ROBUST IMAGE REGISTRATION USING LOG-POLAR TRANSFORMS

by

Siavash Zokai

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

2004

UMI Number: 3115306

UMI®

UMI Microform 3115306

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and approved by the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

1/23/04

Date

George Wolberg

Professor GEORGE WOLBERG

Chair of Examining Committee

1/23/04

Date

Professor TED BROWN

Executive Officer

Professor SAM FENSTER

Professor ROBERT HARALICK

Professor JIE WEI

Dr. CARL WEIMAN

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

ROBUST IMAGE REGISTRATION USING LOG-POLAR TRANSFORMS

Siavash Zokai

Adviser: Professor George Wolberg

Digital image registration is a branch of computer vision that deals with the geometric alignment of a set of images. A large body of research has been drawn to this area due to its importance in remote sensing, medical imaging, computer graphics, and computer vision. In particular, image registration algorithms for applications such as stereopsis, motion estimation, moving object detection, mosaicing, robotics, and vehicle navigation, have received considerable attention. Despite comprehensive research spanning over thirty years, robust techniques to register images in the presence of large deformations remains elusive. Most techniques fail unless the input images are misaligned by moderate deformations.

The primary objective of this thesis is to investigate robust image registration algorithms for geometrically aligning images subjected to large perspective deformations. State-of-the-art techniques are generally limited to small deformations. Although log-polar techniques have been proposed to accommodate rotation and scale, its use in registering images subjected to perspective distortion has not yet been explored. This thesis introduces the use of log-polar techniques to invert perspective deformations among image pairs. We achieve subpixel accuracy through the use of a hybrid algorithm that features log-polar

mappings and nonlinear least squares optimization. The registration process yields the eight parameters of the perspective transformation that best aligns the two input images. Together these eight parameters completely characterize all naturally occurring geometric transformations in the real worlds of natural and artificial perception. In this thesis, the addition of general perspective transformations fills the transformational gap left by previous methods which were limited to samll deformations.

This thesis contributes several results, including: (1) the introduction of log-polar transforms for inverting large-scale geometric transformations; (2) a study of the effects of affine and perspective transformations in the log-polar domain; (3) a review of the use of various similarity measures, including SSD, cross-correlation, correlation coefficient, and mutual information; (4) investigation of the use of color information to enhance registration accuracy; (5) analysis and comparison of the proposed method against state-of-the-art techniques; (6) an efficient multiresolution implementation; and (7) extensive evaluation over 10,000 image pairs.

The scope of this work shall prove useful for various applications, including the registration of aerial images, the formation of image mosaics, motion stabilization, and diminished reality.

DEDICATIONS

Dedicated to my beloved father and mother

ACKNOWLEDGEMENTS

I am sincerely grateful to my advisor, Prof. George Wolberg, for giving me the privilege and honor to work with him over the last six years. Without Prof. Wolberg's constant support, insightful advice, excellent judgment, and more importantly, his demand for top-quality research, this thesis would not be possible. Even after knowing him for many years, I am continuously amazed and humbled by his unmatched knowledge and wisdom.

I would also like to thank the supervisory committee for their guidance and countless hours in fruitful discussions. I also would like to thank Prof. Stanley Habib and Prof. Ted Brown for their kind support over the years.

Furthermore, I would like to thank my friends and fellow students at the City University of New York for participating in numerous discussions and providing valuable feedback on various elements of this work. I am grateful to Gene Yu for furnishing several test images and for sharing his knowledge in the innumerable discussions that we had. Special thanks are given to Megan Paznik for her most appreciated suggestions and editing comments on my manuscripts. I am grateful to Dr. Nassir Navab who gave me the opportunity to be a visiting researcher at the Siemens Corporate Research Lab in Princeton, New Jersey, during the summer of 2002. Under his direction, I was able to work in an exciting area of research that further contributed to my background in image registration. Through that experience I was able to get my first exposure to a major corporate research laboratory.

Finally, a word for my family, including my parents, brother, and sister without whom it would have been extremely difficult to reach this stage. My parents have been instrumental in furthering my education and helping me to get to this point. I would especially like to thank them for sharing their love with me and for their unwavering support and encouragement.

Contents

List of Figures

Chapter 1

DIGITAL IMAGE REGISTRATION

1.1. INTRODUCTION

Image registration refers to the geometric alignment of a set of images. The set may consist of two or more digital images taken of a single scene at different times, from different sensors, or from different viewpoints. The goal of registration is to establish geometric correspondence between the images so that they may be transformed, compared, and analyzed in a common reference frame. This is of practical importance in many fields, including remote sensing, medical imaging, and computer vision. Registration is often necessary for

1. integrating information taken from different sensors (i.e., multisensor data fusion),
2. finding changes in images taken at different times or under different conditions
3. inferring three-dimensional information from images in which either the camera or the objects in the scene have moved
4. for model-based object recognition.

The most common task associated with image registration is the generation of large panoramic images for viewing and analysis. Image mosaics, created by warping and blending together several overlapping images, are central to this process. Other common registration tasks include producing super-resolution images from multiple images of the same scene, change detection, motion stabilization, topographic mapping, and multisensor image fusion.

This work attempts to register two images using one global perspective transformation. In general, images of a 3D scene do not differ by just one perspective transformation because the depth between the camera and the objects introduces parallax. A global transformation cannot align all features in such cases. We must therefore place constraints on camera motion and/or our 3D scene to produce images that are free of parallax. One constraint requires the camera motion to be limited to rotation, pan, tilt, and zoom about a fixed point, e.g, on a tripod. If this constraint is not satisfied, then we may still have images free of parallax if the object's 3D points (x, y, z) in the scene are far away from the camera, i.e., $x, y \ll z$. This means that the scene is flat and we are looking at a planar object. In either case, we assume that the scene is static and the lighting is fixed between images. Nevertheless, we have relaxed these conditions.

The scope of this work shall prove useful for various applications, including the registration of aerial images, the formation of image mosaics, and super-resolution. Note that aerial imagery may be acquired from uncalibrated airborne cameras subjected to yaw, pitch, and roll at various altitudes. Since the terrain appears flat from moderately high altitude, it is an ideal candidate for registration using a single perspective transformation.

1.2. REGISTRATION FRAMEWORK

A recent survey by Brown [20] introduces a framework in which all registration techniques can be understood. The framework consists of the feature space, similarity measure, search space, and search strategy. The *feature space* extracts the information in the images that will be used for matching. The *search space* is the class of transformations, or deformation models, that is capable of aligning the images. The *search strategy* decides how to choose the next transformation from this space, to be tested in the search for the optimal transformation. The *similarity measure* determines the relative merit for each test. Search continues according to the search strategy until a transformation is found whose similarity measure is satisfactory.

For image registration, we need to recover the geometric transformation and/or intensity function. Let $I_1(u, v)$ and $I_2(x, y)$ be the reference and observed images, respectively. The relationship between these images is $I_1(u, v) = I_2(T\{(x, y)\})$, where T is a 2D geometric transformation operator that relates the (x, y) coordinates in I_2 to the (u, v) coordinates in I_1. With the exception of occlusion, any 3D source point that projects to I_1 also projects to I_2. Images I_1 and I_2 are said to be registered, or in correspondence, when a mapping function exists between them.

1.3. FEATURE SPACE

The feature space refers to the collection of extracted measurements from the input images. This requires a preprocessing step in image registration. Normally, we measure one or more criteria for a pixel or a local region. Then, we stack these measurements as a vector that represents a point in the feature space. The choice of the criteria is dependent on the image domain. The most widely used features are pixel intensities and edge gradients.

Other more complex criteria are possible, such as the output of the Harris corner detector [52], Hough transform, and 2D moments [106].

There has been much research activity in feature extraction for the newly emerging field of digital libraries. Due to the explosive growth of digital multimedia, researchers are studying different criteria for feature selection to efficiently and effectively index images and videos. In this thesis, however, we make no assumptions about image content and so we limit our features to pixel intensities and gradients to permit all pixels to vote for the optimal solution. In general, we use pixel intensities as our image features for moderate illumination variation between the input images. Gradients are used only when substantial illumination variation is present. Note that the use of intensity features requires no preprocessing time and thereby simplifies hardware implementation.

1.4. SEARCH SPACE

The search space consists of deformation model parameters necessary to define the spatial transformations that may apply between any two images, I_1 and I_2, of the same scene. This defines a geometric relationship between each point in both images. The general form for the mapping function induced by the deformation is:

$$[x, y] = [X(u, v), Y(u, v)] \tag{1.1}$$

where $[u, v]$ and $[x, y]$ denote corresponding pixels in I_1 and I_2, respectively, and X and Y, are arbitrary mapping functions that uniquely specify the spatial transformation. In registering I_1 and I_2, we shall be interested in recovering the inverse mapping functions U

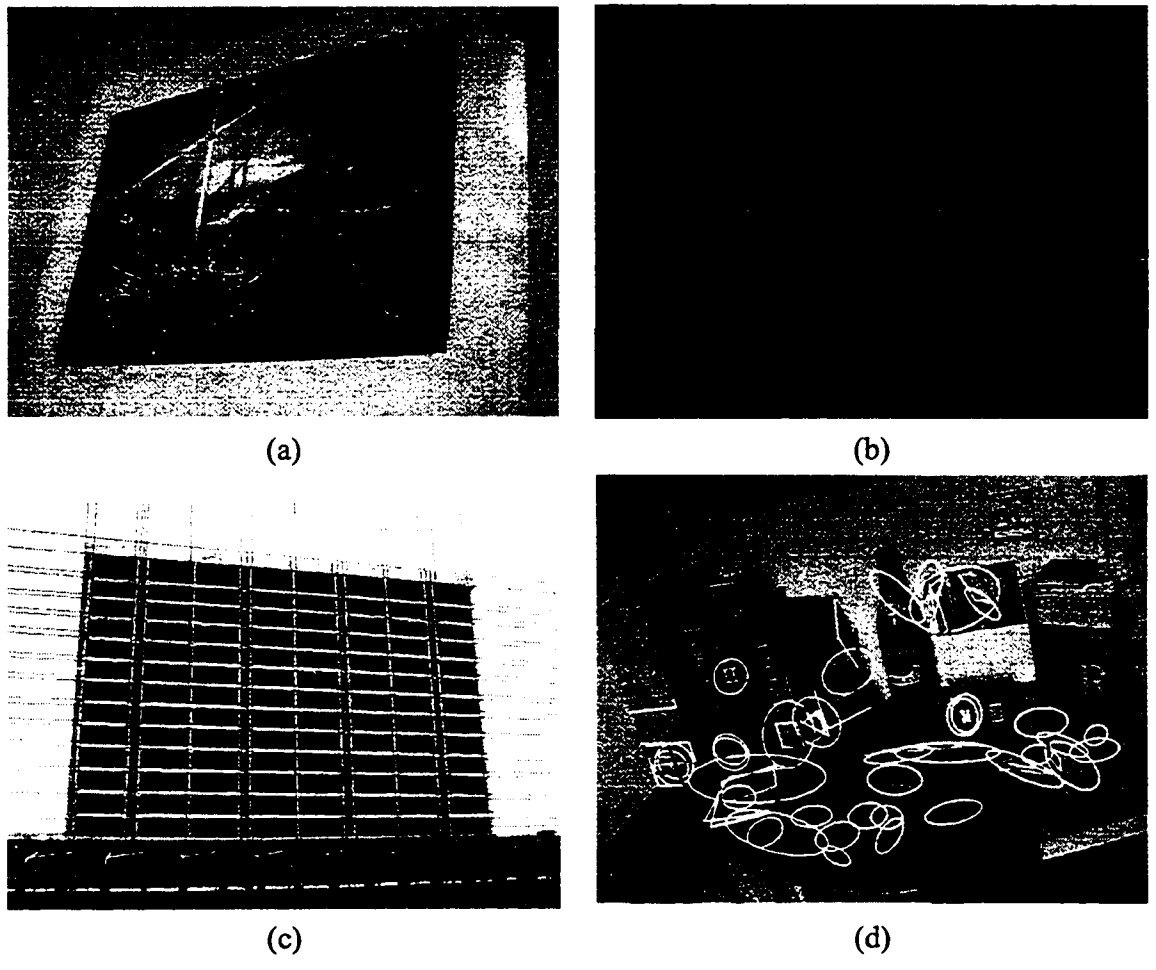


Figure 1.1: Feature space: (a) Harris corner detector (points); (b) gradient image; (c) Hough Transform (lines); and (d) invariant affine region (elliptical regions).

and V that transform I_2 back into I_1:

$$[u, v] = [U(x, y), V(x, y)] \tag{1.2}$$

In this sense, the registration process is akin to geometric correction, whereby we attempt invert the distortion.

Deformation models have been studied in [118]. The choice of deformation model must be closely tied to the imaging process by which the input data set was acquired.

The complexity of the model dictates the choice of registration algorithm. Images which are known to differ by small translations, for example, require only a simple correlation technique to perform registration. Images which differ by elastic deformations, though, may require user-supplied correspondence points. These points, also known as control, fiducial, and tie points, are marked by the user at sparse and irregular landmark sites on the images. Since the mapping function is precisely known at only those points, a scattered data interpolation algorithm must be applied to smoothly propagate the mapping to all other points in the image [65]. Note that in the former case the mapping function was derived directly from the image intensities, while in the latter case the mapping function was constructed from sparse correspondence points.

Geometric correspondence is achieved by determining the mapping function that governs the relationship of all points among a pair of images. There are several common mapping function models in image registration. They include:

- Similarity transformation: translation, rotation, uniform scale (4 parameters)
- Affine transformation: translation, rotation, scale, shear (6 parameters)
- Perspective transformation: affine, perspective foreshortening (8 parameters)
- Local transformation: terrain relief, nonlinear, nonrigid

1.4.1 Similarity Transformation

A similarity transformation preserves angles and changes all distances in the same ratio. It accounts for object or sensor movement in which objects in the images retain their relative shape and size. The transformation is composed of rotation, scale, and translation. These multiple transforms can be collapsed into a single composite transformation by tak-

ing the product of their 3×3 matrices. Note that matrix multiplication is associative, but not commutative.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M_{comp} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.3}$$

where

$$\begin{aligned} M_{comp} &= \begin{bmatrix} 1 & 0 & t_u \\ 0 & 1 & t_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} s\cos\theta & s\sin\theta & t_u \\ -s\sin\theta & s\cos\theta & t_v \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{1.4}$$

Fig. 1.2(a) depicts a similarity transformation, accounting for rotation, uniform scale, and translation of an image.

1.4.2 Affine Transformation

The general representation of an *affine* transformation is:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.5}$$

where a_i are arbitrary real numbers. An affine mapping is characterized by a transformation matrix whose last row is equal to [0 0 1]. This corresponds to an orthographic or parallel plane projection from the source uv-plane onto the target xy-plane. As a result, affine mappings preserve parallel lines, allowing us to avoid foreshortened axes when performing 2D projections. Furthermore, equi-spaced points are preserved and affine transformations accommodate planar mappings. For instance, they can map triangles to triangles. Fig. 1.2(b) depicts an affine transformation, accounting for rotation, scale, translation, and shear.

1.4.3 Perspective

The general representation of a perspective transformation is:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{1.6}$$

where $u = u'/w'$, $v = v'/w'$, and a_i are arbitrary real numbers. A perspective transformation, or projective mapping, is produced when $[a_6 a_7]^T$ is nonzero. It is used in conjunction with a projection onto a viewing plane in what is known a perspective or central projection. Perspective transformations preserve parallel lines only when they are parallel to the projection plane. Otherwise, lines converge to a vanishing point.

Perspective transformations share several important properties with affine transformations. They are planar mappings, and their forward and inverse transforms are single-valued. They preserve lines in all orientations. That is, lines map onto lines, albeit not necessarily of the same orientation. Furthermore, the eight degrees of freedom in the transformation matrix are sufficient to permit planar quadrilateral-to-quadrilateral mappings. In

contrast, affine transformations offer only six degrees of freedom and thereby facilitate only triangle-to triangle mappings. Fig. 1.2(c) depicts a perspective transformation, accounting for rotation, scale, translation, shear, and foreshortening effects.

1.4.4 Polynomial Transformations

Geometric correction requires a spatial transformation that inverts an unknown distortion function. The mapping functions, U and V, have been almost universally chosen to be global bivariate polynomial transformations of the form:

$$u = \sum_{i=0}^{N} \sum_{j=0}^{N-i} a_{ij} x^i y^j \tag{1.7a}$$

$$v = \sum_{i=0}^{N} \sum_{j=0}^{N-i} b_{ij} x^i y^j \tag{1.7b}$$

where a_{ij} and b_{ij} are the constant polynomial coefficients and the number of unknown parameters are $2N(N+1)$. The polynomial transformations given above are low-order global mapping functions operating on the entire image. This form is typical of those used for remotely sensed images. The transformation for $N = 1$ is an affine transformation. Transformations for $N > 1$ are often used to account for internal sensor errors, such as lens distortions and pincushion effects, as well as external distortions, such as earth curvature. An example of a polynomial distortion for $N = 2$ is shown in Fig. 1.2(d).

1.5. SIMILARITY MEASURES

Image registration seeks to find the transformation that most closely aligns, or matches, a pair of images. Central to this goal is the need to quantify what we mean by a "good match." In this section, we review several popular similarity measures that have been in-

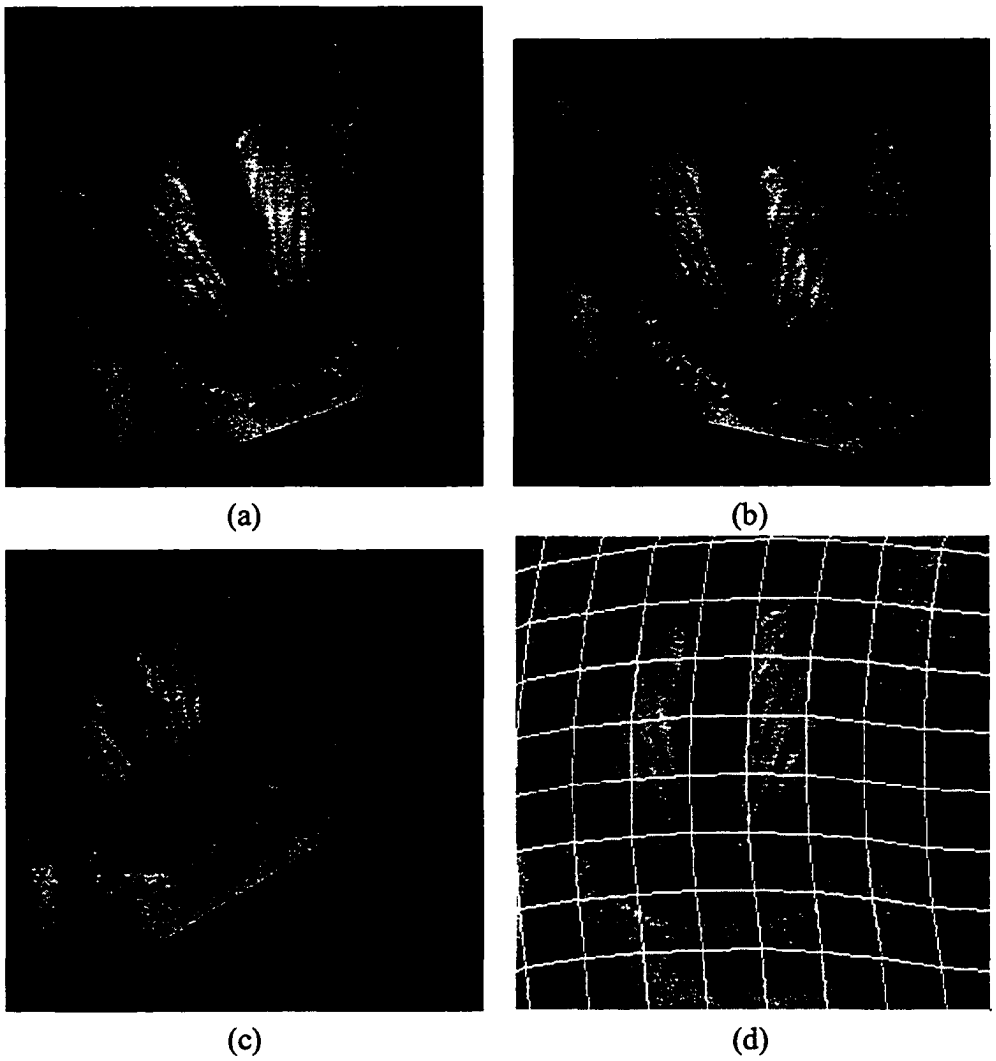


Figure 1.2: Deformation models: (a) similar (RST); (b) affine; (c) perspective; (d) polynomial.

troduced to quantify the distance, or error, between two images.

1.5.1 Correlation Methods

Correlation techniques are among the oldest statistical methods for matching two sets of data. They are used extensively in low-level signal processing and image registration. We begin by introducing several correlation operations, ordered by their computational

complexity: cross-correlation, normalized cross-correlation, and correlation coefficient.

Let I and T denote the reference and template images, respectively. The *cross-correlation* function is defined as:

$$C_{cross}(u,v) = \sum I(x-u, y-v)T(x,y) \tag{1.8}$$

where (x,y) are positions within a neighborhood of (u,v). The *normalized cross-correlation* function is defined as:

$$C_{norm}(u,v) = \frac{\sum I(x-u, y-v)T(x,y)}{\sqrt{\sum I(x-u, y-v)^2 \sum T(x,y)^2}} \tag{1.9}$$

The *correlation coefficient* function is defined as:

$$C_{coef}(u,v) = \frac{\sum (I(x-u, y-v) - \mu_1)(T(x,y) - \mu_2)}{\sqrt{\sum (I(x-u, y-v) - \mu_1)^2 \sum (T(x,y) - \mu_2)^2}} \tag{1.10}$$

where μ is the average image intensity in a neighborhood about (u,v).

If I matches T at translation (u,v), all correlation methods will have their peaks at $C(u,v)$. Fig. 1.3 depicts a geometric interpretation for the correlation coefficient method. If we let α denote the angle between vector $I - \mu_1$ and $T - \mu_2$, then the correlation coefficient is $\cos\alpha$. Due to its robustness in the presence of Gaussian noise and illumination changes, the correlation coefficient method is the most popular correlation method.

Correlation methods can only recover translation (shifts) between two images. Since they are simple to implement in hardware and software, they are widely used [11, 84, 5, 57, 86]. An ideal example is shown in Fig. 1.4, which shows a template image in Fig. 1.4(b) directly cropped from the reference image in Fig. 1.4(a). The associated cross-correlation and normalized cross-correlation surfaces are shown in Fig. 1.5(a) and Fig. 1.5(b), respec-

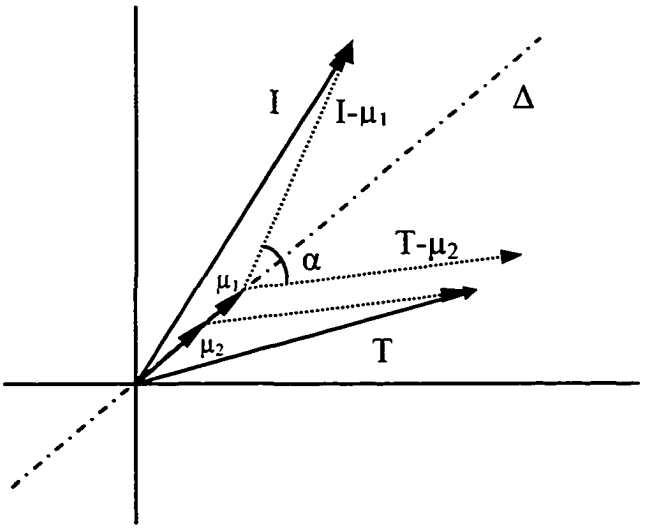


Figure 1.3: Geometric interpretation of the correlation coefficient

tively. The maxima in these images are not readily distinguishable. Note that there are many hills that have similar heights. This condition could lead to false matches. However, the use of the correlation coefficient method gives us a distinguished peak, as shown in Fig. 1.5(c).

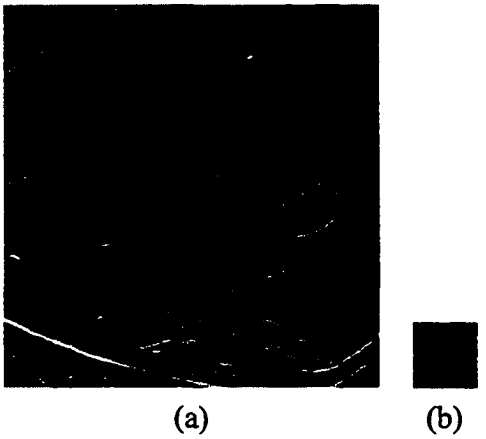


Figure 1.4: Input images: (a) reference I; (b) template T.

We have tested the robustness of the correlation coefficient method under several con-

Figure 1.5: Different correlation surfaces applied to the I and T of Fig. 1.4. (a) Cross-correlation; (b) Normalized cross-correlation; and (c) Correlation coefficient

ditions, including additive Gaussian noise, linear grayscale transformation, and geometric deformation. We have added zero mean Gaussian noise to the reference and template images. The correlation coefficient values are plotted against the signal to noise ratio in Fig. 1.6. We are able to find matches for very low signal to noise ratios (12 db). This tolerance is satisfactory for our applications. An array of images depicting additive Gaussian noise applied to the reference and template images is shown in Fig. 1.7.

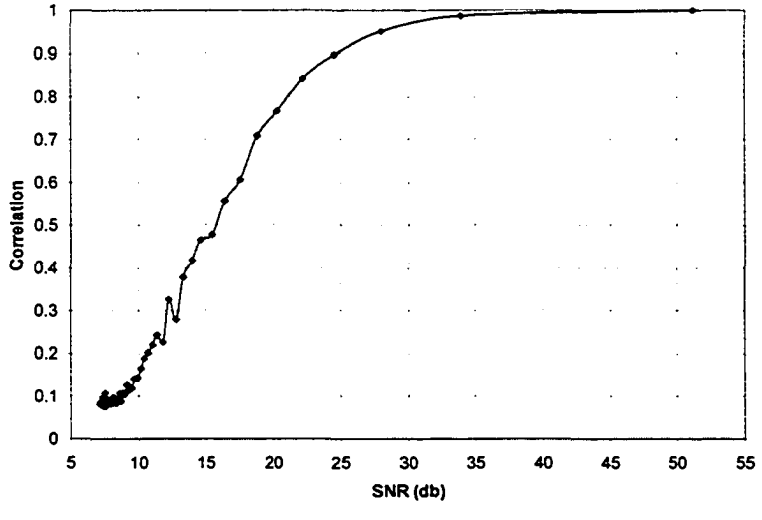


Figure 1.6: Correlation vs. SNR. The correlation values are not significant for very low SNR and the maximum is not detectable.

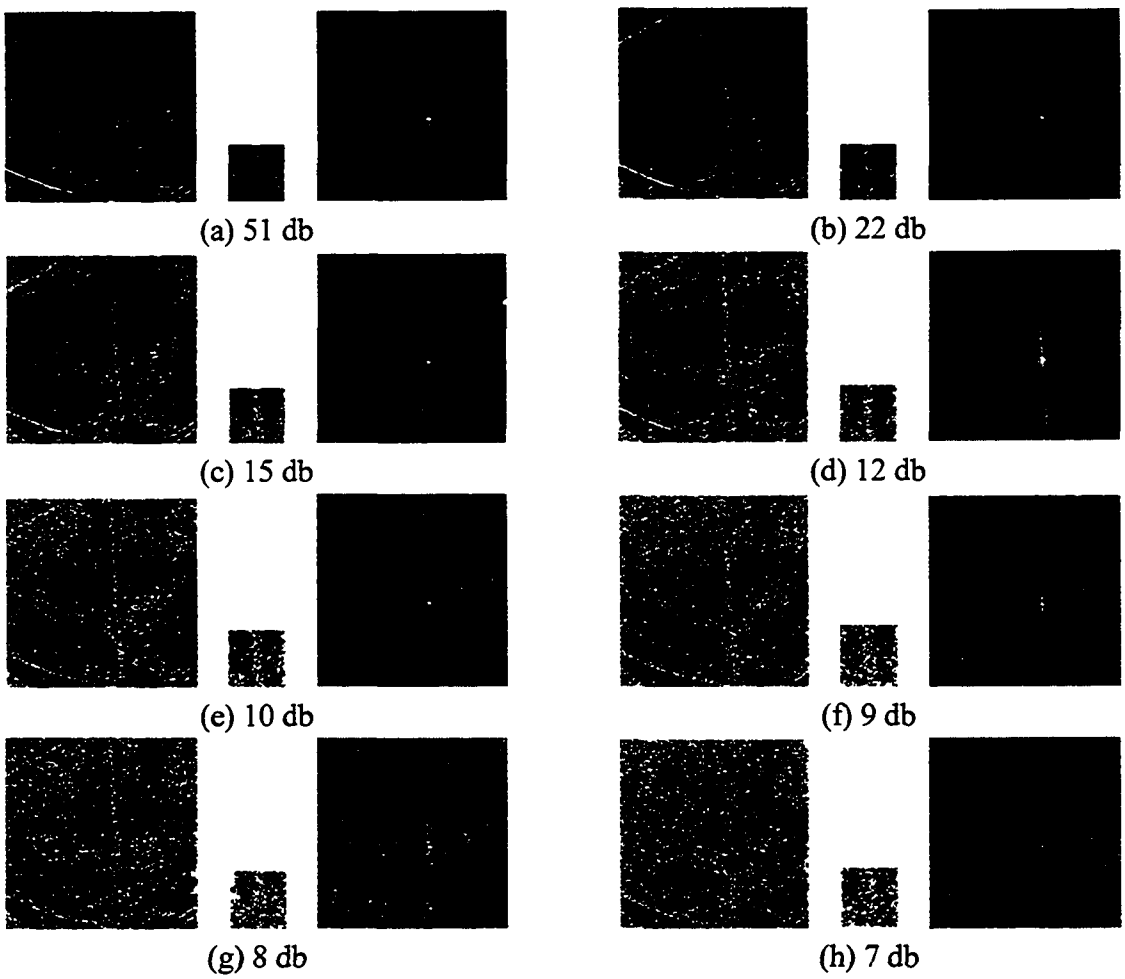


Figure 1.7: Effects of additive Gaussian noise on localizing correlation coefficient maxima.

We have generated a series of reference images by applying the following linear grayscale transformation to the original reference image shown in Fig. 1.4(a):

$$I'_{ref} = aI_{ref} + b \tag{1.11}$$

Fig. 1.8 shows a plot of the correlation values vs. the two parameters (a, b). The plot demonstrates that the correlation coefficient method is insensitive to linear global illumination. This result conforms with results documented in many research papers. An array of linearly transformed reference images and their respective correlation surfaces are shown

in Fig. 1.9.

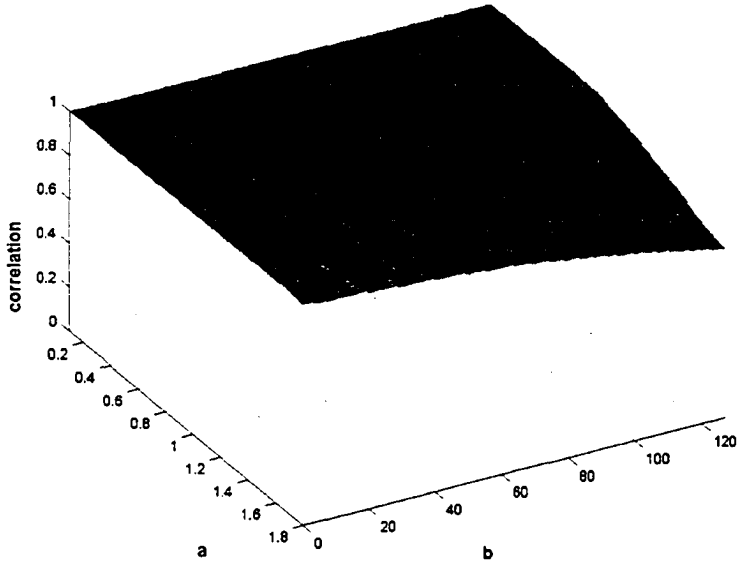


Figure 1.8: The correlation values are plotted for varieties of linear grayscale transformation $(aI_{ref} + b)$.

Finally, we inspect the robustness of the correlation coefficient method under geometric transformations. For instance, we rotate the image plane about the vertical axis between $-90°$ and $90°$. This rotation introduces nonlinear geometric deformations. We have plotted the correlation values vs. rotation in Fig. 1.10. The plot demonstrates that the correlation values quickly degenerate, which yields false matches. After extensive experimentation with a variety of geometric transformations, we have concluded that the correlation coefficient method is not robust in the presence of large geometric transformations. It is important to note that the correlation coefficient method is a metric tool in the Euclidean space. Therefore, slight deformations in the feature space may cause large errors in the Euclidean space.

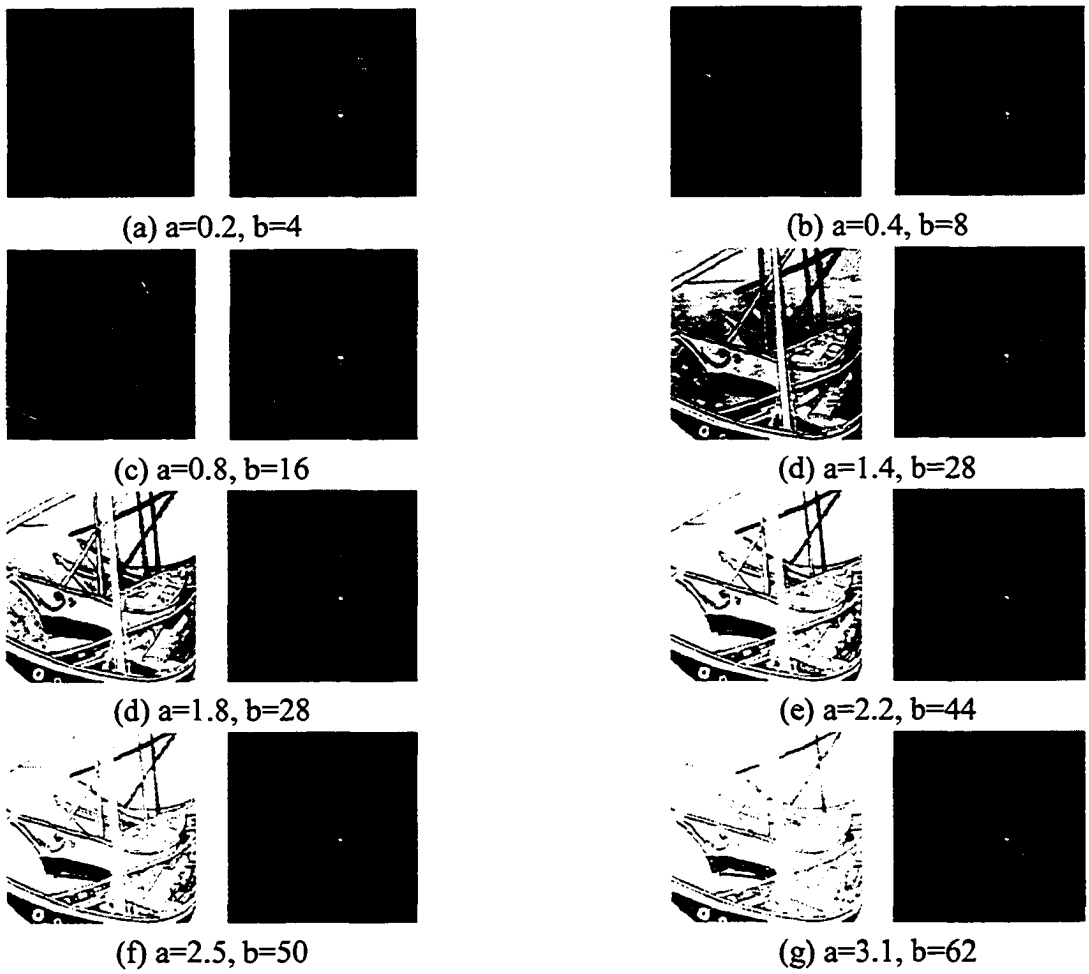


Figure 1.9: Effects of linear grayscale $(aI + b)$ transformation on localizing correlation coefficient maxima. The template is identical to that shown in Fig. 1.4.

1.5.2 Sum of Differences Method

The sum of differences method is another popular similarity measure. It attempts to measure the distance between images, defined in Euclidean space. There are two variants in wide use: the absolute difference (L^1 norm) and the sum of squared differences (L^2 norm). The absolute difference is defined as

$$C_{SAD}(u,v) = \sum |I(x-u, y-v) - T(x,y)| \tag{1.12}$$

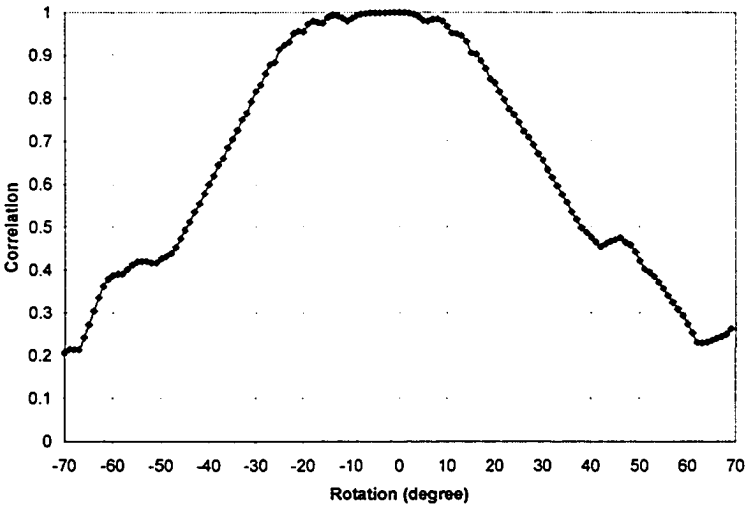


Figure 1.10: The correlation values are plotted for a rotating image plane.

The sum of squared differences is defined as

$$C_{SSD}(u,v) = \sum (I(x-u, y-v) - T(x,y))^2 \tag{1.13}$$

The sum of squared differences is more popular in the registration, optic flow, and stereo matching literature [104, 121, 4, 14, 15, 55, 101, 102]. In these applications, optimization techniques requires the computation of the first derivative of the objective function with respect to the transformation parameters. The square term in the SSD method permits the objective function to be easily differentiated. That same square operation, however, is responsible for exaggerating the role of outliers, thereby rendering the method to be less robust.

The SAD method, however, uses the absolute value operation, permitting it to be more robust than SSD. However, it is less frequently used in many applications that require optimization because the absolute value operation gives rise to a discontinuity in derivative. Both methods, however, work well in the presence of additive white Gaussian noise.

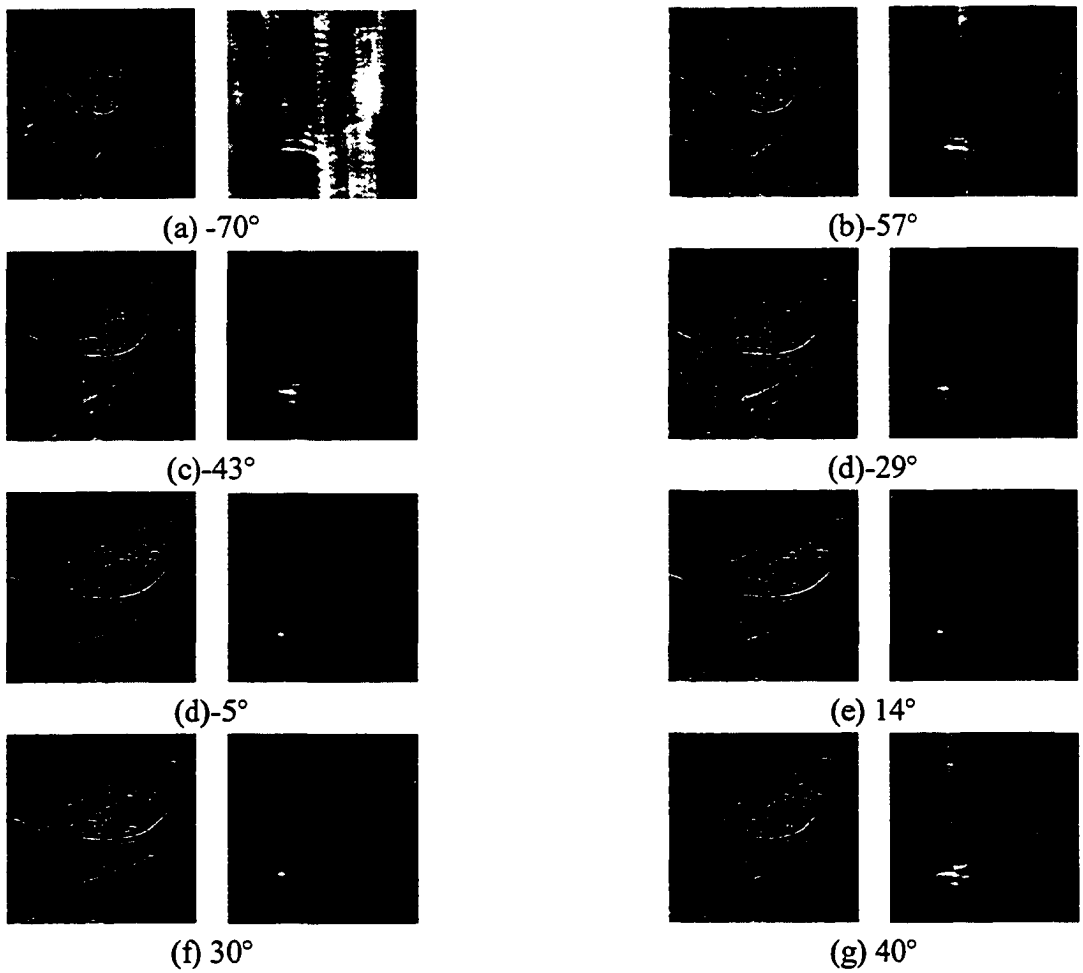


Figure 1.11: Effects of rotation on localizing correlation coefficient maxima. The template is identical to that shown in Fig. 1.4.

1.5.3 Color Similarity Measure

Many researchers have concentrated in the past few decades on devising algorithms for grayscale image understanding. Both the cross-correlation and the sum of squared differences formulations are suitable for grayscale images. We see and capture the real world in color, but we map color images to grayscale images in many computer vision algorithms for simplicity. We should note that we lose information in this conversion and many (R, G, B) colors map to the same gray level Y, whereby $Y = 0.299R + 0.587G + 0.114B$. Thus, two regions with similar texture and different colors will not be distinguishable in

(a) SSD (b) Mutual information (c) Phase Correlation

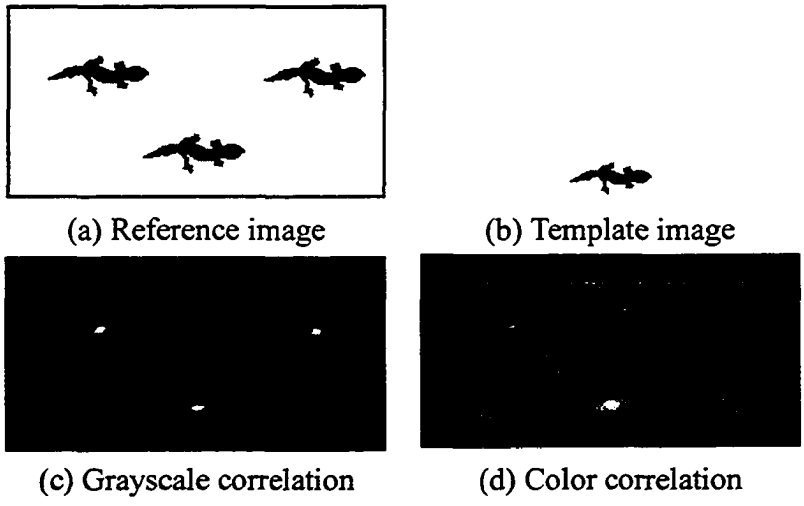
Figure 1.12: Similarity measure surfaces applied to the I and T of Fig. 1.4

the grayscale image. With the advent of powerful personal computers, it is now possible and practical to move to the more computationally intensive realm of color image understanding. We extend these similarity measures to color space for improving accuracy. For instance in Fig. 1.13, color reference and template images are shown. If we perform cross-correlation in grayscale, we will find three easily distinguishable peaks of the same height on the correlation surface, as shown in Fig. 1.13(c). This ambiguity poses a considerable problem in many block matching algorithms. Performing cross-correlation in color space gives us one unique peak, as shown in Fig. 1.13(d). There are many ways to represent color space including RGB, CMYK, YIQ, and CIE La*b* [50]. We chose the CIE La*b* color space because it has been designed to be a perceptually uniform space. A system is perceptually uniform if a small perturbation to a component value is approximately equally perceptible across the range of that value [47]. We define the color version of the normalized cross-correlation and SSD methods as follows:

$$C_{coef}(u,v) = \frac{\sum (I_L T_L + I_{a^*} T_{a^*} + I_{b^*} T_{b^*})}{\sqrt{\sum (I_L^2 + I_{a^*}^2 + I_{b^*}^2) \sum (T_L^2 + T_{a^*}^2 + T_{b^*}^2)}} \quad (1.14)$$

$$C_{SSD}(u,v) = \sum [(I_L - T_L)^2 + (I_{a^*} - T_{a^*})^2 + (I_{b^*} - T_{b^*})^2] \quad (1.15)$$

where it is understood that I_L, I_a-, and I_b- are indexed by $(x-u, y-v)$, and templates T_L, T_a-, and T_b- are indexed by (x, y).



(a) Reference image (b) Template image

(c) Grayscale correlation (d) Color correlation

Figure 1.13: Correlation surfaces computed for grayscale and color images. Note that correlation in the color space properly detects a single match, whereas grayscale correlation detects three matches.

1.5.4 Phase-Correlation Method

Consider two images I_1 and I_2 which differ by a displacement (dx, dy), i.e. $I_1(x, y) = I_2(x - dx, y - dy)$. Their corresponding Fourier transforms I_1 and I_2 are related by:

$$F_1(\omega_x, \omega_y) = F_2(\omega_x, \omega_y) e^{j(\omega_x d_x + \omega_y d_y)} \tag{1.16}$$

or equivalently,

$$\frac{F_1(\omega_x, \omega_y) F_2^*(\omega_x, \omega_y)}{\|F_1(\omega_x, \omega_y) F_2^*(\omega_x, \omega_y)\|} = e^{j(\omega_x d_x + \omega_y d_y)} \tag{1.17}$$

If we compute the inverse Fourier transform of the cross-power spectrum, we get an impulse in the spatial domain that is approximately zero everywhere except at the displacement (dx, dy). This is a fast and robust method for recovering translation, even in the presence of noise [63, 26, 95, 68]. Papadimitriou demonstrated the use of phase correlation for finding disparity maps for stereo images [80]. Virtually all results based on phase correlation have traditionally been computed with integer accuracy. Recent work has been directed at showing how to extend the phase correlation method to subpixel accuracy [98, 96].

Phase correlation is equivalent to cross-correlation in the spatial domain based on the following Fourier property:

$$I_1(\mathbf{X}) \star I_2(-\mathbf{X}) = F_1(\omega)F_2^*(\omega) \tag{1.18}$$

In Fig. 1.5 (c), the phase correlation surface is shown with a peak for the best match (translation) between two images in Fig. 1.4.

1.5.5 Mutual Information

Mutual information is a new similarity measure that has recently been introduced to the digital image processing community. The mutual information of two images is a measure of how much knowledge one gains about one image by being given the other image [83]. Mutual information is a measure originating from information theory [32]. It was first proposed independently by Viola and Collignon in 1995 as a similarity measure for medical image registration [108, 31]. The underlying concept of mutual information is entropy, which can be thought of as a measure of dispersion in the distribution of the image

histogram.

$$H(X) = -\sum P(X) \log P(X) \tag{1.19a}$$

$$H(X,Y) = -\sum P(X,Y) \log P(X,Y) \tag{1.19b}$$

where $P(X,Y)$ is the joint probability distribution of images X and Y. Given images X and Y, the mutual information $I(X,Y)$ of these images is defined as:

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) \tag{1.20}$$

The mutual information similarity measure is useful when there is an unknown functional relation between the intensities of the X and Y. If the functional relation between X and Y is linear, mutual information and the correlation coefficient method are related to each other as follows:

$$MI(X,Y) = \frac{1}{2} \log C_{coef}(X,Y) - 1 \tag{1.21}$$

In Fig. 1.5(b), the mutual information has been calculated for every position in reference image Fig. 1.4(a). Notice that a strong peak coincides with the correct displacement.

1.6. SEARCH STRATEGIES

Given an appropriate deformation model, the registration process searches over all parameters to find the best transformation T that most closely aligns the observed and reference images. In this section, we review two important search strategies useful for deriving T: linear and nonlinear least squares optimization. These search techniques repeatedly

invoke a similarity measure to determine whether the iterative process is converging towards a solution of minimal error. In addition, we review the Fourier-Mellon method for recovering a similar transformation comprising rotation, scale, and translation.

1.6.1 Linear Least-Squares Optimization

This search method has been used for fitting a linear model to a data set. It has been used in the image registration community when the deformation model is linear or affine [3]. Consider two images, I_1 and I_2, that are misaligned by an affine transformation.

$$u = a_1x + a_2y + a_3 \tag{1.22a}$$

$$v = a_4x + a_5y + a_6 \tag{1.22b}$$

The relationship between their respective (u, v) and (x, y) coordinate systems is given in Eq. (1.22). We estimate the local pixel motion between two images by assuming that the intensities of corresponding points in both images must remain the same. Therefore, the mapping between pixels in images I_1 and I_2 is

$$\begin{aligned} I_2(x, y) &= I_1'(x, y) \\ &= I_1(x + dx, y + dy) \end{aligned} \tag{1.23}$$

where

$$dx = u - x = (a_1 - 1)x + a_2y + a_3 \tag{1.24a}$$

$$dy = v - y = a_4x + (a_5 - 1)y + a_6 \tag{1.24b}$$

Eq. (1.23) states that the correspondence for each (x, y) position in I_2 lies at a displacement of (dx, dy) in I_1. The displacement is computed with an affine transformation.

