Intelligent Mining in Image Databases, With Applications to Satellite Imaging and to Web Search

Stephen Gibson[1,2], Vladik Kreinovich[1,2],
Luc Longpré[1], Brian Penn[2], and Scott A. Starks[2]

[1]Department of Computer Science and
[2]NASA Pan-American Center for Earth
and Environmental Sciences (PACES)
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
contact email vladik@cs.utep.edu

Abstract

An important part of our knowledge is in the form of images. For example, a large amount of geophysical and environmental data comes from satellite photos; a large amount of the information stored on the Web is in the form of images, etc. It is therefore desirable to use this image information in data mining. Unfortunately, most existing data mining techniques have been designed for mining numerical data and are thus not well suited for image databases; so, new methods are needed for image mining. In this paper, we show how data mining can be used to find common patterns in several images.

1 Introduction

1.1 It is necessary to apply data mining to images

An important part of our knowledge is in the form of images. For example, a large amount of geophysical and environmental data comes from satellite photos; a large amount of the information stored on the Web is in the form of images, etc. It is therefore desirable to use this image information in data mining. Unfortunately, most existing data mining techniques have been designed for mining numerical data and are thus not well suited for image databases; so, new methods are needed for image mining.

An important part of image mining is finding common patterns in several images. It is difficult to uncover such a pattern, and it is difficult to automatically check whether a new image contains such a pattern. There exist (crisp) FFT-based methods for solving these problems, but often, they fail to detect a clearly visible pattern.

In this paper, we show how the existing methods can be improved. We start with reasonable "expert rules" which describe possible improvements, and describe possible formalizations of these expert rules. Then, we use a group-theoretic technique to find the optimal formalization; this group-theoretic technique have been successfully used to make the choices in fuzzy, neural, and genetic methodologies which turned out to be empirically optimal.

The resulting new pattern-finding and pattern-checking methods are illustrated on two examples:

- analysis of satellite images and
- search for a known pattern (e.g., a known text) in web images.

1.2 First case study: Mosaicking satellite imaging

Satellite photos provide a good description of the Earth areas. Often, we are interested in the area which is covered by several satellite photos, so we need to combine (*mosaic*) these photos into a single image. The problem is that we do not know the exact orientation of the satellite-based camera, so the photos may be shifted and rotated with respect to each other, and we do not know the exact values of these shift and rotation. Therefore, to mosaic two images, we must find how shifted and rotated these images are relative to one another.

At present, mosaicking of satellite images is mainly done manually, by trial and error. This trial-and-error procedure is difficult to automate: for $n \times n$ images, where n can be from 1,000 to 6,000, we have n^2 possible shifts, which, together with $\approx n$ possible rotations and $\approx n$ possible scalings, make for an impossible number of $\approx n^4$ ($\geq 10^{12}$) possible image comparisons. It is therefore necessary to come up with time-saving mosaicking algorithms.

1.3 Second case study: Searching for a pattern in a web image

A similar problem occurs when we search images stored on the web. We may want to find all images which contain a certain pattern (e.g., a certain text), but this pattern may be scaled differently in different web images. So, we must be able to mosaic two images:

- the image which contains the desired pattern, and
- the image which is stored on the web.

We must be able to find the shift, rotation, and scaling after which these two images match in the best possible way.

One particular case of this problem is searching for text in web images. The gaining popularity of the World Wide Web also means increasing security risks. As the World Wide Web has become an affordable way for different political groups to reach a broad audience it is becoming harder to monitor all these web sites for their content. While numerous web search tools can be used to automatically monitor plain text in web pages, search for text in graphical images is still a considerable challenge. This fact is used by designers of such web pages who "hide" their text by placing it inside of graphical images, avoiding detection from regular search engines. At present, the only known way to find all occurrences of suspicious words like "terror" in images is to use character recognition to find and read all the texts in all the images. Performing such a character recognition is still a very computational intensive task which takes a long time, and has to be done for every image, because the only known way to check whether the image contains text is to apply a character recognition to this image. It is therefore desirable to develop faster algorithms for detecting text in web pages.

1.4 The existing FFT-based mosaicking algorithms

To decrease the mosaicking time, researchers have proposed methods based on Fast Fourier Transform (FFT). The best of known FFT-based mosaicking algorithms is presented in [6]. The main ideas behind FFT-based mosaicking in general and this algorithm in particular are as follows.

1.4.1 The simplest case: shift detection in the absence of noise

Let us first consider the case when two images differ only by shift. It is known that if two images $I_1(\vec{x})$ and $I_2(\vec{x})$ differ only by shift, i.e., if $I_2(\vec{x}) = I_1(\vec{x} + \vec{a})$ for some (unknown) shift $\vec{a}$, then their Fourier transforms

$$F_i(\vec{\omega}) = \frac{1}{2\pi} \cdot \int\int I(\vec{x}) \cdot e^{-2\pi \cdot (\vec{x} \cdot \vec{\omega})}\, \mathrm{d}x\mathrm{d}y$$

are related by the following formula:

$$F_2(\vec{\omega}) = e^{2\pi \cdot \mathrm{i} \cdot (\vec{\omega} \cdot \vec{a})} \cdot F_1(\vec{\omega}). \tag{1}$$

