_x \\ +p_y^2 - p_y \cdot q_y)\end{matrix} & (p_y \cdot q_x - q_y \cdot p_x) \\ \begin{matrix}(p_y \cdot q_x \\ -q_y \cdot p_x)\end{matrix} & \begin{matrix}(-p_y \cdot q_y - p_x \cdot q_x \\ +q_x^2 + q_y^2)\end{matrix} & \begin{matrix}(-p_y \cdot q_x \\ +q_y \cdot p_x)\end{matrix} & \begin{matrix}(p_x^2 - p_x \cdot q_x \\ +p_y^2 - p_y \cdot q_y)\end{matrix} \end{pmatrix}.$$

Note that given c and s, we can recover d and θ since $d = \sqrt{c^2 + s^2}$.

7 Rigid Transformations of Digital Images

Suppose we have a rigid transformation τ which we wish to apply it to a digital image I. This section discuss some algorithms for computing a new image I' that is approximately $\tau(I)$. We can view a discrete image $I : \mathbb{R}^2 \to \mathbb{R} \cup \{\uparrow\}$ as a collection of pixels **pixel** p represented as a triple $(x, y; I(x, y))$ where (x, y) is in the domain of I. The **fundamental cell** F is the square region $[0, 1) \times [0, 1)$. The **cell** corresponding to $(x, y) \in \mathbb{Z}^2$ is the region $F_{x,y} := F + (x, y) = \{(x+u, y+v) : (u, v) \in F\}$.

7.1 Algorithm A_0

The simplest solution is as follows: for each pixel $p = (x, y; c)$ in I, we create a new pixel $p' = (x', y'; c)$ where $(x', y') = \lfloor \tau(x, y) \rfloor$. We say that cell (x, y) in I **contributes** to cell (x', y') in I'. There are two details that must be addressed.

(i) First, we may have two grid points $(x_1, y_1) \neq (x_2, y_2)$ such that $(x', y') = \lfloor \tau(x_1, y_1) \rfloor = \lfloor \tau(x_2, y_2) \rfloor$. One solution to this problem is to assume that the color at (x', y') is the average of the values that map to (x', y'). This complicates the simplistic algorithm above because we now need to maintain a counter with each cell of I'. Hence, for simplicity, we just replace the previous color value of (x', y') by the most recent contributed color.

(ii) We may have a cell (x', y') in I' that has no contribution from any cell of I, even though all the neighbors of (x', y') do have contributions. Let us assume that cells with no contributions receive some default color (black in our examples).

Here is an image and its transformation under the algorithm described so far:

Figure 3. Florida: (a) Original, (b) rotated with A_0, (c) refined

The most obvious problem with this problem is that our solution to (ii) above leaves rather obvious "holes". We first prove a lemma to show that such holes are basically isolated cells.

Lemma 1 *Let $B = [0, 2) \times [0, 1)$. For any rigid transformation $\tau = [\theta, a, b]$, $\tau(B)$ contains at least one grid point.*

Proof. It is not hard to see that for the purposes of establishing this lemma, it is sufficient to assume

$$0 < \theta \leq \pi/4, \qquad 0 \leq a, b < 1.$$

It should be clear what is meant by the 'top', 'bottom', 'left' and 'right' edges of B. Similarly, the 'top', etc, edges of $\tau(B)$ refers to the transform 'top', etc, edges of B. Let L be the vertical line $\{x = 1\}$.

Suppose $\tau(0,1) = (\alpha, \alpha')$ and $\tau(1,2) = (\beta, \beta')$. If $\beta \geq 1$ then we see that the top and bottom edges of $\tau(B)$ intersects L. Hence $\tau(B) \cap L$ is a segment of length > 1. This means that $\tau(B) \cap L$ contains a grid point.