The key constraint in Eq. (1.23) is *intensity constancy*. It is fundamental to optic flow techniques, where the intensity is assumed to remain constant along a motion trajectory. This is a reasonable constraint as long as the changing effects of illumination, shadows, and viewing geometry remain small. For this reason, optic flow has proven useful for determining the limited motion present between successive frames in an image sequence. We may now derive the following expression for I_2 in terms of I_1, its gradients, and the affine parameters:

$$\begin{aligned} I_2(x,y) &= I_1(x+dx, y+dy) \\ &= I_1(x,y) + \frac{\partial I_1(x,y)}{\partial x}dx + \frac{\partial I_1(x,y)}{\partial y}dy \\ &= I_1(x,y) + \frac{\partial I_1(x,y)}{\partial x}((a_1-1)x + a_2 y + a_3) + \frac{\partial I_1(x,y)}{\partial y}(a_4 x + (a_5-1)y + a_6) \end{aligned} \tag{1.25}$$

Note that the approximation of I_1 by the first-order expansion of the Taylor series is adequate to recover the affine transformation parameters. Were we attempting to recover a higher-order motion model, then additional terms in the Taylor series would be required. Also, note that dx and dy are defined in terms of x and y, and therefore vary across the image. This furnishes a system of linear equations that expresses the relationship between pixel values $p_i = I_1(u_i, v_i)$ and $q_i = I_2(x_i, y_i)$, positions $[x_i, y_i]$, directional derivatives $[\frac{\partial p_i}{\partial x}, \frac{\partial p_i}{\partial y}]$, and the inverse mapping affine parameters $\mathbf{a} = [a_1, a_2, a_3, a_4, a_5, a_6]^T$. Letting

Eq. (1.25) hold over all N pixels, we can write the equations in matrix form $\mathbf{Y} = \mathbf{XA}$:

$$\begin{bmatrix} q_1 - p_1 \\ q_2 - p_2 \\ \cdots\cdots \\ \cdots\cdots \\ \cdots\cdots \\ q_N - p_N \end{bmatrix} = \begin{bmatrix} \frac{\partial p_1}{\partial x}x & \frac{\partial p_1}{\partial x}y & \frac{\partial p_1}{\partial x} & \frac{\partial p_1}{\partial y}x & \frac{\partial p_1}{\partial y}y & \frac{\partial p_1}{\partial y} \\ \frac{\partial p_2}{\partial x}x & \frac{\partial p_2}{\partial x}y & \frac{\partial p_2}{\partial x} & \frac{\partial p_2}{\partial y}y & \frac{\partial p_2}{\partial y}y & \frac{\partial p_2}{\partial y} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \frac{\partial p_N}{\partial x}x & \frac{\partial p_N}{\partial x}y & \frac{\partial p_N}{\partial x} & \frac{\partial p_N}{\partial y}x & \frac{\partial p_N}{\partial y}y & \frac{\partial p_N}{\partial y} \end{bmatrix} \begin{bmatrix} a_1 - 1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 - 1 \\ a_6 \end{bmatrix} \tag{1.26}$$

This system of equations is overdetermined since N is greater than the six unknown parameters. We can solve for $\mathbf{A}$ by using the pseudoinverse solution of the linear least-squares problem:

$$\mathbf{A} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{1.27}$$

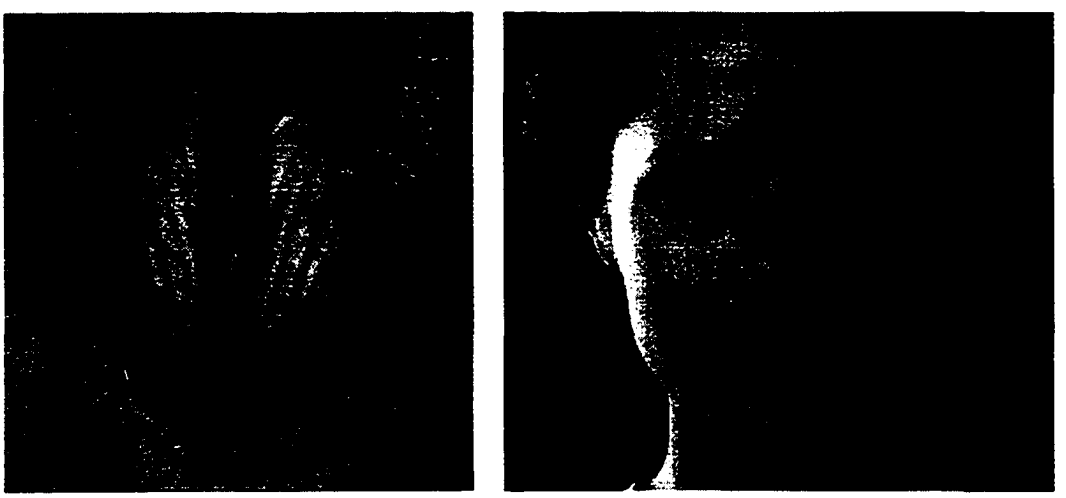
This yields the desired affine parameters **a**. It is well-known that the parameters recovered by the pseudoinverse solution minimize the SSD error measure:

$$\begin{aligned} \chi^2(\mathbf{a}) &= \sum_{i=1}^{N} e_i^2 \\ &= \sum_{i=1}^{N} (I_2(x_i, y_i) - I_1(x_i + dx, y_i + dy))^2 \\ &= \sum_{i=1}^{N} \left((I_2(x_i, y_i) - I_1(x_i, y_i)) - \frac{\partial I_1(x_i, y_i)}{\partial x}dx - \frac{\partial I_1(x_i, y_i)}{\partial y}dy \right)^2 \\ &= \sum_{i=1}^{N} (\Delta I_i - \nabla I_i d\mathbf{x})^2 \end{aligned} \tag{1.28}$$

A review of the linear least-squares method, including a derivation of the pseudoinverse solution, is given in Appendix A. All registration techniques require an objective

criterion to establish a similarity measure between two images. By applying the pseudoinverse solution to solve this linear least-squares problem, we have implicitly selected the SSD as our objective criterion. The recovery of affine parameters a is thereby posed as the solution to an optimization problem that minimizes the SSD error measure.

This registration method suffers from two principal sources of error. One source of error stems from the first-order Taylor series approximation of Eq. (1.25). It is well-known that the approximation is accurate only for small values of dx and dy. Therefore, large transformations cannot be recovered. The second source of error stems from the pseudoinverse solution. Multiplication with $\mathbf{X}^T$ in Eq. (1.26) squares the condition number, thereby reducing the precision of the coefficients by one half. The limitations due to the Taylor series approximation remain the most restrictive in this case. It would be virtually impossible, for instance, to recover affine transformations involving rotation angles beyond 20°, scale changes beyond 20%, and large translations. The examples below demonstrate the range of transformations that can be accommodated with linear least squares.

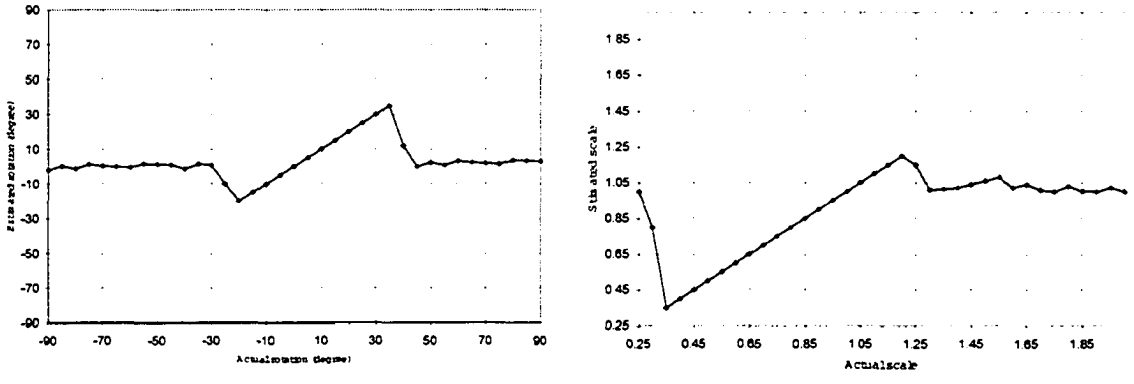


(a) (b)

Figure 1.14: Input images: (a) Mandrill; (b) Tiffany.

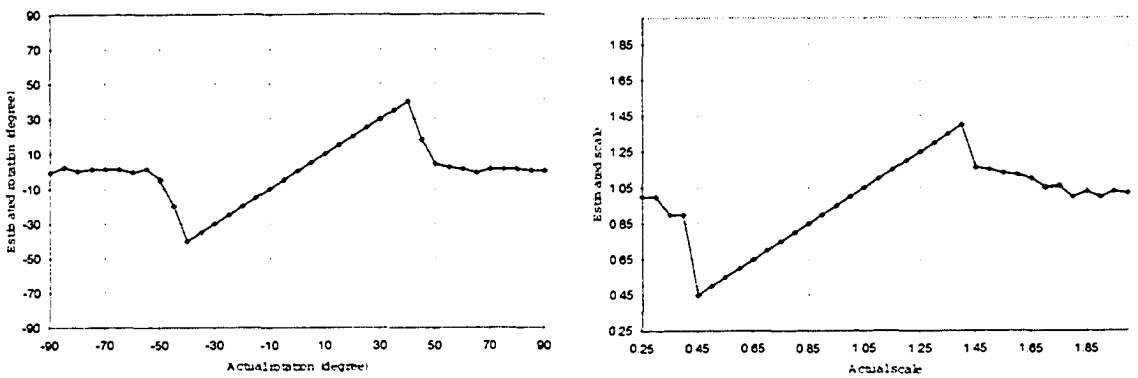
Consider the mandrill and Tiffany images shown in Fig. 1.14. They will serve as our test images for various registration methods. Fig. 1.15 shows the registration accuracy of the linear least-squares method for the case of rotation and scale. In Fig. 1.15(a), the mandrill image was registered against rotated versions of itself. The applied rotation angles varied from $-90°$ to $90°$, in $5°$ increments. The figure shows that the linear least-squares registration method remains accurate only in the $[-25°, 35°]$ range. The algorithm stuck in the initial guess outside of this range. In Fig. 1.15(b), the mandrill image was registered against scaled versions of itself. The applied scale factors varied from 0.25 to 2, in 0.1 increments. In this case, the linear least-squares registration method remains accurate only in the [.35, 1.2] range.

Fig. 1.16 demonstrates similar results on the Tiffany image. The figure shows that the linear least-squares registration method remains accurate in the $[-35°, 35°]$ range for rotation, and [.45, 1.4] range for scale. This improvement is due to the smooth surface of the Tiffany image.



(a) (b)

Figure 1.15: Linear least squares registration accuracy for the mandrill image: (a) rotation; (b) scale.



(a) (b)

Figure 1.16: Linear least squares registration accuracy for the Tiffany image: (a) rotation; (b) scale.

1.6.2 Nonlinear Least-Squares Optimization

In this section, we demonstrate how to search for the affine and perspective parameters using nonlinear least-squares optimization. There is a vast literature of work in the related fields of image registration, motion estimation, image mosaics, and video indexing that have used nonlinear least-squares optimization. Most algorithms exploit a hierarchical approach due to computational efficiency in handling large displacements. Hierarchical motion estimation [14, 4, 15] and image mosaic [30, 55, 101, 102, 103, 97, 72, 33] algorithms usually assume small deformations among image pairs. For instance, a dense image sequence is required to stitch the frames together [102, 72]. The problem of assembling a large set of images into a common reference frame is simplified when the inter-frame deformations are small. We use the sum of squared differences (SSD) as the objective criterion that establishes a similarity measure between two images (or regions):

$$\chi^2(\mathbf{a}) = \int\int_{\mathbf{u}\subset R^2} (I_1(\mathbf{u}) - I_2'(\mathbf{u}))^2 \, d\mathbf{u}$$

$$
\begin{aligned}
&= \iint_{\mathbf{u}\subset R^2} (I_1(\mathbf{u}) - T_{\mathbf{A}}\{I_2(\mathbf{x})\})^2 \, d\mathbf{u} \\
&= \|I_1(\mathbf{u}) - T_{\mathbf{A}}\{I_2(\mathbf{x})\}\|^2
\end{aligned} \tag{1.29}
$$

where T is a geometric transformation applied to observed image I_2 to map it from its $[x, y]$ coordinate system to the $[u, v]$ coordinate system of reference image I_2. The subscript $\mathbf{A}$ denotes that this transformation is affine. For discrete data, we have

$$
\begin{aligned}
\chi^2(\mathbf{a}) &= \sum_{i=1}^{N} [I_1(\mathbf{u}_i) - I_2'(\mathbf{u}_i)]^2 \\
&= \sum_{i=1}^{N} [I_1(\mathbf{u}_i) - T_{\mathbf{A}}\{I_2(\mathbf{x}_i)\}]^2 \\
&= \sum_{i=1}^{N} [I_1(u_i, v_i) - I_2(a_1 x_i + a_2 y_i + a_3, a_4 x_i + a_5 y_i + a_6)]^2
\end{aligned} \tag{1.30}
$$

Newton-Raphson

The minimum of a function occurs at points where the first derivatives of the function vanishes to zero. We now solve for:

$$
B_k(\mathbf{a}) = \frac{\partial \chi^2(\mathbf{a})}{\partial a_k} = -2 \sum_{i=1}^{N} [I_1(\mathbf{u}_i) - I_2'(\mathbf{u}_i)] \frac{\partial I_2'(\mathbf{u}_i)}{\partial a_k} = 0 \tag{1.31}
$$

where $k = 1, 2, ..., 6$. This gives us a system of six nonlinear equations:

$$
B_k(\mathbf{a}) = 0 \quad k = 1, 2, ..., 6 \tag{1.32}
$$

The linear part of the Taylor series gives us:

$$
B_k(\mathbf{a} + \Delta\mathbf{a}) \approx B_k(\mathbf{a}) + \sum_{l=1}^{6} \frac{\partial B_k(\mathbf{a})}{\partial a_l} \Delta a_l \tag{1.33}
$$

Six unknown variables and six linear equations gives us:

$$\mathbf{B}(\mathbf{a}+\Delta\mathbf{a}) = \mathbf{B}(\mathbf{a}) + \mathbf{H}(\mathbf{a})\Delta\mathbf{a} \tag{1.34}$$

If $(\mathbf{a}+\Delta\mathbf{a})$ is the solution, then $\mathbf{B}(\mathbf{a}+\Delta\mathbf{a}) = 0$ and we have:

$$\mathbf{H}(\mathbf{a})\Delta\mathbf{a} = -\mathbf{B}(\mathbf{a}) \tag{1.35}$$

$\mathbf{H}(\mathbf{a})$ is a 6×6 matrix and is known as the *Hessian* matrix:

$$h_{kl} = \frac{\partial B_k(\mathbf{a})}{\partial a_l} = \frac{\partial^2 \chi^2(\mathbf{a})}{\partial a_k \partial a_l} = 2\sum_{i=1}^{N}\left[\frac{\partial I_2'(\mathbf{u}_i)}{\partial a_k}\frac{\partial I_2'(\mathbf{u}_i)}{\partial a_l} - [I_1(\mathbf{u}_i) - I_2'(\mathbf{u}_i)]\frac{\partial^2 I_2'(\mathbf{u}_i)}{\partial a_k \partial a_l}\right] \tag{1.36}$$

Due to symmetry, $h_{kl} = h_{lk}$. We then solve for six values in $\Delta\mathbf{a}$ by Gauss elimination. The advantage of this method is fast convergence near the solution. Its disadvantages include difficulty in finding a good initial estimate, and possibly slow convergence or oscillations if the function is not well-behaved.

Steepest (Gradient) Descent

In any iterative minimization, the $(i+1)$st iterate is related to the ith iterate by the equation:

$$\mathbf{a}_{i+1} = \mathbf{a}_i + \Delta\mathbf{a}_i \tag{1.37}$$

We call $\{\mathbf{a}_{i+1}\}$ a *descending sequence* if $\chi^2(\mathbf{a}_{i+1}) < \chi^2(\mathbf{a}_i)$. One direction that surely produces a descent is the direction of the negative gradient. Therefore, $\mathbf{a}_{i+1} =$

$\mathbf{a}_i - \mathbf{C}\nabla\chi^2(\mathbf{a})$, where $\mathbf{C}$ controls the step size along each parameter in $\mathbf{a}$.

$$\begin{aligned} \Delta\mathbf{a}_i &= \mathbf{a}_{i+1} - \mathbf{a}_i \\ &= -\mathbf{C}\nabla\chi^2(\mathbf{a}) \\ &= -\mathbf{C}\mathbf{B}(\mathbf{a}) \end{aligned} \tag{1.38}$$

An advantage of this method is that it always converges to a (local) minimum. A disadvantage, though, is its slow convergence.

Levenberg-Marquardt Algorithm

This method is based on the combination of the Newton-Raphson method and the steepest descent method. From the Newton-Raphson method, we have $\mathbf{H}(\mathbf{a})\Delta\mathbf{a} = -\mathbf{B}(\mathbf{a})$. From the steepest descent method, we have $\Delta\mathbf{a} = -\mathbf{C}\mathbf{B}(\mathbf{a})$. If we let $\mathbf{C}_k = \frac{1}{\lambda h_{kk}}$, then $\lambda h_{kk}\Delta\mathbf{a} = -\mathbf{B}(\mathbf{a})$. Marquardt proved that the following hybrid method minimizes $\chi^2(\mathbf{a})$ more robustly [74].

$$(\mathbf{H}(\mathbf{a}) + \lambda\mathbf{I})\Delta\mathbf{a} = -\mathbf{B}(\mathbf{a}) \tag{1.39}$$

Parameter $\lambda \geq 0$ controls the extent to which $\Delta\mathbf{a}$ conforms to either the Newton-Raphson or steepest descent methods. If the previous update succeeds in reducing $\chi^2(\mathbf{a})$, then λ is reduced to adopt the Newton-Raphson update. Otherwise, λ is increased to adopt a steepest descent update. The resulting method, known as the Levenberg-Marquardt algorithm (LMA), is given below. The algorithm requires threshold values T_1 and T_2 to specify stop conditions. The algorithm terminates when the change in $\Delta\chi^2$ falls below T_1 or λ rises

above T_2.

Levenberg-Marquardt Algorithm

begin

Initialize parameters to the identity matrix

Initialize λ with a modest value, e.g., $\lambda = 0.1$

while ($|\Delta\chi^2(\mathbf{a})| > T_1$ or $\lambda < T_2$) **do**

Compute the 6×6 Hessian matrix $\mathbf{H}$

Apply affine transformation on I_2

Compute vector $\mathbf{B}$

Solve linear equations $\mathbf{H}\Delta\mathbf{a} = -\mathbf{B}$ for $\Delta\mathbf{a}$

Evaluate $\chi^2(\mathbf{a} + \Delta\mathbf{a})$

if $\chi^2(\mathbf{a} + \Delta\mathbf{a}) < \chi^2(\mathbf{a})$ **then do**

$\lambda \leftarrow \lambda/10$

$\mathbf{a} \leftarrow \mathbf{a} + \Delta\mathbf{a}$

else do

$\lambda \leftarrow \lambda * 10$

end if

end while

end

Chapter 2

PARAMETER ESTIMATION

2.1. AFFINE PARAMETER ESTIMATION

In this section, we address the problem of estimating the affine transformation parameters by means of nonlinear least squares optimization.

The affine mapping function is given as

$$u = a_1 x + a_2 y + a_3 \tag{2.1a}$$

$$v = a_4 x + a_5 y + a_6 \tag{2.1b}$$

The six unknown parameters relating two input images, I_1 and I_2, will be estimated by minimizing the sum of squared differences between I_1 and the transformed I_2. We use the sum of squared differences (SSD) as the objective criterion that establishes a similarity measure between two images (or regions):

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{N} [I_1(\mathbf{x}_i) - I_2'(\mathbf{x}_i)]^2$$

$$= \sum_{i=1}^{N} [I_1(\mathbf{x}_i) - T_{\mathbf{A}}\{I_2(\mathbf{u}_i)\}]^2$$
$$= \sum_{i=1}^{N} [I_1(x_i, y_i) - I_2(u_i, v_i)]^2 \quad (2.2)$$

where $u_i = a_1 x_i + a_2 y_i + a_3$ and $v_i = a_4 x_i + a_5 y_i + a_6$, respectively. Note that T is a geometric transformation applied to image I_2 to map it from its $[u, v]$ coordinate system to the $[x, y]$ coordinate system of I_1. The subscript **A** denotes that the transformation is affine.

We adopt the Levenberg-Marquardt nonlinear least squares optimization algorithm [104, 85] to estimate the six unknown affine parameters. Fig. 2.1 demonstrates affine registration. Image pair I_1 and I_2 are shown in Figs. 2.1(a) and 2.1(b), respectively. Affine registration estimates the six affine parameters necessary to transform I_2 onto I_1. Fig. 2.1(c) shows the overlay of I_1 and the transformed I_2. Minimal double-exposure (or ghosting) effects illustrate that the image pair is registered with high fidelity.

(a) (b) (c)

Figure 2.1: Affine registration. (a) Image I_1; (b) Image I_2; (c) overlay of registered images.

2.2. IMPLEMENTATION DETAILS

In this section, we describe the implemented method in detail. We consider both intensities and gradients for the feature space. Intensities, or gray values, have been used widely in image registration techniques. Their popularity is attributed to their simplicity and the fact that they avoid expensive preprocessing methods such as corner detection [52], line detection (Hough transform) [53, 37], feature extraction (Gabor wavelets) [62], or segmentation and clustering. In Section 4.1 we will show that the use of gradients sometimes performs better than intensities, particularly when edge alignment is more critical than region matching. The search space for geometric transformation T in this work consists of the six unknown parameters of a global affine transformation. The similarity measure used is the sum of squared differences (SSD) $\chi^2(\mathbf{a})$, given in Eq. (1.29). Recall that T is a warp function that resamples I_2 according to the affine transformation given in Eq. (1.22). The standard Levenberg-Marquardt is applicable to Eq. (1.29) and we can minimize it with respect to parameters $\mathbf{a}$. We can improve this registration method by adding two modifications to the standard Levenberg-Marquardt method. The first modification includes the use of a multiresolution pyramid structure for both reference and target image. The second modification takes place in the Levenberg-Marquardt method for reducing the calculation of the Hessian matrix.

2.2.1 Multiresolution pyramid

A multiresolution pyramid consists of a set of images representing an image in multiple resolutions. The original image, sitting at the base of the pyramid, is downsampled by a constant scale factor in each dimension to form the next level. This is repeated from one level to the next until the tip of the pyramid is reached. Fig. 2.2(a) depicts an image pyra-

mid with a scale factor of two between levels. The image size at level i is reduced from the original by a factor of 2^i in each dimension. Level 0, at the base of the pyramid, is referred to as the finest level. Level $n-1$, at the tip of the pyramid, is known as the coarsest level. Downsizing is usually performed by bandlimiting and subsampling the original image. The bandlimiting filter kernel could vary in quality from a box filter (unweighted averaging) to a high quality windowed sinc function.

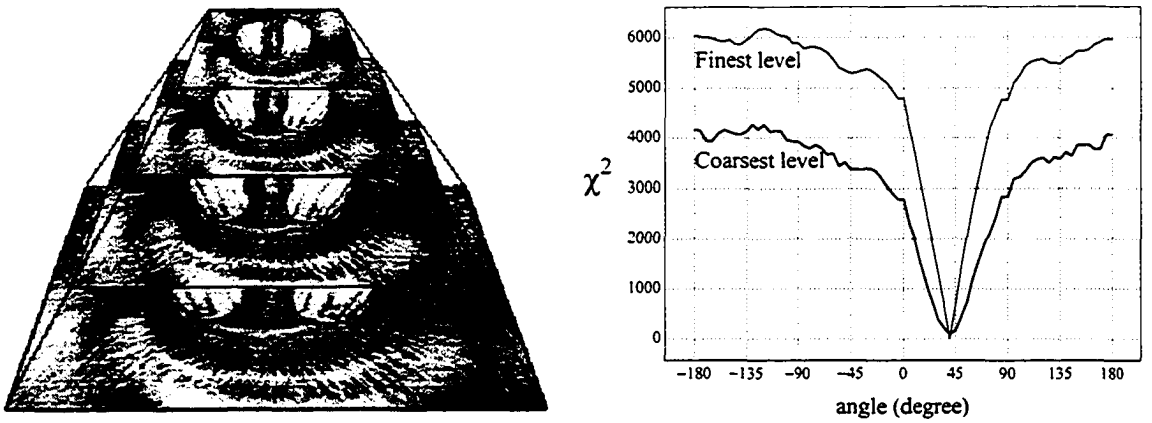


Figure 2.2: (a) Multiresolution pyramid; (b) χ^2 for the coarsest and finest pyramid levels.

Multiresolution pyramids supply us with two major advantages. First, when we apply the Levenberg-Marquardt method to the coarsest level of the pyramid, the number of pixels is reduced by a factor of $2^{2(n-1)}$. We get large computational gains because most of the iterations are executed in the coarsest level, consisting of fewer pixels. Second, the smoothness conditions imposed by successively bandlimiting the pyramid levels causes $\chi^2(\mathbf{a})$ to be computed on smoother images. This smoothness property helps prevent getting trapped in local minimas. An example of $\chi^2(\mathbf{a})$ computed on two different pyramid levels is shown in See Fig. 2.2(b). Since the coarsest level retains large-scale features only, the registration algorithm proceeds from the coarsest level to progressively finer levels, where small corrections due to finer details are integrated. This approach passes the computed parameters

as an initial estimate to the next finer level. The parameters must be scaled properly across successive levels. Let the horizontal and vertical scale factors between adjacent levels be s_1 and s_2, respectively. In level i, the affine transformation is:

$$u^i = a_1^i x^i + a_2^i y^i + a_3^i \tag{2.3a}$$

$$v^i = a_4^i x^i + a_5^i y^i + a_6^i \tag{2.3b}$$

and in level $i+1$ we have:

$$u^{i+1} = a_1^{i+1} x^{i+1} + a_2^{i+1} y^{i+1} + a_3^{i+1} \tag{2.4a}$$

$$v^{i+1} = a_4^{i+1} x^{i+1} + a_5^{i+1} y^{i+1} + a_6^{i+1} \tag{2.4b}$$

The relation between level i and $i+1$ is:

$$u^{i+1} = s_1 u^i \tag{2.5a}$$

$$v^{i+1} = s_2 v^i \tag{2.5b}$$

Applying these scale factors to Eq. (2.3) and substituting into Eq. (2.4) yields the following relation between parameters:

$$\begin{array}{lllll} a_1^{i+1} = a_1^i & a_2^{i+1} = \frac{s1}{s2} a_2^i & a_3^{i+1} = s_1 a_3^i \\ a_4^{i+1} = \frac{s2}{s1} a_4^i & a_5^{i+1} = a_5^i & a_6^{i+1} = s_2 a_6^i \end{array} \tag{2.6}$$

In our case, $s_1 = s_2 = 2$ so only the translation parameters, a_3 and a_6 are multiplied by two.

2.2.2 Modified Levenberg-Marquardt algorithm

In the standard LMA, we calculate the $\mathbf{B}_{6\times 1}$ vector and Hessian matrix $[\alpha]_{6\times 6}$ in each iteration. In this section, we review a modified LMA that realizes performance gains by eliminating the calculation of the Hessian matrix at each iteration.

Consider the following objective function to establish a similarity measure between I_1 and I_2.

$$\chi^2(\mathbf{a}) = \|I_1(\mathbf{u}) - T_{\mathbf{A}}\{I_2(\mathbf{x})\}\|^2 \tag{2.7}$$

We shall assume that I_2 is mapped to I_1 after a series of affine transformations $\mathbf{A}$. During the iterative process, new estimates for $\mathbf{A}$ are computed as follows:

$$\mathbf{A}_{i+1} = \mathbf{A}_i + \Delta\mathbf{A}_i \tag{2.8}$$

where $\mathbf{A}_i = \mathbf{I} + \Delta\mathbf{A}_1 + \Delta\mathbf{A}_2 + \cdots + \Delta\mathbf{A}_{i-1}$. Since I_2 is transformed in each iteration, the Hessian matrix must be recomputed because it is a function of the gradient of I_2. The Hessian matrix is responsible for computing the $\Delta\mathbf{A}$ terms above. Fig. 2.3 depicts the series of parameter estimates for the Tiffany image in Fig. 1.14(b) beginning from the initial guess P_0.

The goal of the modified LMA is to eliminate the computation of the Hessian matrix. This is achieved by casting this problem into one where I_1 is transformed into I_2, leaving I_2 unchanged from one iteration to the next. This permits the Hessian matrix to be computed only once, i.e., in the first iteration.

In order to determine the new estimates for $\mathbf{A}$ in the modified LMA, we must express χ^2 in terms of a transformation that maps I_1 to I_2. The fundamental difference between

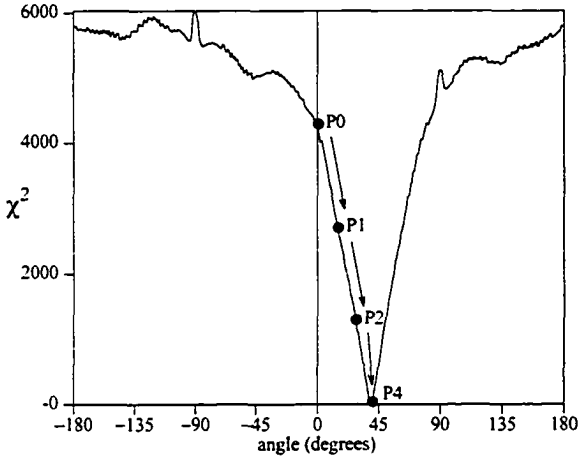
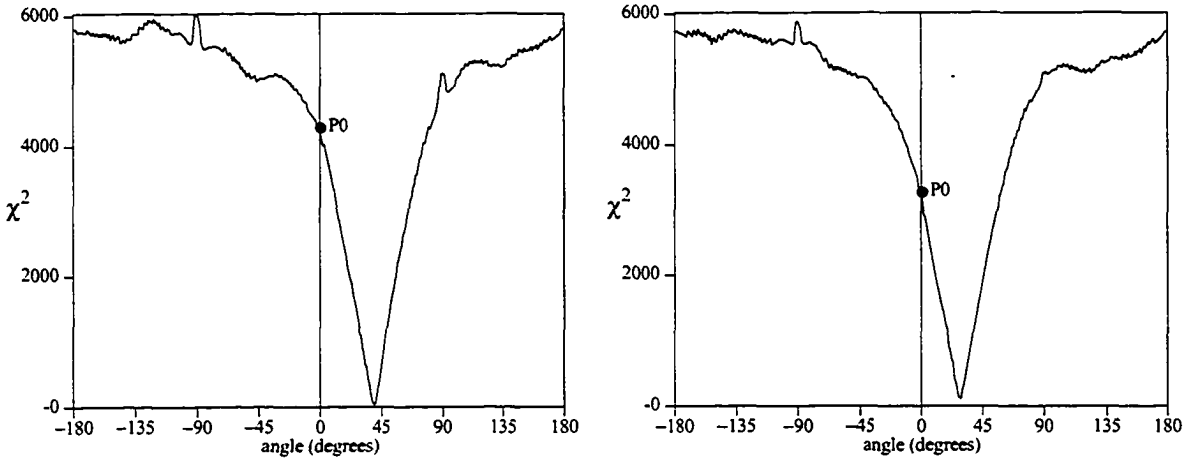
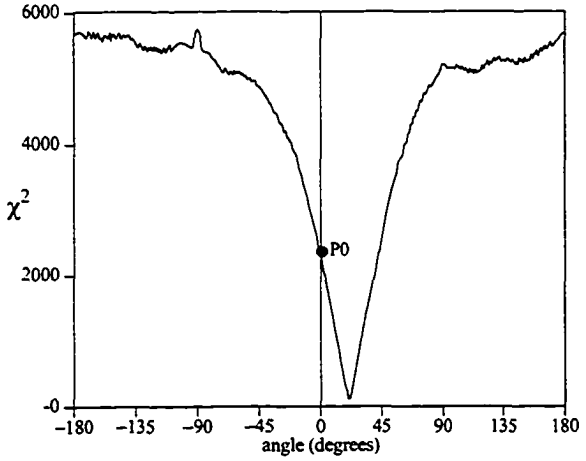


Figure 2.3: χ^2 curve for rotation (standard LMA)

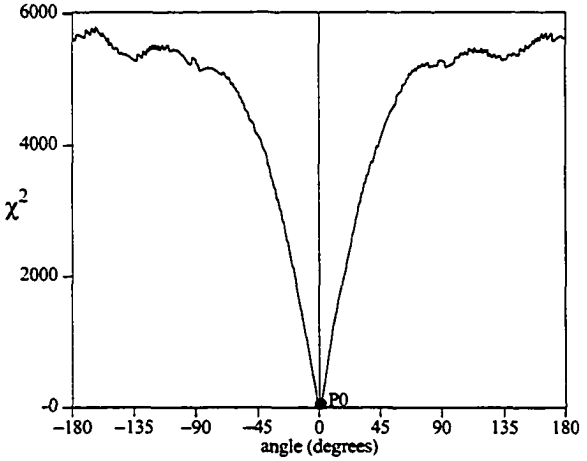


(a) 0 iterations

(b) 10 iterations



(c) 20 iterations



(d) 30 iterations

Figure 2.4: χ^2 curve for rotation (modified LMA)

the standard LMA and the modified LMA is that the standard LMA updates the current estimate by changing the initial guess towards the global minima, while the modified LMA brings the global minima towards the initial guess. Fig. 2.4 depicts several snapshots of the χ^2 curve after 0, 10, 20, and 30 iterations, respectively. The consequence of this formulation can be summarized with the following update rule for the modified LMA:

$$\begin{aligned} \mathbf{A}_{i+1}^{-1} &= (\mathbf{I}+\Delta\mathbf{A}_i)^{-1}\cdots(\mathbf{I}+\Delta\mathbf{A}_2)^{-1}(\mathbf{I}+\Delta\mathbf{A}_1)^{-1} \\ &= (\mathbf{I}+\Delta\mathbf{A}_i)^{-1}\mathbf{A}_i^{-1} \end{aligned} \tag{2.9}$$

An important distinction between the standard and modified LMA methods lie in the manner in which the unknown parameters are updated in each iteration. In the standard LMA, the initial estimates for the unknown parameters are chosen using identity matrix $\mathbf{I}$ as the initial guess for point P_0. Then, we calculate the directional derivatives of I_2, g_x and g_y, where $g_x = \frac{\partial I_2}{\partial x}$ and $g_y = \frac{\partial I_2}{\partial y}$. These processes are in the order of $O(N \times 3 \times 3)$, where N is the number of pixels and 3×3 is the size of the kernel. The standard LMA gives us the $\Delta\mathbf{A}$ that we use to add to the initial guess $\mathbf{I}$ to move from point P_0 to P_1 on the $\chi^2(\mathbf{a})$ curve. In the next iteration, because the image I_2 is warped by $\mathbf{A}+\Delta\mathbf{A}$, we need to compute g_x and g_y again to find a new $\Delta\mathbf{A}$. Therefore, in the standard LMA, the optimal solution point P_i slides on the $\chi^2(\mathbf{a})$ curve. However, in the modified LMA, we shift the $\chi^2(\mathbf{a})$ curve toward the initial guess P_0. This is achieved by resampling I_1 with the inverse transformation $(\mathbf{I}+\Delta\mathbf{A}_i)^{-1}\mathbf{A}_i^{-1}$. Consequently, image I_1' is brought closer to I_2. The new image I_1' and image I_2 are now used to minimize $\chi^2(\mathbf{a})$. The result produces a new $\Delta\mathbf{A}$ that is always added to $\mathbf{I}$, the initial guess point P_0. Since I_2 does not change, we do not need to compute g_x and g_y. In Fig. 2.4, we see how the $\chi^2(\mathbf{a})$ graph is sliding towards the initial guess P_0.

We shall find it useful to rewrite χ^2 in terms of a forward mapping as well as an inverse mapping. This decomposition will enable us to apply a substantial part of the transformation to $I_1(\mathbf{u})$. As a result, the small inverse transformation that remains for $I_2(\mathbf{x})$ will permit us to drop the need to compute the Hessian.

Suppose that we decompose the $T\{\}$ transformation into two transformations $T_{\mathbf{A}}T_{\mathbf{I}+\Delta\mathbf{A}}\{\}$. $T_{\mathbf{A}}\{\}$ is the transformation from the previous iteration and $T_{\mathbf{I}+\Delta\mathbf{A}}\{\}$ is the small transformation from the Levenberg-Marquardt method that minimizes $\chi^2(\mathbf{a})$:

$$\begin{aligned}
\chi^2(\mathbf{a}) &= \|I_1(\mathbf{u}) - T_{\mathbf{A}}\{T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}\}\|^2 \\
&= \frac{1}{|\mathbf{A}|}\|T_{\mathbf{A}^{-1}}\{I_1(\mathbf{u})\} - T_{\mathbf{A}^{-1}}\{T_{\mathbf{A}}\{T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}\}\}\|^2 \\
&= \frac{1}{|\mathbf{A}|}\|T_{\mathbf{A}^{-1}}\{I_1(\mathbf{u})\} - T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}\|^2 \\
&= \frac{1}{|\mathbf{A}(\mathbf{I}+\Delta\mathbf{A})|}\|T_{(\mathbf{I}+\Delta\mathbf{A})^{-1}\mathbf{A}^{-1}}\{I_1(\mathbf{u})\} - I_2(\mathbf{x})\|^2 \\
&= \frac{1}{|\mathbf{A}(\mathbf{I}+\Delta\mathbf{A})|}\|T^{-1}\{I_1(\mathbf{u})\} - I_2(\mathbf{x})\|^2
\end{aligned} \tag{2.10}$$

In the modified LMA, we need to define the $\partial T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}/\partial\Delta\mathbf{a}$ and the update rule for each transformation parameter. We can decompose $\partial T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}/\partial\Delta\mathbf{a}$ as follows:

$$\frac{\partial T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}}{\partial\Delta\mathbf{a}} = \frac{\partial T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\Delta\mathbf{a}} \tag{2.11}$$

and because transformation $\Delta\mathbf{a}$ is small, then $T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\} \approx I_2$ and $\frac{\partial\mathbf{x}}{\partial\Delta\mathbf{a}} \approx \frac{\partial\mathbf{u}}{\partial\Delta\mathbf{a}}$. This yields

$$\frac{\partial T_{\mathbf{I}+\Delta\mathbf{A}}\{I_2(\mathbf{x})\}}{\partial\Delta\mathbf{a}} \approx \frac{\partial I_2}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\Delta\mathbf{a}} \approx \frac{\partial I_2}{\partial\mathbf{x}}\frac{\partial\mathbf{u}}{\partial\Delta\mathbf{a}} \tag{2.12}$$

where $\frac{\partial I_2}{\partial \mathbf{x}}$ is the gradient of I_2. Thus, $\frac{\partial I_2(\mathbf{x})}{\partial \Delta a_k}$ for the six parameters are:

$$\frac{\partial I_2}{\partial \Delta a_1} = \frac{\partial I_2}{\partial x}\frac{\partial}{\partial \Delta a_1}((\Delta a_1 + 1)x + \Delta a_2 y + \Delta a_3) = g_x x \quad (2.13a)$$

$$\frac{\partial I_2}{\partial \Delta a_2} = \frac{\partial I_2}{\partial x}\frac{\partial}{\partial \Delta a_2}((\Delta a_1 + 1)x + \Delta a_2 y + \Delta a_3) = g_x y \quad (2.13b)$$

$$\frac{\partial I_2}{\partial \Delta a_3} = \frac{\partial I_2}{\partial x}\frac{\partial}{\partial \Delta a_3}((\Delta a_1 + 1)x + \Delta a_2 y + \Delta a_3) = g_x \quad (2.13c)$$

$$\frac{\partial I_2}{\partial \Delta a_4} = \frac{\partial I_2}{\partial y}\frac{\partial}{\partial \Delta a_4}(\Delta a_4 x + (\Delta a_5 + 1)y + \Delta a_6) = g_y x \quad (2.13d)$$

$$\frac{\partial I_2}{\partial \Delta a_5} = \frac{\partial I_2}{\partial y}\frac{\partial}{\partial \Delta a_5}(\Delta a_4 x + (\Delta a_5 + 1)y + \Delta a_6) = g_y y \quad (2.13e)$$

$$\frac{\partial I_2}{\partial \Delta a_6} = \frac{\partial I_2}{\partial y}\frac{\partial}{\partial \Delta a_6}(\Delta a_4 x + (\Delta a_5 + 1)y + \Delta a_6) = g_y \quad (2.13f)$$

To compute the gradient images g_x and g_y, we convolved I_2 with two separable 3×3 kernels:

$$mask_x = \begin{bmatrix} 1 & 0 & -1 \\ 4 & 0 & -4 \\ 1 & 0 & -1 \end{bmatrix} \quad (2.14a)$$

$$mask_y = \begin{bmatrix} 1 & 4 & 1 \\ 0 & 0 & 0 \\ -1 & -4 & -1 \end{bmatrix} \quad (2.14b)$$

In the standard Levenberg-Marquardt algorithm, the update rule is as follows:

$$\mathbf{A}_{new} = \mathbf{A}_{old} + \Delta\mathbf{A} \quad (2.15)$$

From Eq. (2.15), the updating rule for parameters is:

$$\mathbf{A}_{new} = \mathbf{A}_{old}(\mathbf{I} + \Delta\mathbf{A})$$

$$= \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta a_1 + 1 & \Delta a_2 & \Delta a_3 \\ \Delta a_4 & \Delta a_5 + 1 & \Delta a_6 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.16}$$

Now, we should address the bandlimiting filter and resampling algorithm for geometric transformations. In this implementation, we use cubic convolution filter for prefiltering the images in the process of building the multiresolution pyramid. We used Fant's resmapling method, which uses linear interpolation for reconstruction and the unweighted averaging for antialiasing. For further details about resampling see [118]. We can achieve good results with this simple filter and this algorithm takes less time than the version implemented in [104]. Their filter is a least-squared spline of order 3. Now, we present the pseudocode for this method:

Modified Levenberg-Marquardt Algorithm

Build multiresolution pyramid for images I_1 and I_2

Initialize parameters to the identity matrix

Initialize λ with a modest value, e.g., $\lambda = 0.1$

for $i = L$ to 0 **do** /* L is the coarsest pyramid level */

Compute directional gradients: $\partial I_2^i/\partial x$ and $\partial I_2^i/\partial y$

Compute the 6×6 Hessian matrix **H**

while ($|\Delta\chi^2(\mathbf{a})| > T_1$ or $\lambda < T_2$) **do**

Apply affine transformation on I_1 in level i

Compute vector **B**

Solve linear equations $\mathbf{H}\Delta\mathbf{a} = \mathbf{B}$ for $\Delta\mathbf{a}$

Evaluate $\chi^2(\mathbf{a} + \Delta\mathbf{a})$

if $\chi^2(\mathbf{a} + \Delta\mathbf{a}) < \chi^2(\mathbf{a})$ **then do**

$\lambda \leftarrow \lambda/10$

$\mathbf{a} \leftarrow \mathbf{a} + \Delta\mathbf{a}$

else do

$\lambda \leftarrow \lambda * 10$

end if

end while

end for

Consider the mandrill and Tiffany images shown in Fig. 1.14. Fig. 2.5 shows the registration accuracy of the Levenberg-Marquardt method for the case of rotation and scale. In Fig. 2.5(a), the mandrill image was registered against rotated versions of itself. The applied rotation angles varied from $-90°$ to $90°$, in $5°$ increments. The figure shows that the Levenberg-Marquardt registration method remains accurate only in the $[-75°, 45°]$ range. The algorithm stuck in the initial guess outside of this range. In Fig. 2.5(b), the mandrill image was registered against scaled versions of itself. The applied scale factors varied from 0.25 to 2, in 0.1 increments. In this case, the linear least-squares registration method remains accurate only in the [.25, 1.8] range. Fig. 2.6 demonstrates similar results on the Tiffany image. The figure shows that the linear least-squares registration method remains accurate in the $[-55°, 60°]$ range for rotation, and [.35, 1.7] range for scale.

2.3. PERSPECTIVE PARAMETER ESTIMATION

In this section, we extend the results of affine registration based on the Levenberg-Marquardt algorithm to handle perspective transformations. We estimate eight unknown parameters of a planar perspective between two images.

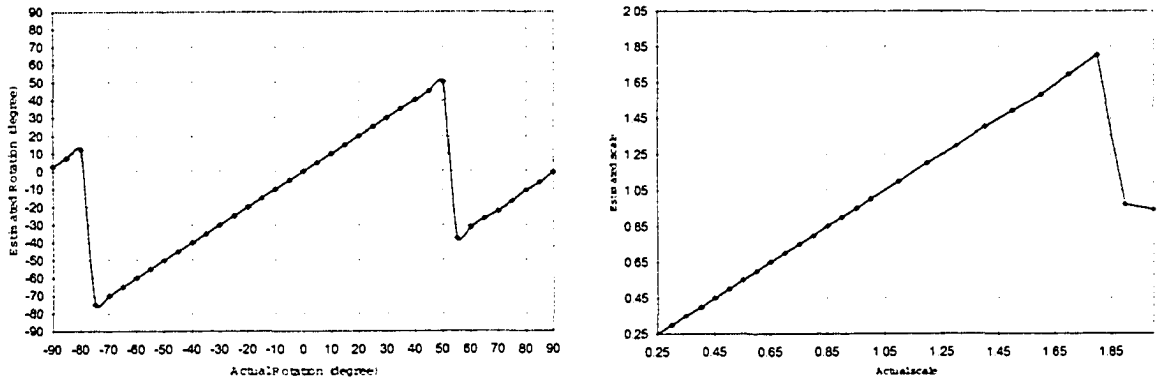


Figure 2.5: Levenberg-Marquardt registration accuracy for the mandrill image: (a) rotation; (b) scale.

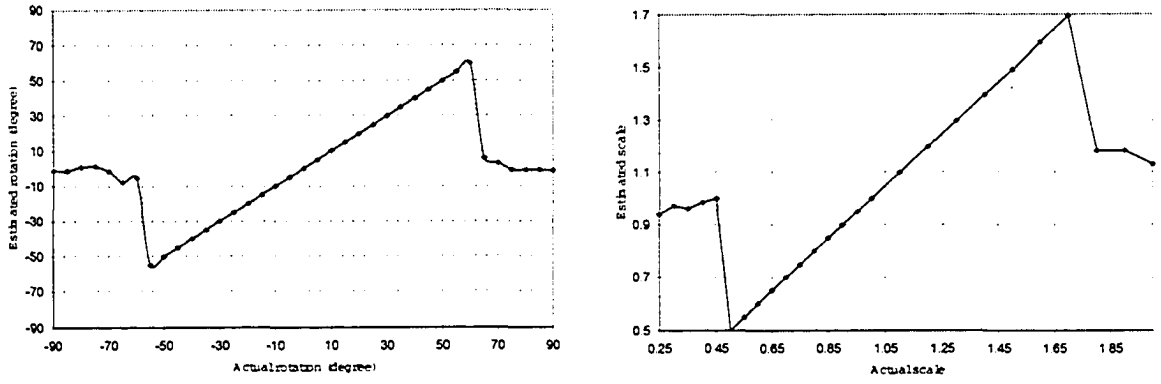


Figure 2.6: Levenberg-Marquardt registration accuracy for the Tiffany image: (a) rotation; (b) scale.

$$u = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} \tag{2.17a}$$

$$v = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1} \tag{2.17b}$$

We described the necessary steps for affine parameter estimation using the modified LMA in Section 2.2.2. The same producers are now needed for the perspective deforma-

tion. First, we compute the derivatives of $I_2(x, y)$ respect to the parameters as follows:

$$\frac{\partial I_2}{\partial \Delta a_1} = \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_1} = g_x \frac{x}{a_7 x + a_8 y + 1} \tag{2.18a}$$

$$\frac{\partial I_2}{\partial \Delta a_2} = \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_2} = g_x \frac{x}{a_7 x + a_8 y + 1} \tag{2.18b}$$

$$\frac{\partial I_2}{\partial \Delta a_3} = \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_3} = g_x \tag{2.18c}$$

$$\frac{\partial I_2}{\partial \Delta a_4} = \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_4} = g_y \frac{x}{a_7 x + a_8 y + 1} \tag{2.18d}$$

$$\frac{\partial I_2}{\partial \Delta a_5} = \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_5} = g_y \frac{y}{a_7 x + a_8 y + 1} \tag{2.18e}$$

$$\frac{\partial I_2}{\partial \Delta a_6} = \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_6} = g_y \tag{2.18f}$$

$$\begin{aligned}\frac{\partial I_2}{\partial \Delta a_7} &= \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_7} + \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_7} \\ &= -x\left(g_x \frac{a_1 x + a_2 y + a_2}{(a_7 x + a_8 y + 1)^2} + g_y \frac{a_4 x + a_5 y + a_6}{(a_7 x + a_8 y + 1)^2}\right)\end{aligned} \tag{2.18g}$$

$$\begin{aligned}\frac{\partial I_2}{\partial \Delta a_8} &= \frac{\partial I_2}{\partial x}\frac{\partial u}{\partial \Delta a_8} + \frac{\partial I_2}{\partial y}\frac{\partial v}{\partial \Delta a_8} \\ &= -y\left(g_x \frac{a_1 x + a_2 y + a_2}{(a_7 x + a_8 y + 1)^2} + g_y \frac{a_4 x + a_5 y + a_6}{(a_7 x + a_8 y + 1)^2}\right)\end{aligned} \tag{2.18h}$$

Second, we compute the update rule for the modified LMA. If the matrix $\mathbf{A}_1$ transforms quadrilateral q_1 to quadrilateral q_2 and the matrix $\mathbf{A}_2$ transforms quadrilateral q_2 to quadrilateral q_3, then we can go directly from q_1 to q_3 by the matrix $\mathbf{A}$= $\mathbf{A}_1\mathbf{A}_2$. Therefore, the update rule is as follows:

$$\begin{aligned}\mathbf{A}_{new} &= \mathbf{A}_{old}(\mathbf{I} + \Delta\mathbf{A}) \\ &= \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & 1 \end{bmatrix}\begin{bmatrix} \Delta a_1 + 1 & \Delta a_2 & \Delta a_3 \\ \Delta a_4 & \Delta a_5 + 1 & \Delta a_6 \\ \Delta a_7 & \Delta a_8 & 1 \end{bmatrix}\end{aligned} \tag{2.19}$$

Third, since the modified LMA is implemented in a multiresolution fashion, we must find

the scale factors needed to propagate the parameters across pyramid levels. Let the scale factor between the levels be s: $x^{i+1} = sx^i$, $y^{i+1} = sy^i$, $u^{i+1} = su^i$, and $v^{i+1} = sv^i$, where

$$u^i = \frac{a_1^i x^i + a_2^i y^i + a_3^i}{a_7^i x^i + a_8^i y^i + 1} \tag{2.20a}$$

$$v^i = \frac{a_4^i x^i + a_5^i y^i + a_6^i}{a_7^i x^i + a_8^i y^i + 1} \tag{2.20b}$$

Substituting the coordinates of the next finer level into the above equations yields:

$$\frac{u^{i+1}}{s} = \frac{a_1^i \frac{x^{i+1}}{s} + a_2^i \frac{y^{i+1}}{s} + a_3^i}{a_7^i \frac{x^{i+1}}{s} + a_8^i \frac{y^{i+1}}{s} + 1} \tag{2.21a}$$

$$\frac{v^{i+1}}{s} = \frac{a_4^i \frac{x^{i+1}}{s} + a_5^i \frac{y^{i+1}}{s} + a_6^i}{a_7^i \frac{x^{i+1}}{s} + a_8^i \frac{y^{i+1}}{s} + 1} \tag{2.21b}$$

Multiplying both sides by s gives us:

$$\frac{a_1^{i+1} x^{i+1} + a_2^{i+1} y^{i+1} + a_3^{i+1}}{a_7^{i+1} x^{i+1} + a_8^{i+1} y^{i+1} + 1} = u^{i+1} = \frac{a_1^i x^{i+1} + a_2^i y^{i+1} + s a_3^i}{a_7^i \frac{x^{i+1}}{s} + a_8^i \frac{y^{i+1}}{s} + 1} \tag{2.22a}$$

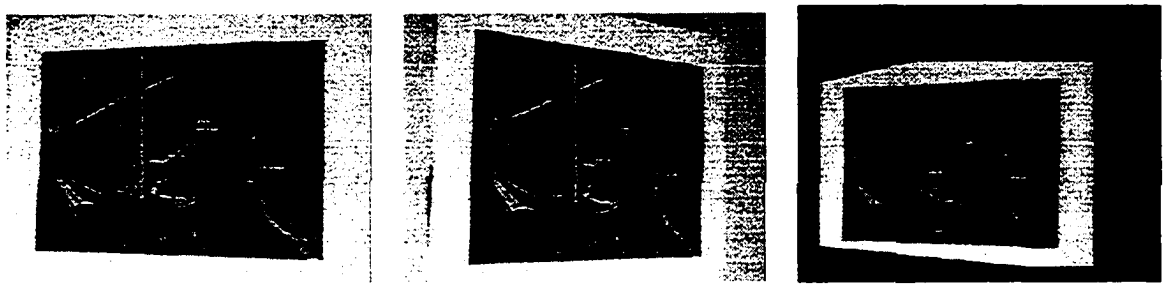
$$\frac{a_4^{i+1} x^{i+1} + a_5^{i+1} y^{i+1} + a_6^{i+1}}{a_7^{i+1} x^{i+1} + a_8^{i+1} y^{i+1} + 1} = v^{i+1} = \frac{a_4^i x^{i+1} + a_5^i y^{i+1} + s a_6^i}{a_7^i \frac{x^{i+1}}{s} + a_8^i \frac{y^{i+1}}{s} + 1} \tag{2.22b}$$

Thus, the relation between parameters is:

$$\begin{aligned} a_1^{i+1} &= a_1^i \quad a_2^{i+1} = a_2^i \quad a_3^{i+1} = s a_3^i \\ a_4^{i+1} &= a_4^i \quad a_5^{i+1} = a_5^i \quad a_6^{i+1} = s a_6^i \\ a_7^{i+1} &= \frac{a_7^i}{s} \quad a_8^{i+1} = \frac{a_8^i}{s} \end{aligned} \tag{2.23}$$

Fig. 2.7 demonstrates perspective registration. Image pair I_1 and I_2 are shown in Figs. 2.7(a) and 2.7(b), respectively. Perspective registration estimates the eight parameters necessary to transform I_2 onto I_1. Fig. 2.7(c) shows the overlay of I_1 and the transformed

I_2. Minimal double-exposure (ghosting) effects illustrate that the image pair is registered with high fidelity.