Therefore, if the images are indeed obtained from each other by shift, then we have

$$M_2(\vec{\omega}) = M_1(\vec{\omega}), \tag{2}$$

where we denoted

$$M_i(\vec{\omega}) = |F_i(\vec{\omega})|. \tag{3}$$

The actual value of the shift $\vec{a}$ can be obtained if we use the formula (1) to compute the value of the following ratio:

$$R(\vec{\omega}) = \frac{F_1^*(\vec{\omega}) \cdot F_2(\vec{\omega})}{|F_1^*(\vec{\omega}) \cdot F_2(\vec{\omega})|}. \tag{4}$$

Substituting (1) into (4), we get

$$R(\vec{\omega}) = e^{2\pi \cdot \mathrm{i} \cdot (\vec{\omega} \cdot \vec{a})}. \tag{5}$$

Therefore, the inverse Fourier transform $P(\vec{x})$ of this ratio is equal to the delta-function $\delta(\vec{x} - \vec{a})$.

In other words, in the ideal no-noise situation, this inverse Fourier transform $P(\vec{x})$ is equal to 0 everywhere except for the point $\vec{x} = \vec{a}$; so, from $P(\vec{x})$, we can easily determine the desired shift by using the following algorithm:

- first, we apply FFT to the original images $I_1(\vec{x})$ and $I_2(\vec{x})$ and compute their Fourier transforms $F_1(\omega)$ and $F_2(\omega)$;
- on the second step, we compute the ratio (4);
- on the third step, we apply the inverse FFT to the ratio $R(\vec{\omega})$ and compute its inverse Fourier transform $P(\vec{x})$;
- finally, on the fourth step, we determine the desired shift $\vec{a}$ as the only value $\vec{a}$ for which $P(\vec{a}) \neq 0$.

1.4.2 Shift detection in the presence of noise

In the presence of noise, we expect the values of $P(\vec{x})$ to be slightly different from the delta-function, but still, the value $|P(\vec{a})|$ should be much larger than all the other values of this function. So, to determine the shift $\vec{a}$, we can use the same algorithm as above but with a different final step:

- on the fourth step, we determine the desired shift $\vec{a}$ as the point for which $|P(\vec{x})|$ takes the largest possible value.

1.4.3 Reducing rotation and scaling to shift

If, in addition to shift, we also have rotation and scaling, then the absolute values $M_i(\vec{\omega})$ of the corresponding Fourier transforms are not equal, but differ from each by the corresponding rotation and scaling.

If we go from Cartesian to polar coordinates (r, θ) in the $\vec{\omega}$-plane, then rotation by an angle θ_0 is described by a simple shift-like formula $\theta \to \theta + \theta_0$.

In these same coordinates, scaling is also simple, but not shift-like: $r \to \lambda \cdot r$. If we go to *log-polar* coordinates (ρ, θ), where $\rho = \log(r)$, then scaling also becomes shift-like: $\rho \to \rho + b$, where $b = \log(\lambda)$. So, in log-polar coordinates, both rotation and scaling are described by a shift.

1.4.4 How to determine rotation and scaling

In view of the above reduction, in order to determine the rotation and scaling between M_1 and M_2, we can do the following:

- transform both images from the original Cartesian coordinates to log-polar coordinates;
- use the above FFT-based algorithm to determine the corresponding shift $(\theta_0, \log(\lambda))$;
- from the corresponding "shift" values, reconstruct the rotation angle θ_0 and the scaling coefficient λ.

The main computational problem with the transformation to log-polar coordinates is that we need values $M(\xi, \eta)$ on a rectangular grid in log-polar space $(\log(\rho), \theta)$, but computing $(\log(\rho), \theta)$ for the original grid points leads to points outside that grid. So, we need interpolation to find the values $M(\xi, \eta)$ on the desired grid. One possibility is to use *bilinear* interpolation. Let (x, y) be a rectangular point corresponding to the desired grid point $(\log(\rho), \theta)$, i.e.,

$$x = e^{\log(\rho)} \cdot \cos(\theta), \quad y = e^{\log(\rho)} \cdot \sin(\theta).$$

To find the value $M(x, y)$, we look at the intensities M_{jk}, $M_{j+1,k}$, $M_{j,k+1}$, and $M_{j+1,k+1}$ of the four grid points (j, k), $(j+1, k)$, $(j, k+1)$, and $(j+1, k+1)$ surrounding (x, y). Then, we can interpolate $M(x, y)$ as follows:

$$M(x, y) = (1-t) \cdot (1-u) \cdot M_{jk} + t \cdot (1-u) \cdot M_{j+1,k} + (1-t) \cdot u \cdot M_{j,k+1} + t \cdot u \cdot M_{j+1,k+1},$$

where t is a fractional part of x and u is a fractional part of y.