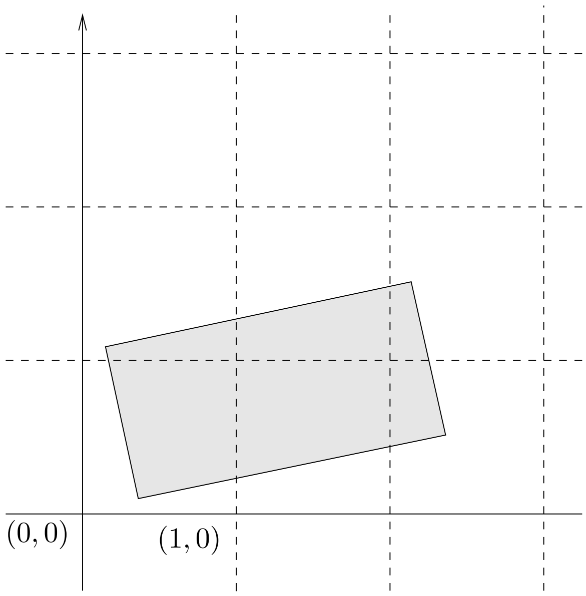

Figure 4: Transformed rectangle $\tau(B)$.

Hence assume $\beta < 1$. Next, observe that if the point $(a, b) = \tau(0,0)$ lies on or below the line $\{x = y\}$ then $\tau(B)$ contains the grid point $(1,1)$. So assume (a, b) lies above the line $\{x = y\}$.

Let L' be the line $\{x = y - 1\}$. We first claim that the lower edge of $\tau(B)$ lies below L'. This is clear because $\tau(0,0) = (a, b)$ lies below L' and $\theta \leq \pi/4$. Next we claim that the upper edge of $\tau(B)$ lies above the line L'. Again, this follows from the fact that $\tau(2,1)$, the upper right corner of $\tau(B)$, lies above the line L'. This is easy to see because of the assumption $\beta < 1$. From these two claims, it follows that $\tau(B) \cap L'$ has length at least 2. But every interval in L' of length more than $\sqrt{2}$ contains a grid point. This concludes our proof. **Q.E.D.**

Assuming I is a nice shape such as a rectangle, this means that except at the boundary of I', no two adjacent cells of I' can have no contribution. In view of this lemma, we propose to make another pass over the image I' so that at isolated hole (x, y) of I' we give it the average color of its four neighbors. This is our final algorithm A_0, and the resulting image I' based on the same input as above is show here:

Figure 5: No Figure Yet: Results from Algorithm A_0

7.2 Another method of filling holes

Another method of avoiding holds is to split each cell (i, j) of I into four "half-cells". More precisely, let $H = [0, \frac{1}{2}) \times [0, \frac{1}{2})$ be the 'half-fundamental region'. A point $(x, y) \in (\frac{1}{2}\mathbb{Z})^2$ represents the **half-cell** $H + (x, y)$. Thus the original cell (i, j) is a union of the four half-cells represented by

$$(i, j), (i + \frac{1}{2}, j), (i, j + \frac{1}{2}), (i + \frac{1}{2}, j + \frac{1}{2}).$$

A half-pixel is $(x, y; c)$ where (x, y) represents a half-cell. For each half-pixel $(x, y; c)$, we again computes its contribution to each cell in I' as follows: the value $c/4$ is added to the cell $(x', y') = \lfloor \tau(x, y) \rfloor$ in I'.

Note that this approach, unlike the previous one of averaging, can lead to color saturation (as when five or more half-cells contribute to one cell of I'). This approach does not leave any holes in I'. In fact, as an immediate corollary of the previous lemma, we have:

Corollary 2 *very cell in I' receives contribution from at least two half-cells of I.*

We have not implemented this algorithm.

7.3 A Weighted Contribution Algorithm

We describe a more sophisticated algorithm for computing the color of each cell in I'. For each cell (i, j) in I, we say its weighted contribution is the function $C_{i,j} : \mathbb{Z}^2 \to \mathbb{R}$ where $C_{i,j}(u, v)$ is the area of $(F_{u,v} \cap \tau(F_{i,j}))$. Thus, $C_{i,j}(u, v) = 0$ means that cell (i, j) in I does not contribute to cell (u, v) in I'.