(a) (b) (c)

Figure 2.7: Perspective registration. (a) Image I_1; (b) Image I_2; and (c) overlay of registered images.

This method cannot, in general, register images subjected to large perspective distortion. The primary difficulty is that the initial guess may not lie close to the solution and the minimization procedure may get stuck in a local minima. This problem is more severe for perspective than affine because perspective transformations are nonlinear. In the next section we propose to circumvent this problem by approximating perspective to be piecewise affine.

2.4. PIECEWISE AFFINE PARAMETERS ESTIMATION

We describe a piecewise method to recover perspective parameters. In this manner, we leverage the robust affine registration algorithm to handle the more difficult perspective registration problem. Recently, a number of researchers have investigated techniques for finding local regions that are invariant to affine transformations. Zisserman et. al. detect quadrilateral and elliptical locally affine regions for finding the fundamental matrix in wide-baseline stereo images [90]. Van Gool et. al. look for locally affine regions. They compute

several degrees of the moments in these regions to build feature vectors for wide-baseline stereoscopy [45] and image retrieval [106]. A local affine approximation is suggested by expanding the perspective transformation function (Eq. (1.6)) about a point using a first-order Taylor series [119].

The approximation holds in a small neighborhood about point (x_0, y_0):

$$\begin{aligned} u &= U(x,y) \\ &= U(x_0,y_0) + \frac{\partial U(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial U(x_0,y_0)}{\partial y}(y-y_0) \\ &= A_1 x + A_2 y + A_3 \end{aligned} \tag{2.24}$$

where $A_1 = \frac{\partial U(x_0,y_0)}{\partial x}$, $A_2 = \frac{\partial U(x_0,y_0)}{\partial y}$, and $A_3 = U(x_0,y_0) - A_1 x_0 - A_2 y_0$. The expression for v is

$$\begin{aligned} v &= V(x,y) \\ &= V(x_0,y_0) + \frac{\partial V(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial V(x_0,y_0)}{\partial y}(y-y_0) \\ &= A_4 x + A_5 y + A_6 \end{aligned} \tag{2.25}$$

where $A_4 = \frac{\partial V(x_0,y_0)}{\partial x}$, $A_5 = \frac{\partial V(x_0,y_0)}{\partial y}$, and $A_6 = V(x_0,y_0) - A_4 x_0 - A_5 y_0$. Notice that the expressions for u and v match the linear form required of affine transformations.

In the next two sections, we review the process of determining the affine transformation for a four-corner (tile-to-tile) mapping and its use in inferring perspective parameters. We have already reviewed how to perform affine registration on a rectangular image domain, e.g., tile. For the purpose of demonstrating the feasibility of piecewise affine approximation, we will show how to infer the affine parameters given the correspondence of the tile corners. This permits us to analyze the proposed technique without concern for errors

that may be due to the affine registration process using the Levenberg-Marquardt algorithm.

2.4.1 Inferring Affine Parameters

Consider a tile in I_1 and its corresponding quadrilateral in I_2. (Fig. 2.8).

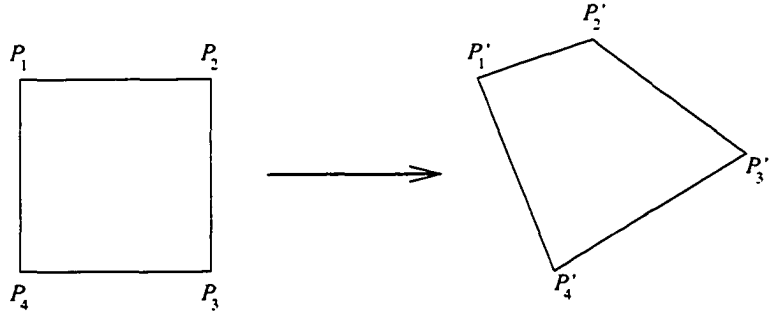


Figure 2.8: Four corner mapping.

Although a perspective transformation can fully account for this four corner mapping, we are interested in finding the best affine transformation that approximates this mapping. Given the four corners of a tile in observed image I_2 and their correspondences in reference image I_1, we may solve for the best affine fit by using the least squares approach. Let the affine mapping be given as:

$$u = a_1x + a_2y + a_3 \tag{2.26a}$$

$$v = a_4x + a_5y + a_6 \tag{2.26b}$$

We solve for the affine parameters by minimizing the expression for χ^2 below.

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{4} \left[(u_i - a_1x_i - a_2y_i - a_3)^2 + (v_i - a_4x_i - a_5y_i - a_6)^2\right] \tag{2.27}$$

We may relate these correspondences in the form $U = WA$:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ x_4 & y_4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \\ 0 & 0 & 0 & x_4 & y_4 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \tag{2.28}$$

The pseudoinverse solution $A = (W^T W)^{-1} W^T U$ is computed to solve for the six affine coefficients.

2.4.2 Inferring Perspective Parameters

The affine parameters computed in Eq. (2.28) apply to a single tile pair. We must infer the affine parameters for each of the N tile pairs in I_1 and I_2, and apply them to all tile centers. This establishes N point correspondences between I_1 and I_2, The eight unknown perspective parameters can be inferred by solving the resulting system of equations. Again, we may relate the correspondences in the form $U = WA$ by rewriting Eq. (1.6) in the form

$$u = a_1 x + a_2 y + a_3 - a_7 ux - a_8 uy \tag{2.29a}$$

$$v = a_4 x + a_5 y + a_6 - a_7 vx - a_8 vy \tag{2.29b}$$

This yields the following overdetermined system of equations.

$$\begin{bmatrix} u_0^i \\ \vdots \\ v_0^i \\ \vdots \end{bmatrix}_{2N\times 1} = \begin{bmatrix} x_0^i & y_0^i & 1 & 0 & 0 & 0 & -u_0^i x_0^i & -u_0^i y_0^i \\ & & & & \vdots & & & \\ 0 & 0 & 0 & x_0^i & y_0^i & 1 & -v_0^i x_0^i & -v_0^i y_0^i \\ & & & & \vdots & & & \end{bmatrix}_{2N\times 8} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix}_{8\times 1} \tag{2.30}$$

where (u_0^i, v_0^i) and (x_0^i, y_0^i) denote the centers of tile i in I_1 and I_2 for $1 \leq i \leq N$. The pseudoinverse solution $A = (W^T W)^{-1} W^T U$ is computed to solve for the eight perspective coefficients.

We may augment these results by including additional constraints in the least squares estimation. In addition to considering the correspondence of tile centers, we may also consider the local affine parameter estimates. From Eqs. (2.24) and (2.25), we know that the affine approximation at a tile is given as

$$u = A_1 x + A_2 y + A_3 \tag{2.31a}$$

$$v = A_4 x + A_5 y + A_6 \tag{2.31b}$$

where

$$\begin{aligned} A_1 &= \frac{\partial U(x_0, y_0)}{\partial x} = \frac{a_1(a_7 x_0 + a_8 y_0 + 1) - a_7(a_1 x_0 + a_2 y_0 + a_3)}{(a_7 x_0 + a_8 y_0 + 1)^2} = \frac{a_1 - a_7 u_0}{a_7 x_0 + a_8 y_0 + 1} \\ &= a_1 - a_7(A_1 x_0 + u_0) - a_8 A_1 y_0 \end{aligned}$$

$$
\begin{aligned}
A_2 &= \frac{\partial U(x_0,y_0)}{\partial y} = \frac{a_2(a_7x_0+a_8y_0+1)-a_8(a_1x_0+a_2y_0+a_3)}{(a_7x_0+a_8y_0+1)^2} = \frac{a_2-a_8u_0}{a_7x_0+a_8y_0+1} \\
&= a_2 - a_7A_2x_0 - a_8(A_2y_0+u_0) \\
A_3 &= U(x_0,y_0) - A_1x_0 - A_2y_0 = \frac{a_1x_0+a_2y_0+a_3}{a_7x_0+a_8y_0+1} - A_1x_0 - A_2y_0 \\
&= a_1x_0+a_2y_0+a_3 - a_7(A_1x_0^2+A_2x_0y_0+A_3x_0) - a_8(A_1x_0y_0+A_2y_0^2+A_3y_0) \\
&\quad -(A_1x_0+A_2y_0) \\
&= a_1x_0+a_2y_0+a_3 - a_7u_0x_0 - a_8u_0y_0 - (u_0-A_3) \\
A_4 &= \frac{\partial V(x_0,y_0)}{\partial x} = \frac{a_4(a_7x_0+a_8y_0+1)-a_7(a_4x_0+a_5y_0+a_6)}{(a_7x_0+a_8y_0+1)^2} = \frac{a_4-a_7v_0}{a_7x_0+a_8y_0+1} \\
&= a_4 - a_7(A_4x_0+v_0) - a_8A_4y_0 \\
A_5 &= \frac{\partial V(x_0,y_0)}{\partial y} = \frac{a_5(a_7x_0+a_8y_0+1)-a_8(a_4x_0+a_5y_0+a_6)}{(a_7x_0+a_8y_0+1)^2} = \frac{a_5-a_8v_0}{a_7x_0+a_8y_0+1} \\
&= a_5 - a_7A_5x_0 - a_8(A_5y_0+v_0) \\
A_6 &= V(x_0,y_0) - A_4x_0 - A_5y_0 = \frac{a_4x_0+a_5y_0+a_6}{a_7x_0+a_8y_0+1} - A_4x_0 - A_5y_0 \\
&= a_4x_0+a_5y_0+a_6 - a_7(A_4x_0^2+A_5x_0y_0+A_6x_0) - a_8(A_4x_0y_0+A_5y_0^2+A_6y_0) \\
&\quad -(A_4x_0+A_5y_0) \\
&= a_4x_0+a_5y_0+a_6 - a_7v_0x_0 - a_8v_0y_0 - (v_0-A_6)
\end{aligned}
$$

Notice that the terms for A_3 and A_6 actually produce equations of the form $u_0 = a_1x_0 + a_2y_0 + a_3 - a_7u_0x_0 - a_8u_0y_0$ and $v_0 = a_4x_0 + a_5y_0 + a_6 - a_7v_0x_0 - a_8v_0y_0$, respectively. These are the same point correspondence relations defined in Eq. (2.30). We may collect the expressions given above to yield the following system of equations that relates the affine parameters at all N tiles with the unknown parameters of the global perspective

transformation.

$$\begin{bmatrix} A_1^i \\ \vdots \\ A_2^i \\ \vdots \\ u_0^i \\ \vdots \\ A_4^i \\ \vdots \\ A_5^i \\ \vdots \\ v_0^i \\ \vdots \end{bmatrix}_{6N\times 1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -(A_1^i x_0^i + u_0^i) & -A_1^i y_0^i \\ & & & & & \vdots & & \\ 0 & 1 & 0 & 0 & 0 & 0 & -A_2^i x_0^i & -(A_2^i y_0^i + u_0^i) \\ & & & & & \vdots & & \\ x_0^i & y_0^i & 1 & 0 & 0 & 0 & -u_0^i x_0^i & -u_0^i y_0^i \\ & & & & & \vdots & & \\ 0 & 0 & 0 & 1 & 0 & 0 & -(A_4^i x_0^i + v_0^i) & -A_4^i y_0^i \\ & & & & & \vdots & & \\ 0 & 0 & 0 & 0 & 1 & 0 & -A_5^i x_0^i & -(A_5^i y_0^i + v_0^i) \\ & & & & & \vdots & & \\ 0 & 0 & 0 & x_0^i & y_0^i & 1 & -v_0^i x_0^i & -v_0^i y_0^i \\ & & & & & \vdots & & \end{bmatrix}_{6N\times 8} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix}_{8\times 1} \quad (2.32)$$

where (u_0^i, v_0^i) and (x_0^i, y_0^i) denote the centers of tile i in I_1 and I_2 for $1 \leq i \leq N$. Note that Eq. (2.32) is a superset of Eq. (2.30). Again, the pseudoinverse solution is computed to solve for the eight perspective parameters. The solution to Eq. (2.32) is normally not stable if pixel coordinates are directly used. We therefore perform a simple normalization to bring the pixel coordinates into the [0,1] range.

2.4.3 Piecewise Affine Approximation

Consider an observed image I_2 subdivided into a regular grid of tiles. For each tile T_2^i, we perform affine registration using the Levenberg-Marquardt algorithm to search for the most similar tile T_1^i in reference image I_1. This yields a collection of affine parameters

and tile centers. Note that all tiles must share the same dimensions in order to compute χ^2. The algorithm is outlined below.

Piecewise Affine Approximation Algorithm

begin

Partition I_2 into a collection of N tiles: T_2^i, for $1 \leq i \leq N$

$i \leftarrow 1$

while $(i < N)$ **do**

Select tile T_2^i

for all positions (x, y) in I_1 **do**

Crop tile T_1^i in I_1

Register T_2^i with T_1^i using LMA

if $\chi^2(\mathbf{a})$ is minimum **then do**

$x^i \leftarrow x$

$y^i \leftarrow y$

$\mathbf{a}^i \leftarrow \mathbf{a}$

end if

end for

$i \leftarrow i + 1$

end while

Solve Eq. (2.30) or Eq. (2.32) for perspective parameters using pseudoinverse solution

end

2.4.4 Examples

Fig. 2.9 demonstrates the proposed algorithm. The top row of the figure depicts an image subdivided into a uniform grid of 2×2, 4×4, and 8×8 tiles, respectively. We consider these three resolutions to demonstrate the relationship between tile size and approximation error.

The second row shows the effects of a perspective warp applied to the input images. The overlaid grids and the marked tiles centers clearly illustrate the foreshortening effects of perspective.

The piecewise affine approximation of a perspective transformation is shown in the third row. Notice that each input tile has undergone an affine warp that best approximates the four corner mapping to its corresponding tile in the warped image. The true perspective grids (second row) are superimposed to illustrate the error with respect to the ground truth.

The positions of the tile centers from the input (first row) and piecewise affine images (third row) are supplied to Eq. (2.30) to infer a global perspective transformation. The results are shown in the fourth row, with the true perspective grids (and tile centers) overlaid on the figure. The mapping is due to the perspective transformation inferred directly from the correspondences of the tile centers. The last row of the figure shows the improved approximation due to Eq. (2.32), when the local affine parameter estimates are used alongside the point correspondences.

The Taylor series approximation is adequate in only a small neighborhood around a point, i.e., tile center. Clearly, a 2×2 tile subdivision is too coarse, as depicted in the first column of Fig. 2.9. Finer subdivision into 16 and 64 tiles, as shown in the second and third columns of the figure, produce greatly improved results. The use of small tiles thus enforces the assumption implicit in the Taylor series, and thereby helps reduce errors.

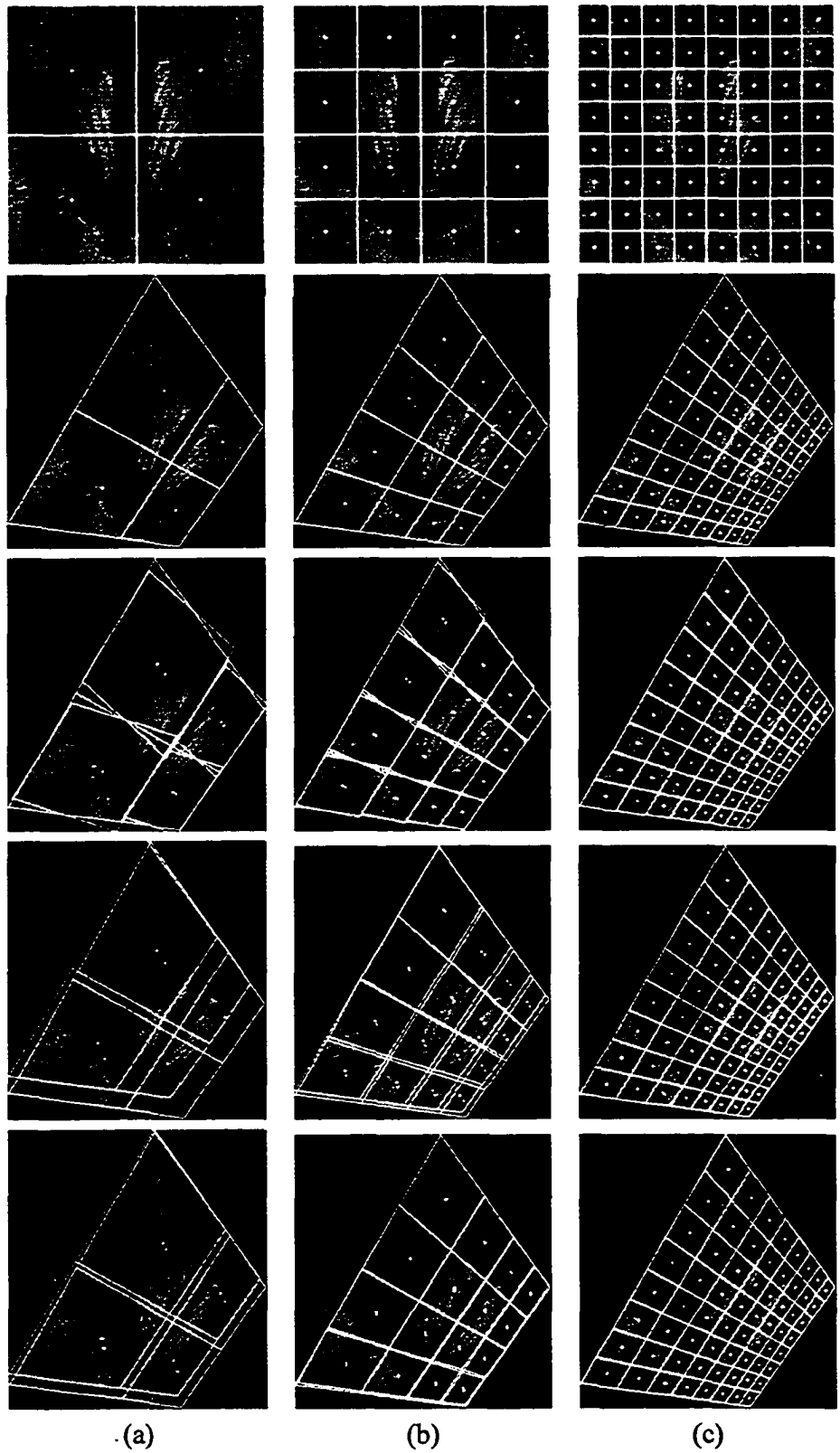


Figure 2.9: Piecewise affine approximation of perspective with (a) 2×2 tiles, (b) 4×4 tiles, and (c) 8×8 tiles.

Fig. 2.10 demonstrates the algorithm on a pair of images subjected to a large perspective deformation. Fig. 2.10(c) illustrates a set of manually chosen tiles drawn from Fig. 2.10(a). These tiles then undergo affine registration to establish correspondence with Fig. 2.10(b). The point correspondences of the tile centers were sufficient to achieve the perspective registration shown in Fig. 2.10(d). Notice that the minimal double-exposure effect in the figure is evidence of a good match.

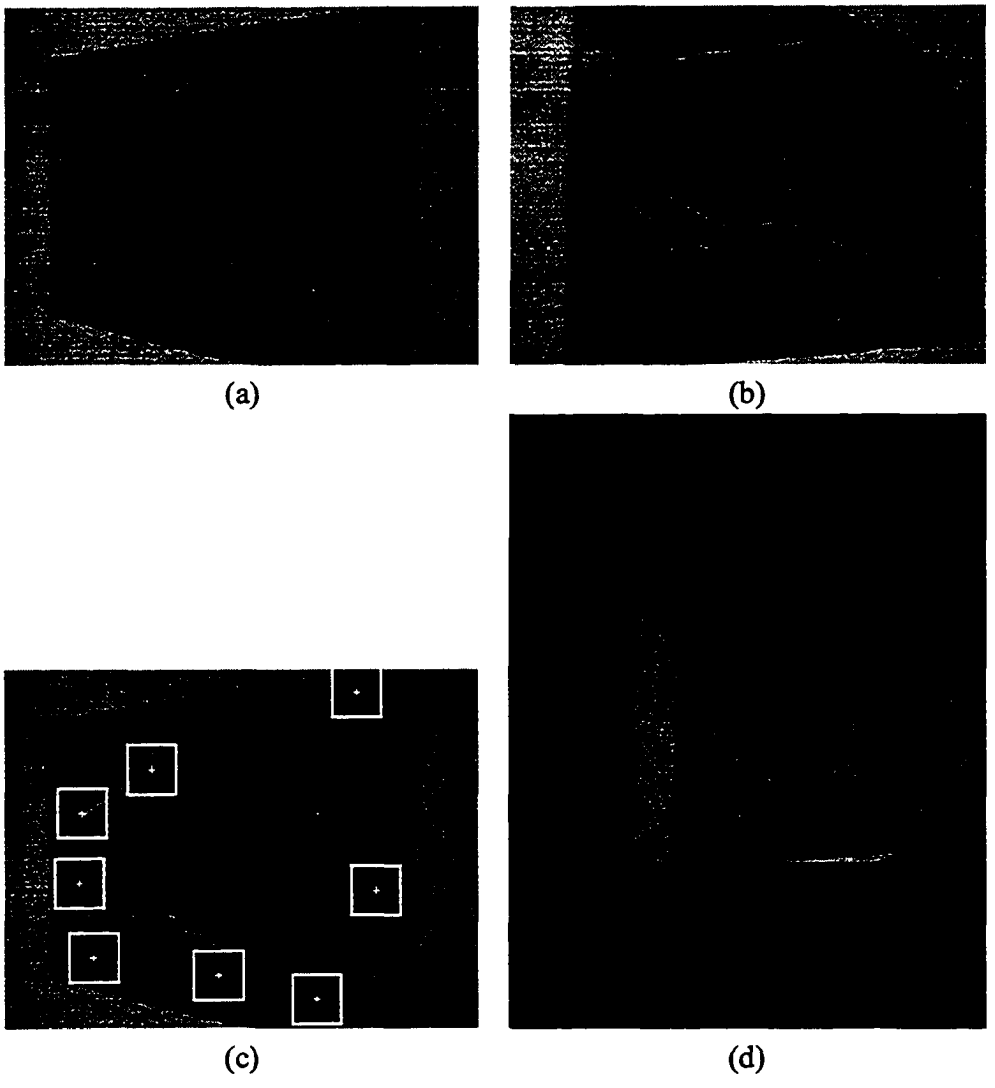


Figure 2.10: Perspective registration. (a) Image I_1; (b) Image I_2; (c) Selected tiles; (d) Overlay of registered I_1 on I_2.

The choice of tiles is important to the process. The perspective transformation computed above was applied to a regular grid. A piecewise affine approximation was applied to the warped grid to yield Fig. 2.11. Notice that many of the image tiles are lacking proper visual structure that would have made accurate matches possible. Therefore, only tiles replete with statistically significant data must be registered. The tiles shown in Fig. 2.10 were selected for their rich visual content.

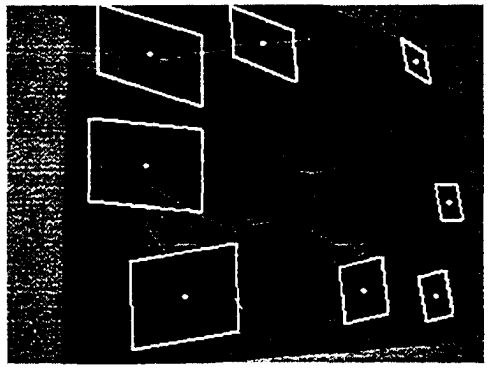


Figure 2.11: Piecewise affine approximation.

2.5. SUMMARY

A method to register image pairs subjected to large perspective deformations has been presented. The images are assumed to be acquired under a weak perspective camera model or a fixed center of projection. These are necessary conditions to avoid parallax and facilitate registration in the presence of a global perspective transformation.

We have demonstrated the feasibility of approximating a perspective transformation as piecewise affine. A reference image is subdivided into tiles and affine registration is applied to find the best matching tiles in the target image. These tile pairs are used to estimate a global perspective transformation. In this manner, we exploit recent results in affine registration to solve the more difficult perspective registration problem.

In affine registration, we estimate the affine parameters necessary to register any two digital images misaligned due to rotation, scale, shear, and translation. The parameters are selected to minimize the sum of squared differences between the two images. They are computed iteratively in a coarse-to-fine hierarchical framework using a variation of the Levenberg-Marquardt nonlinear least squares optimization method. This approach yields a robust solution that precisely registers images with subpixel accuracy.

After applying affine registration to all tiles, the correspondence of tile centers are used to solve for the eight perspective parameters. The accuracy of the technique is further refined by considering the local affine parameter estimates.

A critical component to this work is the affine registration among tiles. The choice of tiles is important to the process. Proper tile selection is the major drawback to this approach. There are several key issues that must be resolved in order for this method to be viable. The minimal set of tiles necessary to achieve accurate registration must be determined. The tile size is also critical to the accuracy and computational performance of the algorithm. There is a tradeoff between registration accuracy and the Taylor series approximation error. Although the use of small tiles conforms more closely to the Taylor series approximation, it lowers the registration accuracy because it offers less structure and visual detail. An automated method of determining this tradeoff must be resolved. Due to these difficult problems, this thesis explores a more promising approach based on log-polar transformations for recovering large perspective deformations.

Chapter 3

LOG-POLAR REGISTRATION

3.1. INTRODUCTION

Despite the large body of literature devoted to image registration, the registration of images subjected to large perspective distortions remains a challenging problem. Most registration algorithms are limited to small linear or affine distortions. Our objective in this thesis is to investigate robust image registration algorithms for geometrically aligning images subjected to large perspective deformations. We seek to introduce the use of log-polar techniques to invert perspective deformations among image pairs. We propose to achieve subpixel accuracy through the use of a hybrid algorithm that features log-polar mappings and nonlinear least squares optimization. The registration process will yield the eight parameters of the perspective transformation that best aligns the two input images.

In this section, we introduce a log-polar registration module that serves as a preprocess to the parameter estimation module. Although the parameter estimation method features subpixel accuracy, the two images to be registered must first be fairly close in scale (within a factor of two), rotation (within 45°), and translation. The purpose of the newly added module is to account for large geometric transformations, bringing images into close align-

ment even in the presence of large (ten-fold) scale changes, as well as arbitrary rotations and translations. In practice, we don't expect scale changes beyond factors of five, however arbitrary rotation angles and translations must surely be addressed.

We review log-polar coordinate transformations and describe their use in registration. Section 3.2 begins with a review of polar coordinate transformations and their use in recovering rotation. Then, we present log-polar transformations in Section 3.3 for the purpose of recovering scale, as well as rotation. Section 3.5 discusses Fourier-Mellin method, its limitations, and the review of related work. Section 3.6 discusses a method to recover global rotation, scale, and translation. Section 3.8 examines the effect of similar, affine, and perspective deformations in the log-polar domain. This serves to motivate the proposed registration technique.

3.2. POLAR TRANSFORM

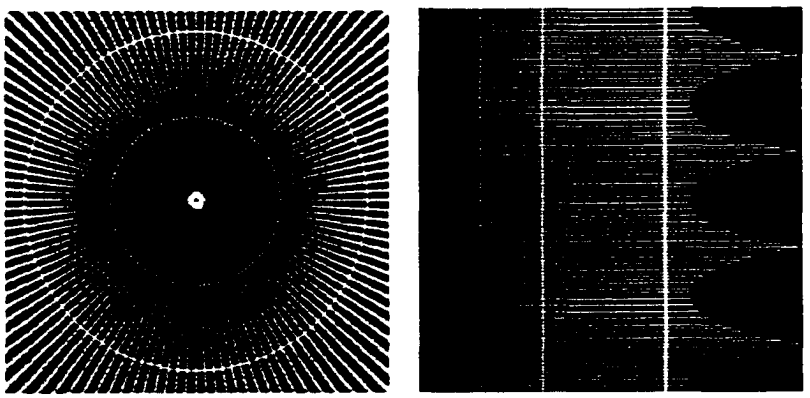
Consider the polar (r, θ) coordinate system, where r denotes radial distance from the center (x_c, y_c) and θ denotes angle. Any (x, y) point can be represented in polar coordinates:

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} \tag{3.1a}$$

$$\theta = \tan^{-1}\left(\frac{y - y_c}{x - x_c}\right) \tag{3.1b}$$

Applying a polar coordinate transformation to an image I maps radial lines in Cartesian space to horizontal lines in the polar coordinate space. We shall denote the transformed image I_p. If we assume that r and θ lie along the horizontal and vertical axes, respectively, then image I shown in Fig. 3.1(a) will be mapped to image I_p in Fig. 3.1(b) after a polar

coordinate transformation. Note that the origin of I_p lies in the upper left corner. Also, the lobes on the right side of I_p are due to the varying extent of r across all angles. Note, for instance, that the longest rows in I_p correspond to the four radial lines in I that extend to the four corners of the image.

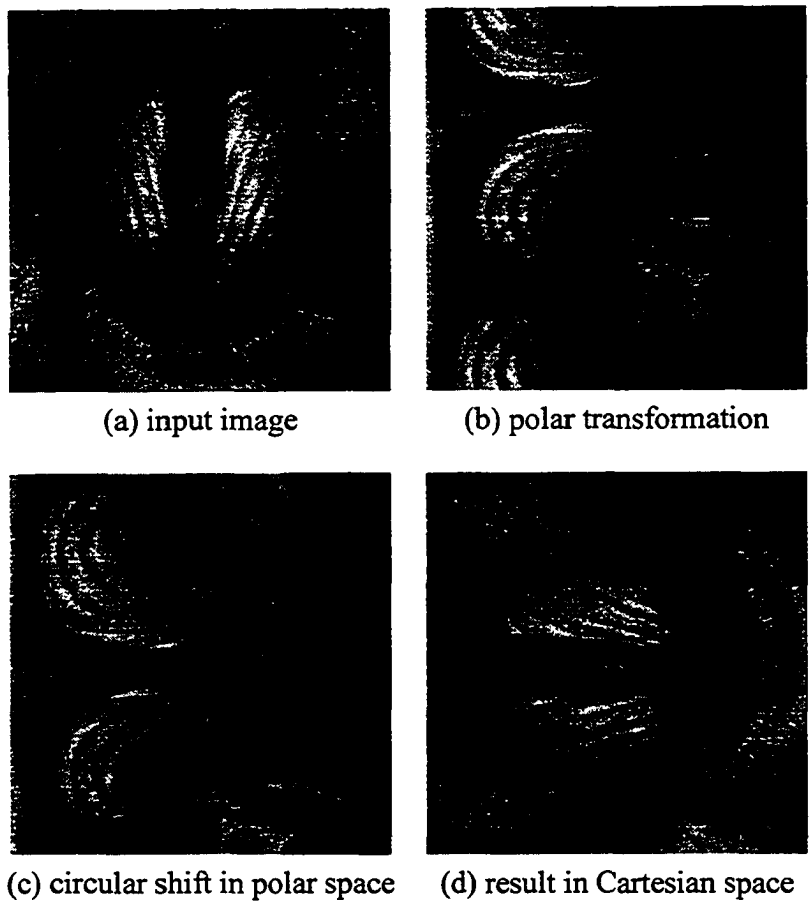


(a) input image (b) polar transformation

Figure 3.1: Polar coordinate transformation.

The benefit of the polar coordinate system is that simple rotation may be induced by modifying the θ data. That is, a circular shift along the θ-axis in I_p induces a rotation of I. Recall, afterall, that the data stored along the rows in $r\theta$-space represent radial data in xy. By moving the I_p rows up or down, that radial data will map to a new (rotated) set of radial lines. Consider images I and I_p shown in Figs. 3.2(a) and 3.2(b), respectively. Fig. 3.2(c) shows a circular shift applied to Fig. 3.2(b). The resulting image in Cartesian space is shown in Fig. 3.2(d). Notice that since we shifted the 512-row image in Fig. 3.2(b) by 128 rows, the image in Fig. 3.2(a) has been rotated by 90°. Similarly, resampling in the r-axis can effect scale changes.

Consider two images I_1 and I_2 and their polar counterparts I_{1p} and I_{2p}. If I_2 is a rotated version of I_1, then I_{2p} will be a circularly shifted version of I_{1p} along the θ-axis. By applying cross-correlation to I_{1p} and I_{2p}, the offset $\Delta\theta$ can be found that best matches



(a) input image (b) polar transformation

(c) circular shift in polar space (d) result in Cartesian space

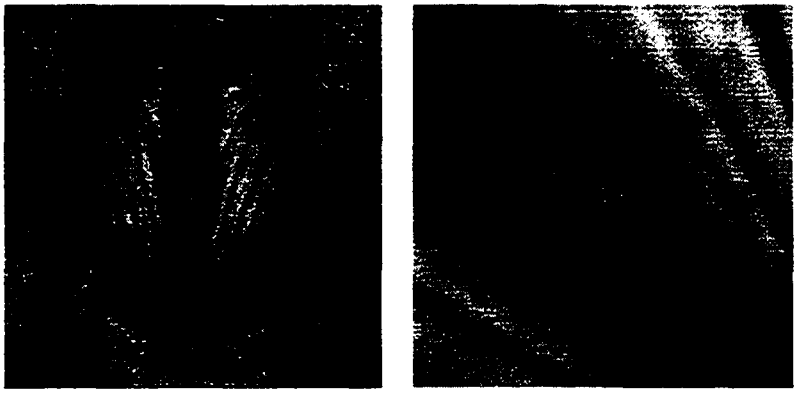
Figure 3.2: Polar coordinate transformation.

the two images. Recall that cross-correlation is commonly used for finding translational offsets (phase shifts) between two images. It does not work well in the presence of scale or rotation. However, in the polar coordinate space, finding the translational component $\Delta\theta$ between I_{1p} and I_{2p} corresponds to finding the rotation between I_1 and I_2. There would be, for instance, no problem in computing the shift between Figs. 3.2(b) and 3.2(c).

3.3. LOG-POLAR TRANSFORM

The log-polar coordinate system is useful for recovering scale, as well as rotation. Consider a four-fold magnification of image I_1, yielding an enlarged image I_2. All points

(x, y) in I_1 now map to $(4x, 4y)$ in I_2. To determine the scale factor, we introduce the use of logarithms. In log-space, $(x, y) \rightarrow (\log x, \log y)$, and $(4x, 4y) \rightarrow (\log 4x, \log 4y) \rightarrow (\log x + \log 4, \log y + \log 4)$. It now becomes apparent that in log-space, the introduction of a scale factor manifests itself as a shift in the log-transformed image. Therefore, instead of mapping an image into (r, θ) coordinate space, it becomes useful to map it into $(\log r, \theta)$ coordinate space using a log-polar transformation. Ordinary cross-correlation in the log-polar space now determines the best $\Delta(\log r)$ and $\Delta\theta$ phase shifts, which translates to scale and rotation in Cartesian space.



(a) input image (b) scale=4; rotation=45°

(c) log-polar transform of (a) (d) log-polar transform of (b)

Figure 3.3: Log-polar coordinate transformation.

Fig. 3.3 shows an example. Figs. 3.3(a) and 3.3(b) depict two input images. The latter

figure is a scaled, rotated, and cropped version of Fig. 3.3(a). The result is consistent with sensor movement that causes a four-fold zoom and a 45° rotation. Their log-polar transformations are shown in Figs. 3.3(c) and 3.3(d). Clearly, they differ by a phase shift. Compare, for instance, the left half of Fig. 3.3(c) with the right half of Fig. 3.3(d). When ordinary cross-correlation is applied to those two images, the computed shift accurately reflects the four-fold scale factor and 45° rotation that constitutes the transformation between the two input images.

We have used the log-polar transformation to recover differences as large as ten-fold scale changes for synthetically enlarged images. Given this superior performance over the state-of-the-art in frequency domain solutions, we believe that this approach should be exploited. There is one difficulty, though, that remains to be solved: finding the origin of the image from which the radial distance is measured. In the examples above, we have assumed that the origin of both images lie at their geometric centers. In fact, their centers can be displaced and unless that correspondence (translation) is known, the information derived from the polar transformation is of limited value.

3.4. BIOLOGICAL MOTIVATION FOR LOG-POLAR TRANSFORM

The log-polar transformation is a nonlinear and nonuniform sampling of the spatial domain. Nonlinearity is introduced by polar mapping, while nonuniform sampling is the result of logarithmic scaling. Despite the difficulties of nonlinear processing for computer vision applications, the log-polar transform has received considerable attention. The motivation for considering the log-polar transform stems from its biological origins. The first reported discoveries of log-polar mappings in the primate visual system were reported in [75] and [54]. The log-polar mapping is an accepted model of the representation of the

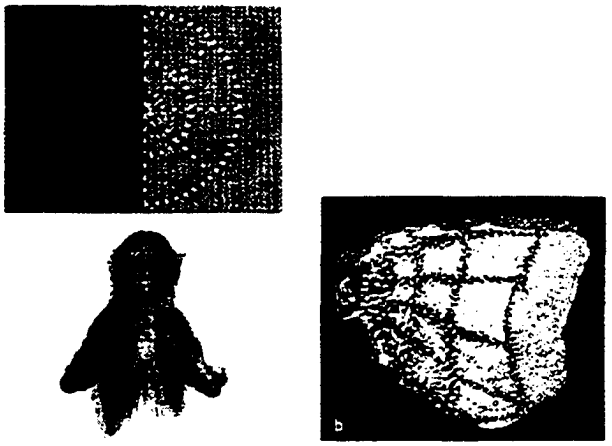
retina in the primary visual cortex in primates, also known as V1 [93, 92, 115]. This has been verified in several experiments [42, 105, 36]. In the experiments described in [92], a stimulus pattern was displayed to a monkey and the electrical activities of neurons in the primary visual cortex was captured (Fig. 3.4).

The nonuniform sampling that simulates logarithmic scale takes place in the retina (Fig. 3.5). There are two groups of photosensitive cells on the retina: cones and rods. Rods are about 500 times more sensitive to light than cones, but cones give us color vision. Rods and cones are arranged side by side like the pile of a carpet, with 120 million rods and 6 million cones comprising the human photoreceptor layer. The distribution of these photoreceptors varies across the surface of the retina. Note, for instance, that there are no rods in the fovea, and very few cones are found at the periphery, where rods predominate. As we move towards the peripheral edge, rods become shorter and wider (Fig. 3.7).

The nerve endings from the retina are connected to the visual cortex by a special mapping. This mapping realizes the polar transformation by a simple rewiring. The radial nerve endings are connected horizontally to the visual cortex. Due to these biological origins, the log-polar transform has often been referred to as the retino-cortical transform [116].

In Section 3.3 we presented one of the principal advantages of log-polar transform: rotation- and scale-invariance. The spatially-varying sampling in the retina has a secondary advantage: it is the evolutionary solution to reduce the amount of information traversing the optical nerve while maintaining high resolution in the fovea and capturing a wide field of view. This bandwidth reduction helps us process a high resolution image only at the focus of attention while aware of a wider field of view. Although the inherent bandwidth reduction of the log-polar transform is compelling, this thesis concentrates on the geometric properties of the log-polar transform to achieve registration in the presence of large-scale

geometric changes.



(a) experiment setup (b) neurons activities

Figure 3.4: Log-Polar mapping in the primates visual cortex. From *Scientific American*, vol. 12, no. 1, August 2002.

Several researchers have designed log-polar sensors for active and real-time vision applications [44, 117, 8, 19]. These efforts sought to make the leap from biological hardware to VLSI hardware. A VLSI space-variant sensor based on the complex-log map has been developed by Van der Spiegel and his colleagues [35, 43]. The design is based on Charged Coupled Device (CCD) technology (Fig. 3.6). Another real-time log-polar sensor designed with a commercially off-the-shelf (COTS) CCD camera and a programable DSP is described in [13].

Weiman laid a theoretical groundwork for using log-polar images to detect lines in log-Hough space [114, 48, 112] and Young [122] extended it for finding lines and circles. Sandini et. al. have used the log-polar transform in different computer vision areas. First, recognizing 2D objects that were arbitrarily rotated or scaled [89][44]. Second, tracking a moving object [24]. Third, estimation of time-to-impact from optical flow. Fourth, finding disparity map in stereo images [73]. Pardo et. al. have designed a system using log-polar transforms to calculate time-to-crash for mobile vehicles [81]. Bernardino has designed a

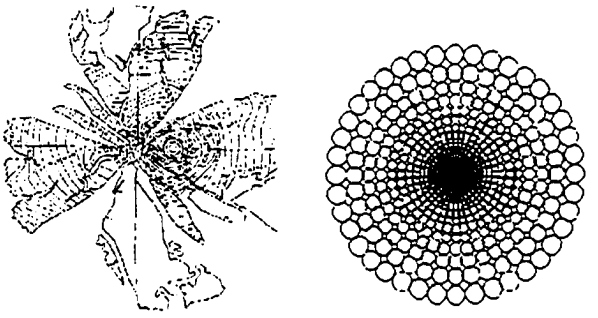


Figure 3.5: The figure shows contours of cone cell density. In fact, the density of the ganglion cells which transmit information out of the retina is very close to logarithmic. From Osterberg, G. (1935) "Topography of the layer of rods and cones in the human retina."

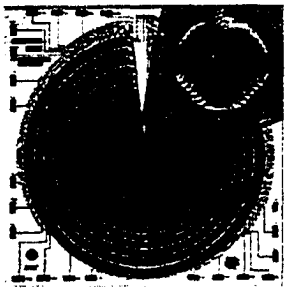


Figure 3.6: Picture of the VLSI log-polar sensor designed by Van der Spiegel.

foveated binocular stereo system using log-polar [16].

Weiman showed that log-polar coordinates yield time-to-collision without knowledge of pixel coordinates nor 3D target position or velocity, and that log-polar coordinates maintain much wider image registration than Cartesian coordinates for binocular disparity [111]. Weiman applied these properties in the design of a real-time binocular motion platform video-rate remapper delivered to the NASA Johnson Space Center on a mobile robot for docking and tracking [113]. Vincze and Weiman showed that space-variant systems such as those typified by log-polar or pyramidal transformations actually improve tracking performance as window size is increased, unlike Cartesian rasters where processing time crosses over to degrade performance [107]. Together, these properties provide a rationale for the

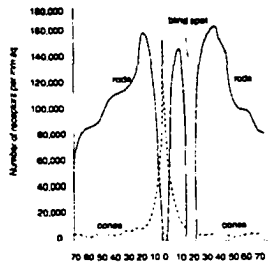


Figure 3.7: Density distributions of rod and cone receptors across the retinal surface. Adapted from Pirenne (1967).

adoptation of log-polar coordinates in the advanced biological vision systems of primates.

3.5. FOURIER-MELLIN METHOD

The Fourier-Mellin registration method is based on phase correlation and the properties of Fourier analysis. As explained in Section 1.5.4, the phase correlation method can find the translation between two images. The Fourier-Mellin transform extends phase correlation to handle images related by both translation and rotation [25, 26, 29, 87, 68, 28, 69, 70]. According to the rotation and translation properties of the Fourier transform, the transforms will be related by

$$I_1(x,y) = I_2(x\cos\theta_0 + y\sin\theta_0 - x_0, -x\sin\theta_0 + y\cos\theta_0 - y_0) \quad (3.2a)$$

$$F_1(\omega_x,\omega_y) = F_2(\omega_x\cos\theta_0 + \omega_y\sin\theta_0, \omega_x\sin\theta_0 + \omega_y\cos\theta_0)e^{j(\omega_x x_0+\omega_y y_0)} \quad (3.2b)$$

We can see that the magnitude of spectra $|F_1|$ is a rotated replica of $|F_2|$, as depicted in the first row of Fig. 3.11. Both spectrum share the same center of rotation. We can recover

this rotation by representing the spectra $|F_1|$ and $|F_2|$ in polar coordinates.

$$|F_1(r,\theta)| = |F_2(r,\theta-\theta_0)| \tag{3.3}$$

The Fourier magnitude in polar coordinates differs only by translation. We can use the phase-correlation method to find this translation and estimate θ_0. This method has been extended to find scale by mapping Fourier magnitude to log-polar coordinates. The effects of the rotation and scale on the Fourier magnitude are shown in Fig. 3.11. Therefore, one finds scale and rotation by phase-correlation, which recovers the amount of shifts in $(\log r, \theta)$ space. One advantage of this method is that it tolerates additive noise. The method, however, can only recover moderate scales and rotations. This difficulty can be understood by realizing that large rotation and scale changes exacerbate the border effects when computing the Fourier transform. These problems are minimized in the rare case when the images are periodic. Therefore, a large translation, or scale introduces additional pixel information that can dramatically alter the Fourier coefficients.

In early papers on Fourier-Mellin, the border problems were not investigated. They were, however, reported recently in [98, 99], where the authors showed that rotation and scale introduce aliasing in the low frequencies. They have suggested that two preprocessing steps are needed to alleviate the aliasing problem. First, we multiply the image by a radial mask consisting of a 2D Gaussian function (Fig. 3.8). This produces the output shown in Fig. 3.9(c). Second, we find the Laplace transform of the input images to remove the offending low frequencies, as shown in Fig. 3.9(e).

In the Fourier-Mellin method, first we recover the rotation and scale independent of translation. After warping the target image based on the recovered rotation and scale, phase correlation is used to search for translation. We have tested this method for ideal cases,

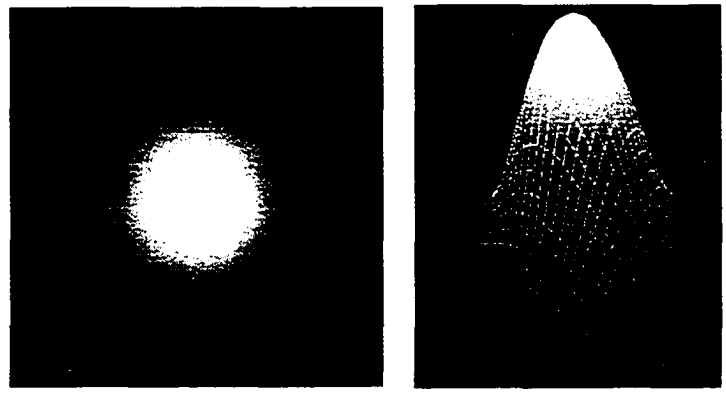
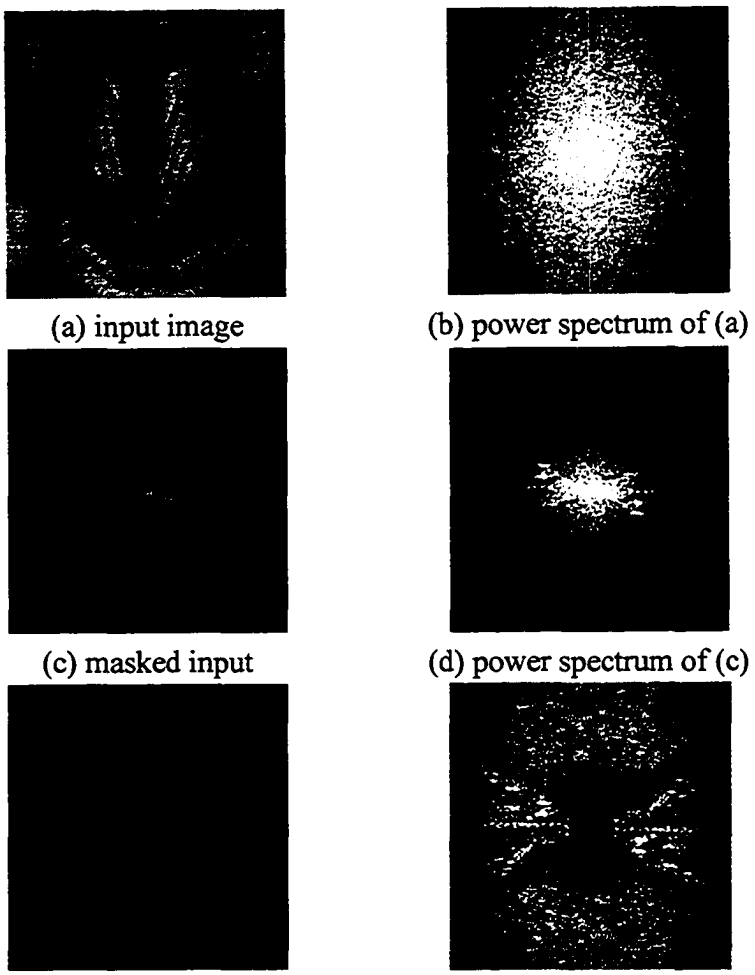


Figure 3.8: The shape of the mask to reduce the border problem.

whereby known rotation and scale parameters are applied to reference images. In the first example, we rotate and scale the reference image about its center. We show phase correlation surfaces for a variety of rotation and scale factors ($\theta \in [0 \cdots 180°]$, $s \in [1.0 \cdots 7.0]$)) in Fig. 3.12. The location of the strong peak in these images corresponds to the amount of the rotation and scale. As we increase the zoom, the maximum peak gets weaker compared to the rest of the coefficients and eventually will be lost. It is clear that rotation does not adversely affect peak detection. A circle around the peak represents the correct position for the amount of rotation and scale applied to the reference image. A crosshair on the maximum peak means the phase-correlation has failed to recovered the rotation and scale. The system is able to recover scale factors not exceeding 4.0, and arbitrary rotation when there are no translations (Fig. 3.12). This method starts to degenerate when the input images do not share a common center and there are translations Fig. 3.13. When the reference image is translated 20 pixels, we are able to recover scale factors up to 3.0. Our application of a radial mask permitted us to capture more accurate results than those described in [87], where it was reported that they recovered scale factors up to 1.8 and 80° rotations. Recently, [58] noted that they were able to recover scale factor up to 4.0.

It is important to note that the literature is replete with synthetic examples for the



(a) input image (b) power spectrum of (a)

(c) masked input (d) power spectrum of (c)

(e) masked input and Laplace transform (f) power spectrum of (e)

Figure 3.9: Steps to improve the Fourier-Mellin method for registering images.

Fourier-Mellin registration method. In particular, a reference image is always matched against a scaled and rotated version of itself. This serves to defer the problem of handling the fine detail due to an actual optical zoom. Conversely, when the image undergoes minification, translation, or rotation, additional real data seeps into the target image, not just black pixels. Note that artificial black backgrounds can help register two images because it ensures that we consider the same underlying content.

An example demonstrating the differences between digital and optical zoom is shown in Fig. 3.10. As is expected, the shape of the spectrum in Fig. 3.10(c) conforms to the

inverse relationship between space and frequency. However, the spectra of Fig. 3.10(b) reflects the fact that the images were taken with optical zoom and minor perspective distortion due to real hand movement. Applying the Fourier-Mellin transform to these image pairs defies recovery because of the lack of similarity in their spectra. However, we are able to correctly register the synthetic versions shown in Fig. 3.10(c).