1.4.5 Final algorithm: determining shift, rotation, and scaling

- First, we apply FFT to the original images $I_1(\vec{x})$ and $I_2(\vec{x})$ and compute their Fourier transforms $F_1(\omega)$ and $F_2(\omega)$.
- Then, we compute the absolute values $M_1(\vec{\omega}) = |F_1(\vec{\omega})|$ and $M_2(\vec{\omega}) = |F_2(\vec{\omega})|$ of these Fourier transforms.
- By applying the above algorithm and scaling detection algorithm to the functions $M_1(\omega)$ and $M_2(\omega)$, we can determine the rotation angle θ_0 and the scaling coefficient λ.
- Now, we can apply the corresponding rotation and scaling to one of the original images, e.g., to the first image $I_1(\vec{x})$. As a result, we get a new image $\widetilde{I}_1(\vec{x})$.
- Since we rotated and re-scaled one of the images, the images $\widetilde{I}_1(\vec{x})$ and $I_2(\vec{x})$ are already aligned in terms of rotation and scaling, and the only difference between them is in an (unknown) shift. So, we can again apply the above described FFT-based algorithm for determining shift: this time, actually to determine shift.

As a result, we get the desired values of shift, rotation, and scaling; hence, we get the desired mosaicking.

1.5 Problems with the existing FFT-based algorithm

In many real life situations, this algorithm works well. However, when we tried to implement this algorithm on several test images, we encountered the following two problems:

- In some cases, we could not complete the algorithm because of a "division by zero" error message.
- In some other cases, although the algorithm worked, the resulting rotation angle was reconstructed with a large inaccuracy even for images with no noise added. For example, when we compared two simple 64×64 pixel images which were obtained from each other by an exact rotation, the inaccuracy in reconstructing the rotation angle was sometimes as high as 1.5 degrees. Sometimes we do not get any reconstruction at all.

It is therefore necessary to modify the above algorithm so as to avoid these two problems. In this paper, we describe the desired improvement of this algorithm. The details are given in [1].

2 Analysis of the problems

In order to avoid the above problems, we must first find out what causes these problems. Let us describe the result of our analysis.

2.1 "Divide by zero" problem: analysis

2.1.1 Experimental analysis

The first problem that we analyzed was the problem of dividing by zero. This problem did not occur for images used in [6], but it did occur in some of our images.

In order to find out what causes this problem, we first tried to find something in common between the different images in which this problem occurred. It turns out that, in our tests, this problem occurs exclusively in simple images. This observation explains why this problem was never encountered before: because the algorithm was always tested on rather complex, real-life image.

2.1.2 Conclusion

- If all we wanted to do was mosaic satellite images, then we would not have to worry about this problem, because it occurs only in simple images.
- However, since one of our major application areas is detecting text in web images, and web images are often very simple, this problem becomes more important.

2.1.3 Theoretical analysis

The "division by zero" error comes from computing the expression (4), one of one of the values $F_i(\vec{\omega})$ of the Fourier transforms is equal to 0. In this case, both the numerator and the denominator of (4) becomes equal to 0, so we have a 0/0 problem.

In general, a Fourier transform $F(\vec{\omega})$ of an image $I(\vec{x})$ is a linear combination of the image's intensity values $I(\vec{x})$ at different pixels $\vec{x}$ with the complex coefficients depending on $\vec{x}$ and $\vec{\omega}$. To get zero, we need these terms to exactly compensate each other.

- For a complex image, especially for a real-life image, the values $I(\vec{x})$ corresponding to different pixels $\vec{x}$ are different and unrelated, so it is unlikely that they will add up to exactly zero.
- However, for a simple image, the values $I(\vec{x})$ can be described by a simple formula, and the intensity values $I(\vec{x})$ corresponding to different pixels $\vec{x}$ are closely related.

It is therefore quite possible that, for simple images, with these related values, we get $F(\vec{\omega})$ for some $\vec{\omega}$.

2.1.4 Example

This possibility can be illustrated by a simple two-pixel image, in which two neighboring pixels $-x_0$ and x_0 on both sides of the central point 0 have equal intensity: $I(x) = I_0 \cdot \delta(x - x_0) + I_0 \cdot \delta(x + x_0)$. The Fourier transform of this image is equal to $2I_0 \cdot \cos(2\pi \cdot \xi \cdot x_0)$, and for certain values ξ, we get division by 0.

In a discrete case, we see that division by zero occurs when one of the images has a unit intensity equally distributed between two neighboring points, e.g., if the intensities are

$$0, 0, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 0, 0, 0, 0, 0, 0, 0$$

2.2 Accuracy problem: analysis

- In the previously known algorithm, we determine the shift as the point $\vec{x}$ *on the grid* for which $|P(\vec{x})|$ attains the largest possible value.
- The actual values of rotation angle θ_0 and log-scaling $\log(\lambda)$ may *not* be *exactly on the grid*.

As a result, when we use the FFT-based shift-detection algorithm to determine rotation and scaling, we do not determine them exactly. Hence, the alignment made by these approximate values of rotation angle and scaling is not exact.

For noisy images, the additional distortion produced by this mis-alignment often prevents the shift-detecting algorithm from finding the shift between the images I_1 and I_2.

To decrease this distortion, in shift detection, we would like to be able to find a more accurate estimate of the shift, even when its actual value is not from the grid.

3 The new mosaicking algorithm

3.1 Main idea

- We have mentioned that in the existing algorithm, we determine the shift as the point $\vec{x}$ on a grid for which $|P(\vec{x})|$ attains the largest possible value.
- To improve the accuracy of mosaicking, it is desirable we would like to be able to find a more accurate estimate of the shift, even when its actual value is not from the grid.