Lemma 3 *At most two corners of $\tau(F)$ can lie in one common cell.*

Proof. Otherwise, a diagonal of $\tau(F)$ lies in that cell, and $\tau(F)$ coincides with that cell. **Q.E.D.**

Lemma 4 *If two corners of $\tau(F)$ lie in a common cell, then $\tau(F) \cap \mathbb{Z}^2$ is non-empty.*

Proof. Without loss of generality, let the bottom edge of $\tau(F)$ lie in F, and assume $\tau = [\theta, a, b]$ where $0 < \theta \leq \pi/4$. It is sufficient to prove that $(0,1) \in \tau(F)$. First we slide $\tau(F)$ in the negative y-direction until the bottom left corner of $\tau(F)$ touches the bottom edge of F. Call this position $\tau'(F)$.

Next, slide $\tau'(F)$ in the positive x-direction until the bottom right corner of $\tau'(F)$ touches the right edge of F. Call this position $\tau''(F)$. Clearly: (a) the left edge of $\tau''(F)$ intersects the left edge of F, and (b) the right edge of $\tau''(F)$ intersects the top edge of F.

But in fact, (a) and (b) is also true for $\tau(F)$ itself!

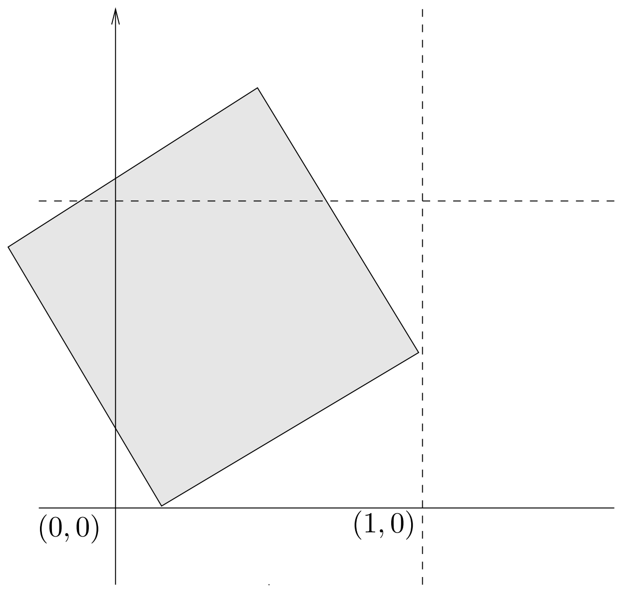

Figure 6: Two corners of $\tau(F)$ in F.

It is then enough to show that the top edge of $\tau(F)$ intersects the Y-axis at $(0,h)$ with $h > 1$. For, by the convexity of $\tau''(F)$ we can conclude that $(0,1) \in \tau''(F)$.

Here we use the following ploy: first assume that $(a,b) = (0,0)$. Then the result is true. Note that $\tau(0,1)$ lies in $F_{-1,0}$ and in fact,

$$\tau(0,1) = (-u, 1-v), u > v > 0.$$

But observe that $\tau(1,0) = (1-v, u)$. Next, if $a \neq 0$, this means that $a < v$ (otherwise, $\tau(1,0)$ would not be inside F). On the other hand, if $a = u$, then

the top edge of $\tau(F)$ would just touch $(0,1)$. Hence for $a < u$, we have $h > 1$.
PICTURE!

Q.E.D.

Corollary 5 *If $\tau(F) \cap \mathbb{Z}^2$ is empty, then each corner of $\tau(F)$ lies in a distinct cell.*

Lemma 6 *Each cell can contribute to 6 target cells, but no more.*

Proof. To see that a cell can contribute to 6 cells, let τ be a pure rotation by a sufficiently small angle θ. Then $\tau(F)$ is seen to intersect the four cells

$$F_{0,0}, F_{-1,0}, F_{-1,1}, F_{0,1}.$$

But the bottom left corner of $\tau(F)$ is at the origin. If we translate $\tau(F)$ slightly to the south-west, we see that $F_{-1,-1}$ and $F_{0,-1}$ would also be intersected.