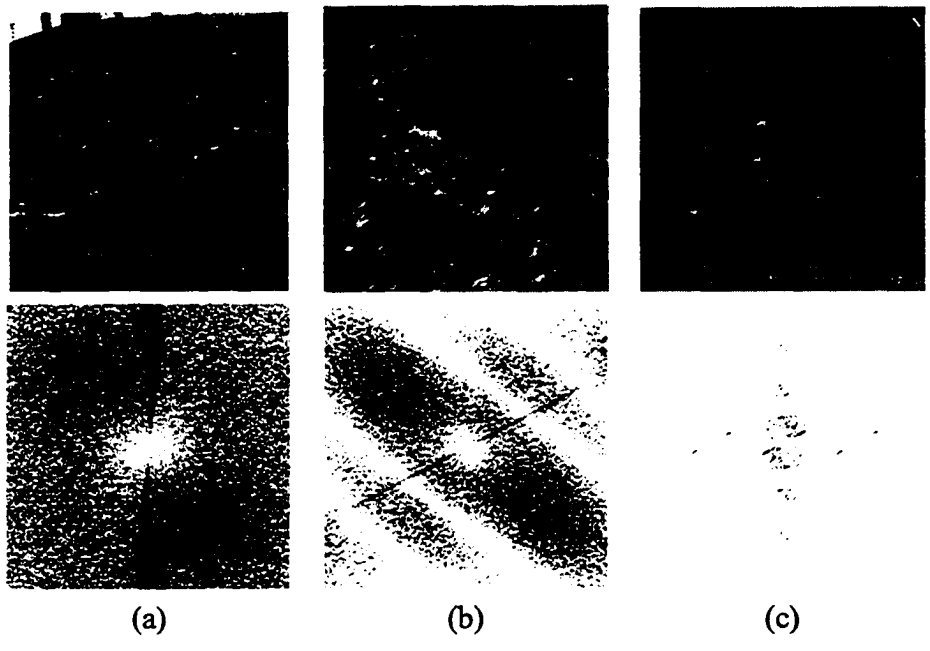


Figure 3.10: (a) Reference image; (b) Target image (real); and (c) Target image (synthetic).

An important contribution of this thesis is that we introduce a method based on the log-polar transform in the *spatial domain* that works robustly with real images.

3.6. GLOBAL REGISTRATION USING LOG-POLAR TRANSFORM

3.6.1 Description

We have implemented a new algorithm for automatically finding the translation between both input images in the presence of large scale and rotation [120]. The search

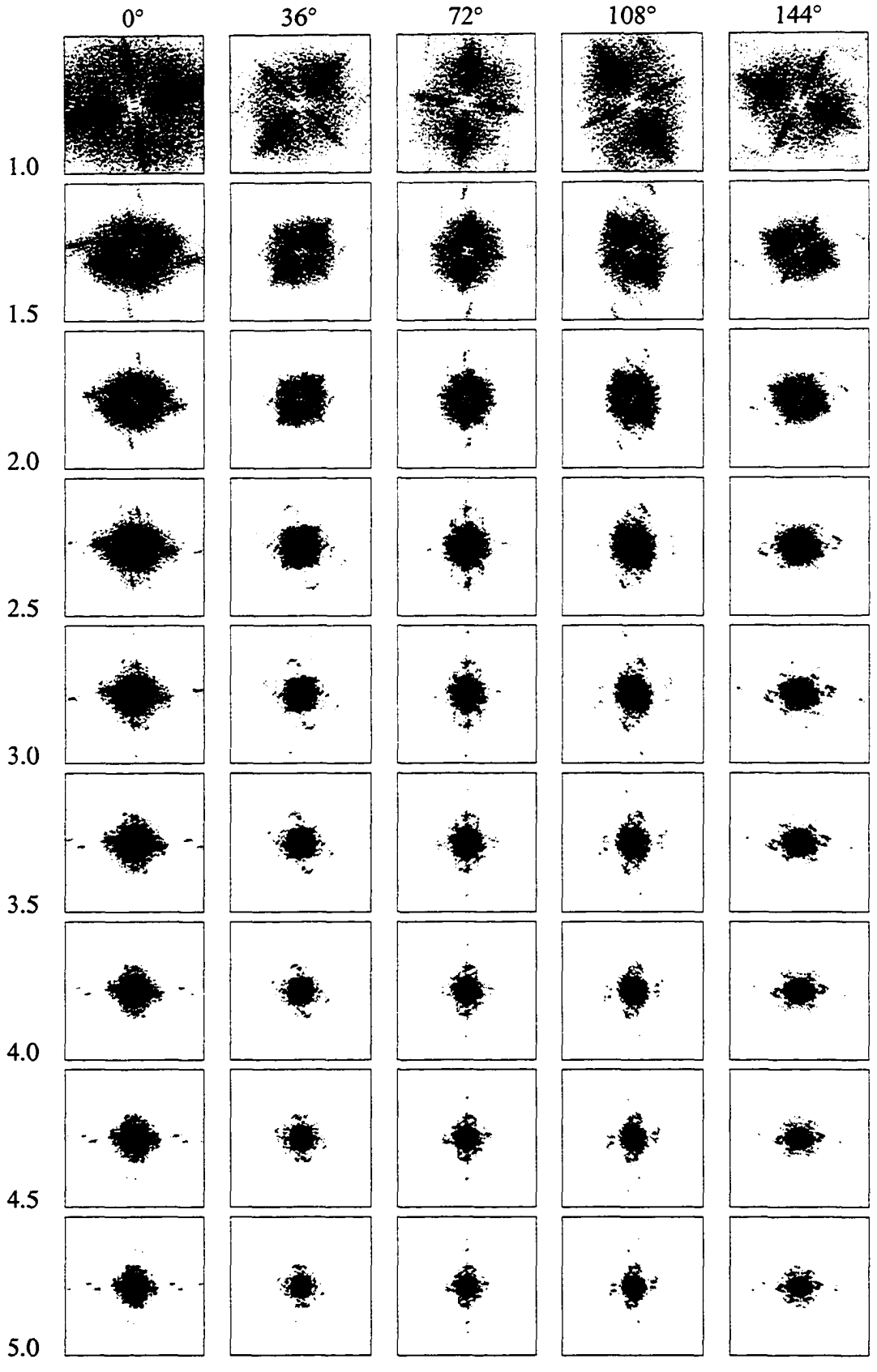


Figure 3.11: The effects of the rotation and scale on the power spectrum.

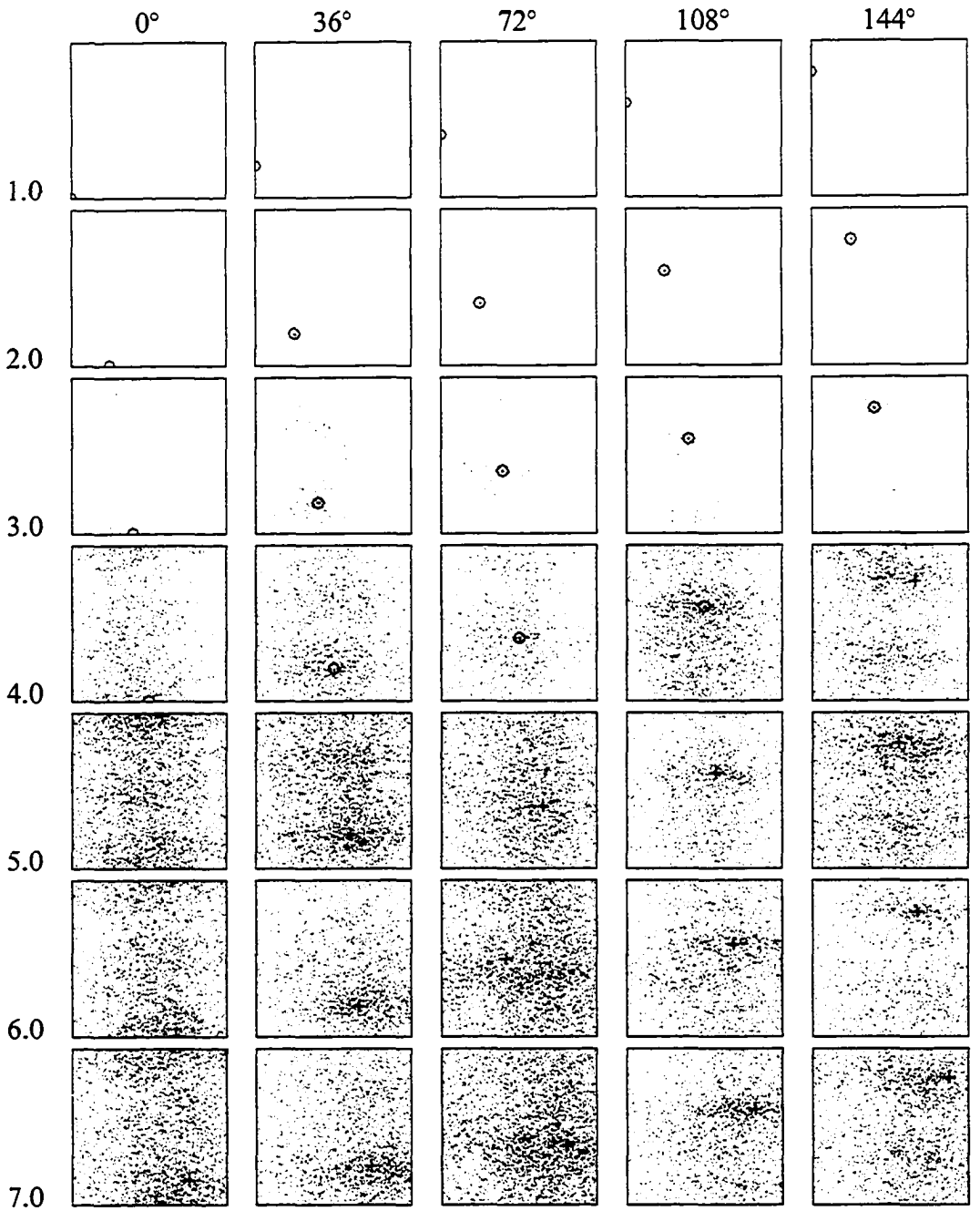


Figure 3.12: Recovered rotation and scale for the ideal case without translation. Peak with circles means correct rotation and scale recovered.

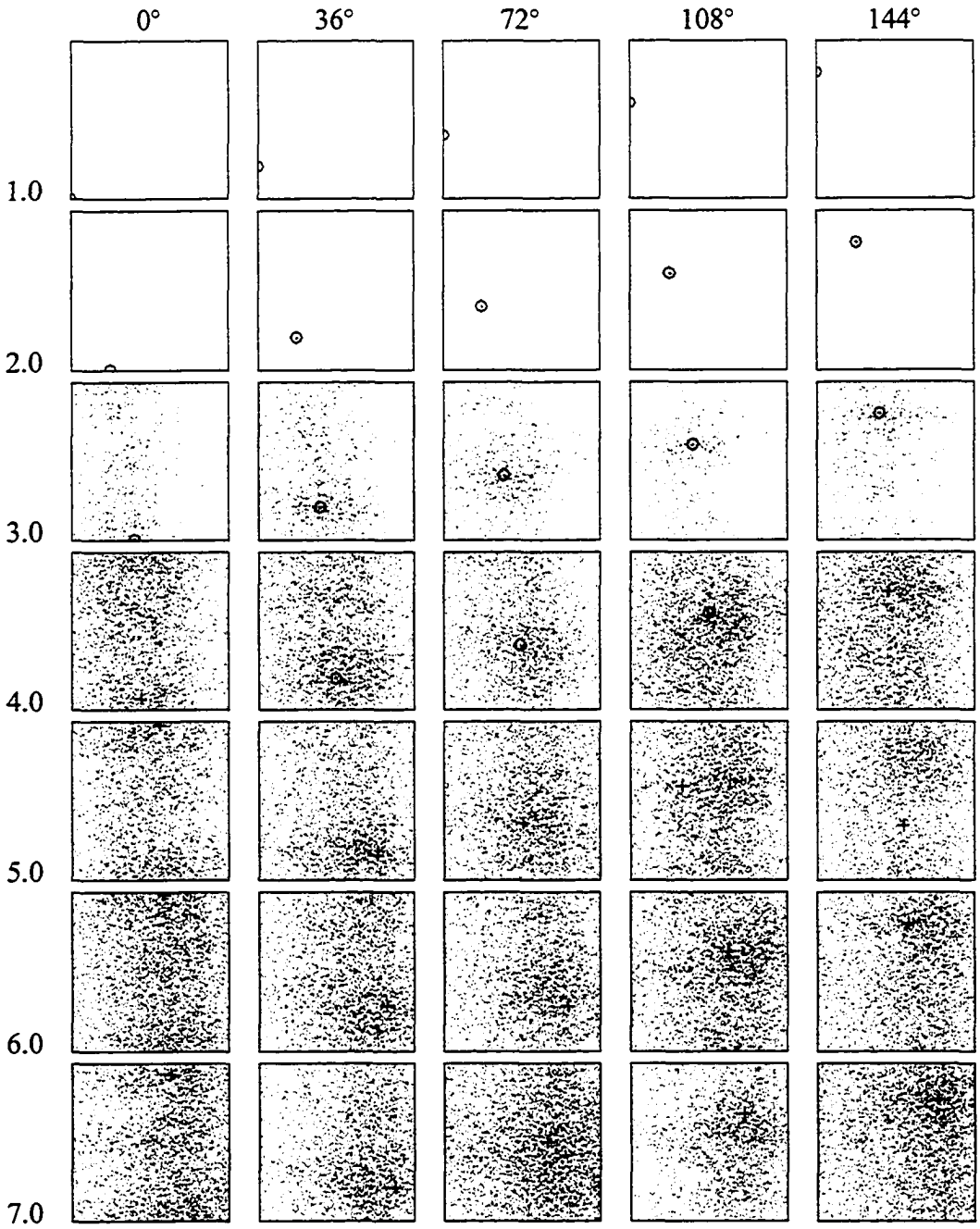


Figure 3.13: Recovered rotation and scale for the ideal case with translation. Peak with circles means correct rotation and scale recovered.

space has four dimensions. The new method is based on multiresolution log-polar transformations to simultaneously find the best scale, rotation, and translation parameters. The coarse-to-fine multiresolution framework accelerates the process by permitting estimates computed in the low resolution images to serve as initial guesses to the higher resolution images. One of the benefits of the discrete log-polar transform is that we quantize the scale and rotation axes. Therefore, we have a finite number of points to search and this number is small at the coarsest level.

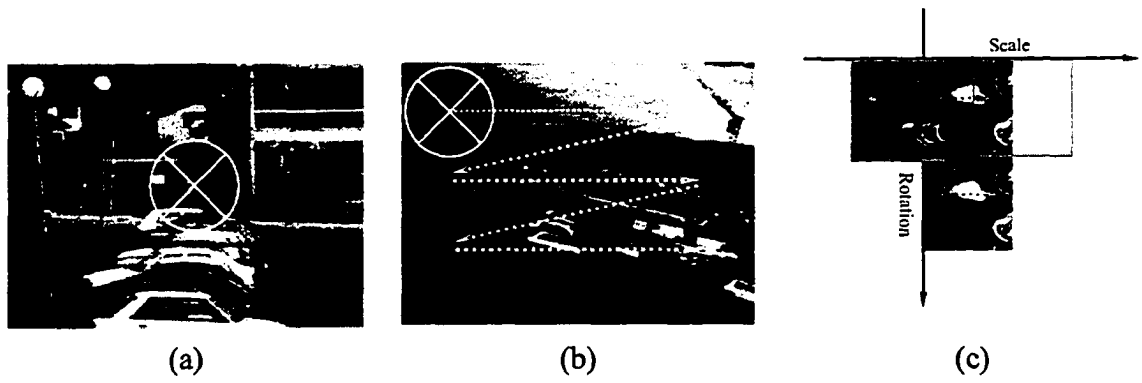


Figure 3.14: 4D search strategy: (a) a circular template from the center of the reference image is cropped; (b) For every position in the target image, a circular region is selected and compared against the circular template in (a) to find the best (T_x, T_y); and (c) search for (R, S) in the log-polar domain.

We crop a circular template from the reference image and compute the log-polar transformation of it. The radius and the center of the template are given by the user. The radius varies from 25% to 10% of the image width Fig. 3.14(a). The default value for the radius is 25% of the input image width and the center of the template is the center of the reference image. Then, for all positions in the target image, we crop a circular image and compute the log-polar transformation Fig. 3.14(b). In the log-polar space, we map $[0° \cdots 360°]$ to $[0 \cdots h]$, where h is the height of the template. Note that we pad the template by wrapping around Fig. 3.14(c). We compute the *base* of logarithm for log-polar transformation as

follows:

$$base = e^{\frac{\log w}{w}} \tag{3.4}$$

where w is the width of the input image ($w = h$ = diameter). The choice of the $base$ is arbitrary. However, we compute the $base$ in Eq. (3.4) to set the width of the log-polar image to that of the input image. We can compute the maximum translation in log-polar space as follows:

$$\frac{\log S_{max}}{\log base} \tag{3.5}$$

we set the maximum scale factor S_{max} to 5.0 to limit the search in the scale direction. We use the normalized correlation coefficient Eq. (1.10) as the similarity measure. The approach at any given level is outlined below.

1. Crop central region I_1' from I_1
2. Compute I_{1p}', the log-polar transformation of I_1'
3. For all positions (x, y) in I_2:

   Crop region I_2'

   Compute I_{2p}'

   Cross-correlate I_{1p}' and $I_{2p}' \rightarrow (dx, dy)$

   If maximum correlation, save (x, y) and (dx, dy)
4. Scale $\leftarrow dx$
5. Rotation $\leftarrow dy$

6. Translation $\leftarrow (x, y)$

The procedure outlined above recovers the origin of the log-polar transform, as well as the global scale and rotation. An example is given in Fig. 3.15. The log-polar transform of the two input images are shown in Figs. 3.15(c) and 3.15(d). Since the input images do not share the same origins, their log-polar images cannot be directly registered. Fig. 3.15(e) shows the log-polar transform of Fig. 3.15(b) after its origin has been translated to coincide with that of Fig. 3.15(a). The resulting log-polar image in Fig. 3.15(e) is now a viable candidate for correlation with Fig. 3.15(c). It is important to note that the translation that brings the origins into alignment was determined as the shift in I_2 necessary to achieve the highest correlation in the log-polar domain. In this manner, we simultaneously search for translation, rotation, and scale. We achieve large performance gains by computing the solution in a multiresolution framework.

3.6.2 Robustness

Fig. 3.16 demonstrates the robustness of the above algorithm for various examples, including several which have moderate perspective deformations. Note that corresponding origins and template outlines are shown in the figure. We will demonstrate the robustness of this algorithm with more examples in chapter 4.

3.6.3 Complexity

Consider two input images with dimensions $m_0 = n_0 \times n_0$. The image will be represented in a pyramidal data structure. The size of the coarsest pyramid level is denoted as $m_L = n_L \times n_L$. By default, we use $n_L = 32$. The number of pyramid levels is $L = log_2 \frac{n_0}{n_L}$. We begin the computation in the coarsest level by cropping a window from the coarsest

(a) I_1

(b) I_2

(c) Log-polar transform of I_1

(d) Log-polar transform of I_2

(e) Log-polar transform of shifted I_2

Figure 3.15: Effect of origin alignment in log-polar images.

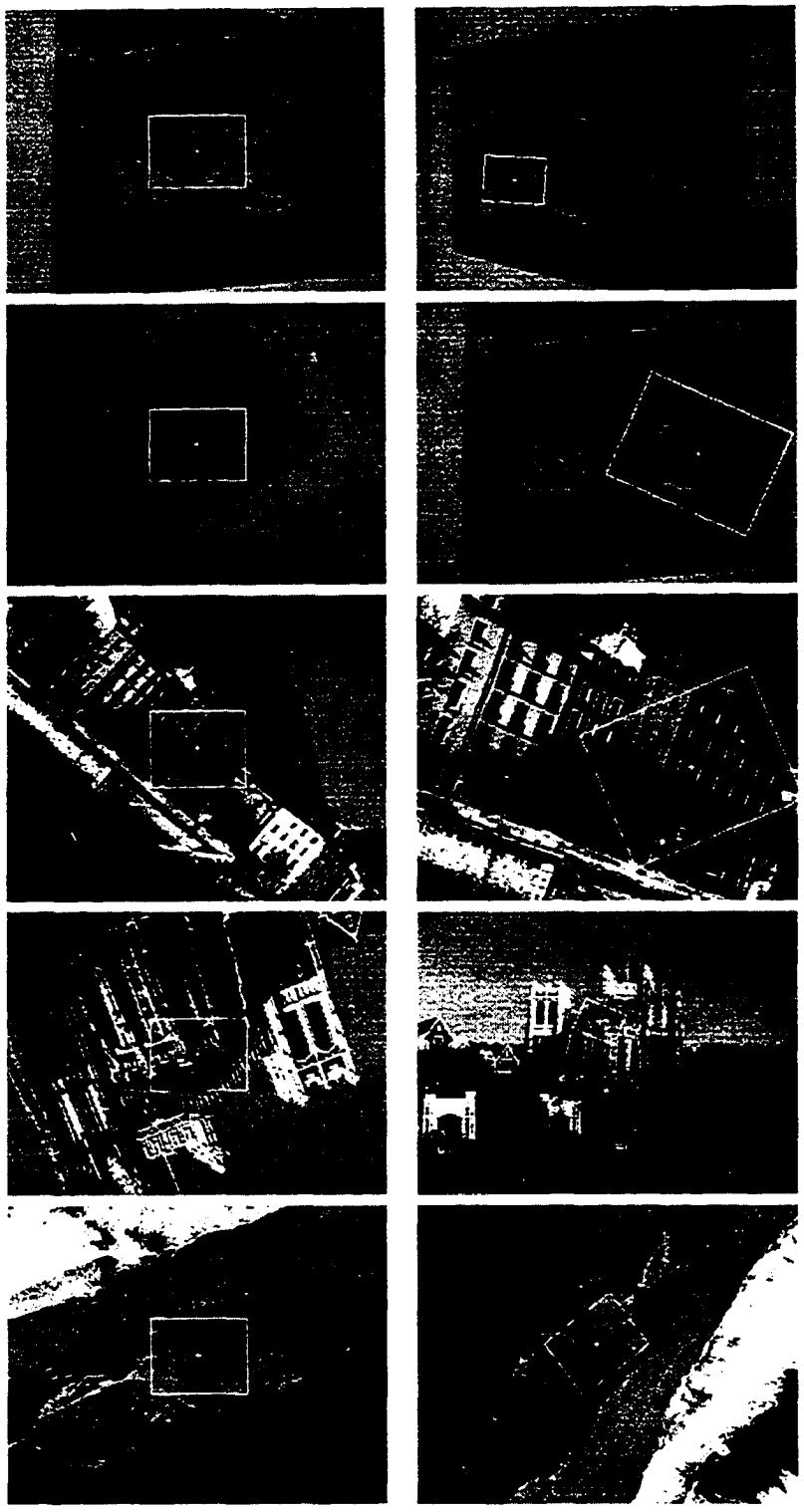


Figure 3.16: Examples of recovering the log-polar origin, scale, and rotation.

level that is 1/4 of n_L. The number of all possible positions of the cropped window in the target image is $(n_L - n_L/4)^2 = 9n_L^2/16$. Then, for each cropped window of size $n_L^2/16$, we apply the log-polar transform and perform cross-correlation. Since the nature of log-polar resampling and cross-correlation is convolution, we assume that their complexities are comparable and on the order of $O(n_L^2/16)$. The complexity of the proposed registration algorithm in the coarsest level is thereby $O(n_L^4) = n_L^2 O(n_L^2)$.

After finding the origin of the log-polar mapping in the coarsest level, we refine the estimates by searching within a small neighborhood around the origin. We linearly reduce this neighborhood between level$_i$ and level$_{i+1}$. The complexity of the whole algorithm is a function of the number of pyramid levels:

$$O(n_L^4) + \sum_{i=L-1}^{0} (i+2)^2 O(n_i^2) \tag{3.6}$$

$$O(m_L^2) + \sum_{i=L-1}^{0} (i+2)^2 O(m_i) \tag{3.7}$$

The number of pixels at level$_i$ is related to the number of pixels at the coarsest level$_L$ as follows: $m_i = m_L(2^{i-L})^2$. Thus the complexity becomes:

$$O(m_L^2) + O(m_L) \sum_{i=L-1}^{0} (i+2)^2 (2^{i-L})^2 \tag{3.8}$$

The left term is the dominant portion of the complexity. Typically, we have three or four levels for our test images. Therefore, the right term in Eq. (3.8) is not a major factor in the performance of our algorithm. If the input image is large enough to generate six or seven levels, the exponential rate in the right term becomes dominant and reduces the performance of our algorithm.

3.7. FEATURE-BASED REGISTRATION: RELATED WORK

Finding local and invariant features is an important tool for detecting correspondences between different views of a scene. Recently, several researchers at INRIA developed a method for recovering large-scale deformation based on scale-space theory [38, 77, 39]. They compute interest points at different scales, calculating at each scale a set of local descriptors that are invariant to rotation, translation, and illumination. The Mahalanobis distance is then used to find the corresponding interest points between two images. In order to remove the outliers, they use the RANSAC algorithm with constraints based on collections of points.

3.7.1 Interest Point Detection for Image Matching

An affine-invariant operator is needed to detect interest points in an image. It has been claimed that scale-invariant operators that detect image features are hard to formulate in practice. One popular operator is the Harris corner detector [52], which is invariant to rotation, translation, and illumination [91]. The Harris corner detector computes the locally averaged auto-correlation matrix. It operates on the image gradients to yield a 2×2 matrix A at each pixel $P(x, y)$. It then combines the two eigenvalues of A to compute a corner strength. If this strength exceeds a threshold, then pixel P is deemed to be a corner point. Matrix A averages signal derivatives in a window w around point (x, y):

$$A(x,y) = \begin{bmatrix} \sum_{(x_i,y_i)\in w} I_x(x_i,y_i)^2 & \sum_{(x_i,y_i)\in w} I_x(x_i,y_i)I_y(x_i,y_i) \\ \sum_{(x_i,y_i)\in w} I_y(x_i,y_i)I_x(x_i,y_i) & \sum_{(x_i,y_i)\in w} I_y(x_i,y_i)^2 \end{bmatrix} \quad (3.9)$$

where $I(x, y)$ is the image and (x_i, y_i) are the points in the window w around (x, y). This matrix captures the structure of the neighborhood. If this matrix is of rank two, that is both of its eigenvalues are large, an interest point is detected. A matrix of rank one indicates that the point belongs to an edge. A matrix of rank zero indicates that the point is part of a homogeneous region. The standard Harris detector computes the derivatives of A by convolution with the mask $[-2\ -1\ 0\ 1\ 2]$. To avoid selecting a noisy point, A is convolved with a Gaussian blurring kernel ($\sigma = 2$).

$$M(\mathbf{x}, \sigma) = G(\sigma, \mathbf{x}) * A(\mathbf{x}) \tag{3.10}$$

The strength of an interest point is measured by Harris function $C(\mathbf{x}, \sigma) = det(M) - \alpha\ trace(M)^2$, where α is normally set to 0.06. Points whose Harris function values exceed a threshold are deemed to be corner points. The threshold is set to 1% of the maximum observed corner point strength. The Harris function has been modified to work in scale-space by varying σ. It has been shown that the Harris function is scale invariant upon proper normalization. The modified Harris function becomes:

$$C_s(\mathbf{x}, \sigma) = s^4(det(M) - \alpha\ trace(M)^2) \tag{3.11}$$

where s is the scale factor. The number of interest points decreases by increasing σ Fig. 3.17.

Without loss of generality, while the reference image I_r is represented at one scale, the target image I_t is represented at eight different scales $[\sigma, 2\sigma, \ldots, 8\sigma]$. At each scale s_i, interest points are extracted using Eq. (3.11).

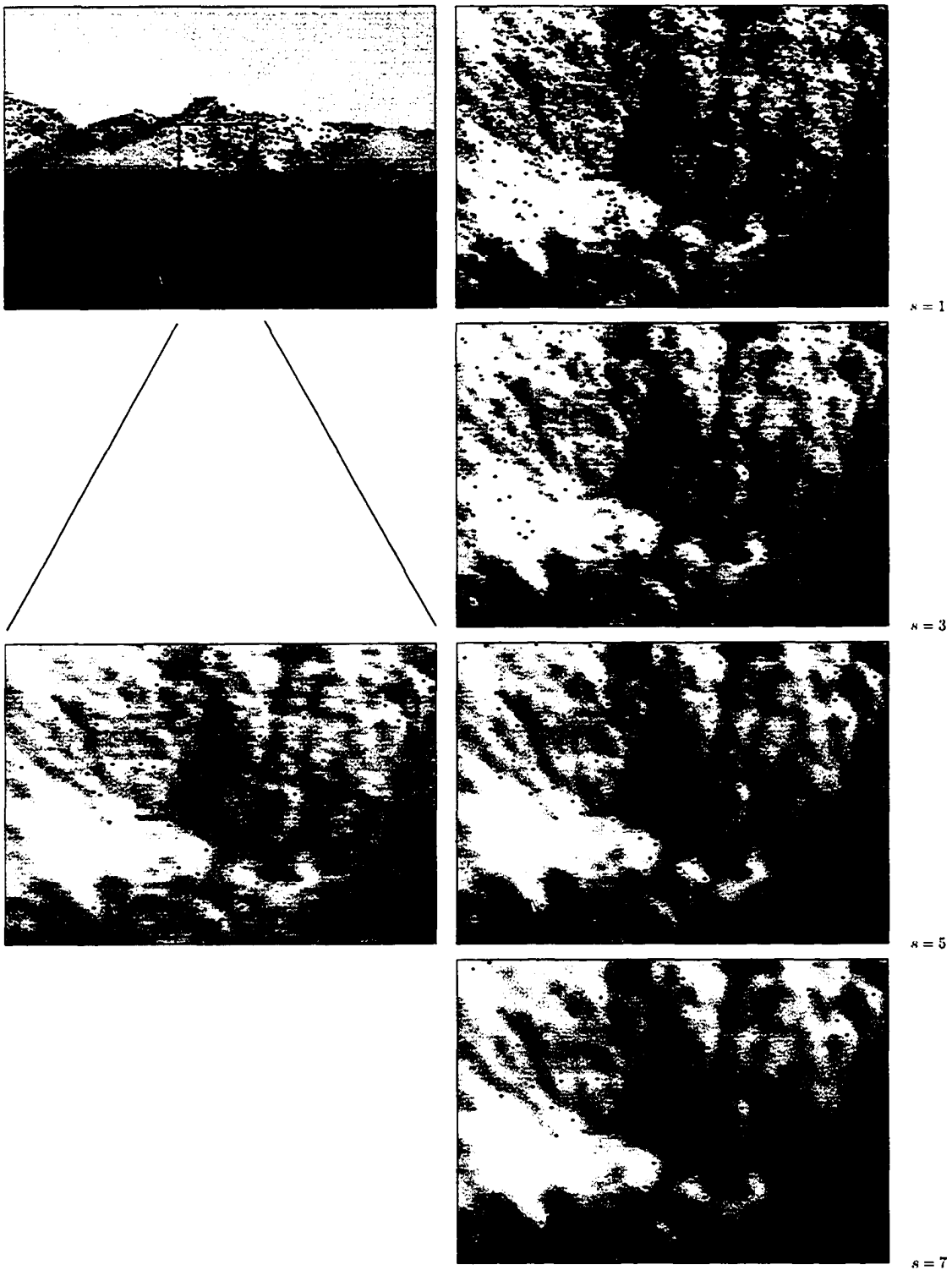


Figure 3.17: Interest points detected at 4 scales (left) and the points detected in the corresponding zoomed image (right).

3.7.2 Feature Vector

All interest points in both images and at all scales are characterized by a description vector whose elements are differential invariants. Differential invariants have been studied theoretically by Koenderink [61] and Romeny et al. [88]. Let us assume $L_{i_1 \cdot i_n}(\mathbf{x}, \sigma)$ are the derivatives of image I and are computed by convolving image I with the derivatives of a Gaussian kernel

$G_{i_1 \cdot i_n}(\mathbf{x}, \sigma)$, where $i_k \in (x, y)$. The following features are computed at each interest point:

$$\nu_i = \begin{bmatrix} L_x^2 + L_y^2 \\ L_{xx}L_x^2 + 2L_{xy}L_xL_y + L_{yy}L_y^2 \\ L_{xx} + L_{yy} \\ L_{xx}^2 + 2L_{xy}^2 + L_{yy}^2 \end{bmatrix} \tag{3.12}$$

The feature vectors are only computed once for the interest points in the reference image and they are computed for each level of scale in the target image.

3.7.3 Similarity Measure

The next step is to find the distance between these features for finding corresponding points. One measure is Euclidean distance. The Euclidian distance between two points $\mathbf{x} = (x_1, \ldots, x_p)$ and $\mathbf{y} = (y_1, \ldots, y_p)$ in the p-dimensional space R^p is defined as:

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \ldots + (x_p - y_p)^2} = \sqrt{(x - y)^t(x - y)} \tag{3.13}$$

$d_E(\mathbf{x}, 0) = \|\mathbf{x}\|_2 = \sqrt{(x_1^2 + \ldots + x_p^2} = \sqrt{\mathbf{x}^t\mathbf{x}}$ is the Euclidian norm of $\mathbf{x}$. Euclidean distance has two disadvantages. First, all points with the same distance from the origin $\|\mathbf{x}\|_2 = c$ satisfy $x_1^2 + \ldots + x_p^2 = c^2$ which is the equation of a spheroid. This means that all components of an observation $\mathbf{x}$ contribute equally to the Euclidian distance of $\mathbf{x}$ from the center. Furthermore, two similar points are no longer considered similar if one of them has undergone an affine transformation. We prefer a distance that takes the variance of that variable into account when determining its distance from the center. Components with high variance should receive less weight than components with low variance. This can be obtained by rescaling the components:

$$\mathbf{u} = (\frac{x_1}{s_1}, \ldots, \frac{x_p}{s_p}) \text{ and } \mathbf{v} = (\frac{y_1}{s_1}, \ldots, \frac{y_p}{s_p}) \tag{3.14}$$

Then, we define the Mahalanobis distance between $\mathbf{x}$ and $\mathbf{y}$ as

$$d_M(\mathbf{x}, \mathbf{y}) = d_E(\mathbf{u}, \mathbf{v}) = \sqrt{(\frac{x_1 - y_1}{s_1})^2 + \ldots + (\frac{x_p - y_p}{s_p})^2} = \sqrt{(\mathbf{x} - \mathbf{y})^t S^{-1} (\mathbf{x} - \mathbf{y})} \tag{3.15}$$

where $S = diag(s_1^2, \ldots, s_p^2)$. All points with the same distance from the origin $\|\mathbf{x}\|_2 = c$ satisfy $(\frac{x_1}{s_1})^2 + \ldots + (\frac{x_p}{s_p})^2 = c^2$, which is the equation of an ellipsoid centered at the origin with principal axes equal to the coordinate axes. Mahalanobis distance is a standard method for modeling the uncertainties in the components of random variables that are themselves modeled with Gaussian distribution. This distance takes into account the different magnitudes as well as the covariance matrix S to select potentially good matches.

The approximate scale is found at level s_i where the percentage of total matches in relation to the number of points extracted is above a threshold. This percentage varies

depending on the image as well as on the type of transformation between the images; the average percentage of matched points is about 50% Fig. 3.18.

Once an approximate scale has been selected using the strategy just described, a robust estimator takes as input the potential one-to-one point assignments, computes the best perspective transformation between the two images, and splits the point assignments into two sets: (1) inliers, i.e. points lying in the small region corresponding to the perspective mapping of the high resolution image onto the low resolution one and (2) outliers, i.e. points that are either outside this region or mismatched points inside the region. The RANSAC [46] algorithm was used for their robust estimation. It allows the user to define in advance the number of potential outliers through the selection of a threshold. Hence, this threshold can be chosen as a function of the scale factor (Fig. 3.19).

3.7.4 Discussions

This feature-based technique documented in [38] was applied to outdoor images with large scale factors (i.e. $s > 4$). Our registration algorithm was able to properly register all of their test images. Their method consists of a series of complex stages that is not prone to direct hardware implementation. These stages include corner detection, conversion to invariant descriptors, matching based on the Mahalanobis distance, and outlier removal using the RANSAC algorithm. Whereas the method was designed to operate under highly textured regions, the approach fails in smooth environments.

The authors in [38] did not supply performance data. In our case, a pair of 640×480 images are registered in approximately 15 seconds on a Pentium IV 3.06 GHz machine. The computational complexity of our algorithm is largely dependent on the resolution of the coarsest level of the pyramid. The user typically sets this to be 64×64 or 32×32.

Thereafter, the search is conducted in a very small neighborhood (e.g., tapering linearly from 8×8 to 2×2).

The foundation of our algorithm is simple. Since we do not search for any specific feature, we register images from any domain, including faces and fingerprints. The proposed algorithm can be implemented simply in hardware [44, 117, 8, 19].

3.8. DEFORMATION PROPERTIES IN THE LOG-POLAR DOMAIN

In this section, we show how various deformation models map into the log-polar domain. We consider the transformation of a circle in the Cartesian space under rigid, affine, and perspective transformations. The results are all conic sections. We study the properties of these conic sections in the log-polar domain.

3.8.1 Similar Transformation

Consider the following similar transformation comprised of rotation, isotropic scale, and translation:

$$u = (s\cos\theta)x + (s\sin\theta)y + t_x \tag{3.16a}$$

$$v = (-s\sin\theta)x + (s\cos\theta)y + t_y \tag{3.16b}$$

where s, θ, and (t_x, t_y) are the scale factor, rotation angle, and translation, respectively. From classic geometry, we know that a circle maps to another circle under a similar transformation.

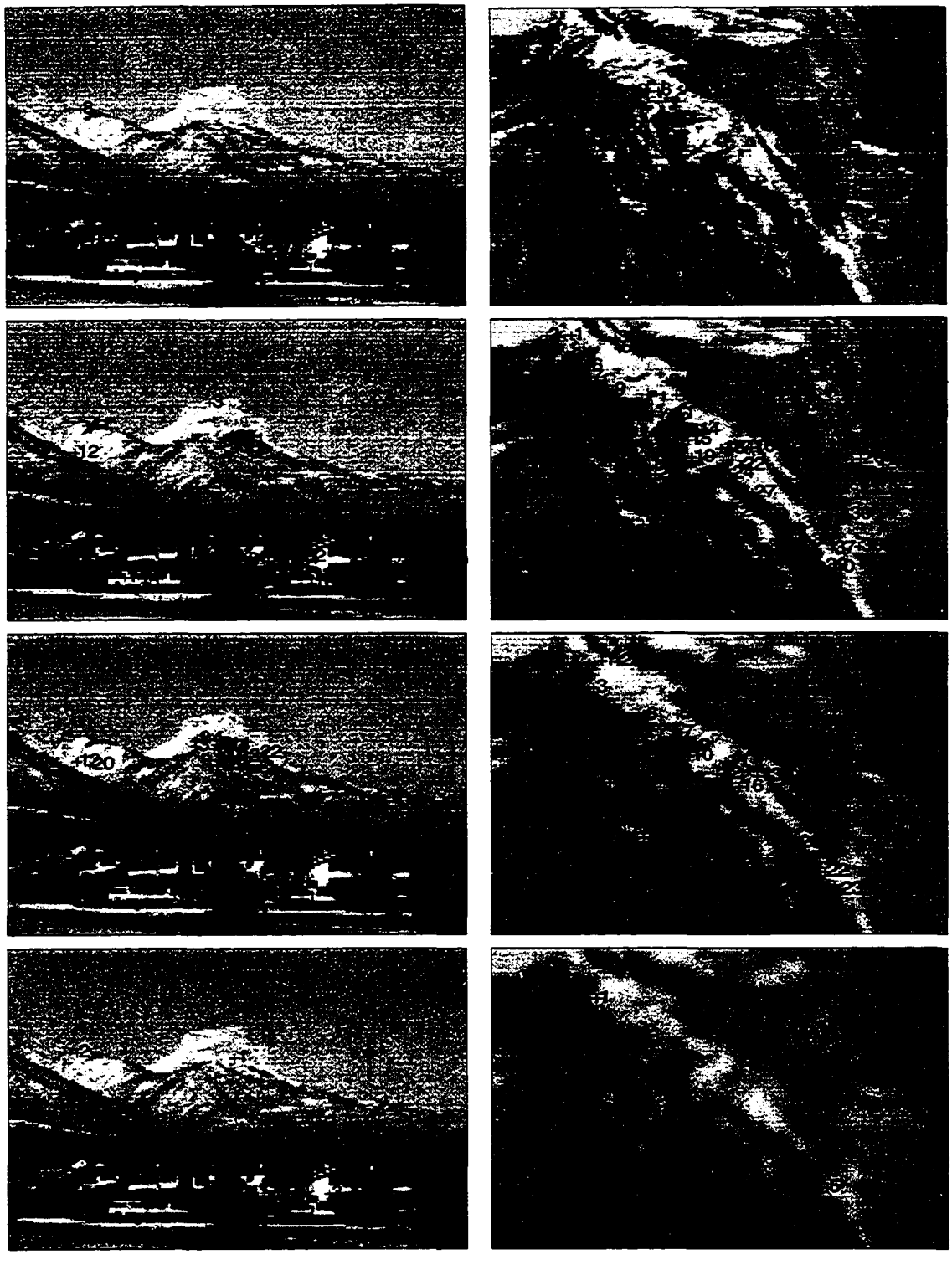


Figure 3.18: Initial point-to-point assignments obtained at four scales (1,3,5,8). The true resolution factor between the two images is 5.

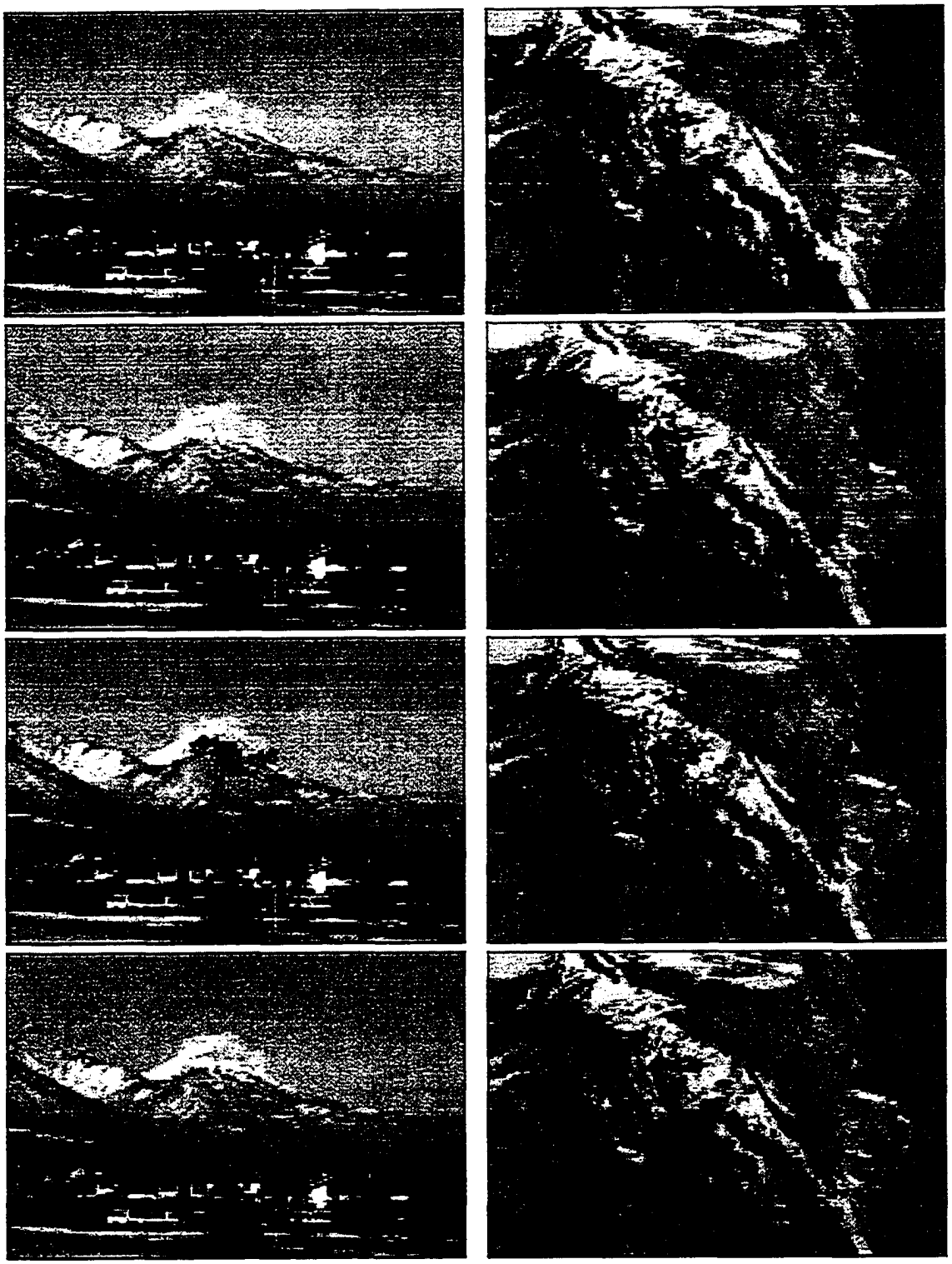


Figure 3.19: Inliers after applying the local constraints and the robust estimator to the previous results.

Proof. Consider a circle with radius R in the (u, v) coordinate system. Substituting Eq. (3.16) into the equation of a circle, we get:

$$\begin{aligned} u^2 + v^2 &= R^2 \\ [(s\cos\theta)x + (s\sin\theta)y + t_x]^2 + [(-s\sin\theta)x + (s\cos\theta)y + t_y]^2 &= R^2 \end{aligned} \quad (3.17)$$

After simplification, we have:

$$\begin{aligned} s^2x^2 + s^2y^2 + 2s(t_x\cos\theta - t_y\sin\theta)x + 2s(t_x\sin\theta + t_y\cos\theta)y + t_x^2 + t_y^2 &= R^2 \\ (x + x_0)^2 + (y + y_0)^2 &= \frac{R^2}{s^2} \end{aligned}$$

where

$$x_0 = \frac{(t_x\cos\theta - t_y\sin\theta)}{s} \quad (3.18a)$$

$$y_0 = \frac{(t_x\sin\theta + t_y\cos\theta)}{s} \quad (3.18b)$$

It is readily apparent that the transformation maps a circle of radius R onto a circle of radius R/s that is shifted by (x_0, y_0). ■

3.8.2 Affine Transformation

Consider an affine transformation $U = AX$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.19)$$

From classic geometry, we know that a circle maps to an ellipse under an affine transformation.

Proof. Consider a circle with radius R in the (u, v) coordinate system. Substituting Eq. (3.19) into the equation of a circle, we get:

$$\begin{aligned} u^2 + v^2 &= R^2 \\ (a_0x + a_1y + a_2)^2 + (a_3x + a_4y + a_5)^2 &= R^2 \end{aligned} \tag{3.20}$$

After simplification, we have:

$$ax^2 + 2bxy + cy^2 + 2dx + 2fy + g = 0 \tag{3.21}$$

where

$$\begin{aligned} a &= a_0^2 + a_3^2 \\ b &= a_0a_1 + a_3a_4 \\ c &= a_1^2 + a_4^2 \\ d &= a_0a_2 + a_3a_5 \\ f &= a_1a_2 + a_4a_5 \\ g &= a_2^2 + a_5^2 - R^2 \end{aligned}$$

Eq. (3.21) is the equation of a conic section. By formulating the equation in matrix form, we shall prove that it represents one particular class of conic sections: an ellipse. We begin

by writing the equation of a circle as:

$$\mathbf{U^T}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -R^2 \end{bmatrix}\mathbf{U} = \mathbf{0} \tag{3.22}$$

where $\mathbf{U} = [u\ v\ 1]^T$. Substituting Eq. (3.19) into Eq. (3.22) yields

$$\mathbf{X^T A^T}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & R^2 \end{bmatrix}\mathbf{AX} = \mathbf{0}$$

$$\mathbf{X^T}\begin{bmatrix} a_0 & a_3 & 0 \\ a_1 & a_4 & 0 \\ a_2 & a_5 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & R^2 \end{bmatrix}\begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{X} = \mathbf{0} \tag{3.23}$$

$$\mathbf{X^T}\begin{bmatrix} a_0^2 + a_3^2 & a_0a_1 + a_3a_4 & a_0a_2 + a_3a_5 \\ a_0a_1 + a_3a_4 & a_1^2 + a_4^2 & a_1a_2 + a_4a_5 \\ a_0a_2 + a_3a_5 & a_1a_2 + a_4a_5 & a_2^2 + a_5^2 - R^2 \end{bmatrix}\mathbf{X} = \mathbf{0}$$

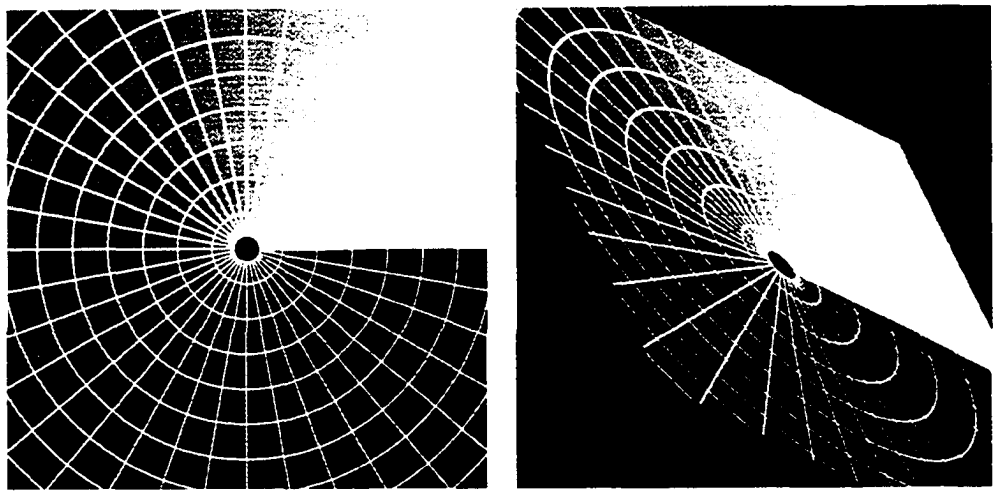
Let C denote the matrix, given above. Note that the elements of C agree with those in Eq. (3.22). C must satisfy three conditions to produce an ellipse [17]:

1. $|C| \neq 0$.

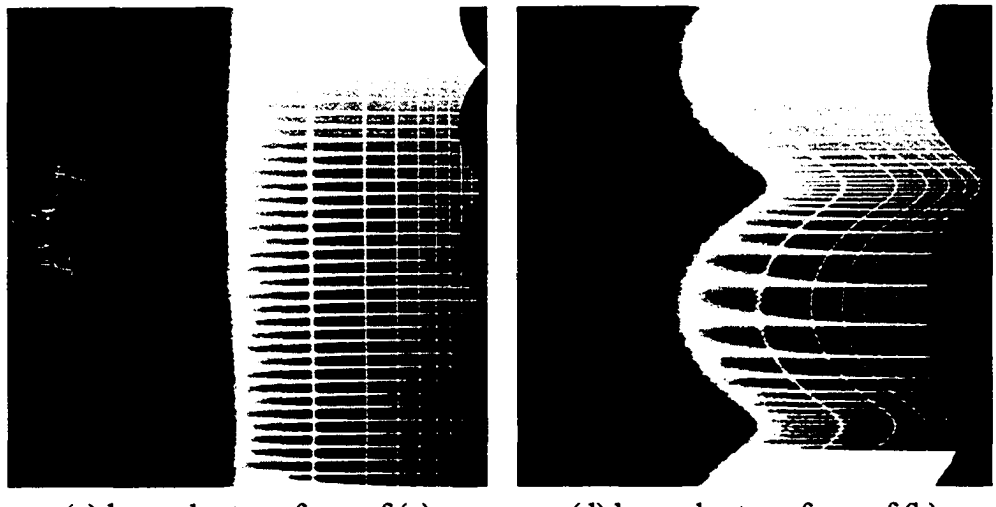
2. $|C|/(a + c) < 0$, where a and c are given in Eq. (3.22).