In 1-D case:

- If the function $|P(x)|$ has a large maximum at a point a and is equal to 0 for all $x \neq a$, then, of course, the actual value of the shift is a.
- However, if the value $|P(x)|$ is large for two sequential points x_1 and x_2, then probably the actual shift is somewhere between x_1 and x_2. In other words, the actual shift should be equal to $x = w_1 \cdot x_1 + w_2 \cdot x_2$ for some weights $w_1 + w_2 = 1$.

The larger $|P(x_i)|$, the closer the actual shift point to x_i; so, the larger the weight w_i should be.

3.2 Toward formalizing this idea

The above idea is formulated in terms of words from natural language, like "large". Let us formalize this idea.

3.2.1 1-D case: first guess

We have already mentioned that the larger $|P(x_i)|$, the closer the actual shift point to x_i; so, the larger the weight w_i should be. At first glance, it therefore seems reasonable to take $w_i = f(|P(x_i)|)$ for some monotonically increasing function $f(z)$.

For this choice, however, we cannot guarantee that $w_1 + w_2 = 1$.

3.2.2 1-D case: corrected guess

A natural way to avoid the above problem is to *normalize* these weights, i.e., to take

$$x = \frac{f(|P(x_1)|) \cdot x_1 + f(|P(x_2)|) \cdot x_2}{f(|P(x_1)|) + f(|P(x_2)|)}. \tag{6}$$

3.2.3 2-D case

In a 2-D case, we can similarly take two points x_1, x_2, y_1, y_2 in each of the grid's directions, and use the sums of the corresponding values $f(|P|)$ as the weights:

$$x = \frac{w_{x1} \cdot x_1 + w_{x2} \cdot x_2}{w_{x1} + w_{x2}}; \tag{7}$$

$$y = \frac{w_{y1} \cdot y_1 + w_{y2} \cdot y_2}{w_{y1} + w_{y2}}, \tag{8}$$

where

$$w_{xi} = f(|P(x_i, y_1)|) + f(|P(x_i, y_2)|); \tag{9}$$

$$w_{yi} = f(|P(x_1, y_i)|) + f(|P(x_2, y_i)|). \tag{10}$$

3.2.4 Remaining problem

To finalize this formalization, we must select a function $f(z)$.

This selection is very important, because numerical experiments show that different choices lead to drastically different efficiency of the resulting method; so, to increase the algorithm's efficiency, we would like to choose the best possible function $f(z)$.

3.3 Best in what sense?

What do we mean by "the best"? It is not so difficult to come up with different criteria for choosing a function $f(z)$:

- We may want to choose the function $f(z)$ for which the resulting location error is, on average, the smallest possible $P(f) \to \min$ (i.e., for which the *quality of the answer* is, on average, the best).
- We may also want to choose the function $f(z)$ for which the *average computation time* $C(f)$ is the smallest (average in the same of some reasonable probability distribution on the set of all problems).

At first glance, the situation seems hopeless: we cannot estimate these numerical criteria even for a single function $f(z)$, so it may look like we therefore cannot undertake an even more ambitious task of finding the *optimal* function $f(z)$. Hopefully, the situation is not as hopeless as it may seem, because there is a symmetry-based formalism (actively used in the foundations of fuzzy, neural, genetic computations, see, e.g., [5]) which will enable us to find the optimal function $f(z)$ for our situation too. (Our application will be mathematically similar to the optimal choice of a non-linear scaling function in genetic algorithms [4, 5].)

Before we make a formal definition, let us make two comments.

- The first comment is that our goal is to find the weights. The weights are always non-negative numbers, so the function $f(z)$ must also take only non-negative values.

- The second comment is that all we want from the function $f(z)$ is the weights. These probabilities are computed according to the formulas (6–8). From these expressions (6–8), one can easily see that if we multiply all the values of this function $f(z)$ by an arbitrary constant C, i.e., if we consider a new function $\widetilde{f}(z) = C \cdot f(z)$, then this new function will lead (after the normalization involved in (6–8)), to exactly the same values of the weights. Thus, whether we choose $f(z)$ or $\widetilde{f}(z) = C \cdot f(z)$, does not matter. So, what we are really choosing is not a *single* function $f(z)$, but a *family* of functions $\{C \cdot f(z)\}$ (characterized by a parameter $C > 0$).

In the following text, we will denote families of functions by capital letters, such as F, F', G, etc.

3.4 An optimality criterion can be non-numeric

Traditionally, optimality criteria are *numerical*, i.e., to every family F, we assign some value $J(F)$ expressing its quality, and choose a family for which this value is minimal (i.e., when $J(F) \leq J(G)$ for every other alternative G). However, it is not necessary to restrict ourselves to such numeric criteria only.

For example, if we have several different families F that have the same average location error $P(F)$, we can choose between them the one that has the minimal computational time $C(F)$. In this case, the actual criterion that we use to compare two families is not numeric, but more complicated: A family F_1 is better than the family F_2 if and only if either $P(F_1) < P(F_2)$, or $P(F_1) = P(F_2)$ and $C(F_1) < C(F_2)$.

The only thing that a criterion *must* do is to allow us, for every pair of families (F_1, F_2), to make one of the following conclusions:

- the first family is better with respect to this criterion (we'll denote it by $F_1 \succ F_2$, or $F_2 \prec F_1$);
- with respect to the given criterion, the second family is better ($F_2 \succ F_1$);
- with respect to this criterion, the two families have the same quality (we'll denote it by $F_1 \sim F_2$);
- this criterion does not allow us to compare the two families.