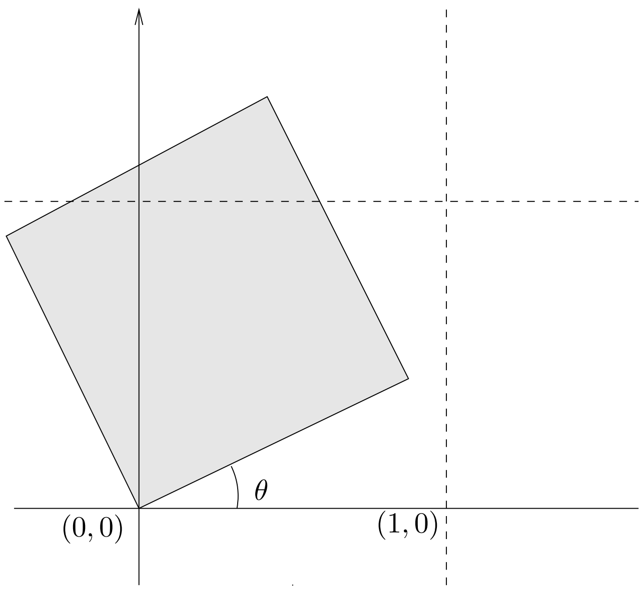

Figure 7: Contribution to Six Cells.

To see that we cannot contribute to more than 6 target cells, suppose $\tau(F) \cap \mathbb{Z}^2$ is empty. In this case, we claim that $\tau(F)$ contributes to at most 5 target cells. First, each corner of $\tau(F)$ must lie in distinct cells.

Now suppose $\tau(F)$

Q.E.D.

8 Dilated Rigid Transformations

The weighted contribution algorithm clearly can be generalized to allow dilations as well as rigid transformations. Conceptually, this is rather simple. But it is rather nontrivial to implement efficiently. We will briefly describe our implementation of this algorithm:

- We want to process the input image by rows. So we do not need to have the whole input image in main memory.
- We assume that the output image will be decomposed into a collection of tiles. For each input row of pixels, the collection of tiles that will be affected by this row will be maintained in main memory. Note: tiles are by assumption relatively small, but we further assume that we can afford to have all these tiles in memory.
- For efficiency, we do not compute the transformation of each grid point in the source image. Rather, we use linear interpolation to compute these points from the four corners of the source image.
- To process an input pixel P, we compute its contribution to each pixel column in the output image. The number of such columns is proportional to the dilation factor d.
- To begin processing of P, we first compute all the intersection points between its transformed boundary with the grid lines of the target image. These intersection points are organized as two lists, CWISE_LIST and CCWISE_LIST (sharing their starting and ending points). The starting points of these lists is the leftmost point of the transformed P – the leftmost point is usually unique but if not, we choose the lowest leftmost point. Similarly, the ending points of these lists are the rightmost point of the transformed P (we choose the highest rightmost point if this is not unique). Using these two lists we compute the area of overlap between the transformed P with each an output pixel (column by column).
- We need to handle special cases for the output pixel columns: a "leftcolumn" is one that contain the leftmost point. a "rightcolumn" is one that contain the leftmost point. Sometimes, the leftcolumn and the rightcolumn may coincide.

9 Local Transformations

In practice, there will be local distortions and no single (global) linear transformation for an image will suffice. The approach for solving this problem in the freely distributed USGS Mini Image Processing System (MIPS) [5, 8, 9] is to construct a grid of correspondences from the the “slave” to the “master” images, and to interpolate the correspondences using some weighted interpolation function. Although the grid in MIPS can be adaptive, it is still relatively dense and regular.

Let us outline an approach that is slightly less restrictive. Suppose we are given a collection of correspondences between a source image and some target image. Let $S \subseteq \mathbb{R}^2$ be a finite set of points in the source image and let $\mu : S \to \mathbb{R}^2$ represent this correspondences.

One interesting approach is to use the **order-k Voronoi diagrams** of the set S. Let $V_k(S)$ denote the order-k Voronoi diagram of S.

10 A SuperFile Format

We introduce a new file format, simply called the `BIG` format. A `BIG` file is not really a single file, but a collection of files that are stored in a single file directory. For this reason, it seems more accurate to call `BIG` a “superfile format”.