3. the determinant of the upper 2×2 submatrix of C must be positive.

These conditions can be shown to be always true, therefore a circle maps into an ellipse under an affine transformation. ■



(a) concentric circles (b) concentric ellipses



(c) log-polar transform of (a) (d) log-polar transform of (b)

Figure 3.20: Effect of affine transformation in the log-polar domain.

Fig. 3.20 depicts the effect of an affine transformation on concentric circles. The figure shows that concentric circles map to concentric ellipses under an affine transformation. A proof is given below.

Lemma:Concentric circles map to concentric ellipses by affine transform.

Proof. Let us pick two circles with radius R_1 and R_2. From Section 3.8.2 , we know

these circles map to ellipses with following conic equations:

$$ax^2 + 2bxy + cy^2 + dx + fy + g_1 = 0 \quad (3.24)$$

$$ax^2 + 2bxy + cy^2 + dx + fy + g_2 = 0 \quad (3.25)$$

In the case of affine, g and g' are the only coefficients that are not equal.

$$g_1 = a_2^2 + a_5^2 - R_1^2 \quad (3.26)$$

$$g_2 = a_2^2 + a_5^2 - R_2^2 \quad (3.27)$$

The goal is to reduce conic section Eq. (3.24) and Eq. (3.25) to

$$\frac{(x - x_0)^2}{k_1^2} + \frac{(y - y_0)^2}{k_2^2} = 1$$

we can get rid of xy term by the following method. consider the following 2x2 matrix:

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \quad (3.28)$$

$$|A| = ac - b^2 > 0 \quad (3.29)$$

det $A > 0$ is one the conditions needs to be satisfied for having an ellipse. Let us find the eigenvalue and eigenvector of A. This means solve for:

$$AX = \lambda X \quad (3.30)$$

$$|A - I\lambda| = 0 \quad (3.31)$$

$$\begin{vmatrix} a-\lambda & b \\ b & c-\lambda \end{vmatrix} = 0 \tag{3.32}$$

$$\lambda^2 - (a+c)\lambda + ac - b^2 = 0 \tag{3.33}$$

The above quadradic equation has two real solution λ_1 and λ_2. Now, we find the direction of unit eigenvector. Thus, this vector for λ_1 is:

$$\begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} \tag{3.34}$$

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \cos\theta_1 \\ \sin\theta_1 \end{bmatrix} = \begin{bmatrix} \lambda_1 \cos\theta_1 \\ \lambda_1 \sin\theta_1 \end{bmatrix} \tag{3.35}$$

$$a\cos\theta_1 + b\sin\theta_1 = \lambda_1 \cos\theta_1 \tag{3.36}$$

$$b\cos\theta_1 + c\sin\theta_1 = \lambda_1 \sin\theta_1 \tag{3.37}$$

let us rotate each ellipse around the center by θ_1.

$$x' = x\cos\theta_1 + y\sin\theta_1 \tag{3.38}$$

$$y' = -x\sin\theta_1 + y\cos\theta_1 \tag{3.39}$$

plug above transformation into Eq. (3.24) and Eq. (3.25):

$$\lambda_1 x'^2 + \lambda_2 y'^2 + d'x' + f'y' + g_1 = 0 \tag{3.40}$$

$$\lambda_1 x'^2 + \lambda_2 y'^2 + d'x' + f'y' + g_2 = 0 \tag{3.41}$$

Now, we find the center of each ellipse by getting rid of x' and y' components:

$$\lambda_1(x'^2 + \frac{d'x'}{\lambda_1} + \frac{d'^2}{4\lambda_1^2} - \frac{d'^2}{4\lambda_1^2}) + \lambda_2(y'^2 + \frac{f'y'}{\lambda_2} + \frac{f'^2}{4\lambda_2^2} - \frac{f'^2}{4\lambda_2^2}) + g_1 = 0 \quad (3.42)$$

$$\lambda_1(x'^2 + \frac{d'x'}{\lambda_1} + \frac{d'^2}{4\lambda_1^2} - \frac{d'^2}{4\lambda_1^2}) + \lambda_2(y'^2 + \frac{f'y'}{\lambda_2} + \frac{f'^2}{4\lambda_2^2} - \frac{f'^2}{4\lambda_2^2}) + g_2 = 0 \quad (3.43)$$

$$\lambda_1(x' + \frac{d'}{2\lambda_1})^2 + \lambda_2(y' + \frac{f'}{2\lambda_2})^2 - \frac{d'^2}{4\lambda_1^2} - \frac{f'^2}{4\lambda_2^2} + g_1 = 0 \quad (3.44)$$

$$\lambda_1(x' + \frac{d'}{2\lambda_1})^2 + \lambda_2(y' + \frac{f'}{2\lambda_2})^2 - \frac{d'^2}{4\lambda_1^2} - \frac{f'^2}{4\lambda_2^2} + g_2 = 0 \quad (3.45)$$

(3.46)

As shown in the above equations, the center of both ellipses is $(\frac{d'}{2\lambda_1}, \frac{f'}{2\lambda_2})$ ■

We now consider the implication of these results in the log-polar domain. Let the origin of the log-polar coordinate system be coincident with the centers of the concentric circles. The log-polar transformation maps the set of concentric circles onto a family V of parallel vertical lines. Similarly, the set of concentric ellipses maps to a family C of parallel curves. We seek to find the correspondence of the vertical lines in V to the curves in C. We achieve this correspondence by searching along radial lines in the Cartesian space, which is equivalent to a search along horizontal scanlines in the log-polar domain. The benefit of this approach is that we can exploit the scale-rotation invariance of the log-polar mapping.

We now show that a radial line at angle θ transforms into another radial line at angle θ'. Also, the scale factor along each radial line is uniform. If we set the origin of the log-polar domain at the center of the circle (or ellipse), we have the following affine mapping:

$$x = a_0u + a_1v \quad (3.47a)$$

$$y = a_3u + a_4v \quad (3.47b)$$

Let u and v be given by a parametric representation:

$$u = R\cos\theta \tag{3.48a}$$

$$v = R\sin\theta \tag{3.48b}$$

This yields

$$x = a_0 R\cos\theta + a_1 R\sin\theta \tag{3.49a}$$

$$y = a_3 R\cos\theta + a_4 R\sin\theta \tag{3.49b}$$

We can now express θ' in terms of θ. This accounts for the deformation (wobble) noticeable in Fig. 3.20(d).

$$\tan\theta' = \frac{a_3\cos\theta + a_4\sin\theta}{a_0\cos\theta + a_1\sin\theta} = \frac{a_3 + a_4\tan\theta}{a_0 + a_1\tan\theta} \tag{3.50}$$

Notice that each radial line has a different scale factor. A circle with radius R maps to $r(\theta) = \log R$ in the log-polar domain. The corresponding ellipse maps to

$$r(\theta) = \log(R\sqrt{(a_0\cos\theta + a_1\sin\theta)^2 + (a_3\cos\theta + a_4\sin\theta)^2}) \tag{3.51}$$

Scale in the Cartesian space amounts to a shift in the log-polar domain. Therefore, the amount of shift is a function of θ:

$$S_A(\theta) = \log\sqrt{(a_0\cos\theta + a_1\sin\theta)^2 + (a_3\cos\theta + a_4\sin\theta)^2} \tag{3.52}$$

3.8.3 Perspective Transformation

Consider a perspective transformation $U = PX$:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.53}$$

From classic geometry, we know that a circle maps to an ellipse under a perspective transformation. However, concentric circles do not map onto concentric ellipses.

Proof. Consider a circle with radius R in the (u, v) coordinate system. Substituting Eq. (3.53) into the equation of a circle, we get:

$$\begin{aligned} u^2 + v^2 &= R^2 \\ (a_1x + a_2y + a_3)^2 + (a_4x + a_5y + a_6)^2 &= R^2(a_6x + a_7y + 1)^2 \end{aligned} \tag{3.54}$$

After simplifying, we have:

$$ax^2 + 2bxy + cy^2 + 2dx + 2fy + g = 0 \tag{3.55}$$

where

$$\begin{aligned} a &= a_0^2 + a_3^2 - R^2a_6^2 \\ b &= a_0a_1 + a_3a_4 - R^2a_6a_7 \\ c &= a_1^2 + a_4^2 - R^2a_7^2 \\ d &= a_0a_2 + a_3a_5 - R^2a_6 \\ f &= a_1a_2 + a_4a_5 - R^2a_7 \end{aligned}$$

$$g = a_2^2 + a_5^2 - R^2$$

Eq. (3.55) is the equation of a conic section. Using the same approach we used in the affine case, we show that the equation yields ellipses by formulating Eq. (3.55) in matrix form. Consider the equation of a circle in matrix form:

$$\mathbf{U^T} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -R^2 \end{bmatrix} \mathbf{U} = \mathbf{0} \tag{3.56}$$

where $\mathbf{U} = [u\ v\ 1]^T$. Substituting Eq. (3.53) into Eq. (3.56) yields

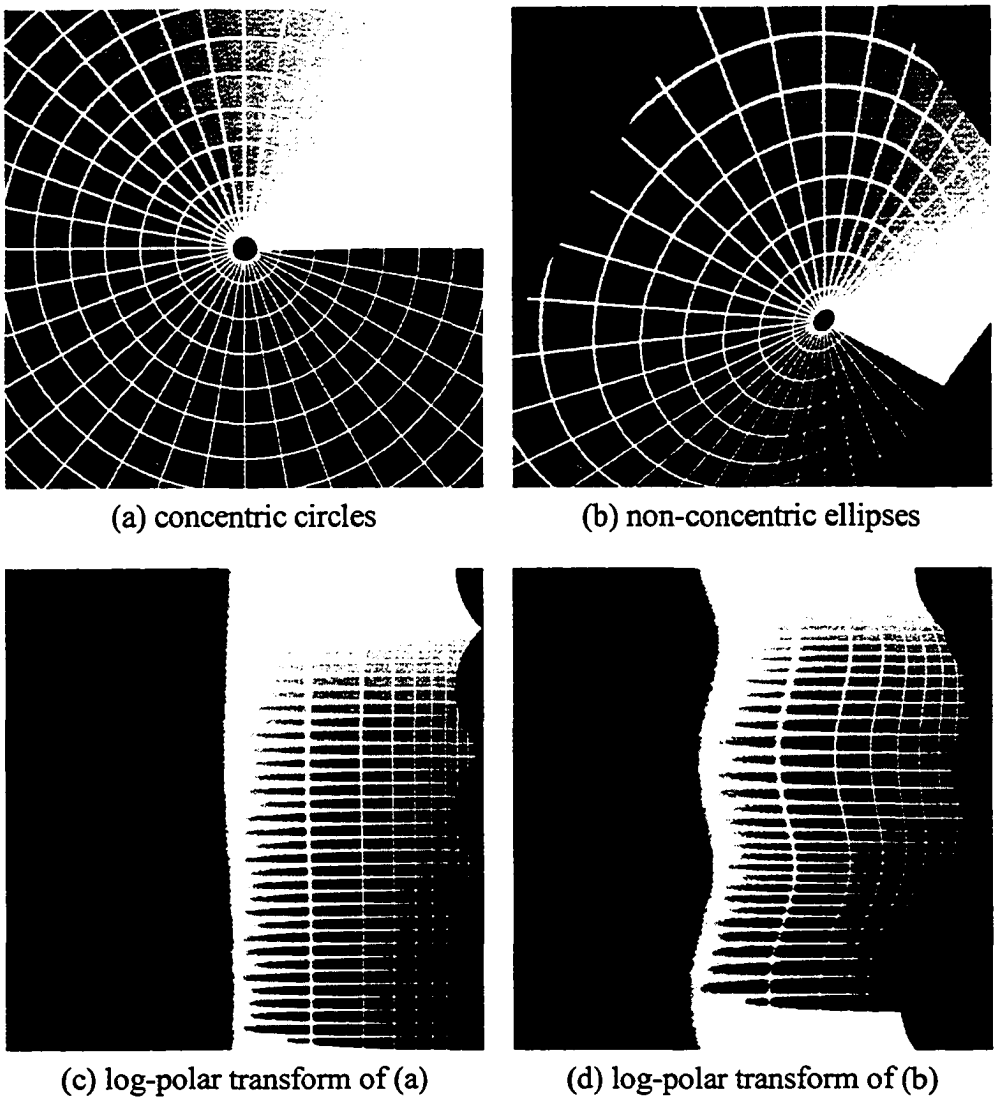
$$\mathbf{X^T P^T} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & R^2 \end{bmatrix} \mathbf{PX} = \mathbf{0}$$

$$\mathbf{X^T} \begin{bmatrix} a_0 & a_3 & a_6 \\ a_1 & a_4 & a_7 \\ a_2 & a_5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & R^2 \end{bmatrix} \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix} \mathbf{X} = \mathbf{0} \tag{3.57}$$

$$\mathbf{X^T} \begin{bmatrix} a_0^2 + a_3^2 - R^2 a_6^2 & a_0 a_1 + a_3 a_4 - R^2 a_6 a_7 & a_0 a_2 + a_3 a_5 - R^2 a_6 \\ a_0 a_1 + a_3 a_4 - R^2 a_6 a_7 & a_1^2 + a_4^2 - R^2 a_7^2 & a_1 a_2 + a_4 a_5 - R^2 a_7 \\ a_0 a_2 + a_3 a_5 - R^2 a_6 & a_1 a_2 + a_4 a_5 - R^2 a_7 & a_2^2 + a_5^2 - R^2 \end{bmatrix} \mathbf{X} = 0$$

■

Fig. 3.21 shows the effect of a perspective transformation on concentric circles. The figure shows that concentric circles map to non-concentric ellipses under a perspective transformation. A proof is given below.



(a) concentric circles (b) non-concentric ellipses

(c) log-polar transform of (a) (d) log-polar transform of (b)

Figure 3.21: Effect of perspective transformation in the log-polar domain.

Lemma: Concentric circles does not map to concentric ellipses by perspective transform.

Proof. Let us Pick two concentric circles with radius R_1 and R_2. Recall from Section 3.8.3, these two circles map following correspondence ellipses:

$$a_1x^2 + 2b_1xy + c_1y^2 + d_1x + f_1y + g_1 = 0 \quad (3.58)$$

$$a_2x^2 + 2b_2xy + c_2y^2 + d_2x + f_2y + g_2 = 0 \quad (3.59)$$

In the case of perspective, all of the coefficents of the conic section are the function of R. Because of (a_1, b_1, c_1) and (a_2, b_2, c_2), we will get two different sets of eigenvalue and rotation angle. After doing the same process we have done for affine case, we will get two different centers, which are $(\frac{d_1'}{2\lambda_1'}, \frac{f_1'}{2\lambda_2'})$, and $(\frac{d_2'}{2\lambda_1''}, \frac{f_2'}{2\lambda_2''})$ ■

We now consider the implication of these results in the log-polar domain. Let the origin of the log-polar coordinate system be coincident with the centers of the concentric circles. The log-polar transformation maps the set of concentric circles onto a family V of parallel vertical lines. Similarly, the set of non-concentric ellipses maps to a family C of non-parallel curves. We seek to find the correspondence of the vertical lines in V to the curves in C. We achieve this correspondence by searching along radial lines in the Cartesian space, which is equivalent to a search along horizontal scanlines in the log-polar domain. The benefit of this approach is that we can exploit the scale-rotation invariance of the log-polar mapping.

We can decompose a perspective transformation into two matrices: $\mathbf{P} = \mathbf{AH}$, where $\mathbf{A}$ is an affine transformation and $\mathbf{H}$ is the projection matrix.

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & 1 \end{bmatrix} = \begin{bmatrix} a_0' & a_1' & a_2 \\ a_3' & a_4' & a_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_6 & a_7 & 1 \end{bmatrix} \tag{3.60}$$

where

$$a_0' = a_0 - a_2 a_6$$

$$a_1' = a_1 - a_2 a_7$$

$$a_3' = a_3 - a_5 a_6$$

$$a'_4 = a_4 - a_5 a_7$$

Each radial line in I_1 maps to a different radial line in I_2. Corresponding radial lines do not share a uniform scale factor. This contributes to a chirping effect along the radial lines. We now show that a radial line at angle θ transforms into another radial line at angle θ' with nonuniform scale. If we set the origin of the log-polar domain at the center of the circle (or ellipse), we have the following perspective mapping:

$$x = \frac{(a_0 u + a_1 v + a_2)}{a_6 u + a_7 v + 1} - a_2 = \frac{(a_0 - a_2 a_6)u + (a_1 - a_2 a_7)v}{a_6 u + a_7 v + 1} \tag{3.61a}$$

$$y = \frac{(a_3 u + a_4 v + a_5)}{a_6 u + a_7 v + 1} - a_5 = \frac{(a_3 - a_5 a_6)u + (a_4 - a_5 a_7)u}{a_6 u + a_7 v + 1} \tag{3.61b}$$

Let u and v be given by a parametric representation:

$$u = R\cos\theta \tag{3.62}$$

$$v = R\sin\theta \tag{3.63}$$

This yields

$$x = \frac{(a_0 - a_2 a_6)R\cos\theta + (a_1 - a_2 a_7)R\sin\theta}{a_6 R\cos\theta + a_7 R\sin\theta + 1} \tag{3.64a}$$

$$y = \frac{(a_3 - a_5 a_6)R\cos\theta + (a_4 - a_5 a_7)R\sin\theta}{a_6 R\cos\theta + a_7 R\sin\theta + 1} \tag{3.64b}$$

We can now express θ' in terms of θ.

$$\tan\theta' = \frac{y}{x} = \frac{a'_3\cos\theta + a'_4\sin\theta}{a'_0\cos\theta + a'_1\sin\theta} = \frac{a'_3 + a'_4\tan\theta}{a'_0 + a'_1\tan\theta} \tag{3.65}$$

Notice that the above function is similar to the affine case in Section 3.8.2. Each radial line

has a different scale factor. A circle with radius R maps to $r(\theta) = \log R$ in the log-polar domain. The corresponding ellipse maps to

$$r(\theta) = \log \frac{R\sqrt{(a'_0 \cos\theta + a'_1 \sin\theta)^2 + (a'_3 \cos\theta + a'_4 \sin\theta)^2}}{a_6 R\cos\theta + a_7 R\sin\theta + 1} \tag{3.66}$$

Scale in the Cartesian space amounts to a shift in the log-polar domain. The amount of shift is a function of θ and R:

$$S_P(\theta, R) = \log\left(\frac{\sqrt{(a'_0 \cos\theta + a'_1 \sin\theta)^2 + (a'_3 \cos\theta + a'_4 \sin\theta)^2}}{a_6 R\cos\theta + a_7 R\sin\theta + 1}\right) \tag{3.67}$$

or

$$S_P(\theta, R) = S_A(\theta) - \log([a_6 \cos\theta + a_7 \sin\theta]R + 1) \tag{3.68}$$

In this case, scale is a function of both θ and R. Our first observation from the above equation is that radial lines remain lines. Secondly, if R is small, then $S_P(\theta, R) \approx S_A(\theta)$. This is consistent with the fact that perspective can be approximated to be locally affine in a small neighborhood.

3.9. MATCHING ALGORITHM IN THE LOG-POLAR DOMAIN

The registration algorithm proposed in this section works in three stages. First, we find the global rotation, scale, and translation. We have already reviewed this procedure in Section 3.6. Second, we find corresponding radial lines in both images by searching around the log-polar origin within a small radius. The purpose of using a small radius is to achieve reliable correlation in the log-polar domain with minimal skewing effect. Finally,

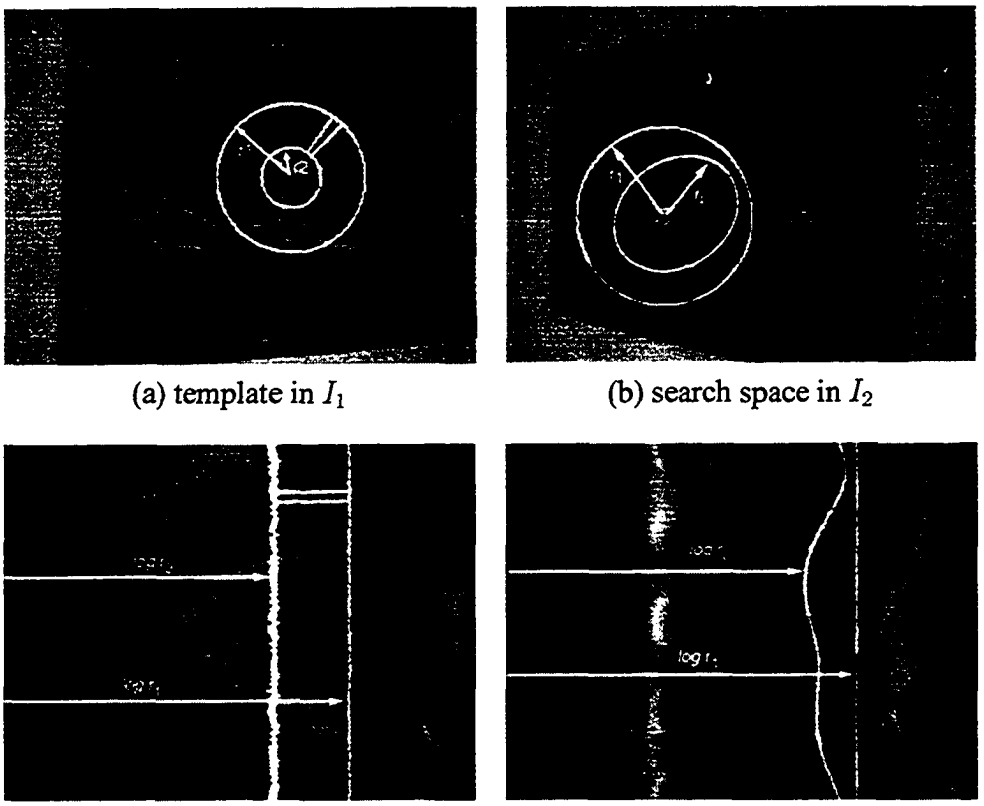
we extend the search along the corresponding radial lines to account for the additional skewing present due to possibly large perspective deformations. It is important to note that identifying the corresponding radial lines in our second step permits us to deal with the harder perspective problem because we can narrow our search to reduce false matches. In this section, we discuss the algorithm for matching corresponding radial lines in the log-polar domain.

3.9.1 Radial Line Correspondence

In Sections 3.8.2 and 3.8.3 we showed that radial lines map to horizontal lines in the $r\theta$-plane after the application of affine and perspective transformations. We also found the relationship between corresponding radial lines. The important observation is that the mapping of θ to θ' is the same for affine and perspective transformations.

Since a perspective transformation can be approximated by a local affine transformation, we seek to establish radial line correspondence for the simpler affine case. That same correspondence will hold when we extend the search for perspective. We begin by centering a circle with a small radius r_1 on the origin of I_1 and searching for corresponding radial lines in I_2. For each radial line in I_1 at angle θ, we search for its corresponding radial line at angle θ' in I_2. The match process makes use of a template, consisting of a radial line segment at angle θ. The search takes place in the log-polar domain, where the template maps into a row segment. The cross-correlation similarity measure is used to find $(\Delta r_i, \Delta\theta_i)$. Note that $\theta_i' = \theta_i + \Delta\theta_i$. We repeat this process for each radial line from 0° to 360°. Fig. 3.22 depicts this process.

The algorithm outlined above will produce a set of correspondence points in the log-polar domain. As a result, point $(\log r_1, \theta_i)$ in I_{p1} maps to point $(\log r_i, \theta_i')$ in I_{p2}. By



(a) template in I_1 (b) search space in I_2

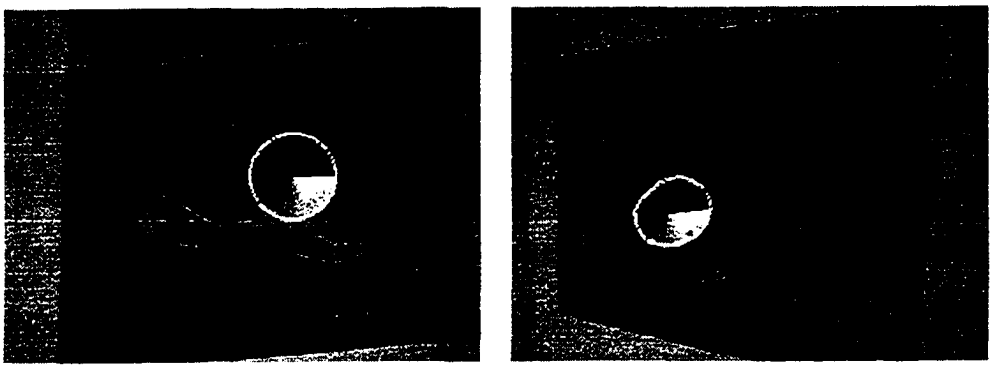
(c) log-polar transform of (a) (d) log-polar transform of (b)

Figure 3.22: Search space for radial line correspondence.

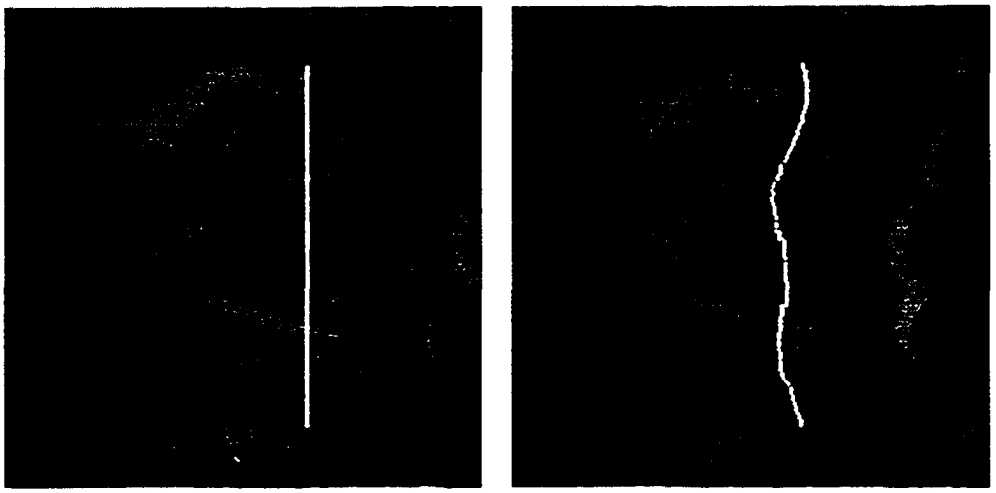
performing an inverse mapping from the log-polar domain to the Cartesian coordinate system, we establish correspondences in the xy-plane between a circle and an ellipse.

These correspondences will comprise the input to a linear system of equations for recovering the best local affine parameters. The solution is overdetermined and so we solve it with the pseudoinverse method. One problem with this approach is outliers. Points in the first image lie in a circle with radius r_1 and we know that the circle maps to an ellipse. We therefore fit an ellipse to the second set of points. Any point having a large Euclidian distance from the ellipse is considered an outlier. Fig. 3.23 shows the result of the above algorithm.

When we establish corresponding radial lines, we can limit our search space from a



(a) radial lines in I_1 (b) corresponding radial lines in I_2



(c) log-polar transform of (a) (d) log-polar transform of (b)

Figure 3.23: Search space for radial line correspondence.

2D to a 1D search and we have to only search along the radial lines for correspondence points. This step has the same analogy as stereo matching for finding dense disparity. First one finds the fundamental matrix and then limits the search from 2D to 1D along epipolar lines.

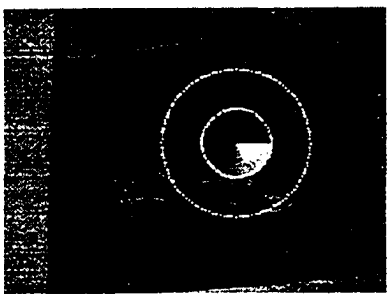
In the next step of the algorithm, we expand the radius of the circle in I_1 and search for small template along correspondence radial line in I_2. The result will be a bigger ellipse that inscribes the smaller ellipse. From correspondence points that lie on the circle and ellipse, we can find the eight perspective parameters. Fig. 3.24 shows the result of above

step.

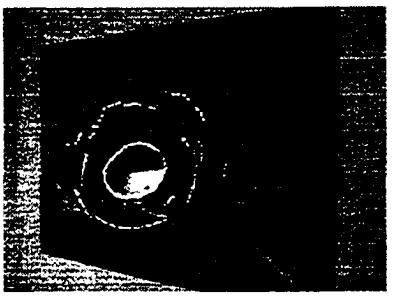
As is shown in Fig. 3.24(e), the final match is not perfect and we have minor misalignment between features. This due to two reasons. First, the log-polar space is a non-uniform sampling space. Second, the search strategy does not offer sub-pixel accuracy. Since the features are very close to each other, we can input the two images to a sub-pixel registration method such as Levenberg-Marquardt to find the final residual transformation for perfect alignment.

Fig. 3.25 presents a flowchart of the described algorithm. As shown in Fig. 3.25, the matching algorithm has three phases. In the first phase, we search for global rotation, scale, and translation. We apply this transformation to image I_1 and bring it closer to image I_2. Fig. 3.26(a) and (b) shows a pair of aerial images. In phase I, we find the rotation, scale, and translation. This result was shown in Fig. 3.26 (c) and (d). Then, the second phase looks for corresponding radial lines. As shown in Fig. 3.26(e), the inner circle has mapped to its corresponding ellipse in Fig. 3.26(f). Due to the height of the trees, we have minor motion parallax. The resulting disparities deform the outer ellipse estimated in phase II.

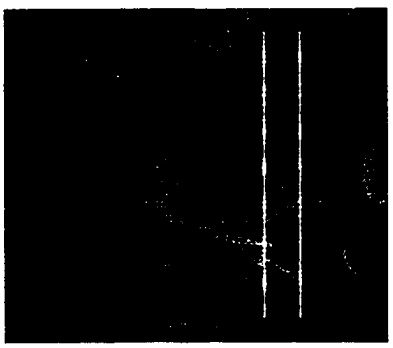
The proposed method may fail under a severe perspective transformation. Fig. 3.28 depicts such an example. The principal problem lies in the nonlinearity introduced in the log-polar domain by the perspective transformation. Such problems can be largely circumvented by limiting the search space of the scale parameter. Fig. 3.28(c) shows an example of registration after recovering rotation and translation alone. Notice that the features have been brought into close alignment. This is then further refined using nonlinear least squares optimization to yield the result shown in Fig. 3.29. It is important to note that the optimization method could not be directly applied to the original inputs because it requires a good initial alignment in order to successfully register the input images to subpixel accuracy.



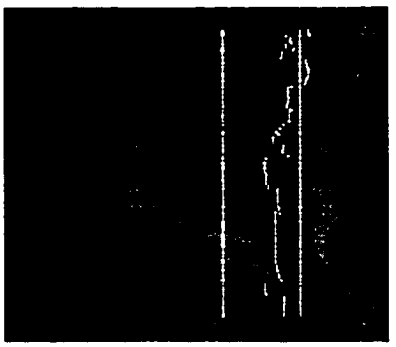
(a) radial lines in I_1



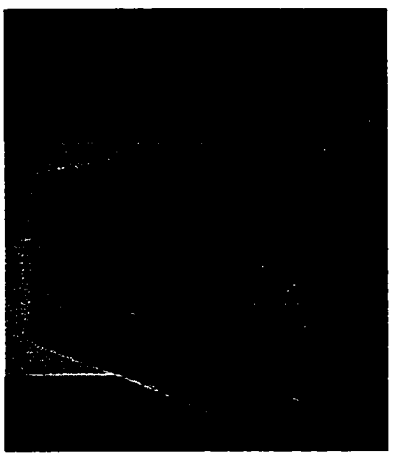
(b) corresponding radial lines in I_2



(c) log-polar transform of (a)



(d) log-polar transform of (b)



(e) final registration result

Figure 3.24: Perspective registration in the log-polar domain.

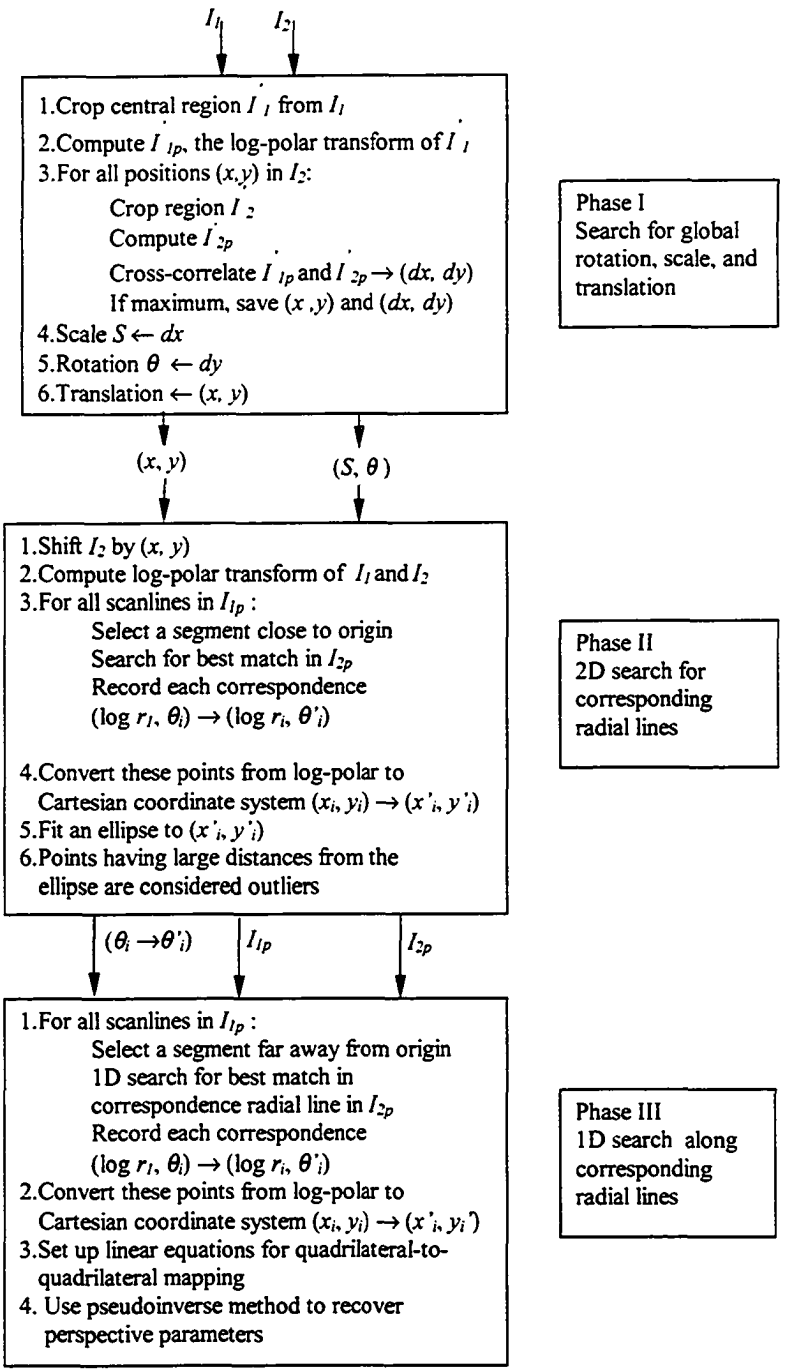
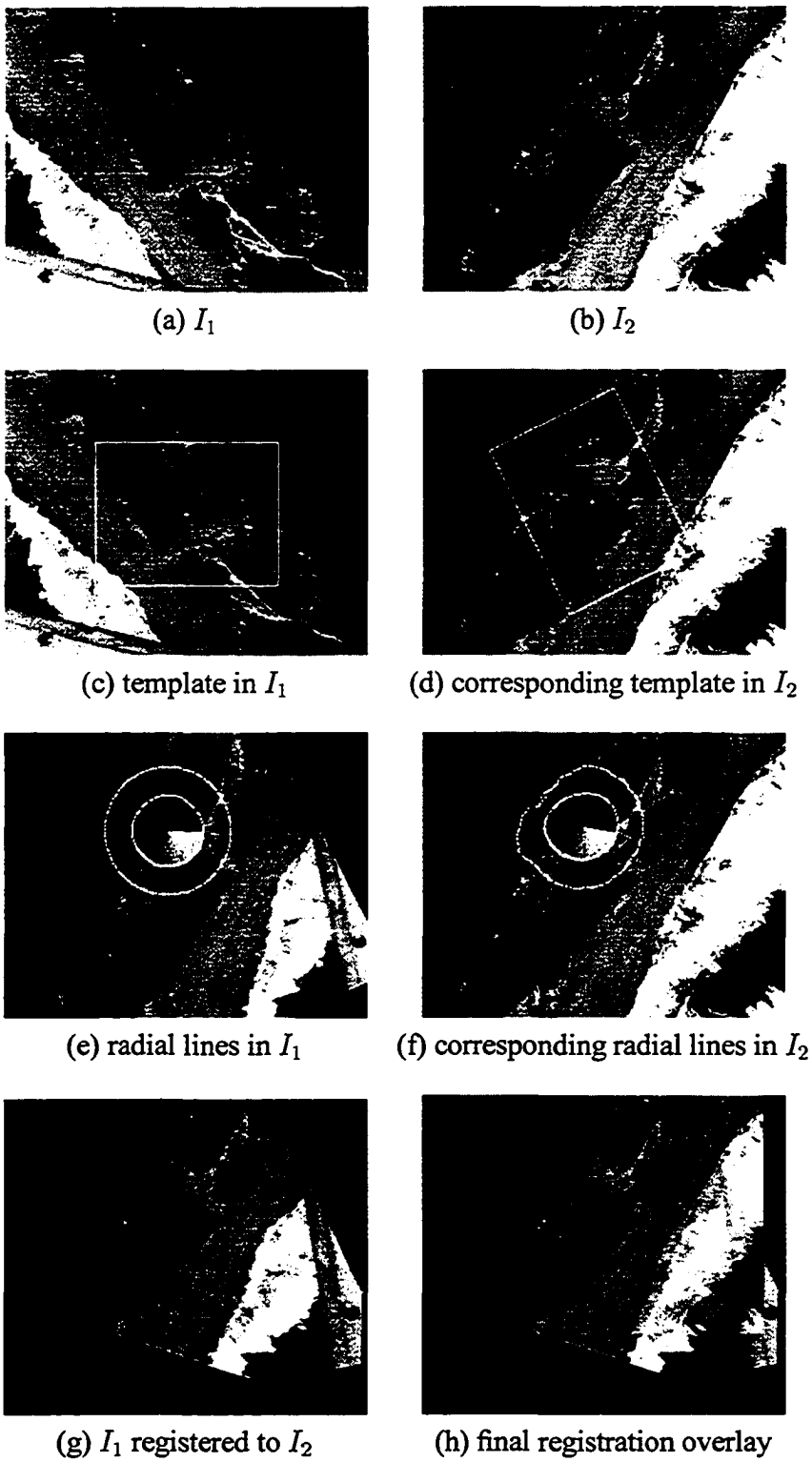


Figure 3.25: Flow chart of the registration method.



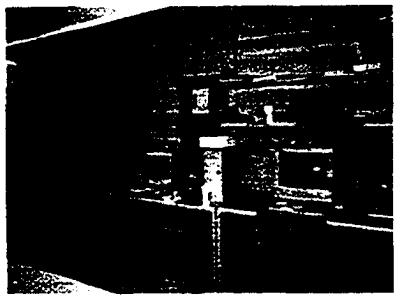
(a) I_1 (b) I_2

(c) template in I_1 (d) corresponding template in I_2

(e) radial lines in I_1 (f) corresponding radial lines in I_2

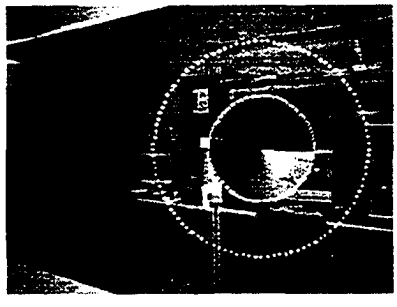
(g) I_1 registered to I_2 (h) final registration overlay

Figure 3.26: Perspective registration.

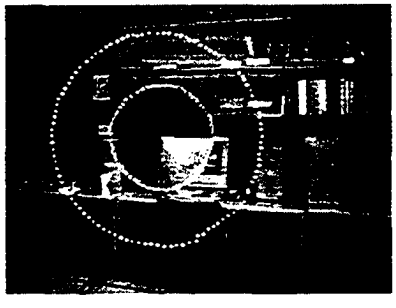


(a) I_1

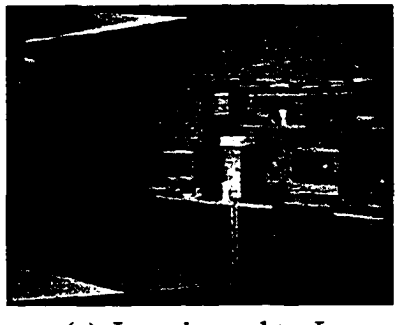
(b) I_2



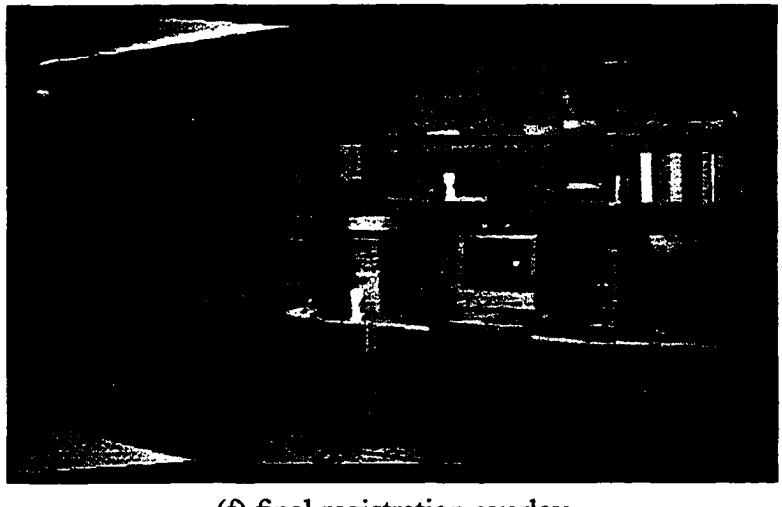
(c) radial lines in I_1



(d) corresponding radial lines in I_2

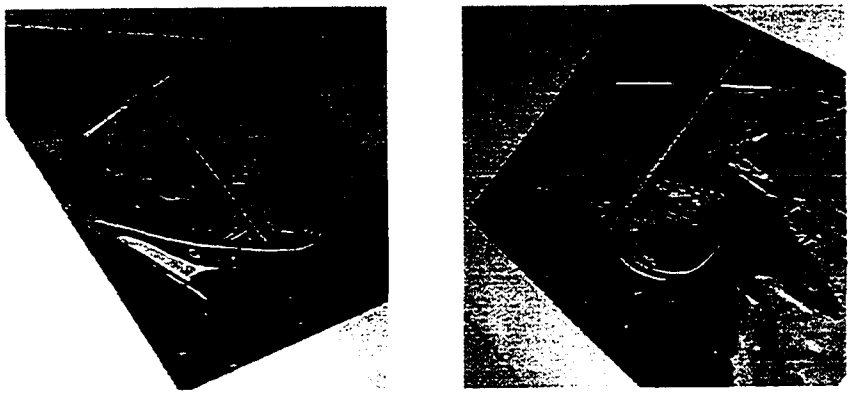


(e) I_1 registered to I_2



(f) final registration overlay

Figure 3.27: Perspective registration.



(a) I_1 (b) I_2

(c) recovered rotation and translation

Figure 3.28: Approximate registration of severe perspective distortion using rotation and translation.

I_1 registered to I_2

Figure 3.29: Refined registration result using nonlinear least squares optimization.

3.10. SUMMARY AND DISCUSSION

In this chapter, we introduced the log-polar mapping, a biologically motivated transformation, for the purpose of registering images subjected to large scale and rotation. We discussed the limitations of the related Fourier-Mellin registration technique. We demonstrated that the Fourier-Mellin transform is not a robust method for registering real images under mild affine and perspective deformations. We proposed a new, superior method that operates in the spatial domain and is robust under challenging large-scale changes in rotation, scale, and translation. Our method was shown to be simpler and more robust than the leading feature-based methods based on scale-space techniques advanced by INRIA.

We further extended the proposed method to invert perspective transformations in the log-polar domain. We derived the nonlinear transformation that maps radial lines in the log-polar domain map to a set of radial lines in the spatial domain, for images subjected to affine and perspective transformations.

Chapter 4

APPLICATIONS

Image registration plays an important role in many applications such as image mosaicing, video mosaicing, image fusion, super-resolution, and video stabilization. In this chapter, we demonstrate the robustness and application of the proposed registration algorithm in various domains. In Section 4.1, we demonstrate the registration of facial images subjected to large changes in scale and rotation. In Section 4.2, we demonstrate the registration of other natural images in which the reference and target images are taken by an uncalibrated camera with optical zoom. In Section 4.3, we test the robustness of our algorithm with 10,000 image pairs whose transformation is known. This is achieved by taking 1,000 images from the Corel Stock Photo Library and applying ten random transformations to each image. The registration algorithm then seeks to recover these transformations. We demonstrate the utility of the log-polar transform in recovering large deformations by comparing registration accuracy with and without the log-polar registration module. A significant increase in correct matches is attributed to our algorithm.

The algorithm proves useful for image enhancement. We apply the registration algorithm to remove undesirable artifacts, such as specular glare. Since the appearance of glare

is view-dependent, registering images acquired from multiple viewpoints enables nonlinear blending to dismiss the bright, saturated highlights. This application is demonstrated in Section 4.4.

The generation of image mosaics from a set of disparate images is described in Section 4.5. Finally, Section 5.1 introduces a novel use of the proposed algorithms to remove unwanted foreground elements from input images of a complex scene. The term *diminished reality* is given to this new application since its goals of object removal stand in contrast to the goals of *augmented reality*, an emerging technique in computer graphics.

4.1. ROBUST IMAGE REGISTRATION ON FACIAL IMAGES

An analytical evaluation of the robustness of image registration algorithms is an elusive task. Performance is highly dependent on the content of the input images. Although image models may exist for particular domains, the deformations and noise functions that may apply to images defy restrictive bounds. Consequently, many proposed image registration algorithms in the literature have limited their published results to the use of a few reference images and their synthetically generated target images. In an effort to broaden our test suite, we chose an empirical approach with a variety of input images from different domains: natural and man-made domains with a wide range of texture diversity.

The first set of test images belong to the face domain. In the following examples, we show that our registration method can register two different faces by aligning their features. The images may differ in their hair styles, eyes, nose, and lip positions. Two sample facial images are shown in Fig. 4.1. An attempt at registering these two images produces a correct match, as shown in the proper head alignment in Fig. 4.1(c).

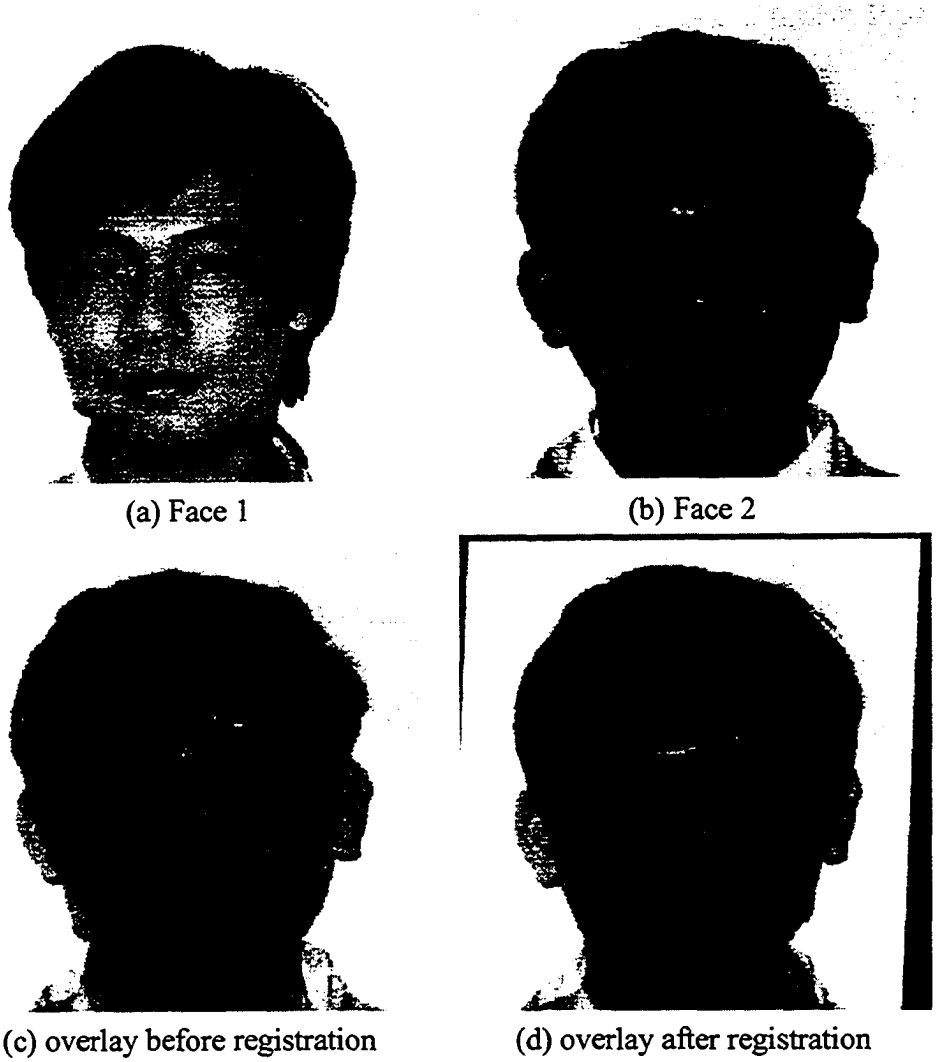
The next example demonstrates how the global affine registration tends to match the

(a) (b) (c)

Figure 4.1: Example of registering two different faces. (a) Face 1; (b) Face 2 rotated by 45°; and (c) Face 2 after alignment.

same facial features. If we overlay the input images in Fig. 4.2(a) and Fig. 4.2(b) we can see double features in Fig. 4.2(c). After affine registration, features like lips, eyes, nose, and hair are moved to nearby correspondence features (Fig. 4.2(d)). Although we do not expect that a global affine transformation is entirely sufficient to align two facial images, the purpose of this exercise is to show the benefits of a global transformation in establishing *initial estimates* for feature correspondence in a subsequent local registration operation.

In the next example, we will see that if we use the gradient image instead of the intensity image, we can improve the alignment of the facial features. We use two different female faces. The hair covers a large dark area in face 3 (Fig. 4.3(a)). After registration using intensity, we can see that affine transformation is trying to match the hair area in order to minimize $\chi^2(a)$, see Fig. 4.3 (d). Clearly, the registration in the lips and chin areas is not acceptable. We can improve this result by using a blurred version of the gradient image. Note that blurring serves to smooth the $\chi^2(a)$, which helps us avoid local minimas. Fig. 4.4 (a) and (b) depict the result of this preprocessing step. The registration based on the gradient tends to align the edges of the features (Fig. 4.4 (d)). The overlaid images show that similar features now lie closer together (Fig. 4.4 (e)).