Of course, it is necessary to demand that these choices be consistent. For example, if $F_1 \succ F_2$ and $F_2 \succ F_3$ then $F_1 \succ F_3$.

3.5 Optimality criterion must be final

A natural demand is that this criterion must choose a *unique* optimal family (i.e., a family that is better with respect to this criterion than any other family). The reason for this demand is very simple:

- If a criterion *does not choose* any family at all, then it is of no use.
- If *several* different families are the best according to this criterion, then we still have the problem of choosing the best among them. Therefore we need some additional criterion for that choice, like in the above example: If several families $F_1, F_2, \ldots$ turn out to have the same average location error ($P(F_1) = P(F_2) = \ldots$), we can choose among them a family with minimal computation time ($C(F_i) \to \min$). So what we actually do in this case is abandon that criterion for which there were several "best" families, and consider a new "composite" criterion instead: F_1 is better than F_2 according to this new criterion if either it was better according to the old criterion, or they had the same quality according to the old criterion and F_1 is better than F_2 according to the additional criterion.

In other words, if a criterion does not allow us to choose a unique best family, it means that this criterion is not final, we'll have to modify it until we come to a final criterion that will have that property.

3.6 The criterion must not change if we change the measuring unit for length

The exact mathematical form of a function $f(z)$ depends on the exact choice of units for measuring length. If we replace this unit by a new unit that is λ times larger, then the same physical value that was previously described by a numerical value $I(x, y)$ will now be described, in the new units, by new numerical values $\widetilde{I}(x, y) = I(x/\lambda, y/\lambda)$, and the corresponding Fourier transform of the ratio will change to $\widetilde{P}(x, y) = P(x, y)/\lambda$. So, for $J(\vec{x}) = |P(\vec{x})|$, we will have $\widetilde{J}(\vec{x}) = J(\vec{x})/\lambda$.

How will the expression for $f(z)$ change if we use the new units? In terms of $\widetilde{J}(\vec{x})$, we have $J(\vec{x}) = \lambda \cdot \widetilde{J}(\vec{x})$. Thus, if we change the measuring unit for $J(\vec{x})$, the same weight $w(\vec{x}) \sim f(J(\vec{x}))$ that was

originally represented by a function $f(z)$, will be described, in the new units, as $w(\vec{x}) \sim f(\lambda \cdot \widetilde{J}(\vec{x}))$, i.e., as $w(\vec{x}) \sim \widetilde{f}(\widetilde{J}(\vec{x}))$, where $\widetilde{f}(z) = f(\lambda \cdot z)$.

There is no reason why one choice of unit should be preferable to the other. Therefore, it is reasonable to assume that the relative quality of different families should not change if we simply change the units, i.e., if the family F is better than a family G, then the transformed family $\widetilde{F}$ should also be better than the family $\widetilde{G}$.

We are now ready for the formal definitions.

3.7 Definitions and the main result

Definition 1. *Let $f(z)$ be a differentiable strictly increasing function from real numbers to non-negative real numbers. By a family that corresponds to this function $f(z)$, we mean a family of all functions of the type $\widetilde{f}(z) = C \cdot f(z)$, where $C > 0$ is an arbitrary positive real number. (Two families are considered equal if they coincide, i.e., consist of the same functions.)*

In the following text, we will denote the set of all possible families by Φ.

Definition 2. *By an optimality criterion, we mean a consistent pair $(\prec, \sim)$ of relations on the set Φ of all alternatives which satisfies the following conditions, for every $F, G, H \in \Phi$:*

(1) *if $F \prec G$ and $G \prec H$ then $F \prec H$;*

(2) *$F \sim F$;*

(3) *if $F \sim G$ then $G \sim F$;*

(4) *if $F \sim G$ and $G \sim H$ then $F \sim H$;*

(5) *if $F \prec G$ and $G \sim H$ then $F \prec H$;*

(6) *if $F \sim G$ and $G \prec H$ then $F \prec H$;*

(7) *if $F \prec G$ then $G \not\prec F$ and $F \not\sim G$.*

Comment. The intended meaning of these relations is as follows:

- $F \prec G$ means that with respect to a given criterion, G is better than F;
- $F \sim G$ means that with respect to a given criterion, F and G are of the same quality.

Under this interpretation, conditions (1)–(7) have simple intuitive meaning; e.g., (1) means that if G is better than F, and H is better than G, then H is better than F.

Definition 3.

- *We say that an alternative F is optimal (or best) with respect to a criterion $(\prec, \sim)$ if for every other alternative G either $F \succ G$ or $F \sim G$.*
- *We say that a criterion is final if there exists an optimal alternative, and this optimal alternative is unique.*

Definition 4. *Let $\lambda > 0$ be a positive real number.*

- *By a λ-rescaling of a function $f(x)$ we mean a function $\widetilde{f}(x) = f(\lambda \cdot x)$.*
- *By a λ-rescaling $R_\lambda(F)$ of a family of functions F we mean the family consisting of λ-rescalings of all functions from F.*

Definition 5. *We say that an optimality criterion on Φ is unit-invariant if for every two families F and G and for every number $\lambda > 0$, the following two conditions are true:*

i) if F is better than G in the sense of this criterion (i.e., $F \succ G$), then $R_\lambda(F) \succ R_\lambda(G)$;

ii) if F is equivalent to G in the sense of this criterion (i.e., $F \sim G$), then $R_\lambda(F) \sim R_\lambda(G)$.