The names and formats of `BIG` files follow the following convention: There is a **catalog file** named $XXX.big$ and any number of **data files** and some auxilliary files. The data file can be stored in any of the standard graphics file formats. For the present work, we choose to use `TIFF` format. Hence, each data file has a name of the form

$$XXX \pm M \pm N.tif$$

where XXX is any user-chosen name which may not contain the characters ‘.’, ‘_’, ‘+’ or ‘$-$’). Here M and N are unsigned integers (that must not begin with a ‘0’ unless the integer is zero). The ‘$\pm$’ symbols represents either ‘+’ or ‘$-$’ (except that we do not allow the subsequence -0). If $XXX = map$, then $map+0+0.tif$, $map-19+2.tif$ and $map+1-11.tif$ are valid data file names but $map+0-0.tif$ and $map+01+2.tif$ are not. The image in each data file represents a “tile”, and the entire image is formed by arranging the tiles in a (M, N)-grid, as suggested by the data file names. There is no requirement that the tiles arrange into a rectangle or even cover a connected region of the plane. The collection of XXX-files constitute a superfile. All the XXX-files must be stored in one directory, but it is possible to have more than one superfile in one directory.

The catalog file is an ascii file. Here is an example:

```
Parameters
# ----------
nrows=128       # the number rows per tile (must be power of 2)
ncols=128       # the number columns per tile (must be power of 2)
xmin=0
xmax=2000
```

```
ymin=-3000
ymax=4000
                # So the nominal dimensions of the image is 7000x1000
                #   (there are no restrictions on these dimensions)
                # Note that the tiles need not fully cover the
                #   every pixel in these 7000x1000 area.  If a pixel is
                #   not covered by a tile, it is given a default value.
```

The character '#' indicates a comment: it and the rest of the line is ignored. The key words "Parameters", "nrows", etc., and the "=" symbols are required. Actually, this example shows only the minimal information required. Below we will describe other information that can be put in the catalog file.

Our superfile format is augmented into a multiresolution hierarchy as follows: the above data files constitute the 0th level of the hierarchy. For $k \geq 1$, the tiles in the kth level hierarchy are named

$$XXX\_k \pm M \pm N.tif.$$

Thus $map_2 + 97 + 21.tif$ is a tile in the 2nd level. We will describe this more fully in the sequel paper.

11 Conclusions

Our theoretical contribution in this work is to isolate the basic task of synthesizing an image from a cover of subimages and to provide efficient algorithms for the problem of computing a dilated rigid transformation of an image. Our practical contribution is to introduce a simple superfile graphics format and to provide a suite of tools for synthesizing images in this format.

The algorithmic investigations of the image synthesis problem can clearly be extended in many ways. For instance, in our introduction, we noted the 3 dimensional version of this problem. Another direction is to move closer to the image registration literature, and study algorithmically the problem of automatically registration for overlapping images.

References

[1] ??? A survey of image registration techniques. *ACM Comp.Surveys*, 24(4), 1992.

[2] E. Chang, C. Yap, and T.-J. Yen. Realtime visualization of large images over a thinwire. In *IEEE Visualization '97 (Late Breaking Hot Topics)*, pages 45–48, 1997. See CD proceedings of conference. Paper from `ftp://cs.nyu.edu/pub/local/yap/visual/thinwire.ps.gz`.

[3] E.-C. Chang and C. K. Yap. A wavelet approach to foveating images. *ACM Symp. on Computational Geometry*, 13:397–399, 1997.

[4] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Pub. Co, Reading, Massachusetts, second edition edition, 1992.

[5] U. MIPS. United states geological survey's mini image processing system, 1999. Information and free distribution from URL `http://TerraWeb.wr.usgs.gov/TRS/software/mips/`.

[6] J. L. Moigne and R. Cromp. The use of wavelets for remote sensing image registration and fusion. Technical Report TR-96-171, Center of Excellence in Space Data and Information Sciences, Goddard Space Flight Center, Greenbelt, MD 20771, 1997.

[7] J. D. Murray and W. vanRyper. *Encyclopedia of Graphics File Formats*. O'Reilly & Associates, Inc, 2nd edition, 1996.

[8] J. P. S. Chavez and J. A. Anderson. Standardization of the USGS Mini Image Processing System. In *Proc. of ASPRS and ACSM Ann. Spring Convention*, pages 364–373, 1987. April 1987, Baltimore, Maryland. Invited paper.

[9] S. Sides, J. Anderson, and J. P.S. Chavez. An interactive image analysis system. In *XVII Congress of the Intl. Soc. for Photogrammetry and Remote Sensing (ISPRS)*, 1992. August 1992, Washington D. C.