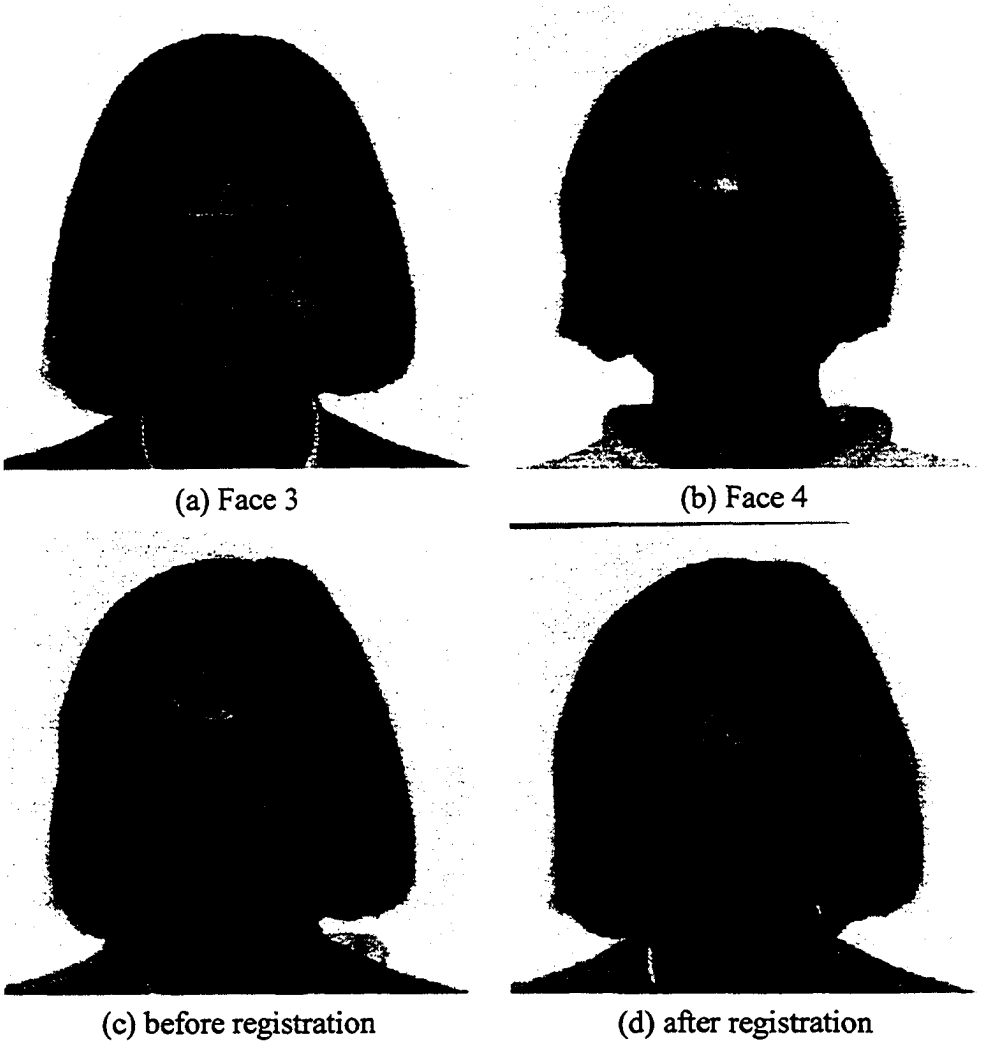


(a) Face 1 (b) Face 2

(c) overlay before registration (d) overlay after registration

Figure 4.2: Face registration using affine transformations.

The next example demonstrates that the use of the Levenberg-Marquardt method alone will fail when there is a large scale factor between the two input images. The log-polar module, however, will correctly register these images. An example is shown in Fig. 4.5. The image in Fig. 4.5(a) is 2.5 times larger than the target image in Fig. 4.5(b). It is important to note that this is a difficult case due to differences in the facial expressions and lighting conditions. Fig. 4.5(c) demonstrates the robustness of the proposed algorithm. Notice that facial features are in approximate alignment, a necessary requirement for any



(a) Face 3 (b) Face 4

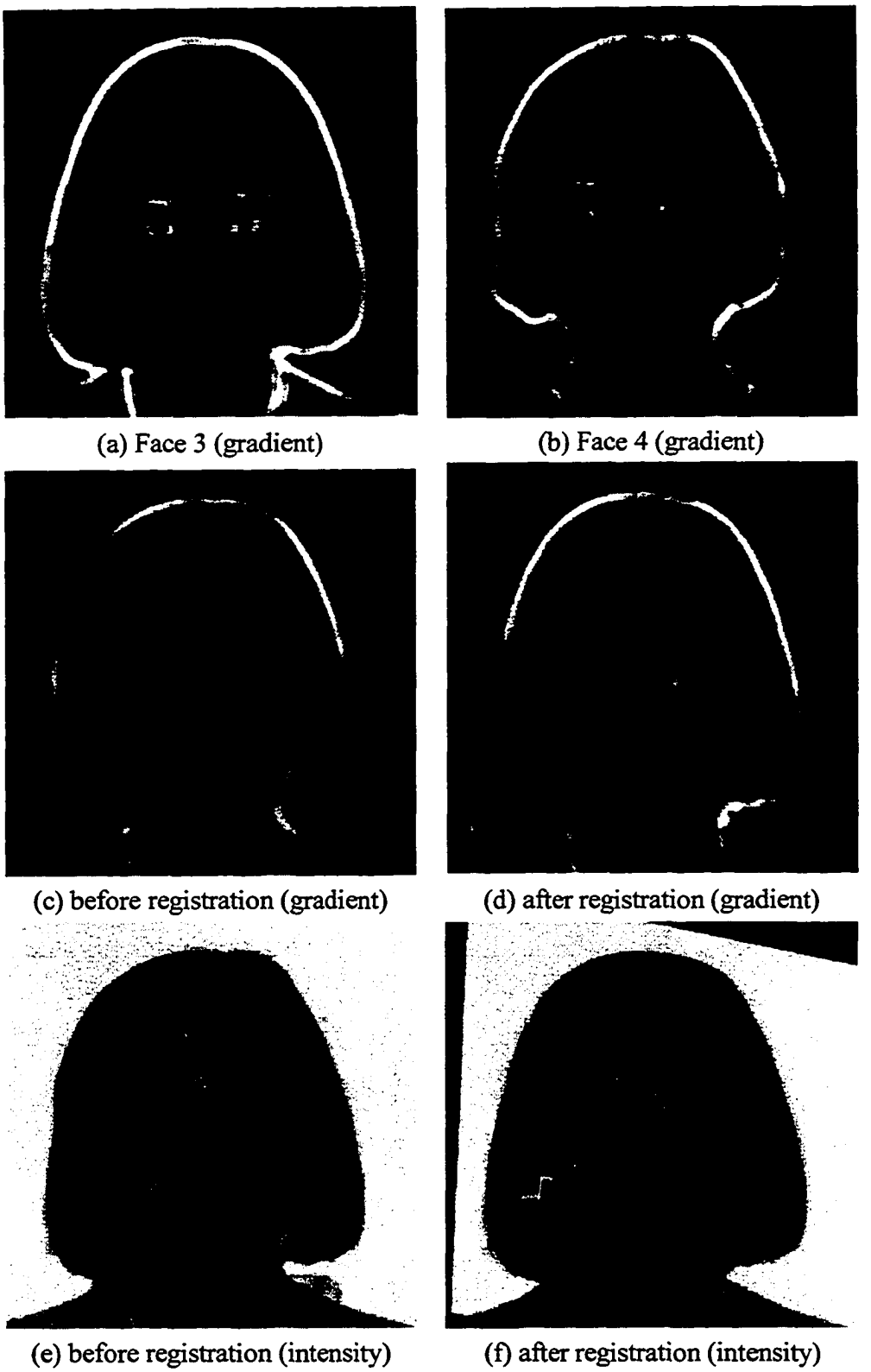
(c) before registration (d) after registration

Figure 4.3: Example of registration using intensity images. Alignment is not satisfactory because emphasis was placed on matching large regions instead of prominent edges.

face recognition system.

4.2. ROBUST IMAGE REGISTRATION ON UNCALIBRATED IMAGES

An uncalibrated Canon PowerShot G3 digital camera with 4x optical zoom was used to capture the next set of test images taken from natural and manmade scenes. The content



(a) Face 3 (gradient) (b) Face 4 (gradient)

(c) before registration (gradient) (d) after registration (gradient)

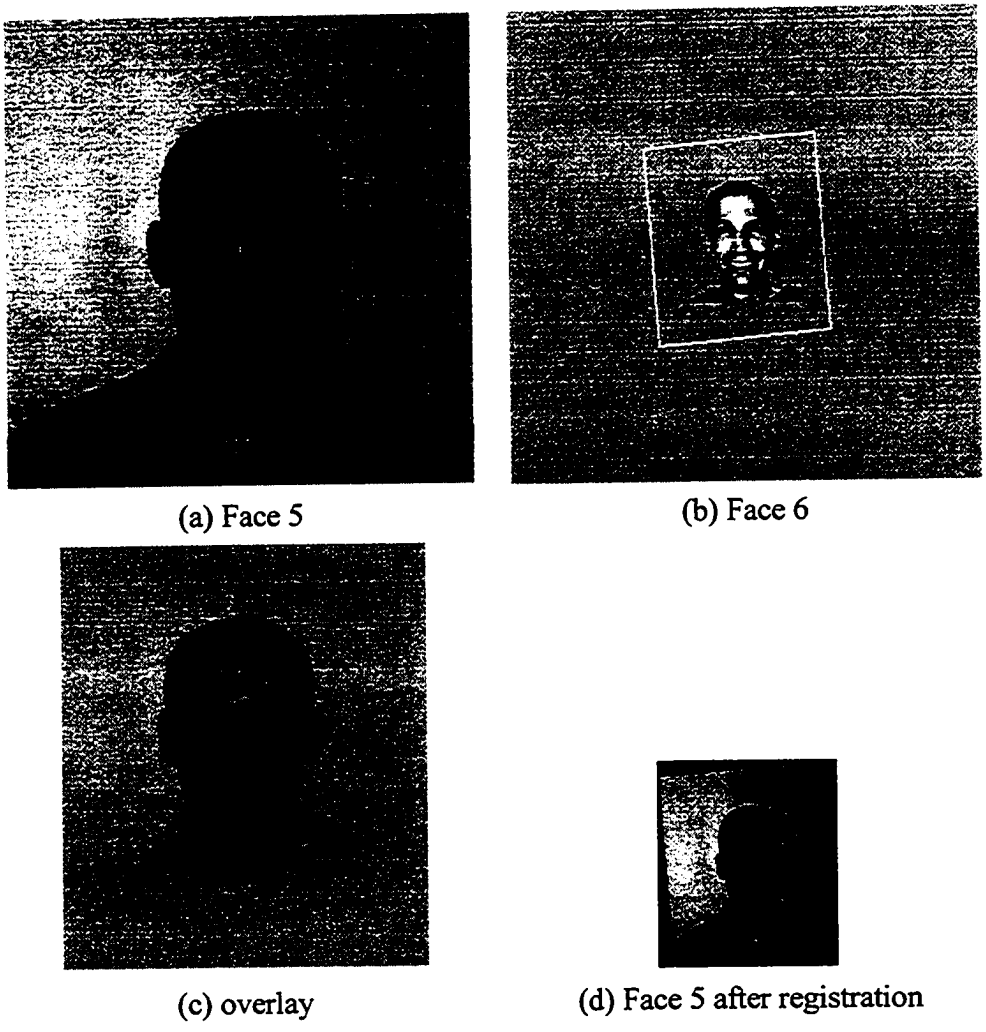
(e) before registration (intensity) (f) after registration (intensity)

Figure 4.4: Example of registration using gradient images. Alignment is now satisfactory because emphasis was placed on matching facial feature boundaries.

of these images varies from very highly textured to minimally textured areas. These input images can be problematic for feature-based methods like the one described in [38]. Our method uses all pixels and does not depend on any specific feature set. This distinctive capability lets us register the test images shown in Figs. 4.6 through 4.8. These images are taken from a single viewpoint. Images were acquired with (a) no magnification, and (b) 4x magnification with unknown rotation about the optical axis. We have identified the boundaries of the magnified images in their counterpart target images with a white rectangle after successful registration. The resulting overlays demonstrate high accuracy, as depicted by the absence of any noticeable visual misalignment. The correlation coefficient values for this set of images are above 0.85, which is very good considering camera noise and subsample filtering.

4.3. ROBUST IMAGE REGISTRATION ON CALIBRATED IMAGES

In the previous section, we have tested our registration algorithm on several facial and natural images taken by an uncalibrated camera with optical zoom. In order to test the robustness of the algorithm against a large set of parameters, we adopt a Monte Carlo simulation. We sample 1000 images from the Corel Stock Photo Library for our test images. The Corel library contains 20,000 royalty-free photographic images on 200 CD-ROMs (http://www.corel.com/). Each CD contains a different category of images. We randomly sampled five images from each CD, for a total of 1,000 images. Then, we randomly generated ten different sets of perspective parameters for each image. In this manner, we uniformly sampled 10,000 points from the parameter space. The range of these parameters are as follows: $\alpha \in [-30° \cdots 30°]$, $\beta \in [-30° \cdots 30°]$, $\gamma \in [0° \cdots 180°]$, $s \in [1.0 \cdots 4.5]$, $t_x \in [-40 \cdots 40]$, $t_y \in [-40 \cdots 40]$, where γ is the rotation about the optical axis, s is



(a) Face 5 (b) Face 6

(c) overlay (d) Face 5 after registration

Figure 4.5: The result of registering the pictures of the same person. Face 5 is 2.5 times larger than Face 6.

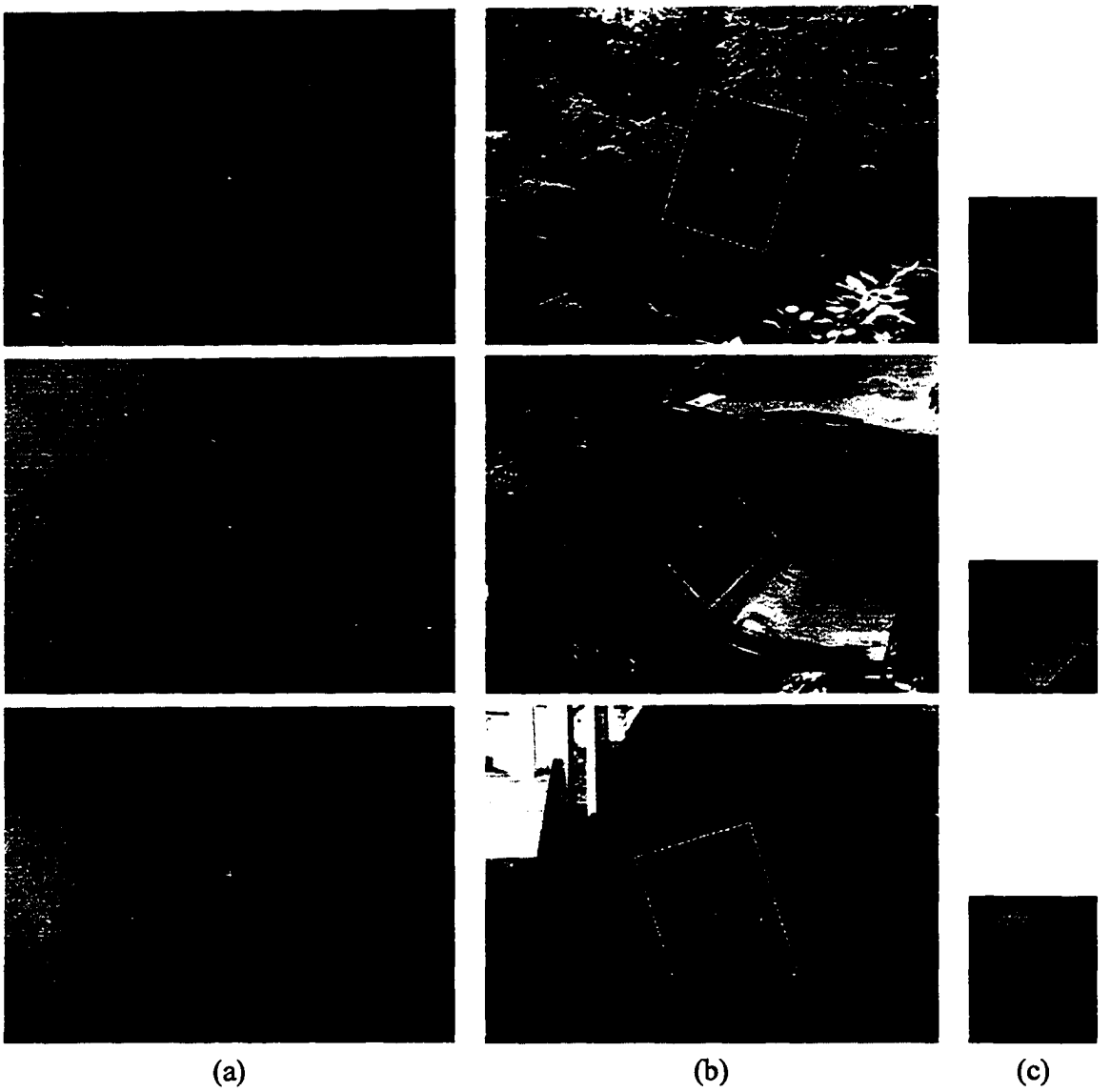


Figure 4.6: These examples demonstrate the registration results for (a) 4x zoom, arbitrary rotation, and mild perspective; (b) no zoom; and (c) registered result.

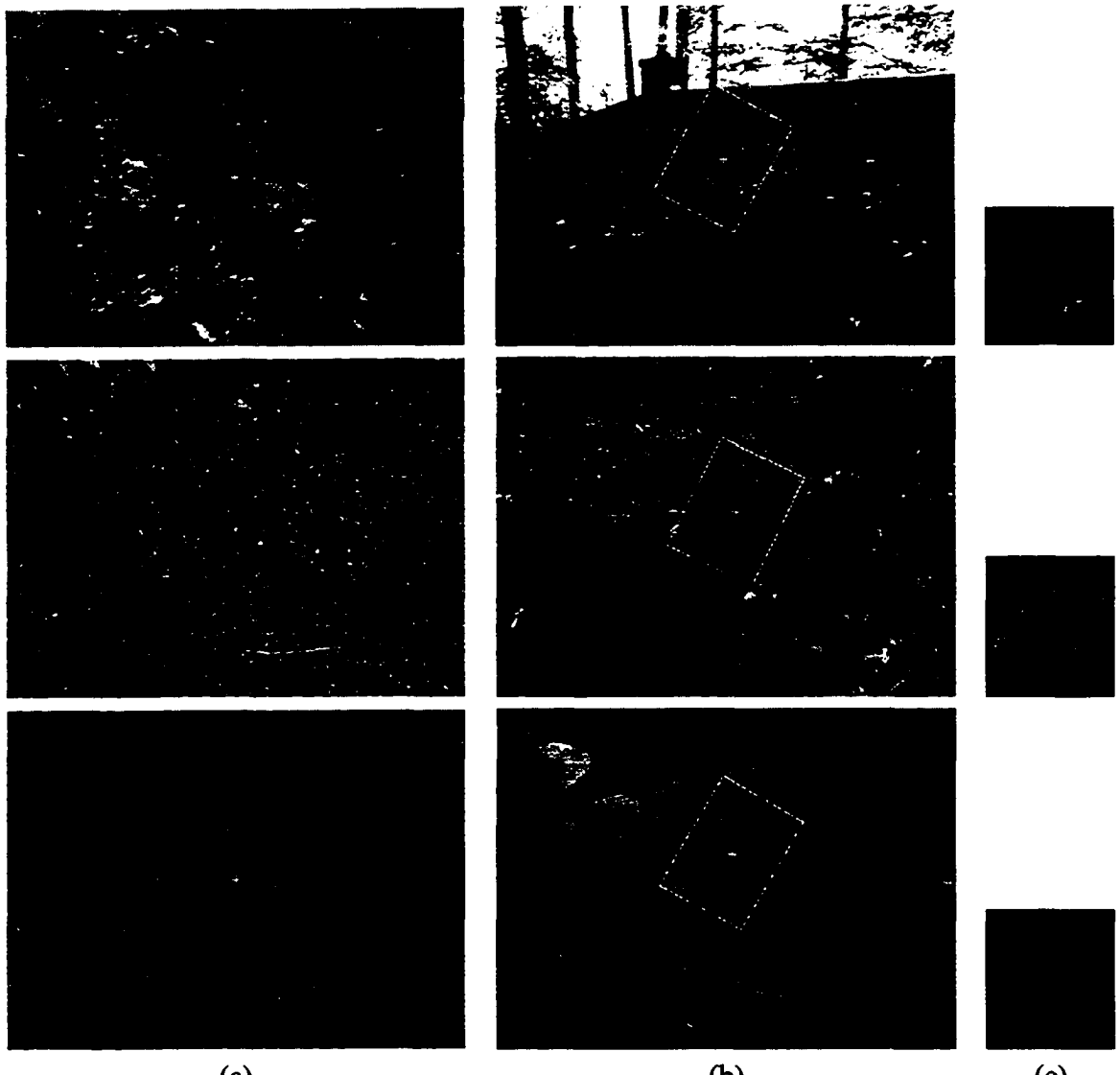


Figure 4.7: These examples demonstrate the registration results for (a) 4x zoom, arbitrary rotation, and mild perspective; (b) no zoom; and (c) registered result.

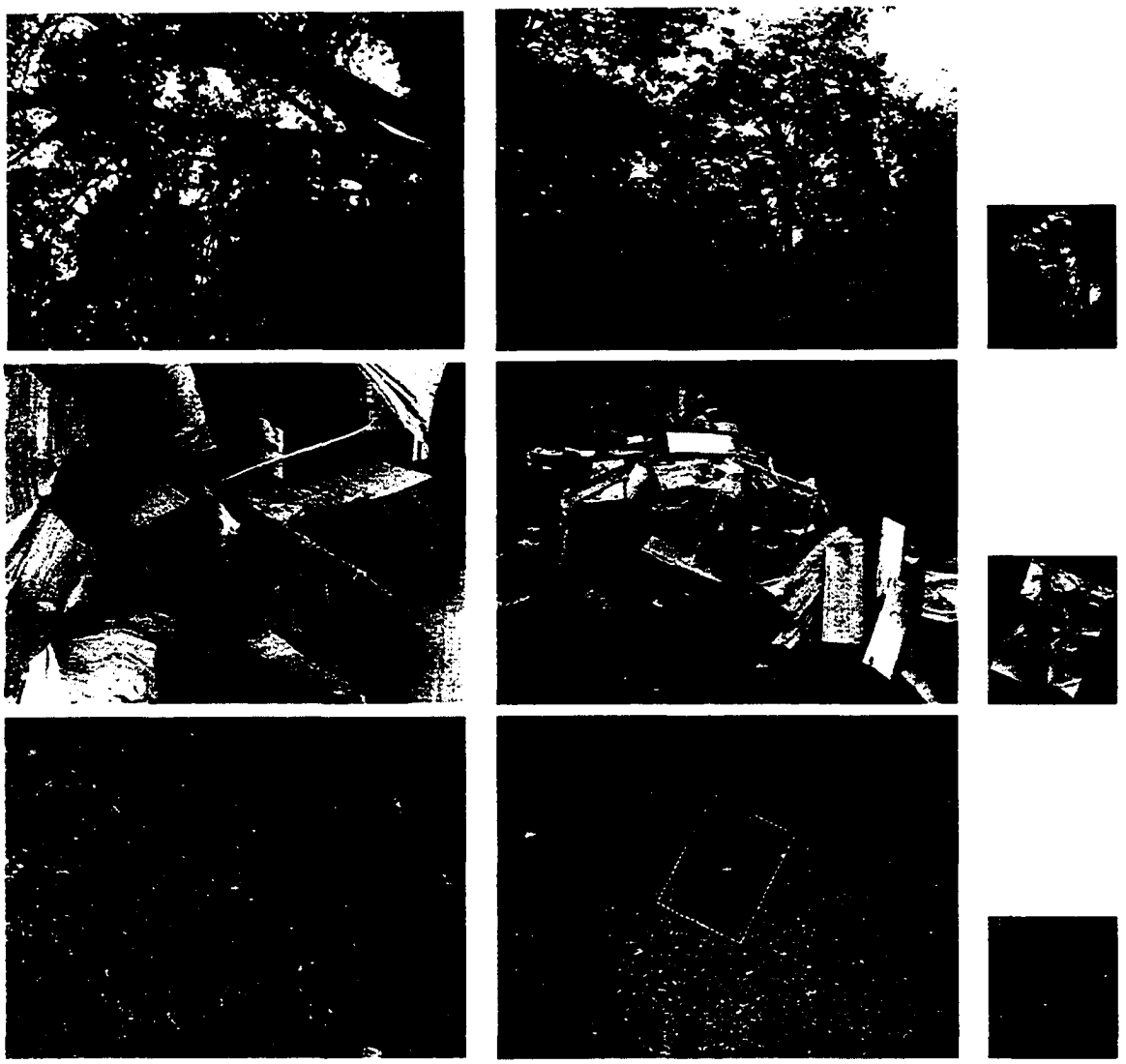
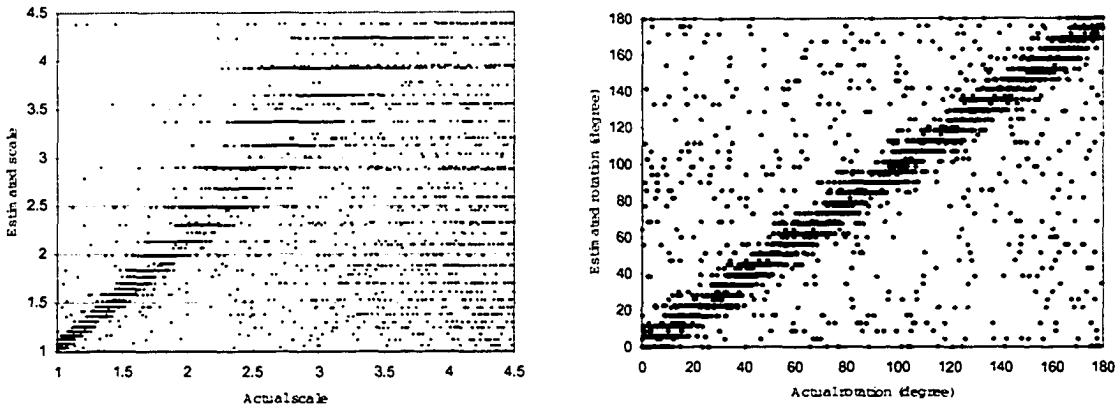


Figure 4.8: These examples demonstrate the registration results for (a) 4x zoom, arbitrary rotation, and mild perspective; (b) no zoom; and (c) registered result.

the scale factor of the digital zoom, and α and β rotate the image plane about the $x-$ and $y-$axes, respectively. These rotations introduce foreshortening effects. We generated 10,000 synthetic target images from these random parameters. First, we used our log-polar module to recover the global rotation, scale, and translations. Fig. 4.9(a) and Fig. 4.9(b) depict the plots of actual rotation and scale vs. estimated rotation and scale.



(a) (b)

Figure 4.9: The result of log-polar registration for 10000 cases. (a) actual scale vs. estimated scale. (b) actual rotation vs. estimated rotation.

These plots must be straight lines in an ideal estimation. However, we have 10% mismatches. These mismatches were largely due to several images in the test set that did not have interesting visual content in the central region. Since the log-polar algorithm crops a central window from the image, the lack of visual information there leads the algorithm to generate a false match. We have shown a few problematic images in Fig. 4.10.

Ideally, the plots in Fig. 4.9 should appear as a ramp function. Instead, a staircase appearance is noted. The reason for this apparent artifact is that the log-polar module quantizes the parameters space. Although the actual parameters are real numbers, the estimated parameters are their quantized counterparts.

The estimated parameters serve as the initial guess for the LMA module, which finds

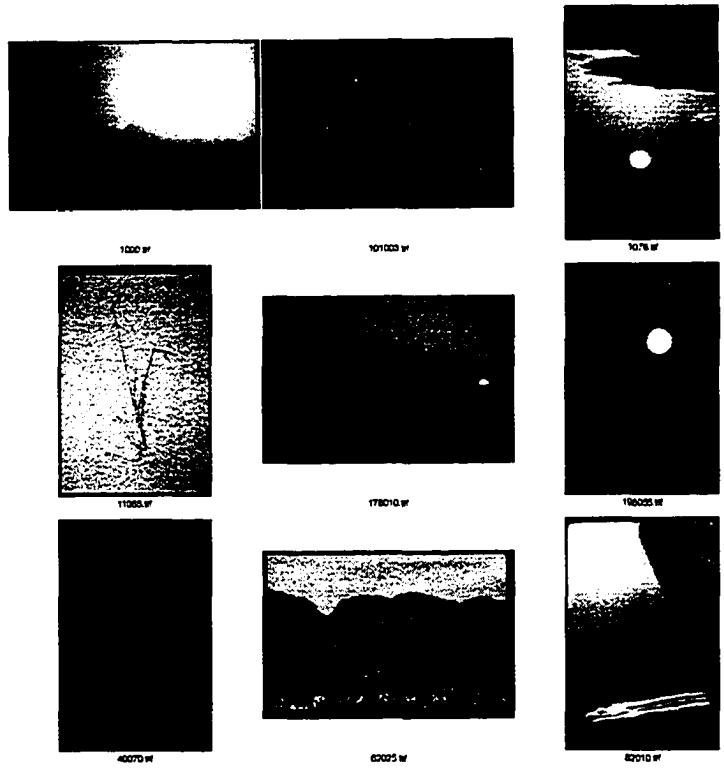


Figure 4.10: The log-polar algorithm failed to register the few images in the database that did not have visual information in their central regions.

the perspective parameters with subpixel accuracy. The registration operations conducted over the set of 10,000 image pairs yielded a 94% success rate. We calculated correlation coefficient values between the eight actual and estimated parameters. The correlation values lie in the $[-1 \cdots 1]$ range. If the correlation value is close to one, the error between the actual and estimated parameters is very small. The correlation coefficient histogram is plotted in Fig. 4.11. Notice that there is a sharp peak above 0.8 and that the majority (94%) of the cases are concentrated in this section.

In order to show the importance of the log-polar module, we ran the LMA without the estimated initial parameters from the log-polar module. The LMA module alone performed poorly, achieving only a 40% success rate. The correlation coefficient histogram is plotted

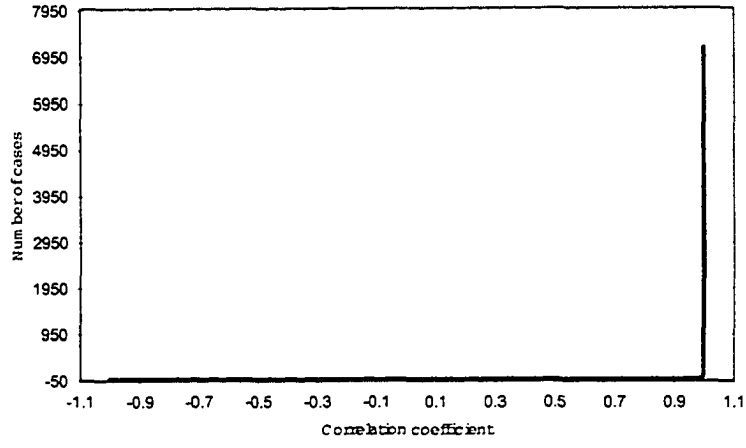


Figure 4.11: The histogram of the correlation coefficient values between the actual and estimated parameters after applying the log-polar and LMA modules.

in Fig. 4.12.

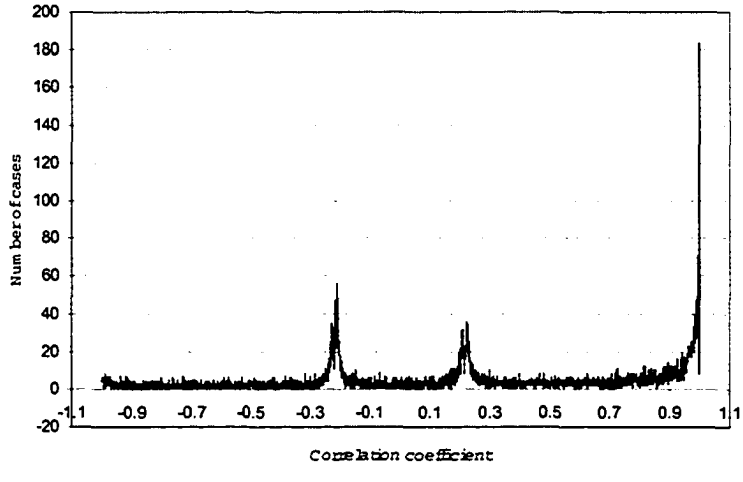


Figure 4.12: The histogram of the correlation coefficient values between the actual and estimated parameters after applying the LMA module alone.

4.4. NOISE (GLARE) REMOVAL

One of the applications of image registration is to improve image resolution. This method is known as super-resolution. Super-resolution is the process of combining multiple low resolution images to form a higher resolution one. Numerous super-resolution algorithms have been proposed in the literature [56, 22, 12, 51, 41, 125, 23]. If there is relative motion between the camera and the scene, then the first step to achieve super-resolution is to register the images. Once registered, the images may be fused, or blended, to enhance the resolution and remove noise.

We developed a method to remove specular glare, a typical problem in photography due to the reflection of light against shiny surfaces in the scene. The basis of this work is the observation that the appearance of specular glare (bright spot) changes based on the camera's viewpoint. Therefore, by acquiring and registering multiple views of a scene, it is possible to eliminate the moving spot. This exposes the underlying image, undegraded by the highlights of specular glare. The following blending function is used to fuse the registered images:

$$f(x) = \begin{cases} min\{I_1, I_2\}, & \text{if } |I_1 - I_2| > T \\ \frac{(I_1+I_2)}{2}, & otherwise \end{cases} \tag{4.1}$$

In Fig. 4.13(a) and (b), two such images are shown. Clearly, they differ by a large perspective deformation. This is readily apparent by noting that the overlay of these two images exhibits great ghosting effects, as shown in Fig. 4.13(c). After finding the necessary perspective parameters, we are able to warp Fig. 4.13(a) into alignment with Fig. 4.13(b), as shown in Fig. 4.13(d). Notice that naive averaging does not improve the result and the

white saturated glare is visible in Fig. 4.13(e). However, using the blending function in Eq. (4.1) substantially removes the glare, yielding a superior fused image in Fig. 4.13(f). Those areas that still retain glare are common in both input views. Additional viewpoints would have further eliminated the glare.

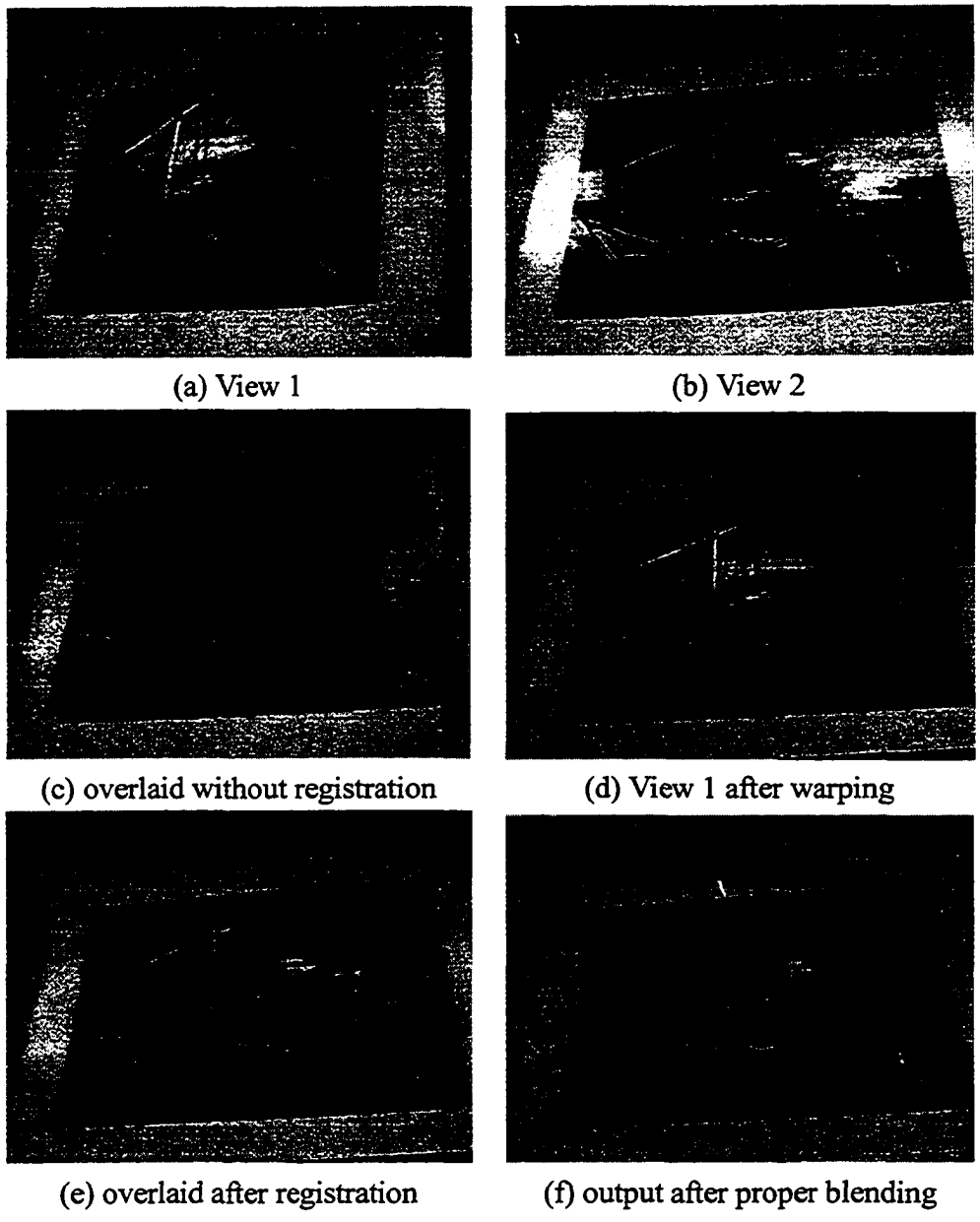
4.5. IMAGE AND VIDEO MOSAICS

A mosaic is a high resolution image that is created by stitching together the low resolution frames from a video sequence or several overlapping images. Mosaic techniques have been used to obtain images with a large field of view. The process involves two steps: (1) alignment of frames in the sequence, and (2) composition (blending) of these frames in order to create the final mosaic image. The alignment of video frames is straightforward, since the motion between adjacent frames is small. Remarkable progress has been documented during the last decade in the area of video mosaicing [56, 102, 82, 72, 55, 33].

We have tested our registration method to create image and video mosaics. In this section, we emphasize registration accuracy. We defer discussion of blending and global color correction, which are important elements in creating seamless mosaics. The reader can refer to [21][110] for information on image blending.

4.5.1 Image Mosaics

We begin by demonstrating our method to create a panoramic image from a sequence of widely disparate input images. That is, successive image pairs need only overlap by at least 20%. Successive image pairs I_i and I_{i+1} are registered independently, for $i = 1, 2, \cdots, N$, where N is the number of images in the sequence. After all perspective parameters are estimated, we can get a set of 3×3 transformation matrices $T_{i,i+1}$ where



(a) View 1 (b) View 2

(c) overlaid without registration (d) View 1 after warping

(e) overlaid after registration (f) output after proper blending

Figure 4.13: Glare removal.

$i = 1, 2, \cdots, N-1$. Each matrix $T_{i,i+1}$ contains all perspective parameters $\{a_1, a_2, \cdots, a_8\}$ of image I_{i+1} with respect to I_i. This set of parameters represents relative motion between two successive images in the sequence.

After the transformations are computed, we select one image in the sequence to be a reference image. A new set of transformation matrices must be computed that relate each image in the sequence to the newly selected reference image. For example, if image I_j is chosen to be the reference image, the motion parameter set will be $\{T_{ij} : i = 1, 2, \cdots, N, i \neq j\}$. Note that T_{jj} is the 3×3 identity matrix.

Figure 4.14: Image set used to create a panorama image.

In Fig. 4.14, a set of input images is presented. The set was acquired from http://www.inrialpes.fr/movi/. Notice that the images differ by large transformations in perspective, scale, and rotation. We have tested two commercial stitching software packages *(Quickstitch™, Ulead COOL 360™)* on this nontrivial example. Both systems failed to register the images. In contrast, our algorithm produced the result shown in Fig. 4.15. In order to best expose any misalignment, we applied unweighted averaging upon the overlapping areas. No advanced feathering technique was used since they can be misleading through their ability to hide minor misalignments.

For those applications that do require seamless mosaics, we implemented a blending function in which the pixels in the overlapping area are weighted based on their distances

Figure 4.15: Mosaic produced by stitching images shown in Fig. 4.14.

from the center of the image. The weighting function is a cosine falloff that gives more weight to the central pixels (Eq. (4.2)). We compare the results of unweighted averaging with the cosine falloff blending function in Fig. 4.18.

$$d_1 = \sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2}$$

$$d_2 = \sqrt{(x_2 - x_c)^2 + (y_2 - y_c)^2}$$

$$w_1 = f(d_1), w_2 = f(d_2)$$

$$I_{out} = \frac{w_1 1 * I_1 + w_2 * I_2}{w_1 + w_2}$$

where $f(t)$ is:

$$f(t) = \frac{1 + cos(t)}{2} \tag{4.2}$$

4.5.2 Video Mosaics

The interframe differences between successive video frames is generally small. This simplifies the registration process, permitting us to defer the log-polar module. In order to produce a challenging video mosaic task, we collected video sequences using a handheld camera subjected to large rotation/zoom operations. Furthermore, we only retained every 20th frame to produce a sparse input sequence. The frames used to generate our video mosaic are shown in Fig. 4.16. The output video mosaic is shown in Fig. 4.17.

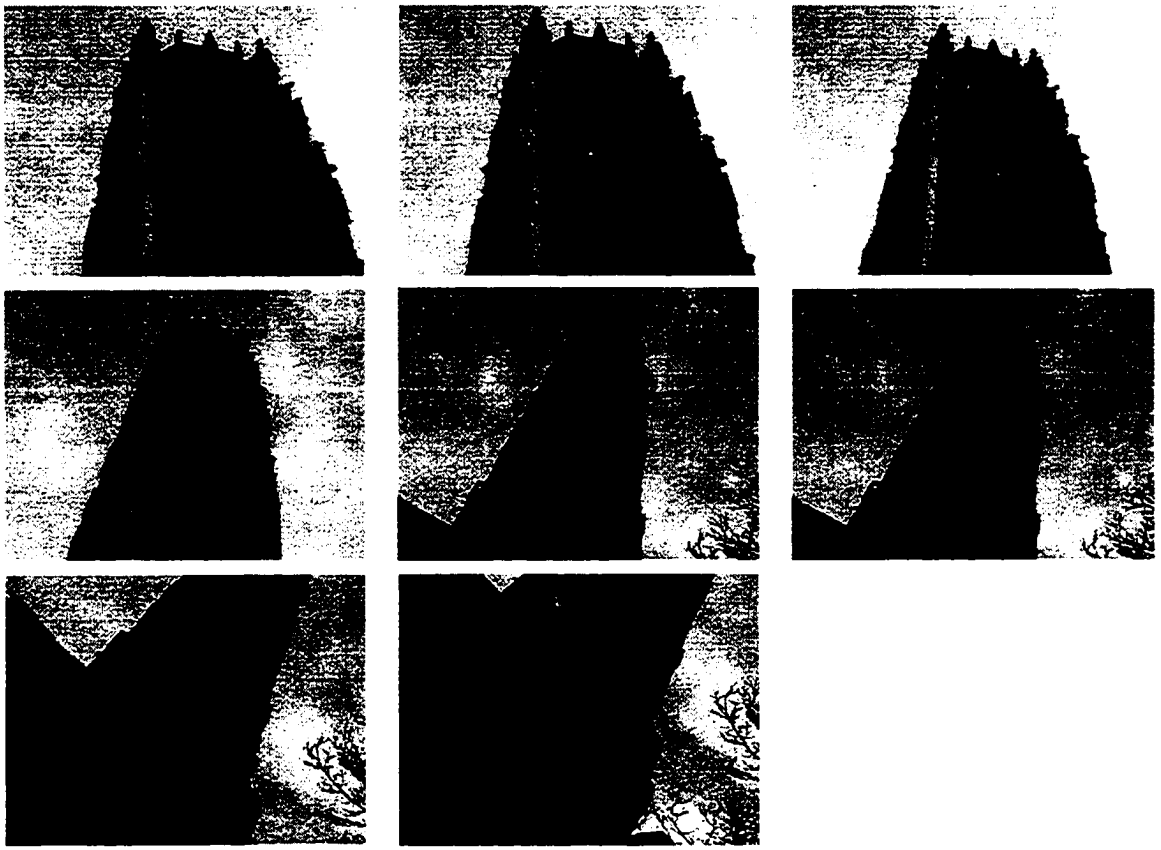


Figure 4.16: The input frames used in the video mosaic.

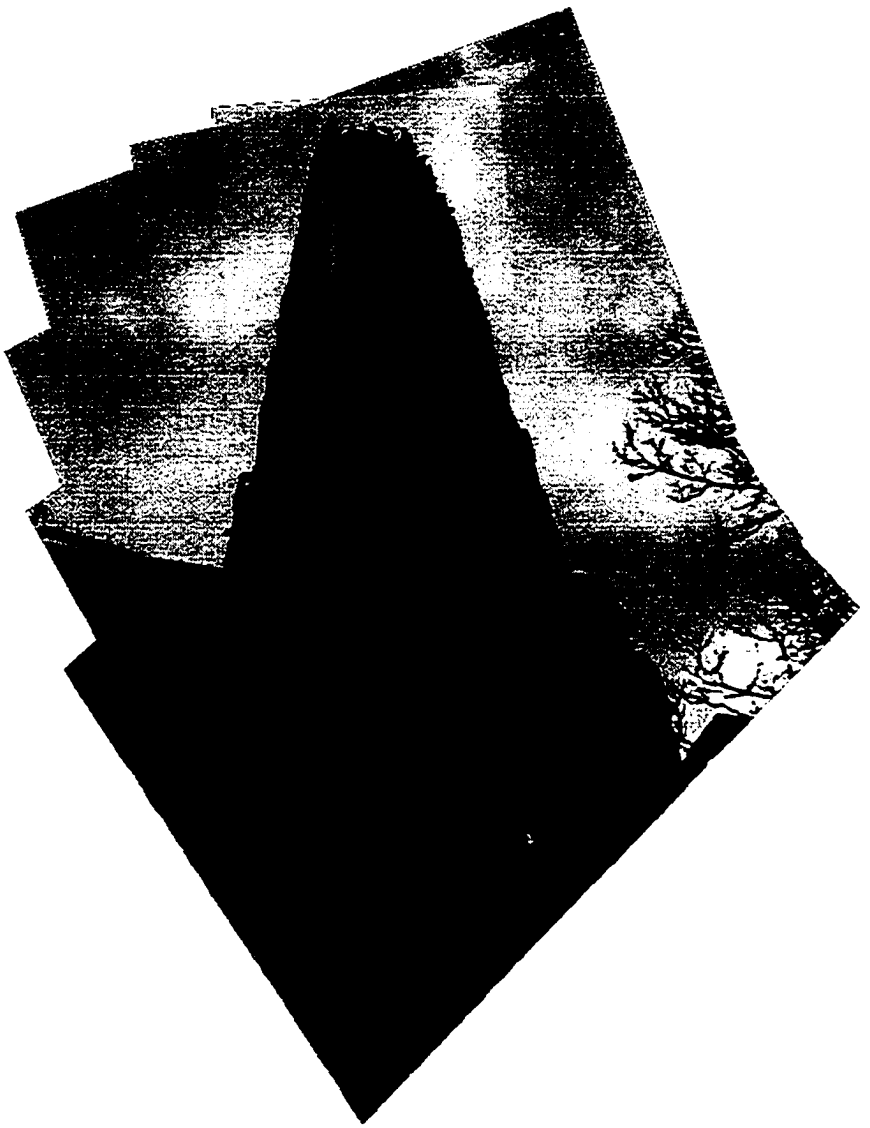


Figure 4.17: Video mosaic blended by unweighted averaging.

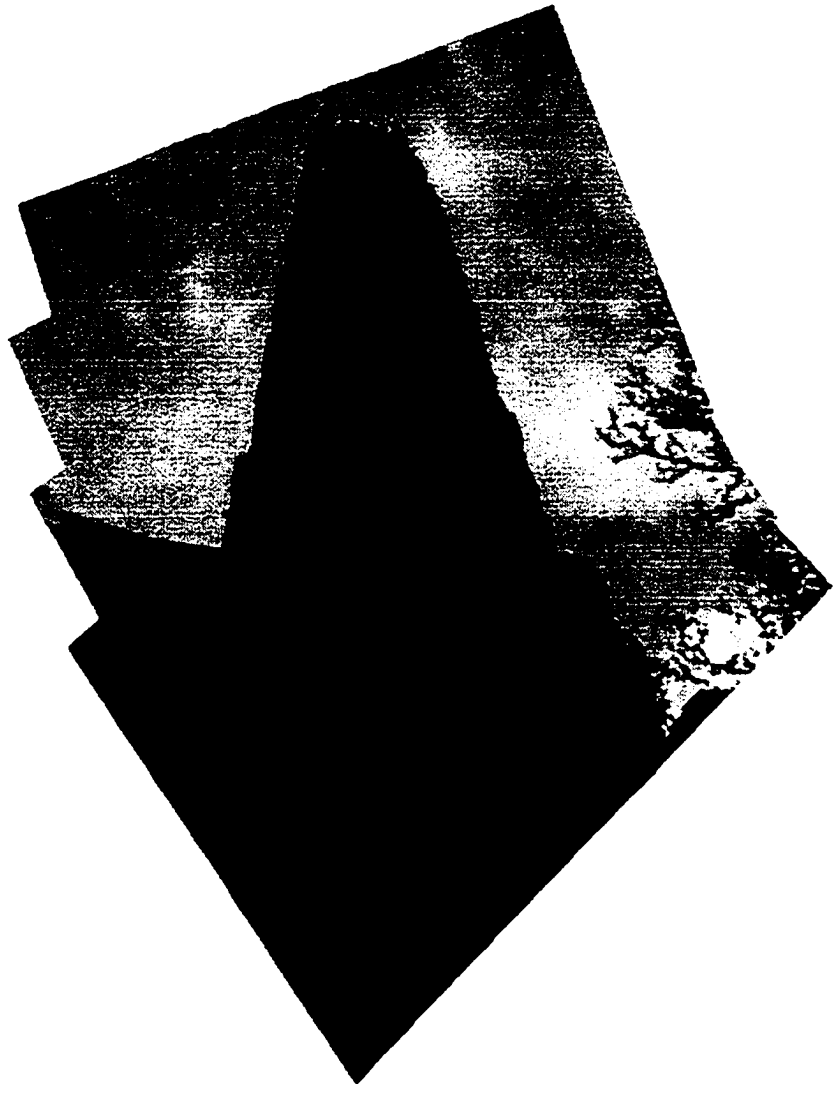


Figure 4.18: Video mosaic blended by Eq. (4.2). Compare with Fig. 4.17.

Chapter 5

DIMINISHED REALITY

5.1. INTRODUCTION

For the past decade, there have been remarkable advances in Augmented Reality (AR) [10, 9]. This emerging technology is beginning to be integrated in many industrial and medical applications [1, 18, 59]. Exemplary systems have shown how Augmented Reality could support the assembly or maintenance of complex equipments in a factory as well as the training of technicians involved in such tasks [27, 123, 60]. Such tools could offer more efficient maintenance with fewer errors and improved safety for the technicians. Augmented Reality not only is beneficial for maintenance, but also can be used for future planning and development in a complex power plant. Navab et al. [79, 7, 6] introduced novel methods to use existing orthographic drawings and calibrated images of a power plant to build 3D models of pipelines and use them for augmentation. The final mixed reality product could help designers to efficiently plan for future developments and for the revamping of refineries, as well as power, chemical and water treatment plants.

A complex plant could be quite cluttered with pipelines, pumps, and valves. When we want to plan and illustrate revamp and modification, it is necessary to remove objects from

existing images or videos. In these cases we need to correctly visualize the background behind them in order to have a reasonable result (Fig. 5.1).