Theorem 1. *If a family F is optimal in the sense of some optimality criterion that is final and unit-invariant, then every function $f(z)$ from this family F has the form $C \cdot z^\alpha$ for some real numbers C and α.*

This theorem was, in effect, proven in [3, 5]. For reader's convenience, all the proofs are given in the special Appendix.

3.8 Tuning the resulting algorithm

The above theorem shows that $f(z) = z^\alpha$, but it does not tell which value α we should choose. To determine the optimal value of α, we analyzed several different images and came up with the following experimental conclusion:

- on the first stage, when we determine rotation and scaling, the optimal value of α is $\alpha_r \approx 1.55$;
- on the second stage, on which we determine the shift, the optimal value of α is $\alpha_s \approx 0.65$.

For these values, we indeed get a pretty good mosaicking.

4 The optimal choice of 0/0

4.1 Main idea

In the previous section, we showed the optimal solution to the second problem – of fractional shifts. To complete the description of an optimal FFT-based algorithm, we must find an optimal solution to the first (0/0) problem. In principle, we can choose an arbitrary complex number as 0/0. Which is the best choice?

In solving this problem, we will use the same theoretical approach as in the previous section: similarly to that section, it is difficult to formulate a numerical criterion for choosing z. So, we will assume that there is a final optimality criterion on the set of all complex numbers, and we will look for the number which is best with respect to this criterion.

Similarly to the previous section, we can formulate natural invariance requirements for this criterion. Namely, the value 0/0 comes from the ratio (4). We have already mentioned that if we shift I_1, then the value of $F_1(\vec{\omega})$ gets multiplied by $e^{2\pi\cdot \mathrm{i}\cdot(\vec{\omega}\cdot\vec{a})}$. Thus, the ratio z determined by the formula (4) gets changed to

$$z \to z \cdot e^{\mathrm{i}\cdot\theta}, \tag{11}$$

where $\theta = 2\pi\cdot(\vec{\omega}\cdot\vec{a})$.

The decision of which value of 0/0 is the best should be universal, and it should not change with an additional shift of I_1. Therefore, it makes sense to assume that the optimality criterion should not change if we apply the transformation (11). Now, we are ready for the formal definitions:

4.2 Definitions and the result

Definition 6. *We say that an optimality criterion on the set* $\mathbf{C}$ *of all complex numbers is invariant if for every two complex numbers* z *and* z' *and for every real number* $\theta > 0$*, the following two conditions are true:*

i) if z *is better than* z' *in the sense of this criterion (i.e.,* $z \succ z'$*), then* $z \cdot e^{\mathrm{i}\cdot\theta} \succ z' \cdot e^{\mathrm{i}\cdot\theta}$.

ii) if z *is equivalent to* z' *in the sense of this criterion (i.e.,* $z \sim z'$*), then* $z \cdot e^{\mathrm{i}\cdot\theta} \sim z' \cdot e^{\mathrm{i}\cdot\theta}$.

Theorem 2. *If a number* z *is optimal in the sense of some optimality criterion that is final and invariant, then* $z = 0$.

4.3 Experimental confirmation

To test this theoretical conclusion, we tested our algorithm, for different values of $z = 0/0$, on the simple 2-pixel image described above. For this image, the error with which we can determine the shift is indeed the smallest for $z = 0$.

5 Summarizing: new algorithm

Combining the above results, we come up with the following new modification of the FFT-based algorithm:

5.1 The simplest case: shift detection

- First, we apply FFT to the original images $I_1(\vec{x})$ and $I_2(\vec{x})$ and compute their Fourier transforms $F_1(\omega)$ and $F_2(\omega)$.
- On the second step, we compute the ratio $R(\vec{\omega})$ by using formula (4); if we denominator is 0, then we take the ratio to be equal to 0 too.
- On the third step, we apply the inverse FFT to the ratio $R(\vec{\omega})$ and compute its inverse Fourier transform $P(\vec{x})$
- Finally, on the fourth step, we do the following:
  - we find the point $\vec{x} = (x_1, y_1)$ for which $|P(\vec{x})|$ takes the largest possible value;
  - then, among 4 points $(x_1 \pm 1, y_1 \pm 1)$, we select a point (x_2, y_2) for which the value $|P(x_2, y_2)|$ is the largest;
  - after that, we apply the formulas (7–10) with $f(z) = z^\alpha$ and $\alpha = 0.65$ to find the coordinates (x, y) of the shift.