Figure 5.1: The portion of the vertical pipe in the top image is removed and the background is exposed in the middle image. Two additional images are used for rendering the background. Bottom image illustrates a possible augmentation of the diminished scene.

We proposed a method to remove an object of interest from a reference view and render it with the proper background. Removal and replacement of an object in an image can be referred to as diminished reality. Removal and replacement means that whatever is behind the object should be rendered when the object is removed. This rendering can be realistic or approximate. Given a set of calibrated images of a real scene, we aim to remove an object from an image (reference image) using two or more other views. We assume that the view of the background can be modelled by a paraperspective projection model, see Appendix B.1. The border of the object is interactively defined by the user in the reference and in at least two other images.

5.2. RELATED WORK

The reconstruction of a scene from a set of calibrated or weakly calibrated images has been studied by the computer vision community for many years and ground breaking algorithms have been developed including stereoscopy [64], Image-Based Modelling [34], Image-Based Rendering [2, 76, 94, 49, 67], and volumetric reconstruction [100, 40].

Laveau and Faugeras [64] use consistency along the epipolar lines in multiple views to render the new image. Seitz and Dyer [94] proceed to image rectification and then use the disparity maps, and McMillan and Bishop [76] use Plenoptic modelling for image based rendering. In these works, a dense disparity map has to be provided and a new image of the whole scene is rendered, which can be computationally expensive. Therefore, a need exists for a fast and practical system and method for removing or replacing an object in an image, where the number of available source images is limited. Several researchers have used the "Diminished Reality" term in the past. Mann and Fun [71] proposed a method for removing the content of a planar object and replacing it with another planar texture in a movie by video orbit. Wang and Adelson [109] proposed a method for segmenting a sequence of video images into multiple layers and rendering the same video when removing one of the layers. Lepetit and Berger [66] proposed a method for tracking a user-defined boundary in a set of moving images and detecting the occlusion to remove the object from the scene. Mourgues et al. [78] designed a system to remove the endoscopic instrument for 3D-endoscopic surgery by using stereoscopy. The above methods use a dense temporal sequence of images taken by video cameras. This allows them to segment and track the objects on their apparent motion in the video sequence. However, this can be computationally expensive and slow. In this work we are interested in a diminished reality solution, where only a few images taken from different viewpoints are used rather than a video sequence [124]. This is quite useful for many applications. In particular, this is of high value in conjunction with as-built reconstruction of industrial sites, where digital cameras are used to acquire multiple views. Industrial sites are very large and video data will not be practical to handle. The digital images are taken from different point of view in order to have large baselines for accurate reconstruction. This is suitable for Diminished Reality, because it increases the chance of seeing what is be hind an object in the other images.

5.3. MULTIVIEW PARAPERSPECTIVE PROJECTION MODEL

Our method, unlike previous methods, does not use a dense temporal sequence of images. Instead, we use widely separated images from different viewpoints. Several researchers have proposed elegant methods for the calibration and recovery of the fundamental matrix of wide baseline images by using locally invariant affine features [106, 90]. We assume that images are already calibrated either using methods described in [106, 90](markerless) or using markers. Fig. 5.2 depicts the setup for image acquisition.

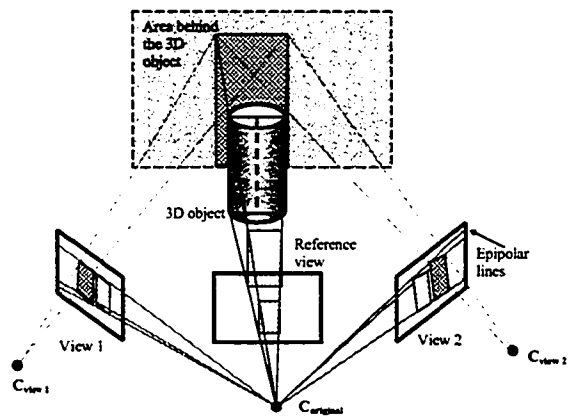


Figure 5.2: Diminished Reality setup.

As we mentioned, a good background reconstruction is very important for diminished reality. If we do not reconstruct the background properly, we will get unpleasant discontinuities at the boundaries of the reconstructed background and the rest of the reference view. In general, we cannot assume that the background is planar. It may include large depth variations. For the general case, the solution falls into the category of stereoscopy. We can alleviate the problem by assuming that imaging of the background can be modeled by a paraperspective projection geometry. This constraint not only facilitates the reconstruction, but also is a practical and reasonable assumption. In order to see the background from multiple views, there needs to be a depth variation between the object to be removed

and its background. If the background is far enough and the region of interest is small, the paraperspective assumption is a valid one. If the region is not relatively small, we suggest an additional step, in which we divide the region into smaller areas. For each area we search for the correct background. In this case, by reducing the size of the region of interest, we increase the validity of our paraperspective projection model.

The user defines the object or a collection of objects to be removed from the reference image. This is done by defining a polygon around the region of interest (ROI) including the object or objects of interest. In the rest of this chapter, we take the ROI to be of rectangular shape only for the sake of simplicity. In theory this ROI can be of any geometrical shape. We then form a frustum originating from the camera center of the reference view and passing through the vertices of the ROI. (Fig. 5.2). We then generate a family of 3D polygons, which we call "*virtual planes*", all parallel to the image plane and bounded by this frustum. The virtual plane is defined uniquely by its depth from the optical center of the reference camera. This plane is a possible candidate for approximating the background, if it is behind the object to be removed. Each plane defines a set of homographic transformations between its images in all the calibrated views. We use these homographic transformations to render the background image in the reference view using the image of the virtual plane in each additional view. As the virtual plane gets closer to the real background, these rendered or warped images from different views exhibit higher correlation. This represents the main motivation for introducing the three methods presented in the next sections.

Fig. 5.3 shows the variation of the correlation between the warped images from two views, when the virtual plane moves away from the image plane. There are two maxima. The first one corresponds to the object since it is in front of the background and, in this example also can be seen from the two views. One could easily verify this by correlating the warped image onto this virtual plane with the ROI in the reference view. The second

maximum correlation value relates to the plane of the background. In our application, we will assume that we know the depth of the object to be removed from the reference image. Since the cameras are calibrated, the approximate depth of the object can be derived easily by triangulation. In this case, we start the process by taking a virtual plane, which is located behind the object of interest. As we move the virtual plane away from the reference camera, we search for the maximum correlation between the images warped onto the virtual plane bounded by the frustum.

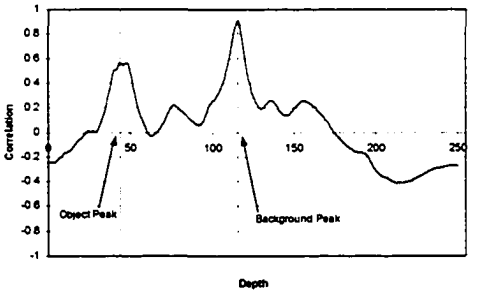


Figure 5.3: The variation of the correlation between the warped images from two views, when the virtual plane moves away from the image plane (Correlation vs. depth).

5.4. DESCRIPTION OF THE ALGORITHM

Let us assume we have three calibrated images:

$$\begin{aligned} view_1 &: \ I_1, \mathbf{A}, \mathbf{M}^1 \\ view_2 &: \ I_2, \mathbf{A}, \mathbf{M}^2 \\ reference\ view &: \ I_o, \mathbf{A}, \mathbf{M}^o \end{aligned} \tag{5.1}$$

$\mathbf{A}$ and $\mathbf{M}^i = [\mathbf{R}^i, -\mathbf{R}^i\mathbf{t}^i]$ define the intrinsic and the extrinsic parameters of the ith camera. Suppose that we know the approximate depth of the object and therefore the

position of the four corners of a "*virtual plane*" $\mathbf{P}_i$, $i \in [1 \cdots 4]$ immediately behind it. We generate a set of virtual planes based on a frustum originating from the center of the reference camera:

$$\mathbf{P}_i' = \mathbf{P}_i + \lambda(\mathbf{P}_i - \mathbf{T}_0) \tag{5.2}$$

where λ is the depth of each virtual plane and acts as a free parameter for moving the virtual plane bounded by the frustum and its four vertices, $\mathbf{P}_i' = \mathbf{P}_i(\lambda)$, back and forth starting behind the object of interest ($\lambda = 0$). These four corners are function of λ : $\mathbf{P}_i' = \mathbf{P}_i(\lambda)$. We project the four corners into all views to get their 2D positions:

$$\begin{aligned} \mathbf{p}_i^1(\lambda) &= \mathbf{A}\mathbf{M}^1\mathbf{P}_i^1(\lambda) \\ \mathbf{p}_i^2(\lambda) &= \mathbf{A}\mathbf{M}^2\mathbf{P}_i^2(\lambda) \\ \mathbf{p}_i^o &= \mathbf{A}\mathbf{M}^o\mathbf{P}_i^o \end{aligned} \tag{5.3}$$

p_i^1 and p_i^2 $i \in [1 \cdots 4]$ define the boundary of quadrilaterals in the views and are functions of λ. p_i^o is obviously not a function of depth λ, since the plane is moving parallel to the image plane of the reference camera. We find a planar perspective transformation, also called a homography, for each view that maps the projected plane onto the reference view:

$$\begin{aligned} \mathbf{H}_1(\lambda) : p_i^1(\lambda) \rightarrow p_i^o \\ \mathbf{H}_2(\lambda) : p_i^2(\lambda) \rightarrow p_i^o \end{aligned} \tag{5.4}$$

We then warp and crop images from the views based on these homographic transformations. This creates candidate background images, $I_{bk1}(\lambda)$ and $I_{bk2}(\lambda)$. This process rectifies both images so that their scan lines are parallel to the reference view. We need a *similarity*

measure to confirm that both warped images agree on a common background. We choose the correlation coefficient for its performance and implementation simplicity:

$$< I_1 \circ I_2 >= \frac{\sum (I_1 - \mu_1)(I_2 - \mu_2)}{\sqrt{\sum (I_1 - \mu_1)^2 \sum (I_2 - \mu_2)^2}} \tag{5.5}$$

Where, μ_i is the average intensity of I_i. Maximizing the following objective function results in recovering the average depth of the background:

$$\underset{\lambda}{arg\ max} < I_1(\lambda) \circ I_2(\lambda) > \tag{5.6}$$

The method searches for λ, which maximizes the correlation. A high correlation indicates that the corresponding virtual plane could represent a unique background for the reference image, see Fig. 5.4.

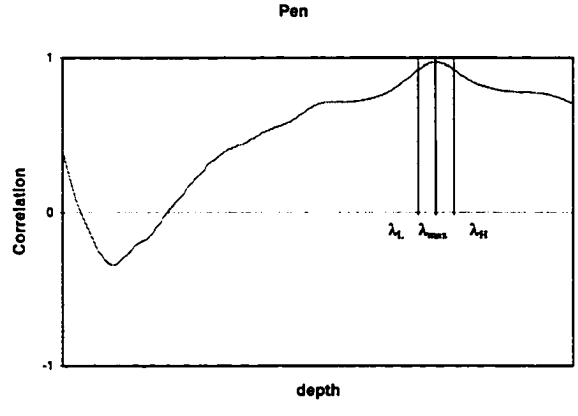


Figure 5.4: Correlation vs. depth for one of the experiments.

Once the depth λ_{max} for the virtual plane corresponding to maximum correlation is determined, we can render the background. The rendering can be done in different ways, see [21] for a classical and [110] for a newer approach. The simplest way is to render the background by warping one of the source images onto the virtual plane. Since the source

images have the maximum correlation, any of these warped images could be a good approximation of the background. One could also take their average to render the background using all the source images. Each pixel on the final image is, in this case, associated with the average of the intensity values of the corresponding pixels in the warped images. Another possibility is to use the weighted average, where the weighting is a function of the relative position and orientation of each camera with respect to the virtual plane. The objective is to give more weight to a source image that is taken by a camera close to the background plane and parallel to it. Such a camera provides an image with higher resolution and lower perspective distortion from the background to be rendered, as compared to other cameras.

The proposed method works perfectly, when the background can be approximated by a plane parallel to the image plane, see Sec. Section 5.5. But this is not always the case and the image plane is not parallel to the the background plane. In Section Section 5.6, we discuss a robust method that recovers the orientation of the background plane by image registration. In Section Section 5.7, a general non-planar background is approximated by a piecewise planar patches.

5.5. PLANAR BACKGROUND PARALLEL TO REFERENCE IMAGE PLANE

If the background is far enough from the camera or does not represent major depth variations, it can be reasonably approximated by a plane parallel to the reference image. This means that the depth variation of the background is very small relative to its average depth λ_{max}. In this case, we can get a perfect rendering, and warped images from all views coincide exactly. In this case all background textures are warped by a set of homographies $(\mathbf{H}_i)$ and blended to capture the correct background, see Fig. 5.9.

5.6. PLANAR BACKGROUND WITH ARBITRARY ORIENTATION

In the second case, we assume that the variation of the depth relative to λ_{max} is small, but the plane is not parallel to the reference image plane and has an unknown orientation (Fig. 5.5).

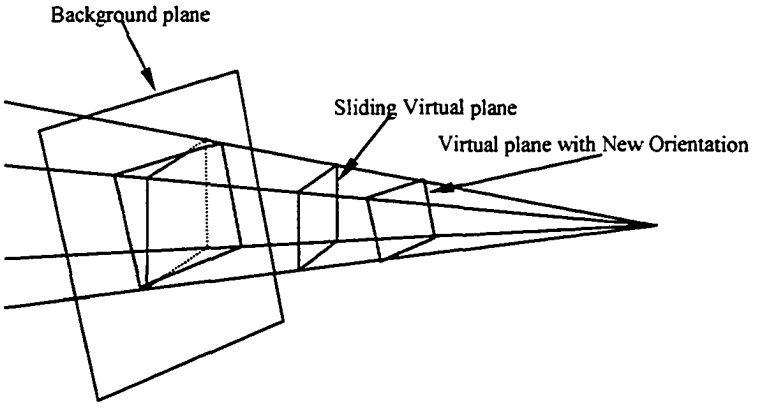


Figure 5.5: Planar background non parallel to the image plane.

When we take the virtual plane to be again parallel to the reference image plane, the maximum correlation is usually found at the correct average depth. However, the maximum correlation score is quite low i.e. < 0.7. This means that even if the two views are looking at the same areas with common textures, our virtual plane cannot represent the background well. In this case, we register the two warped images to find the necessary transformations between the corresponding areas on the two source images. This allows us to establish correspondences between the two areas and to reconstruct the correct background plane.

Fig. 5.6 illustrates different modules of the algorithm. In module (A), after finding proper depth (λ_{max}), We project the vertices of the virtual plane onto the views. In module (B), we estimate homography $\mathbf{H}_1$ and $\mathbf{H}_2$. $\mathbf{H}_1$ and $\mathbf{H}_2$ provide the most similar textures relative to the other virtual plane, however they provide neither a unique nor a uniform background. If we use these homographies for rendering the background, we will introduce

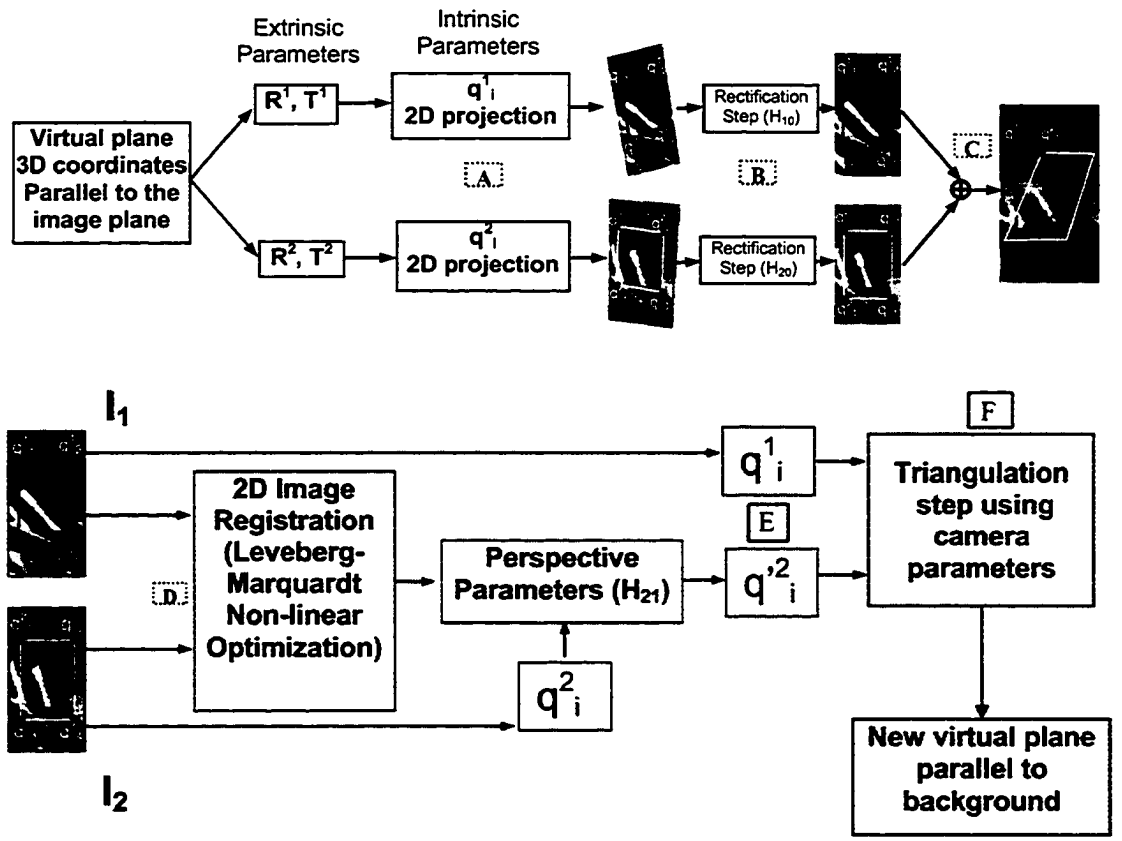


Figure 5.6: Flow chart of proposed method for recovering the plane orientation.

discontinuity and parallax, see Fig. 5.6(C). The image features at q_i^1 and q_i^2 are not aligned.

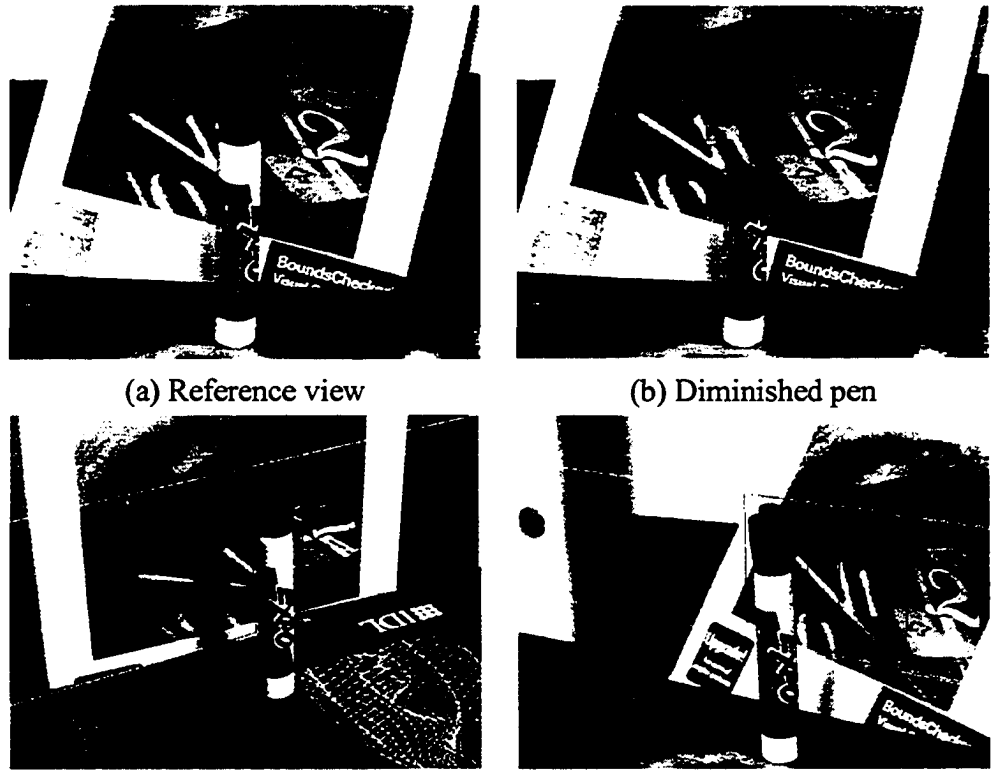
$$q_i^1 = \mathbf{H}_1 p_i^1(\lambda_{max})$$

$$q_i^2 = \mathbf{H}_2 p_i^2(\lambda_{max})$$

$$q_i^1 \neq q_i^2$$

The problem is shown in the Fig. 5.7(b). Clearly, we see double features and ghosting effects in the reconstructed background.

To solve this problem, we use registration techniques to align the two warped im-



(a) Reference view (b) Diminished pen

(c) View$_1$ (d) View$_2$

Figure 5.7: Background is not parallel to the image plane. Unpleasant ghosting effect is seen in the blended background.

ages, I_{bk1} and I_{bk2}, and estimate the residual perspective parameters due to the different orientation of the background Fig. 5.6(D). The registration process is based on our hybrid registration algorithm. We then use estimated perspective parameters, $\mathbf{H}_{12}$ to transform 2D coordinates of each quadrilateral's corners in one of the views, e.g. $q_i'^1 = \mathbf{H}_{12}\, q_i^1$ in view$_1$. Now, 2D coordinates of quadrilaterals in view$_1$ and view$_2$ represent reliable point correspondences (Fig. 5.6(E)). Since the two views are fully calibrated, we can use these point correspondences to correctly reconstruct the optimal virtual plane through triangulation (Fig. 5.6(F)). The virtual plane now is parallel to the background and is bounded inside the object frustum. It therefore results in a satisfactory result for our Diminished Reality application, (Fig. 5.10).

5.7. NONPLANAR BACKGROUND

We cannot assume that the background is always planar. However, this is not our main assumption. Our assumption is that the paraperspective projection model needs to be valid. This model will become a valid one if we look at small 3D patches. In order to find whether we have a planar background or not, we can study the variation of correlation scores as the virtual plane moves through the background. Fig. 5.4 shows that if the background is not a plane parallel to the image, then instead of getting a maximum correlation, we obtain an area where the correlation remains high. In this case, the assumption about paraperspective background is not valid anymore and we do not wish to fit a single plane to this background. In order to approximate this non-planar background, we subdivide the plane in a quad-tree fashion and adjust the depth of each tile separately, (Fig. 5.8). Our method has some interesting similarities to the one known as voxel coloring (space carving) [100, 40]. The goal of voxel coloring is to create a 3D model from a large number of calibrated images. They assume that the scene needs to be outside the convex hull of the cameras viewpoint and the volume covering the object is made of small cubes named voxels. The size of these cubes defines the resolution of the 3D reconstruction. The smaller the voxel, the finer is the rendering resolution. The differences between our proposed method and voxel coloring (space carving) methods is that we reconstruct the background surface by small planar patches and not by voxels. This makes plenty of sense for our application. The space carving method is overkill for our application, since we only aim to render the image from the vantage point of our reference camera. Our method therefore satisfies itself with the reconstruction of planar patches seen by the first camera and not the whole background volume.

We refine the background's surface by subdividing the ROI recursively and recovering the depth of each of its tiles. At each step of this recursive algorithm, each tile uses the

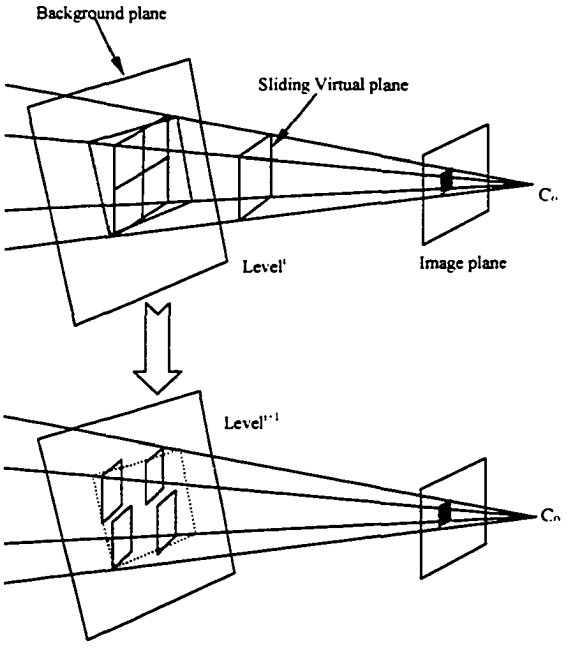


Figure 5.8: Background is approximated as piecewise planar. The projection model remains paraperspective.

recovered depth (λ^i_{max}) from the previous step of the algorithm to initialize the search. As we subdivide the region and get closer to the right depth, we also reduce our search interval Δ^i by a factor $f(i)$:

$$\lambda^{i+1} \in [\lambda^{i+1}_{low} = \lambda^i_{max} - \Delta^i, \lambda^{i+1}_{high} = \lambda^i_{max} + \Delta^i] \tag{5.7}$$

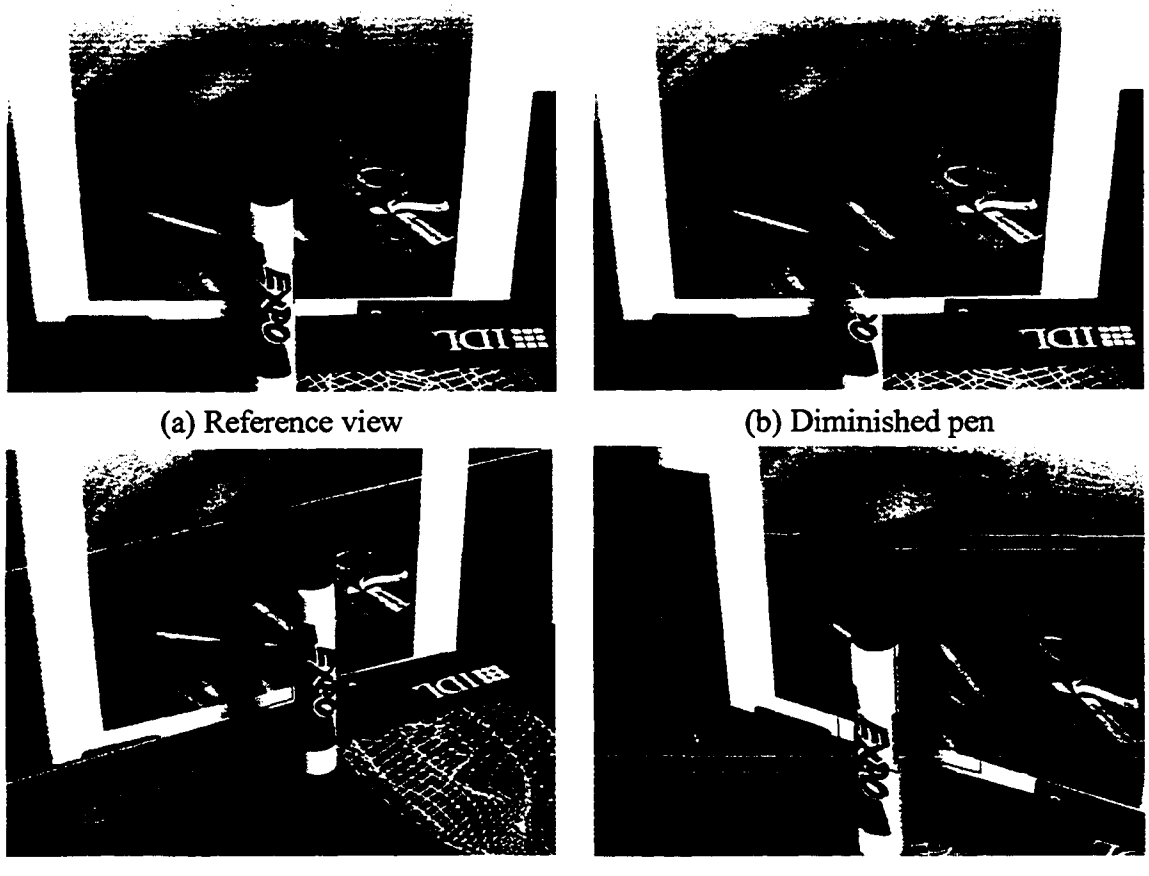
$$\Delta^{i+1} = \Delta^i \downarrow_{f(i)}$$

In our implementation $f(i)$ is a linear function of the recursive subdivisions level.

5.8. EXPERIMENTAL RESULTS

In this section, we first present the results of the application of our diminished reality algorithm to a series of real images obtained under controlled conditions. The objective

of these experiments are to observe the performance of the algorithm for the different scenarios described in Section 5.5, Section 5.6 and Section 5.7. We then conclude the experimental results with its application to two industrial scenes. we created several scenes with simple objects and planar and non-planar backgrounds. In the first example, the reference camera is almost parallel to the planar background behind the object, a red marker. Fig. 5.9 depicts this arrangement.



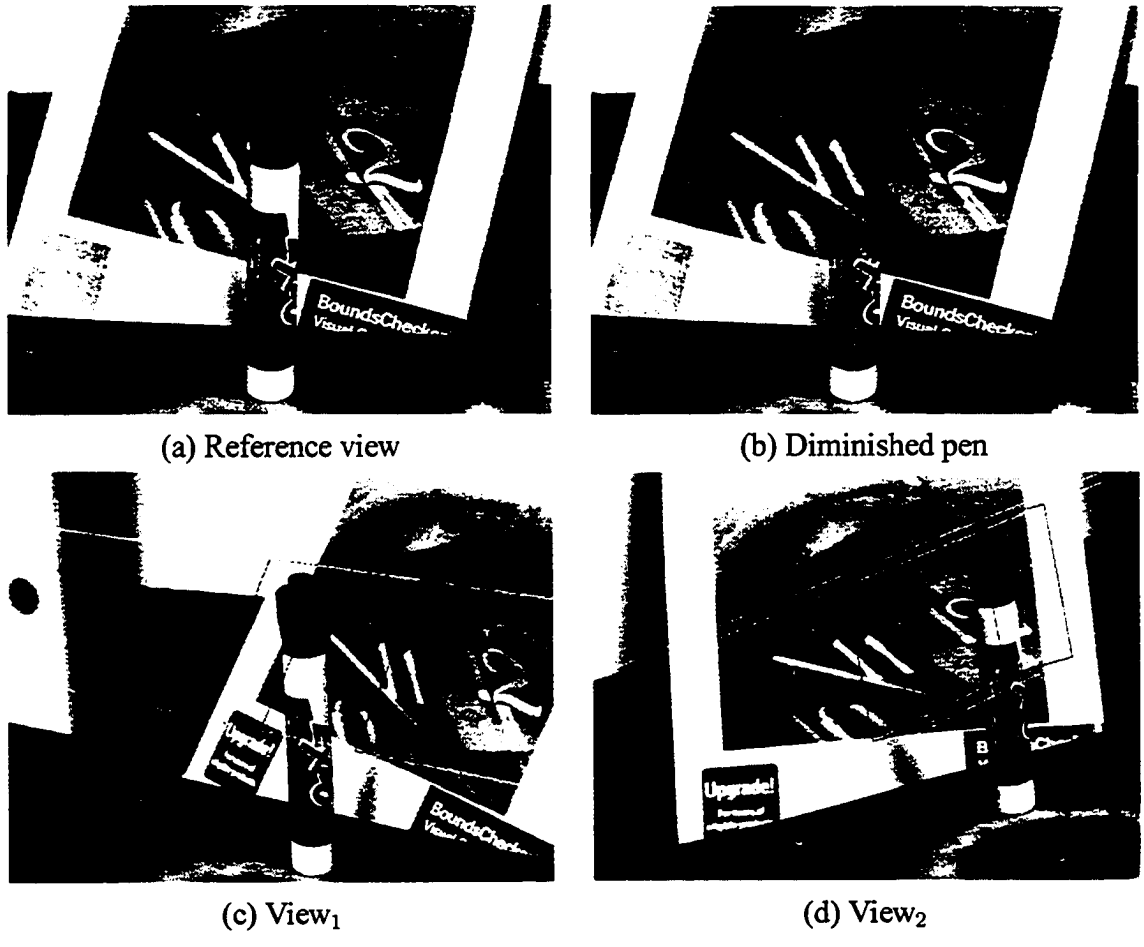
(a) Reference view (b) Diminished pen

(c) $View_1$ (d) $View_2$

Figure 5.9: Reference camera is approximately parallel to the background plane.

The diminished reality result is seamless. The pen is removed and the background is recovered with great continuity. In the second example, we demonstrate the case in which the paraperspective assumption is valid, but the background is not parallel to the image

plane, (Fig. 5.10). The reconstruction of the background is perfect after proper registration and the recovery of the plane orientation (Fig. 5.10(b)).



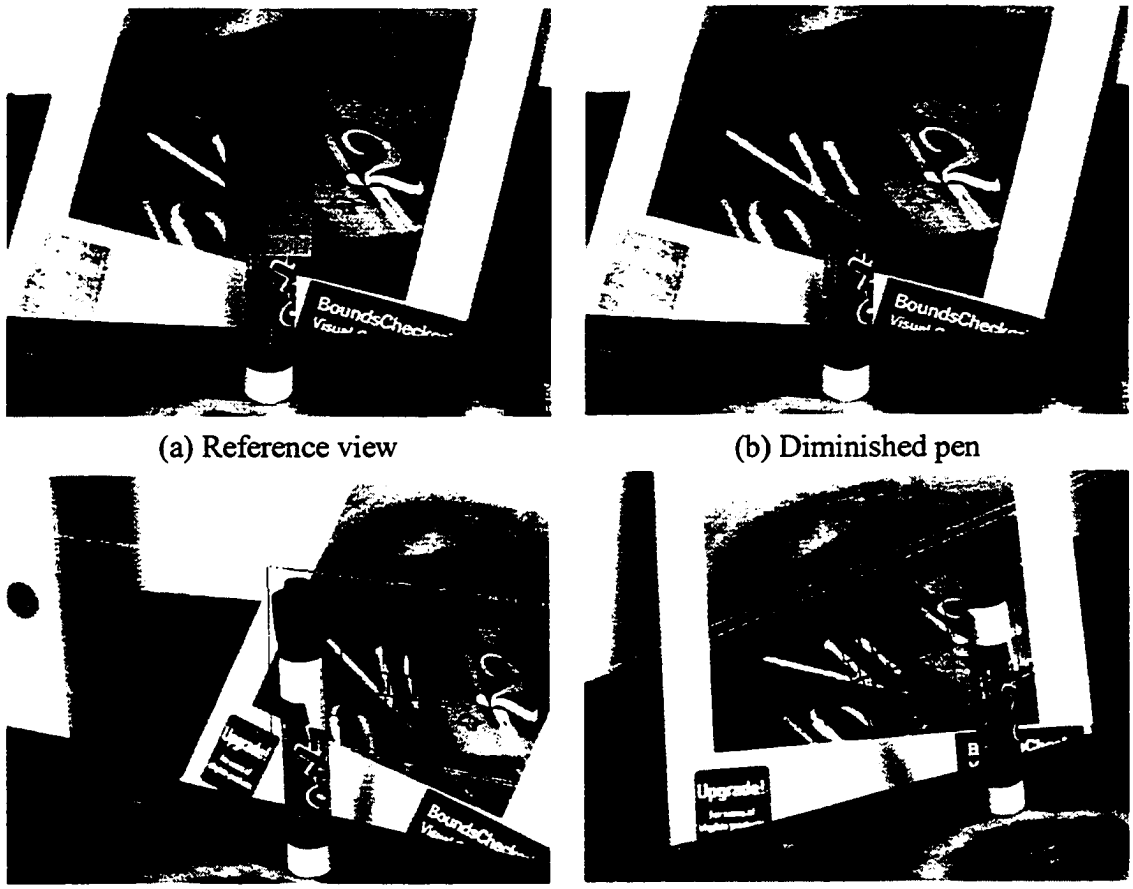
(a) Reference view (b) Diminished pen

(c) $View_1$ (d) $View_2$

Figure 5.10: Reference camera is not parallel to the background. The 3D orientation of the background plane is estimated through an additional image registration step.

In the last set of laboratory experiments presented here, we show that the piecewise (multi-resolution) paraperspective projection model can properly render non-planar backgrounds. Note that this is the general case and we do not assume anything about the background. We first applied this technique to the previous case of a planar background, which is non-parallel to the image plane, (Fig. 5.7).

As we can see in Fig. 5.11 (a) the background plane is reconstructed as a set of smaller



(a) Reference view (b) Diminished pen

(c) $View_1$ (d) $View_2$

Figure 5.11: Background is not parallel to the image plane.

planes at different depth (darker shades are far away and brighter shades are closer). The process of refinement is shown for a curved background surface in Fig. 5.12.

In Fig. 5.12(b), the recovered background is blurry, when we define the ROI as a single "*virtual plane*". Note that we do not use any feathering or weighted blending methods in Fig. 5.12. We render the background as an unweighted average of all warped views. As we subdivide the ROI into smaller tiles, we obtain better image reconstruction results. We applied this method to a background with large discontinuity, and obtained promising result even in the presence of large depth variation (Fig. 5.13).

At each level, the search space for depth becomes smaller. The only problem with this

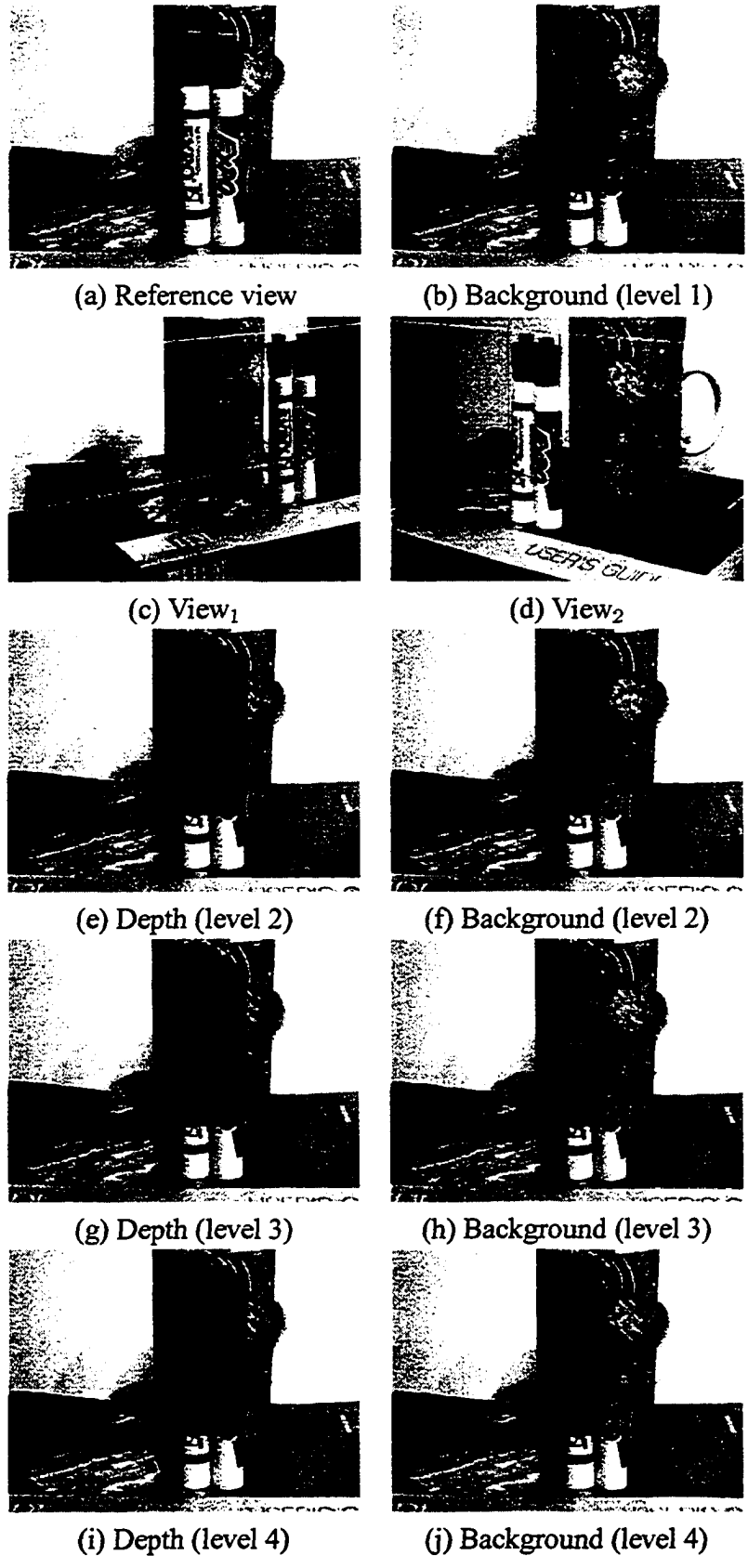


Figure 5.12: Rendering a non-planar (curved) background using piecewise paraperspective model.

approach is possibility of the error propagation. If a subdivision from a coarse level finds a wrong depth (for example due to lack of texture), the next level uses that depth as an initial guess and searches around a smaller neighborhood. Therefore it may not be able to recover the right depth. In order to solve this problem, we aim to use the values of the correlation scores as an indication of success. If the maximum correlation score for a level of subdivision is low, the next level will not reduce the search interval. We do not have a *prior* knowledge about the shape of the background unless it is fully reconstructed. Thus, the algorithm progressively goes from a crude estimation of the background by one virtual plane to very fine background by subdividing virtual plane as described in sections Section 5.5, Section 5.6, and Section 5.7. The algorithm checks if it can improve the correlation value in the next level else stops. Also, the user can intervene by entering the number of the levels. This way, the user trades between accuracy and execution time.

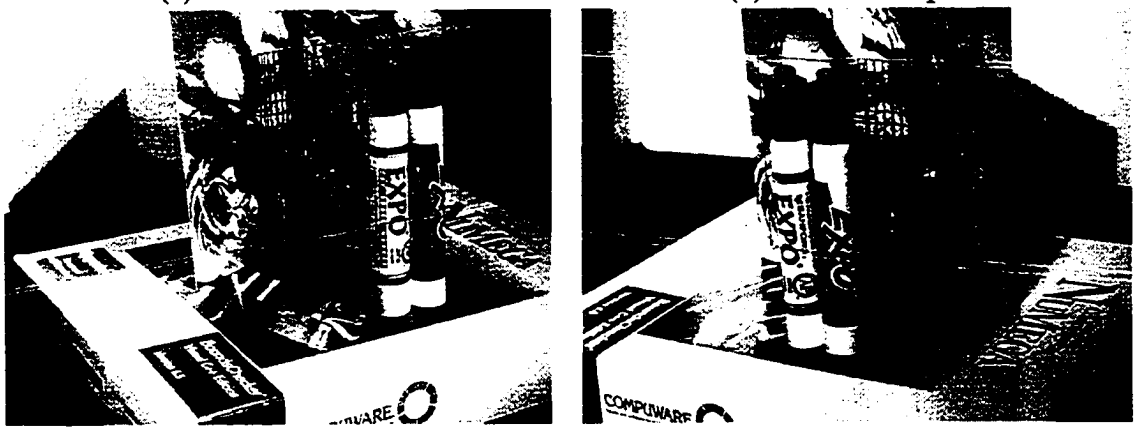
Finally, we have used two sets of calibrated images from a water distribution system and a power plant. In each of these cases, we have removed a piece of pipe. In the first case the background is planar (Fig. 5.1). In the second one the background is non-planar but the region of interest is small and far enough away for the paraperspective model to perform well (Fig. 5.14).

5.9. CONCLUSION

This section presents a well motivated and practical method for Diminished Reality. It uses multiple calibrated views of a scene to remove an object or region of interest and to replace it with the correct background. A series of methods are proposed to deal with planar and non-planar backgrounds. The algorithm uses the paraperspective projection model and has flexibility to recover a crude or fine detailed background. The diminished reality

results on images taken in a controlled laboratory environment show the performance of each of the proposed algorithms. The algorithm also is applied successfully to images from industrial scenes. We hope to see the results of such augmented and/or diminished reality research work used in the design of real revamping and modification procedures within the traditional industries.

(a) Reference view (b) Diminished pens



(c) $View_1$ (d) $View_2$

(c) Background depth (d) 3D reconstruction

Figure 5.13: Background violates planarity assumption. With piecewise paraperspective projection model, we have diminished the pens and rendered the correct background in the reference view.

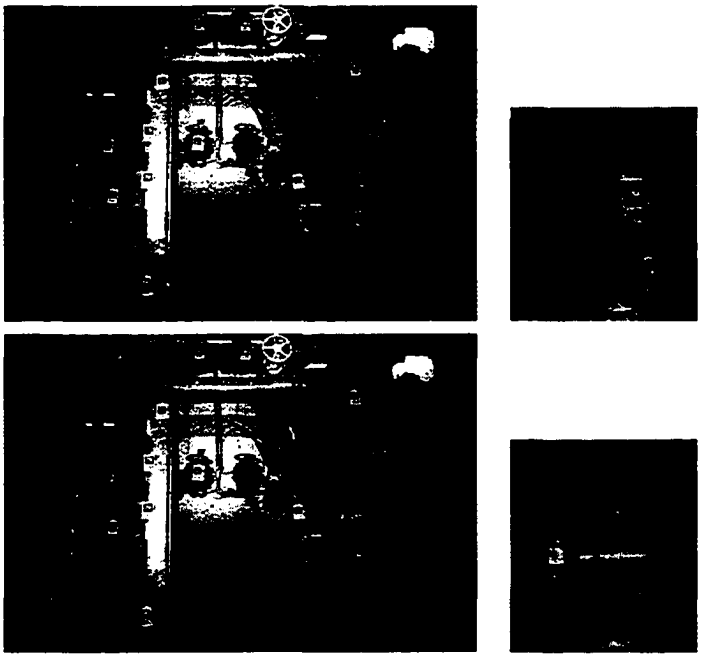


Figure 5.14: A part of the beam is removed and the pipe behind is clearly exposed.

Chapter 6

CONCLUSIONS

6.1. SUMMARY

This thesis has presented a hierarchical image registration algorithm to register any two digital images misaligned due to similar, affine, or perspective transformations. We have introduced four registrations techniques:

1. We extended the modified Levenberg-Marguardt nonlinear optimization algorithm to recover mild perspective transformations. Note that the scale change must not exceed 50%, and rotation must not exceed 45°. The modified Levenberg-Marguardt is faster than the standard version by eliminating the calculation of the Hessian matrix in each iteration. The method is capable of subpixel registration accuracy.

2. We introduced a piecewise affine registration to recover perspective parameters. In this method, we break the global perspective to small tiles under the assumption of first order Taylor approximation. This method can recover larger perspective deformations than the modified Levenberg-Marquardt algorithm. The images, however, must still have small rotation and scale changes.

3. We introduced a new approach based on log-polar transforms to handle similar (RST) transformations. Next, the algorithm couples the log-polar transform with a nonlinear least squares algorithm to estimate the perspective transformation parameters. Although the Fourier-Mellin transform also uses the log-polar transformation to recover rotation and scale, it is limited in use to small scale factors (<2.0). Larger scale changes induce too much distortion to the Fourier coefficients to be useful for affine recovery. Instead, our work operates directly in the spatial domain and recovers the best rotation, scale, and translation by performing correlation on tiles that have been transformed into log-polar space.

   The purpose of the log-polar registration module is to bring two images into alignment using only rotation, scale, and translation. This serves as a fine estimate for the subsequent perspective registration module based on nonlinear least squares optimization. That module, based on the Levenberg-Marquardt algorithm, offers subpixel precision. Coupling the two modules in this manner facilitates the registration of images in the presence of large-scale ($\leq 5\times$) affine transformations and moderate perspective transformations.

4. In this thesis, we extended these results for perspective registration by solving for radial line correspondences in log-polar space. This approach is motivated by the fact that the relationship between corresponding radial lines is the same for affine and perspective transformations in log-polar space. This motivates the hierarchical three-stage algorithm presented in this thesis. First, we find the global rotation, scale, and translation. Second, we find corresponding radial lines in both images by searching around the log-polar origin within a small radius. The purpose of using a small radius is to achieve reliable correlation in the log-polar domain with minimal skewing effect. Finally, we extend the search along the corresponding radial lines to account

for the additional skewing present due to possibly large perspective deformations. It is important to note that identifying the corresponding radial lines in our second step permits us to deal with the harder perspective problem because we can narrow our search to reduce false matches.

This thesis has contributed to our understanding of image registration in the presence of large perspective distortions. Although the rotation- and scale-invariant properties of the log-polar transform have been used in the literature for various applications, its proposed use in this thesis is novel, particularly for the challenging perspective registration problem considered here. We demonstrated the registration techniques in several computer vision applications including image mosaic, video mosaic, glare removal, and diminished reality.

6.2. FUTURE WORK

There are several points which may be investigated in order to extend the results presented in this thesis and apply them to different domains.

- Image resampling poses the largest bottleneck for the log-polar or Levenberg-Marquardt optimization modules. Currently, our image resampling function is implemented in software. With the advent of fast graphic cards and specialized GPUs, we will investigate the use of hardware resampling for improving the performance of our algorithms.

- Currently, correlation in the log-polar domain consists of sliding a cropped window in raster order. It is worthwhile to examine whether this process may be accelerated by positioning the sliding window on areas of high information content. Entropy, variance, or other statistically discriminating techniques can be used to quantify in-

formation content. This can be applied directly in the spatial domain or in the log-polar domain. The benefits of this approach in either domain must be investigated.

- The work proposed in this thesis has been demonstrated on 2D images. Due to the recent emergence of laser range scanners, 3D images consisting of range data (3D point clouds) are becoming increasingly available and popular. These range images need to be registered to create the model of large-scale objects, such as buildings. A fast and robust 3D-to-3D image registration technique is desired. We would like to extend the results of the log-polar 2D registration to 3D range images.

Appendix A

A.1. THE LINEAR LEAST-SQUARES METHOD

Consider a set of N data points, $(\mathbf{x}_i, y_i)$, for $i = 1, 2, ..., N$, where $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{im})$ is an m-dimensional coordinate and y_i is a scalar value. We seek to fit a linear model to the data points:

$$y = \sum_{i=1}^{m} a_i x_i \tag{A.1}$$

where $\mathbf{a} = (a_1, a_2, ..., a_m)$ are the parameters of the model. Note that we may accommodate a constant term a_{m+1} by representing $\mathbf{x}$ in homogeneous coordinates, i.e., $\mathbf{x} = (x_1, x_2, ..., x_m, 1)$. For the purpose of this presentation, it is sufficient to let $\mathbf{x}$ and $\mathbf{a}$ retain m components. The set of N data points gives rise to the following system of linear equations:

$$\begin{bmatrix} y_1 \\ y_2 \\ .. \\ .. \\ y_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ ..... & & & \\ ..... & & & \\ x_{N1} & x_{N2} & \cdots & x_{Nm} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ .. \\ .. \\ a_m \end{bmatrix} \tag{A.2}$$

If $N = m$, then it is possible to solve the linear system of equations for the unknown parameters such that the model will pass through (interpolate) the data points. In practice, though, we often have more equations than unknowns ($N > m$) and the system of equations is overdetermined. In that case, the model is not of sufficiently high order to satisfy all the constraints and guarantee interpolation through all N data points. Consequently, errors will exist between the data points and the model, and a least-squares technique must be applied to compute an approximate solution. Our objective now becomes to determine the coefficients of the model that minimize the sum of the squared error:

$$\begin{aligned} \chi^2(\mathbf{a}) &= \sum_{i=1}^{N} e_i^2 \\ &= \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{m} a_j x_{ij} \right]^2 \end{aligned} \tag{A.3}$$

This is achieved by determining the partial derivatives of χ^2 with respect to coefficients a_k, and setting them to zero. This yields

$$\frac{\partial \chi^2(\mathbf{a})}{\partial a_k} = -2 \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{m} a_j x_{ij} \right] x_{ik} = 0 \tag{A.4}$$

where $k = 1, 2, ..., m$. We thus have

$$\sum_{i=1}^{N} \left[a_1 x_{i1} + a_2 x_{i2} + \cdots + a_m x_{im} \right] x_{ik} = \sum_{i=1}^{N} y_i x_{ik} \tag{A.5}$$

$$\begin{bmatrix} x_{11} & x_{21} & \cdots & x_{N1} \\ x_{12} & x_{22} & \cdots & x_{N2} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ x_{1m} & x_{2m} & \cdots & x_{Nm} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ x_{N1} & x_{N2} & \cdots & x_{Nm} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ .. \\ .. \\ a_m \end{bmatrix} = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{N1} \\ x_{12} & x_{22} & \cdots & x_{N2} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mN} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ .. \\ .. \\ y_N \end{bmatrix}$$

This expression may be written in matrix notation as

$$(\mathbf{X}^T\mathbf{X})\mathbf{A} = \mathbf{X}^T\mathbf{Y} \tag{A.6}$$

Since $(\mathbf{X}^T\mathbf{X})$ is a square matrix, it is readily invertible. The solution for $\mathbf{A}$ now becomes

$$\begin{aligned} \mathbf{A} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \\ &= \mathbf{X}^+\mathbf{Y} \end{aligned} \tag{A.7}$$

This technique is known as the *pseudoinverse solution* of the linear least-squares problem, and $\mathbf{X}^+$ is called the *pseudoinverse matrix*.