5.2 Final algorithm: determining shift, rotation, and scaling

- First, we apply FFT to the original images $I_1(\vec{x})$ and $I_2(\vec{x})$ and compute their Fourier transforms $F_1(\omega)$ and $F_2(\omega)$.
- Then, we compute the absolute values $M_1(\vec{\omega}) = |F_1(\vec{\omega})|$ and $M_2(\vec{\omega}) = |F_2(\vec{\omega})|$ of these Fourier transforms.
- We transform both "images" $M_i(\vec{\omega})$ from the original Cartesian coordinates to log-polar coordinates.
- Then, we use the above FFT-based algorithm, with $\alpha = 1.55$, to determine the corresponding shift $(\theta_0, \log(\lambda))$.
- From the corresponding "shift" values, we reconstruct the rotation angle θ_0 and the scaling coefficient λ.
- Now, we apply the corresponding rotation and scaling to one of the original images, e.g., to the first image $I_1(\vec{x})$. As a result, we get a new image $\widetilde{I}_1(\vec{x})$.
- Since we rotated and re-scaled one of the images, the images $\widetilde{I}_1(\vec{x})$ and $I_2(\vec{x})$ are already aligned in terms of rotation and scaling, and the only difference between them is in an (unknown) shift. So, we can again apply the above described FFT-based algorithm for determining shift: this time, actually to determine shift.

As a result, we get the desired values of shift, rotation, and scaling; hence, we get the desired mosaicking.

6 Experimental testing of the new algorithm

We ran three series of tests:

- First, we checked whether the resulting algorithm indeed solves the problems of the original FFT-based method, i.e., that its accuracy is better and its applicability is wider.
- Second, we tested this algorithm on two overlapping satellite images to see how well the algorithm works with images that have different shading and substantial noise.
- Finally, we applied this algorithm to find text in images.

In all three series, we got good mosaicking results.

Conclusion

In many application areas several images cover a single area and, therefore, it is important to mosaic them into a single image. For that, we need to properly shift, rotate, and re-scale the component images.

Several FFT-based algorithms have been proposed for such mosaicking. Sometimes, however, these algorithms do not work well: for some simple images, these methods do not work at all, while for some more complicated images, the resulting mosaicking accuracy is very low.

We have developed and tested an optimal FFT-based mosaicking algorithm. This algorithm works well on all kinds of images including man-made images, satellite photos, and detection of text in images. In particular, this algorithm works well on the images on which the previously known algorithms failed.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grants No. DUE-9750858 and CDA-9522207, by United Space Alliance, grant No. NAS 9-20000 (PWO C0C67713A6), by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518, and by the National Security Agency under Grant No. MDA904-98-1-0561.

References

[1] S. Gibson, *An optimal FFT-based algorithm for mosaicking images*, Master Thesis, Department of Computer Science, University of Texas at El Paso, December 1999.

[2] L. T. Koczy, V. Kreinovich, Y. Mendoza, H. T. Nguyen, and H. Schulte, "Towards Mathematical Foundations of Information Retrieval: Dependence of Website's Relevance on the Number of Occurrences of a Queried Word", *Proceedings of the Joint Conferences in Information Sciences JCIS'2000*, Atlantic City, NJ, February 27–March 3, 2000 (to appear).

[3] O. Kosheleva, L. Longpré, and R. Osegueda, "Detecting Known Non-Smooth Structures in Images: Fuzzy and Probabilistic Methods, with Applications to Medical Imaging, Non-Destructive Testing, and Detecting Text on Web Pages", *Proceedings of The Eighth International Fuzzy Systems Association World Congress IFSA'99*, Taipei, Taiwan, August 17–20, 1999, pp. 269–273.

[4] V. Kreinovich, C. Quintana, and O. Fuentes. "Genetic algorithms: what fitness scaling is optimal?" *Cybernetics and Systems: an International Journal*, 1993, Vol. 24, No. 1, pp. 9–26.

[5] H. T. Nguyen and V. Kreinovich, *Applications of continuous mathematics to computer science*, Kluwer, Dordrecht, 1997.

[6] B. S. Reddy and B. N. Chatterji, "An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration," *IEEE Transactions on Image Processing*, 1996, Vol. 5, No. 8, pp. 1266–1271.

Appendix: Proofs

6.1 Proof of Theorem 1

This proof is based on the following lemma:

Lemma. *If an optimality criterion is final and unit-invariant, then the optimal family F_{opt} is also unit-invariant, i.e., $R_\lambda(F_{opt}) = F_{opt}$ for every number λ.*

Proof of the Lemma. Since the optimality criterion is final, there exists a unique family F_{opt} that is optimal with respect to this criterion, i.e., for every other F:

- either $F_{opt} \succ F$,
- or $F_{opt} \sim F$.

To prove that $F_{opt} = R_\lambda(F_{opt})$, we will first show that the re-scaled family $R_\lambda(F_{opt})$ is also optimal, i.e., that for every family F:

- either $R_\lambda(F_{opt}) \succ F$,
- or $R_\lambda(F_{opt}) \sim F$.

If we prove this optimality, then the desired equality will follow from the fact that our optimality criterion is final and therefore, there is only one optimal family (so, since the families F_{opt} and $R_\lambda(F_{opt})$ are both optimal, they must be the same family).

Let us show that $R_\lambda(F_{opt})$ is indeed optimal. How can we, e.g., prove that $R_\lambda(F_{opt}) \succ F$? Since the optimality criterion is unit-invariant, the desired relation is equivalent to $F_{opt} \succ R_{\lambda^{-1}}(F)$. Similarly, the relation $R_\lambda(F_{opt}) \sim F$ is equivalent to $F_{opt} \sim R_{\lambda^{-1}}(F)$.

These two equivalences allow us to complete the proof of the lemma. Indeed, since F_{opt} is optimal, we have one of the two possibilities:

- either $F_{opt} \succ R_{\lambda^{-1}}(F)$,

- or $F_{opt} \sim R_{\lambda^{-1}}(F)$.

In the first case, we have $R_\lambda(F_{opt}) \succ F$; in the second case, we have $R_\lambda(F_{opt}) \sim F$.

Thus, whatever family F we take, we always have:

- either $R_\lambda(F_{opt}) \succ F$,
- or $R_\lambda(F_{opt}) \sim F$.

Hence, $R_\lambda(F_{opt})$ is indeed optimal and thence, $R_\lambda(F_{opt}) = F_{opt}$. The lemma is proven.

Let us now prove the theorem. Since the criterion is final, there exists an optimal family $F_{opt} = \{C \cdot f(z)\}$. Due to the lemma, the optimal family is unit-invariant.

From unit-invariance, it follows that for every λ, there exists a real number $A(\lambda)$ for which $f(\lambda \cdot z) = A(\lambda) \cdot f(z)$. Since the function $f(z)$ is differentiable, we can conclude that the ratio

$$A(\lambda) = \frac{f(\lambda \cdot z)}{f(z)}$$

is differentiable as well. Thus, we can differentiate both sides of the above equation with respect to λ, and substitute $\lambda = 1$. As a result, we get the following differential equation for the unknown function $f(z)$:

$$z \cdot \frac{df}{dz} = \alpha \cdot f,$$

where by α, we denoted the value of the derivative

$$\frac{dA}{d\lambda}$$

taken at $\lambda = 1$. Moving terms dz and z to the right-hand side and all the term containing f to the left-hand side, we conclude that

$$\frac{df}{f} = \alpha \cdot \frac{dz}{z}.$$

Integrating both sides of this equation, we conclude that $\ln(f) = \alpha \cdot \ln(z) + C$ for some constant C, and therefore, that $f(z) = \text{const} \cdot z^\alpha$. The theorem is proven.

6.2 Proof of Theorem 2

This proof is based on the following lemma:

Lemma. *If an optimality criterion is final and invariant, then the optimal value z_{opt} is also invariant, i.e., $z_{opt} = z_{opt} \cdot e^{\mathrm{i}\cdot\theta}$ for every real number θ.*

Proof of the Lemma. Since the optimality criterion is final, there exists a unique complex number z_{opt} that is optimal with respect to this criterion, i.e., for every other z,

- either $z_{opt} \succ z$,
- or $z_{opt} \sim z$.

To prove that $z_{opt} = z_{opt} \cdot e^{\mathrm{i}\cdot\theta}$, we will first show that the number $z_{opt} \cdot e^{\mathrm{i}\cdot\theta}$ is also optimal, i.e., that for every number z:

- either $z_{opt} \cdot e^{\mathrm{i}\cdot\theta} \succ z$,
- or $z_{opt} \cdot e^{\mathrm{i}\cdot\theta} \sim z$.

If we prove this optimality, then the desired equality will follow from the fact that our optimality criterion is final and therefore, there is only one optimal number (so, since the numbers z_{opt} and $z_{opt} \cdot e^{\mathrm{i}\cdot\theta}$ are both optimal, they must be the same number).

Let us show that $z_{opt} \cdot e^{\mathrm{i}\cdot\theta}$ is indeed optimal. How can we, e.g., prove that $z_{opt} \cdot e^{\mathrm{i}\cdot\theta} \succ z$? Since the optimality criterion is invariant, the desired relation is equivalent to $z_{opt} \succ z \cdot e^{-\mathrm{i}\cdot\theta}$. Similarly, the relation $z_{opt} \cdot e^{\mathrm{i}\cdot\theta} \sim z$ is equivalent to $z_{opt} \sim z \cdot e^{-\mathrm{i}\cdot\theta}$.

These two equivalences allow us to complete the proof of the lemma. Indeed, since z_{opt} is optimal, we have one of the two possibilities:

- either $z_{opt} \succ z \cdot e^{-\mathrm{i}\cdot\theta}$,
- or $z_{opt} \sim z \cdot e^{-\mathrm{i}\cdot\theta}$.

In the first case, we have $z_{opt} \cdot e^{\mathrm{i}\cdot\theta} \succ z$; in the second case, we have $z_{opt} \cdot e^{\mathrm{i}\cdot\theta} \sim z$.

Thus, whatever number z we take, we always have:

- either $z_{opt} \cdot e^{\mathrm{i} \cdot \theta} \succ z$,
- or $z_{opt} \cdot e^{\mathrm{i} \cdot \theta} \sim z$.

Hence, $z_{opt} \cdot e^{\mathrm{i} \cdot \theta}$ is indeed optimal and thence, $z_{opt} \cdot e^{\mathrm{i} \cdot \theta} = z_{opt}$. The lemma is proven.

Let us now prove the theorem. Since the criterion is final, there exists an optimal number z_{opt}. Due to the lemma, the optimal family is invariant. So, $z_{opt} \cdot e^{\mathrm{i} \cdot \theta} = z_{opt}$ for every real number θ. In particular, for $\theta = \pi$, we have $e^{\mathrm{i} \cdot \theta} = -1$ and hence $z_{opt} = -z_{opt}$, i.e., $z_{opt} = 0$. The theorem is proven.