Appendix B

B.1. PARAPERSPECTIVE MODEL

Assume we have an ideal pin-hole camera, see Fig. B.1. It contains a plane I, known as the retinal or image plane, upon which the image is formed. The image of the 3-D point P undergoes a perspective projection as it passes through the optical center O and is projected to point p on plane I. The distance between O and the normal to I is called the focal length. The line passing through O and lying along the normal to R is called the optical axis. The optical axis intersects the image plane at the principal point.

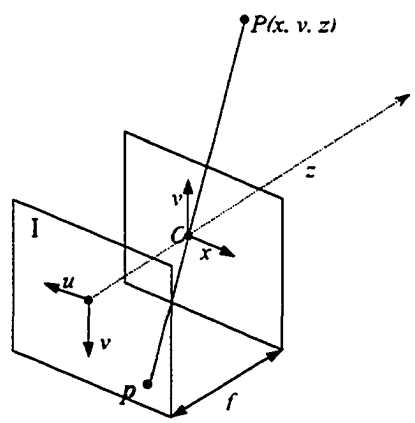


Figure B.1: Pinhole camera model.

Consider the origin of the 3-D reference frame to be at O, and its z-axis to be along

the optical axis. Let (u, v) be the 2-D coordinates of p and (x, y, z) the 3-D coordinates of P. Through the use of similar triangles, it can easily be seen that the following equations hold.

$$\begin{aligned} u &= \frac{fx}{z} \\ v &= \frac{fy}{z} \end{aligned}$$

The above mapping is conveniently written in matrix form:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Commonly, the origin of the image reference frame does not lie at principal point. We shall find it useful to move the origin to the upper left corner of the image. We can account for this shift, as well as nonuniform scaling, by performing a coordinate transformation on the image and rewriting the camera mapping as:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fk_u & fk_\theta & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where k_u and k_v are scaling factors along u and v, respectively, and (u_0, v_0) is the location of the principal point in the image plane. The additional parameter k_θ gives the skew between axes. For most CCD cameras $k_\theta = 0$.

The paraperspective projection model is a simplification of the standard perspective model. When the 3-D world frame is not identical to the camera frame, we consider that the 3-D points undergo a rigid transformation.

$$\lambda \mathbf{p} = \mathbf{A} \begin{bmatrix} & \mathbf{R} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{P}$$

The matrix $\mathbf{R}$ is a 3 rotation matrix and the 3×1 vector $\mathbf{t}$ contains the coordinates of the origin of the world frame expressed in the camera frame: $\lambda \mathbf{p} = \mathbf{A}[\mathbf{R} - \mathbf{R}\mathbf{t}]\mathbf{P}$ or $\lambda \mathbf{p} = \mathbf{AMP}$. $\mathbf{A}$ is a 3×3 matrix that depends on the intrinsic parameters, while $\mathbf{M}$ depends on the extrinsic parameters. The variable λ represents the depth of the object point. This mapping is non-linear due to the scaling factor λ. $u = \frac{f\mathbf{R}_1 \cdot (\mathbf{P}-\mathbf{t})}{\mathbf{R}_3 \cdot (\mathbf{P}-\mathbf{t})} + u_0$ and $v = \frac{f\mathbf{R}_2 \cdot (\mathbf{P}-\mathbf{t})}{\mathbf{R}_3 \cdot (\mathbf{P}-\mathbf{t})} + v_0$. This yields the value for $\lambda = \frac{\mathbf{R}_3 \cdot (\mathbf{P}-\mathbf{t})}{f}$. Assume the principle point, also called image center, to be at $(0, 0)$, and the reference point $\mathbf{P}_0$ and the rest of the visible points $\mathbf{P}$in the scene to be close to $\mathbf{P}_0$.

$$\mathbf{p} = \frac{1}{\lambda}\mathbf{Q} \cdot (\mathbf{P} - \mathbf{t}), \quad \mathbf{Q} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \end{bmatrix}$$

$\mathbf{p}_0$ is the image of reference point $\mathbf{P}_0$. The depth of $\mathbf{P}_0$ is given by $\lambda_0 = \mathbf{R}_3 \cdot (\mathbf{P}_0 - \mathbf{t})/f$. The depth of $\mathbf{P}$ is given by $\lambda = \lambda_0 + \Delta\lambda$, where $\Delta\lambda = \mathbf{R}_3 \cdot (\mathbf{P} - \mathbf{P}_0)/f$.

$$\begin{aligned} \mathbf{p} &= \frac{1}{\lambda}\mathbf{Q} \cdot (\mathbf{P} - \mathbf{t}) \\ &= \frac{1}{\lambda_0(1 + \frac{\Delta\lambda}{\lambda_0})}(\mathbf{Q} \cdot (\mathbf{P}_0 - \mathbf{t}) + \mathbf{Q} \cdot (\mathbf{P} - \mathbf{P}_0)) \end{aligned}$$

To a first-order approximation, we have $\frac{1}{1+\frac{\Delta\lambda}{\lambda_0}} \approx 1 - \frac{\Delta\lambda}{\lambda_0}$ This yields the following approx-

imation:

$$\begin{aligned}\mathbf{p} &\approx \frac{1}{\lambda_0}(1-\frac{\Delta\lambda}{\lambda_0})(\mathbf{Q}\cdot(\mathbf{P}_0-\mathbf{t})+\mathbf{Q}\cdot(\mathbf{P}-\mathbf{P}_0)) \\ &= \frac{f}{\lambda_0}(\mathbf{Q}\cdot(\mathbf{P}_0-\mathbf{t})+\mathbf{Q}\cdot(\mathbf{P}-\mathbf{P}_0)-\frac{\Delta\lambda}{\lambda_0}\mathbf{Q}\cdot(\mathbf{P}_0-\mathbf{t})-\frac{\Delta\lambda}{\lambda_0}\mathbf{Q}\cdot(\mathbf{P}-\mathbf{P}_0))\end{aligned}$$

If $\mathbf{P}$ *lies near the reference point* $\mathbf{P}_0$, *then* $\frac{\Delta\lambda}{\lambda_0}\mathbf{Q}\cdot(\mathbf{P}-\mathbf{P}_0)\approx 0$.

$$\begin{aligned}\mathbf{p} &= \frac{1}{\lambda_0}(\mathbf{Q}\cdot(\mathbf{P}_0-\mathbf{t})+\mathbf{Q}\cdot(\mathbf{P}-\mathbf{P}_0)-\frac{\Delta\lambda}{\lambda_0}\mathbf{Q}\cdot(\mathbf{P}_0-\mathbf{t}) \\ \mathbf{p} &= \mathbf{p}_0+\frac{1}{\lambda_0}(\mathbf{Q}-\mathbf{p}_0\mathbf{R}_3)\cdot(\mathbf{P}-\mathbf{P}_0) \\ \mathbf{p}-\mathbf{p}_0 &= \frac{1}{\lambda_0}[\mathbf{I}_2-\mathbf{p}_0]\mathbf{R}(\mathbf{P}-\mathbf{P}_0)\end{aligned}$$

Letting $\tilde{\mathbf{p}}=\mathbf{p}-\mathbf{p}_0$ and $\tilde{\mathbf{P}}=\mathbf{P}-\mathbf{P}_0$, we have

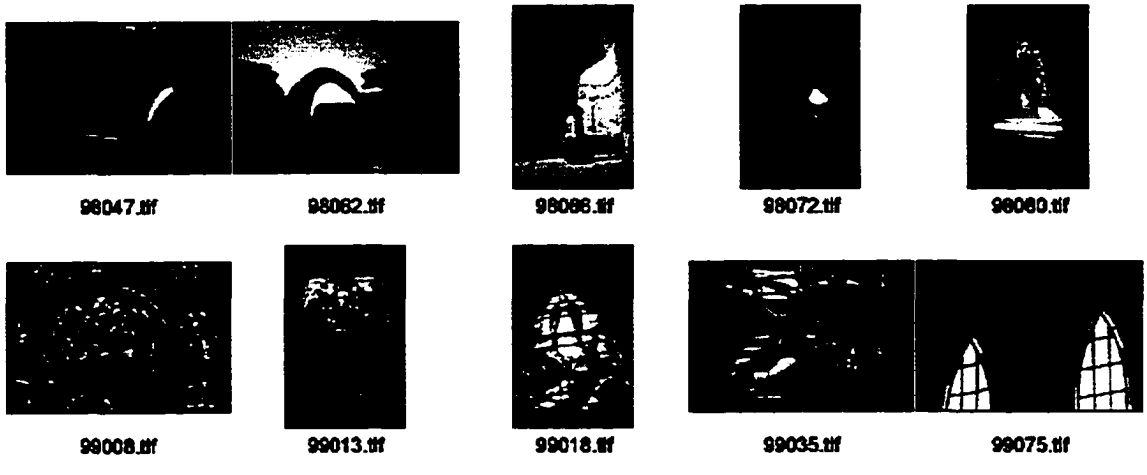
$$\tilde{\mathbf{p}} = \frac{1}{\lambda_0}[\mathbf{I}_2-\mathbf{p}_0]\mathbf{R}\tilde{\mathbf{P}}$$

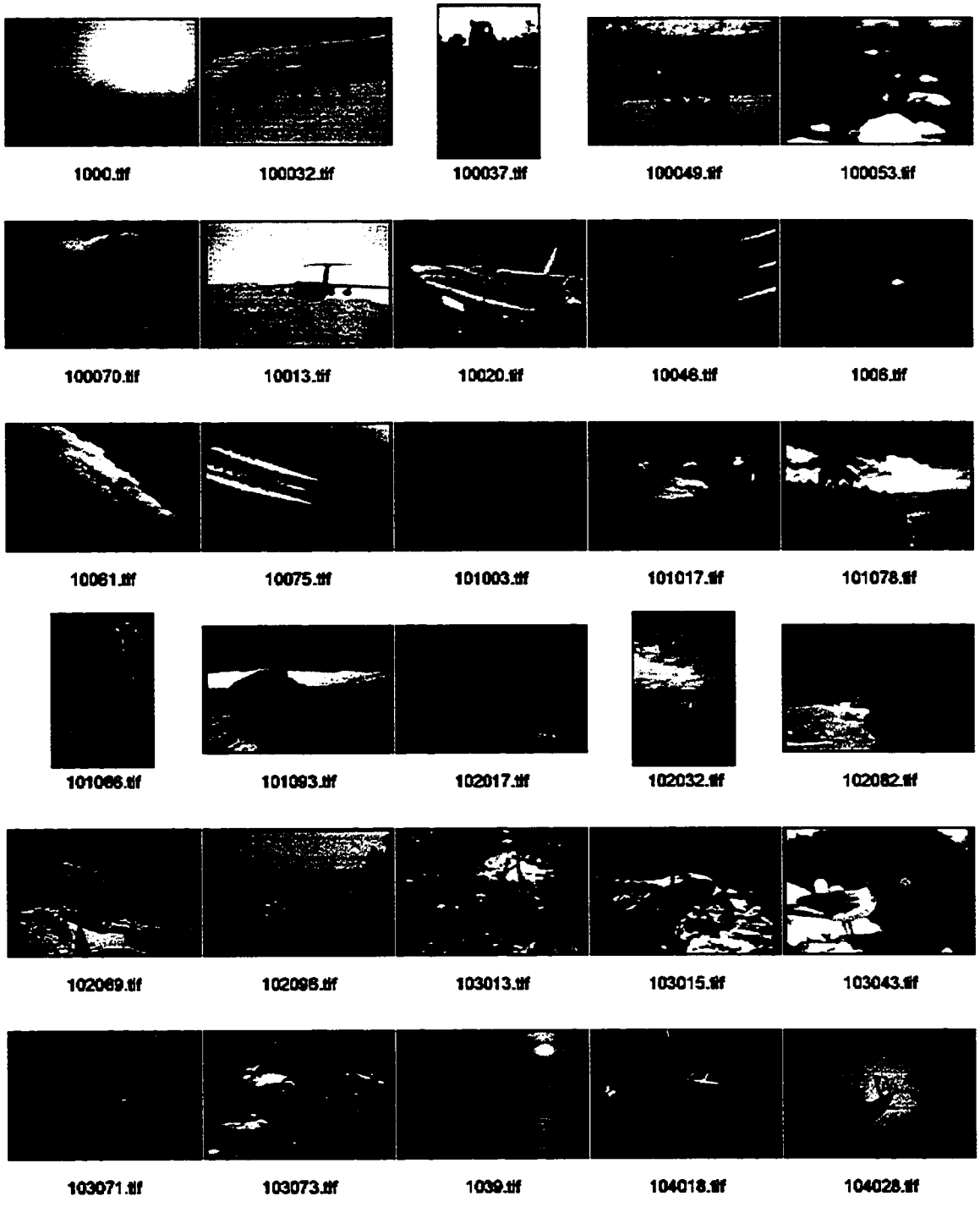
The above equation represents points on a plane with the orientation of $\mathbf{R}$ and the depth of λ_0. This means that all the 3D points are first projected onto this plane and then viewed by the perspective camera.

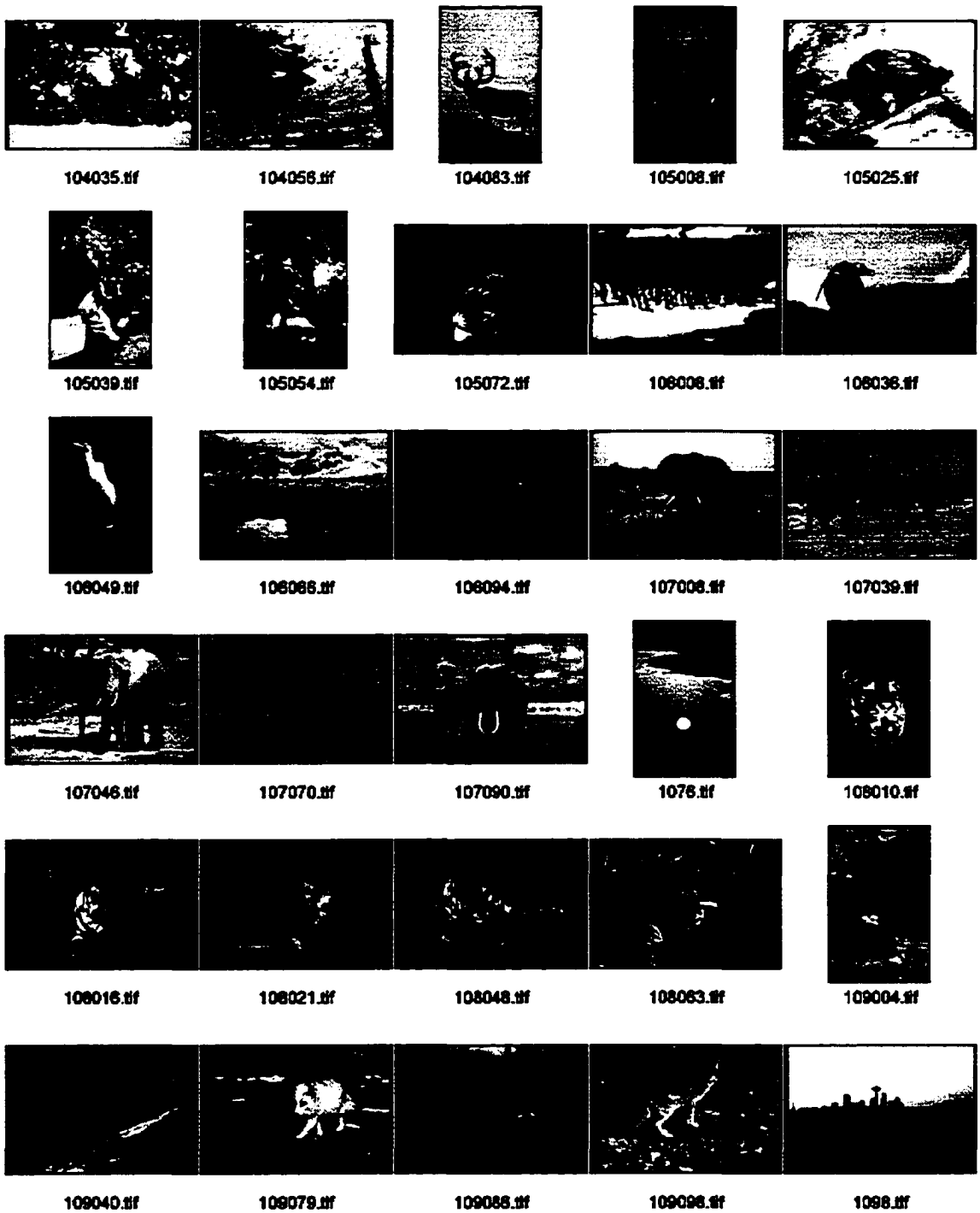
Appendix C

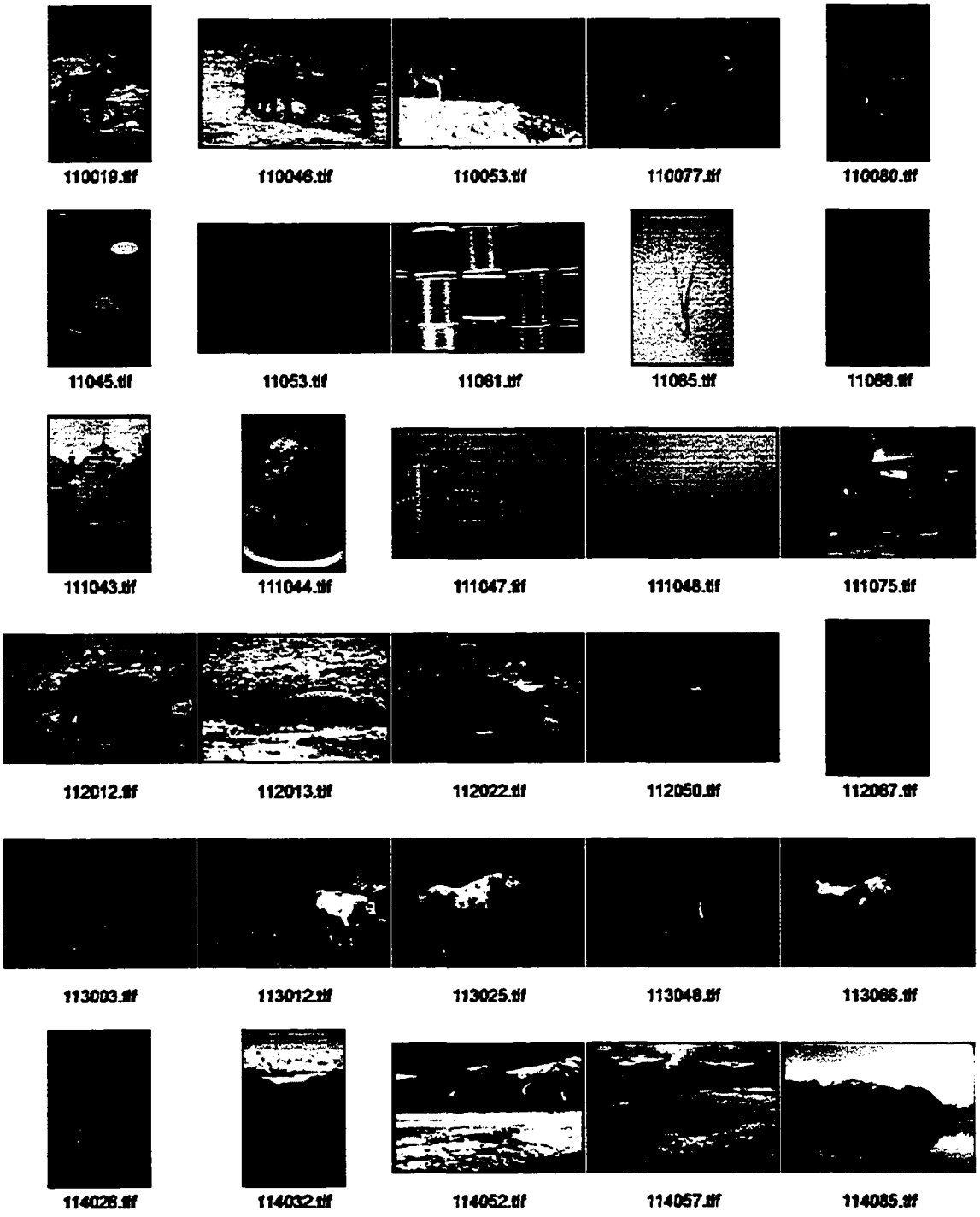
C.1. COREL STOCK PHOTO LIBRARY

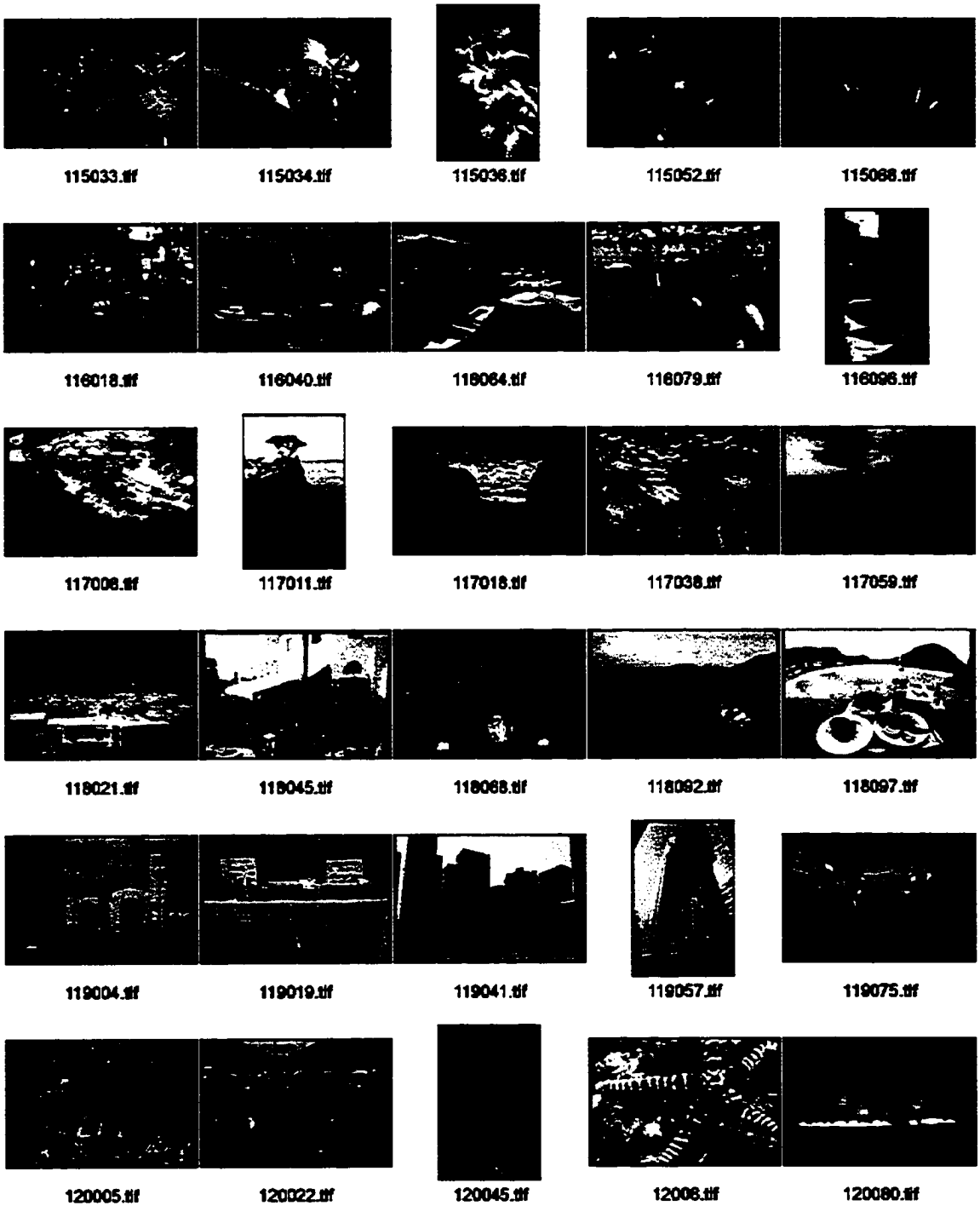
The Corel Stock Photo Library contains 20,000 royalty-free photographic images on 200 CD-ROMs (http://www.corel.com/). We have chosen five images randomly from each CD. The thumbnails of 1000 photos that have been used in the experiment are shown below. This listing is intended to show the classes of images to which the registration algorithm was applied.

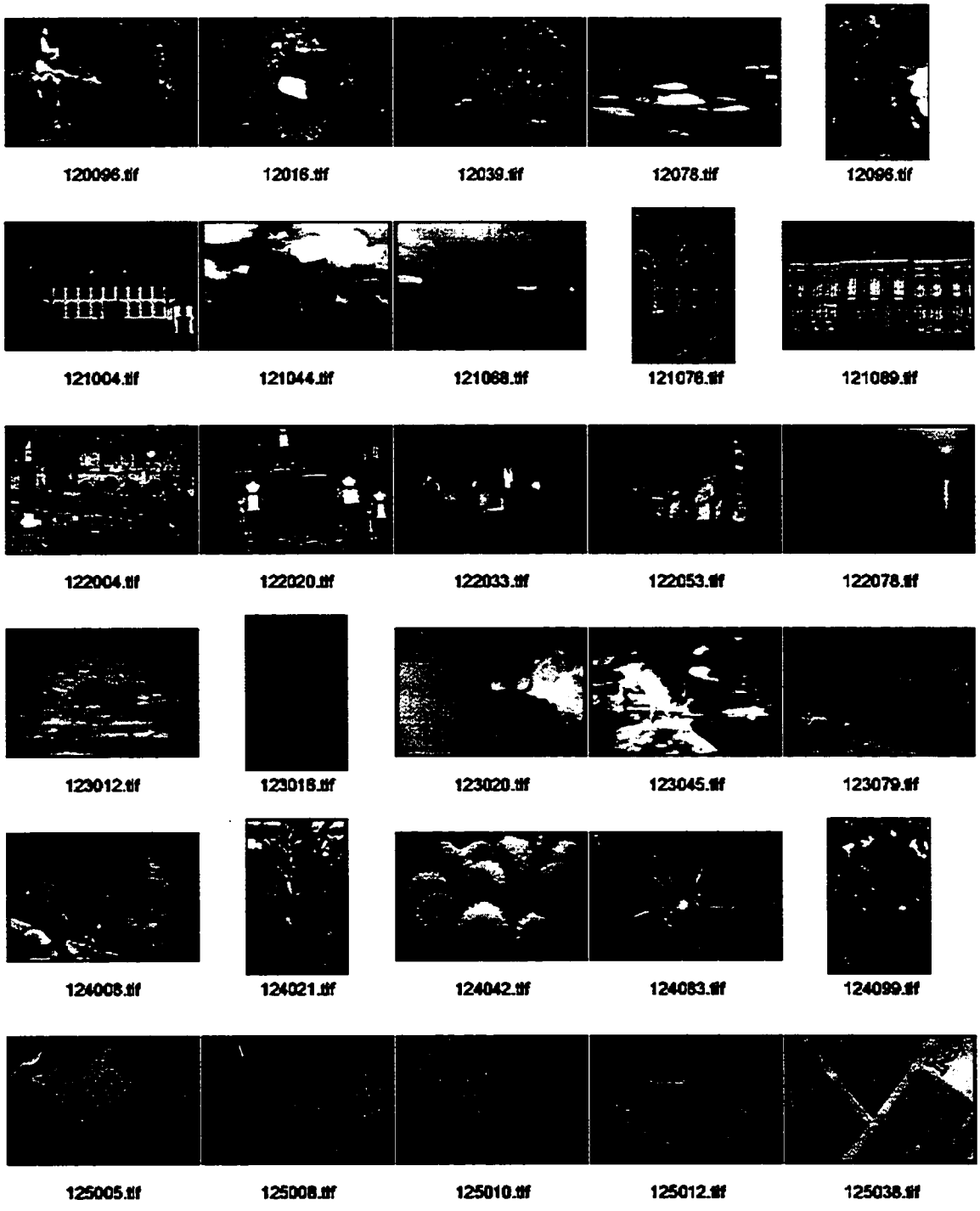


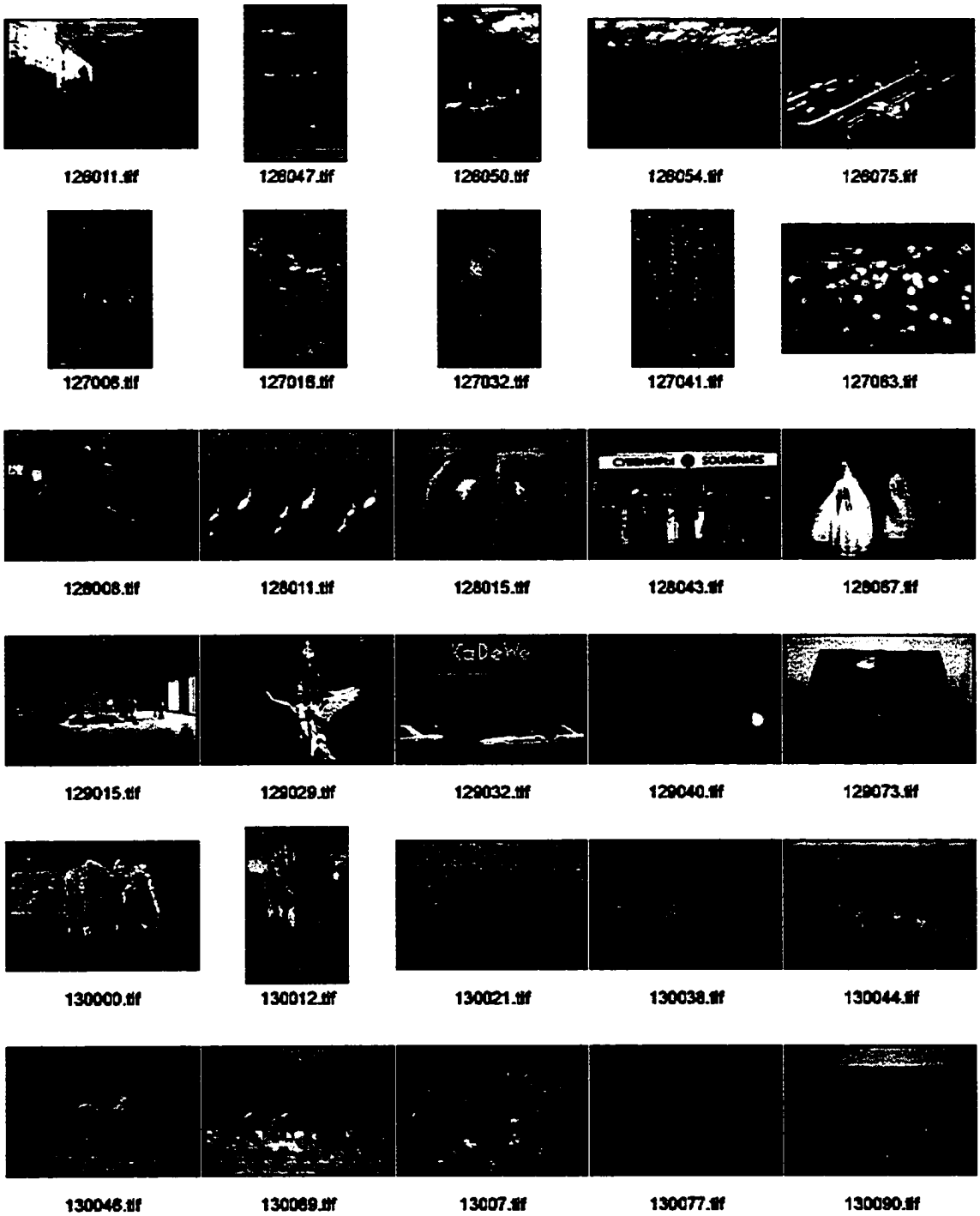


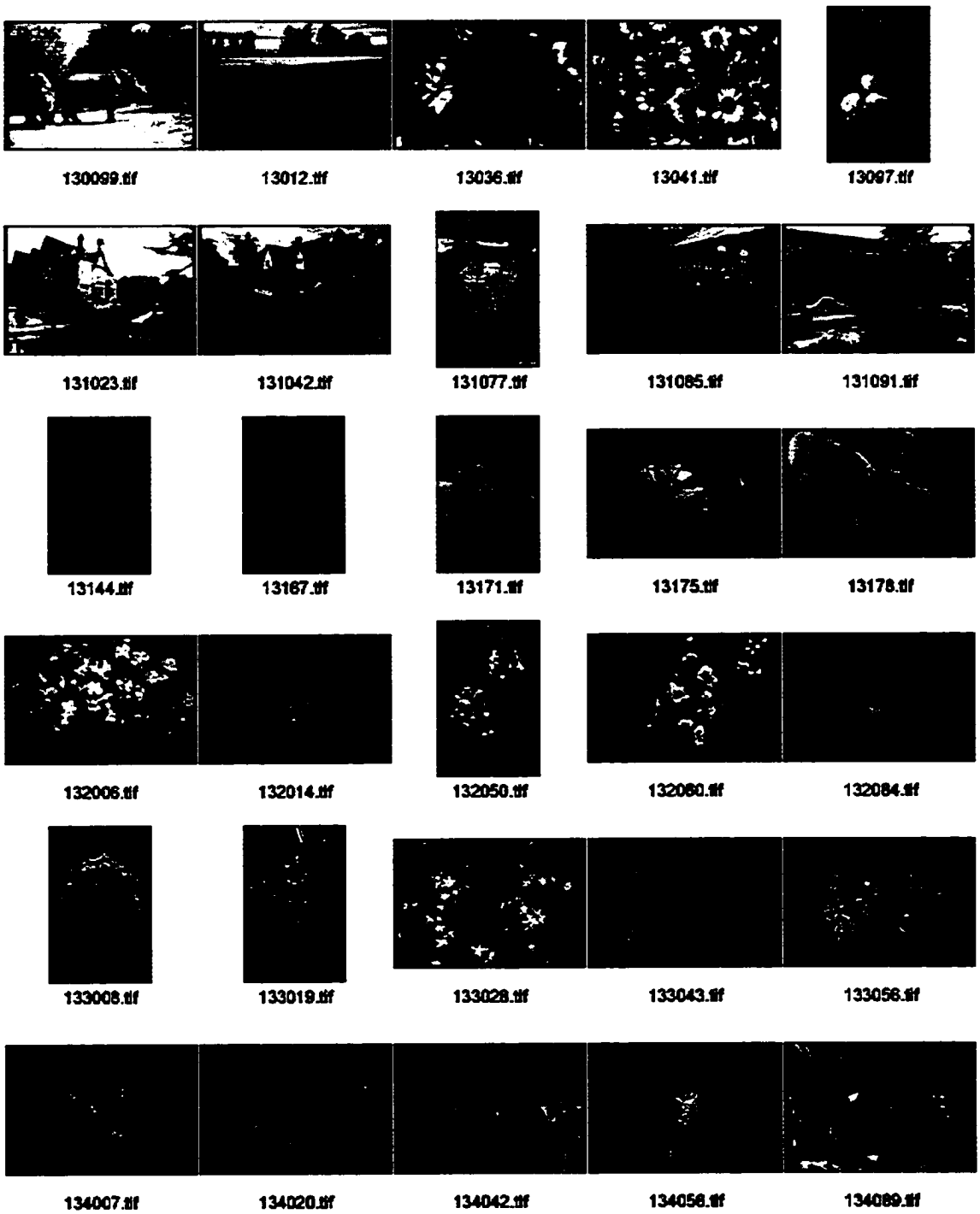


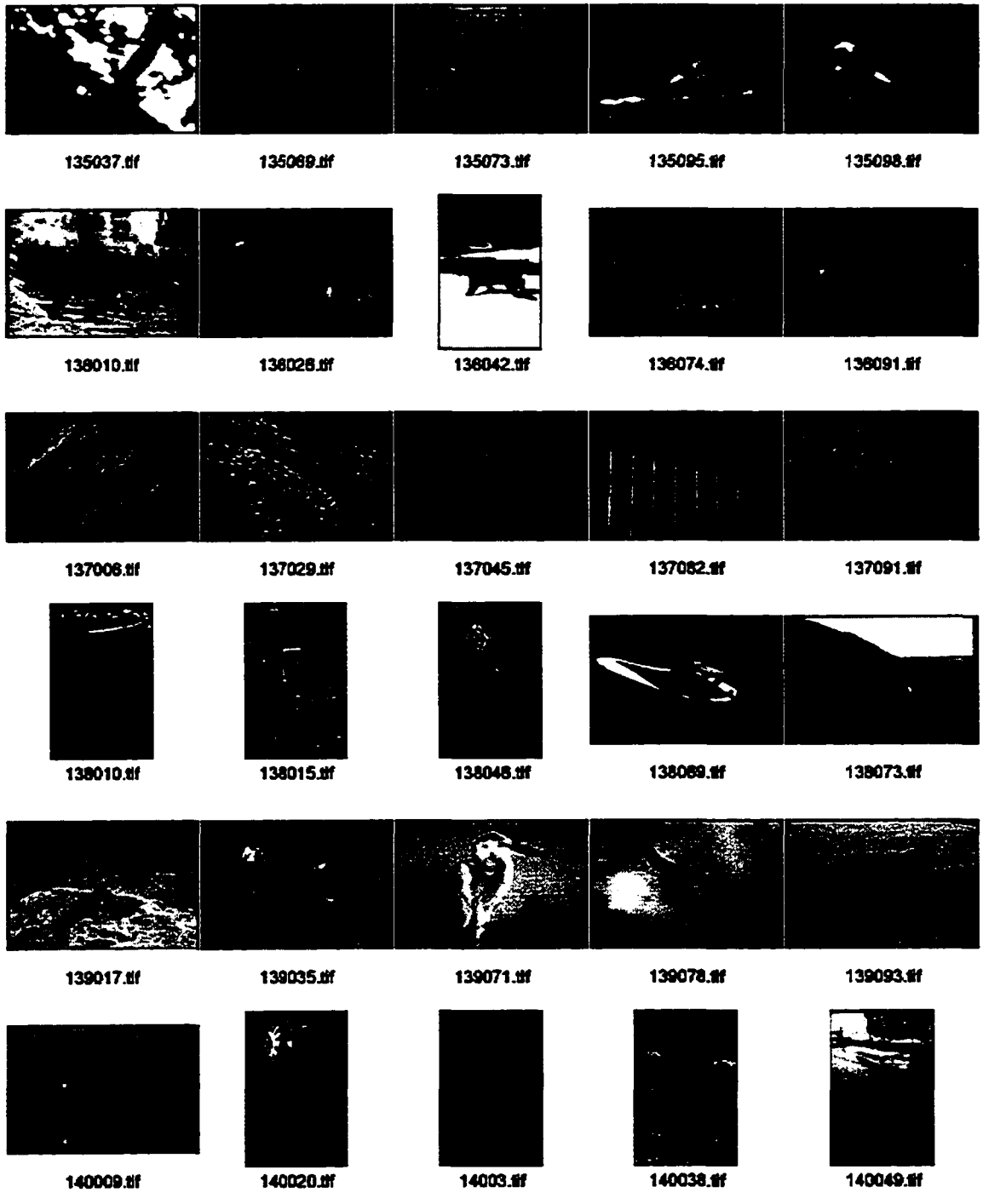


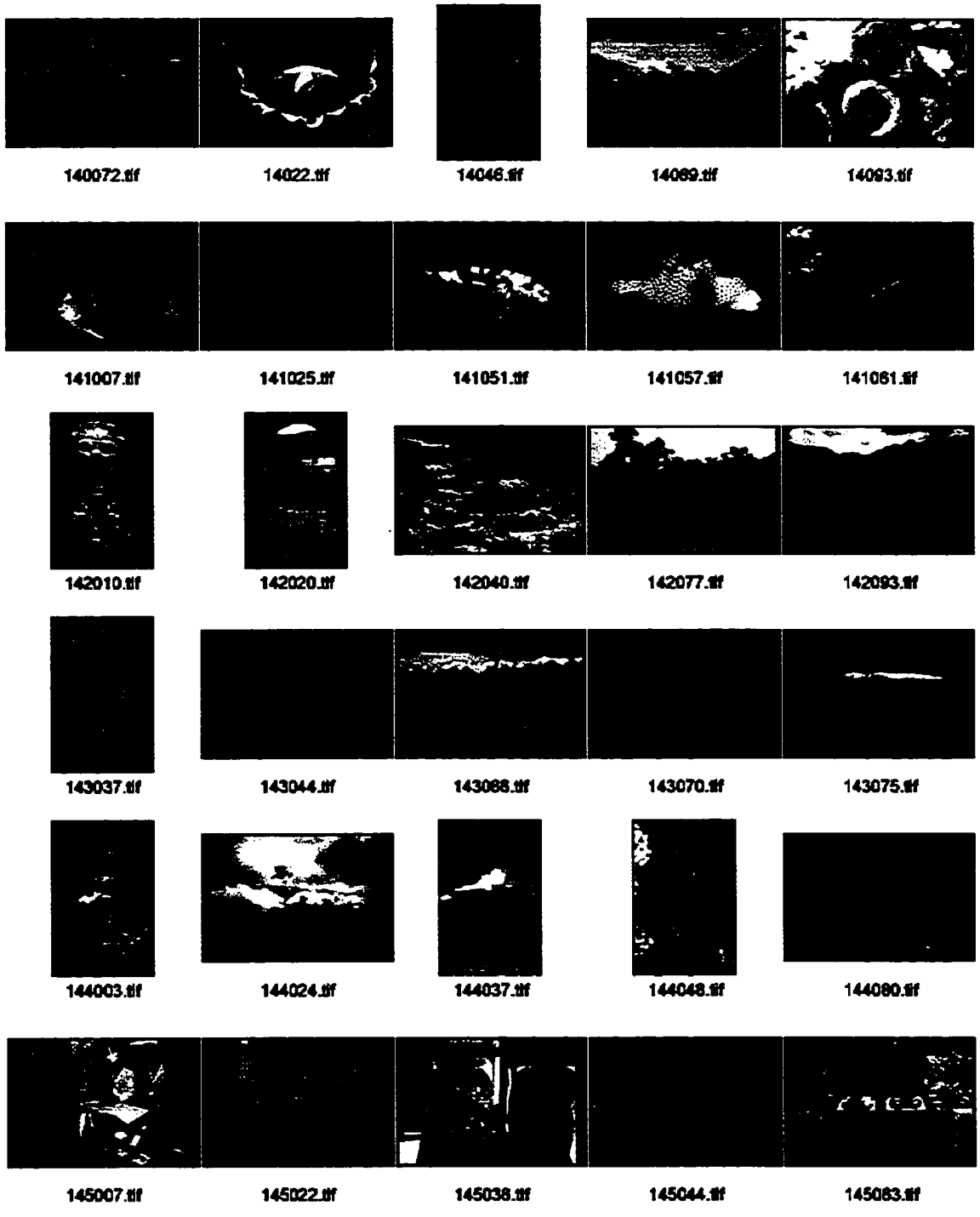


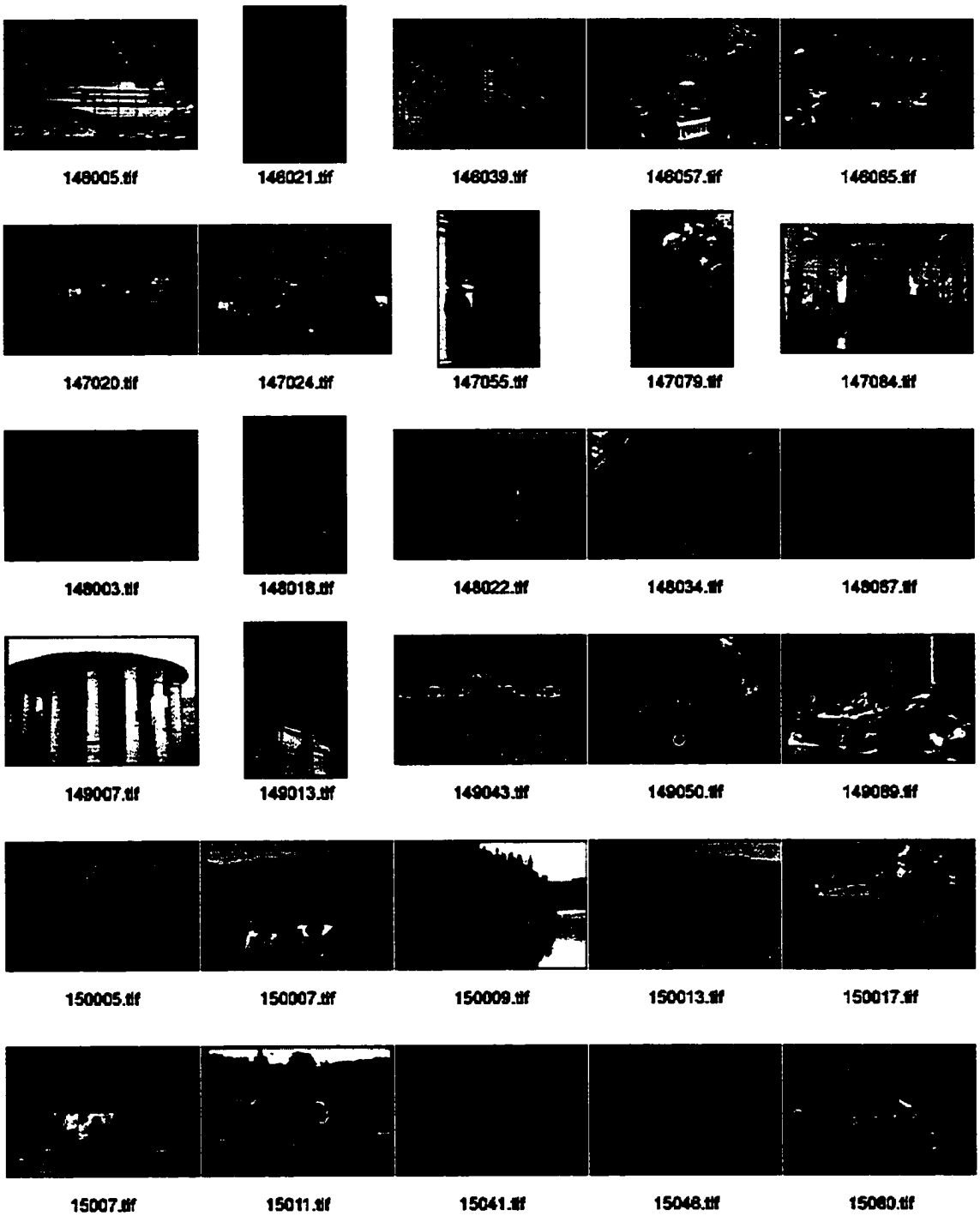


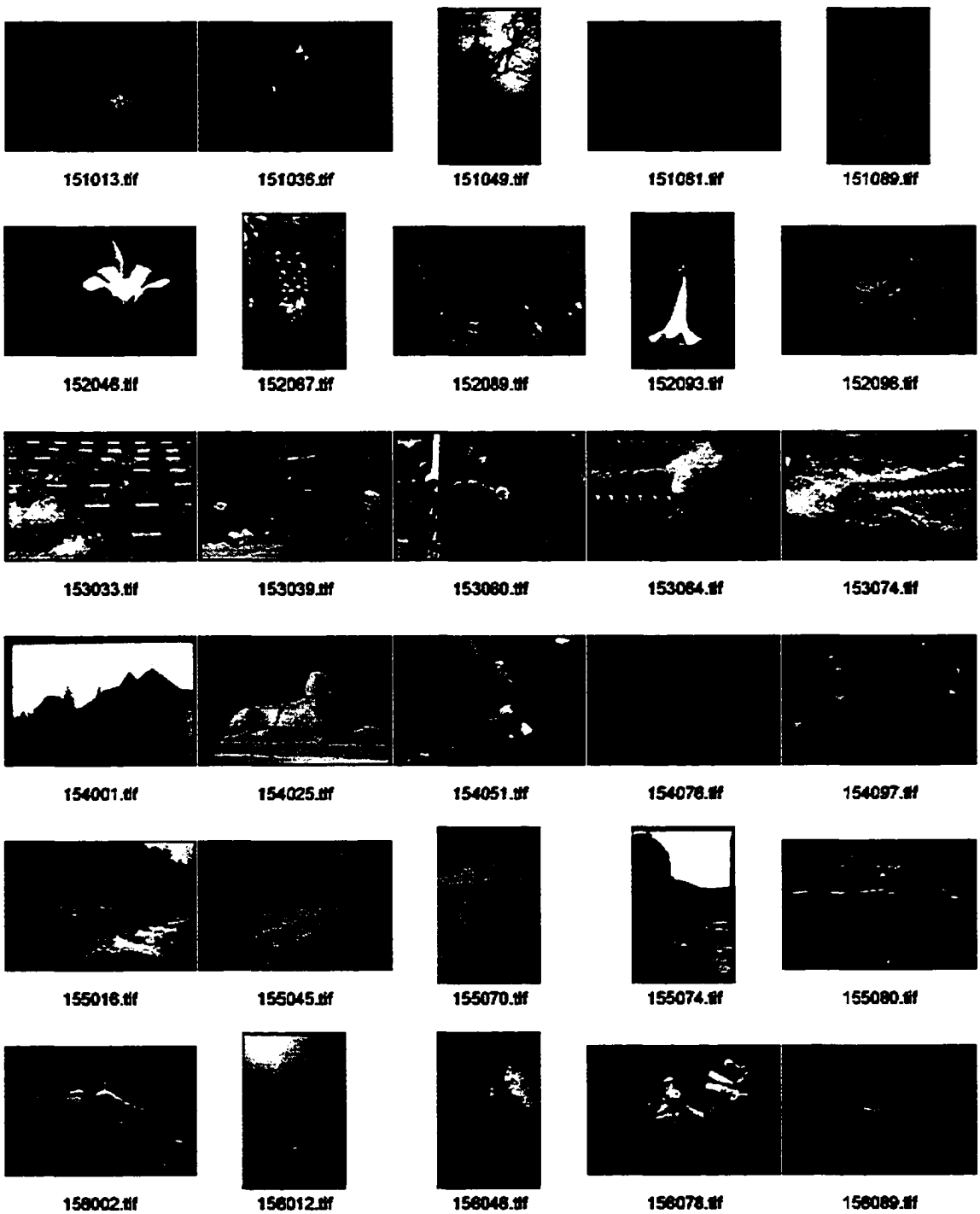


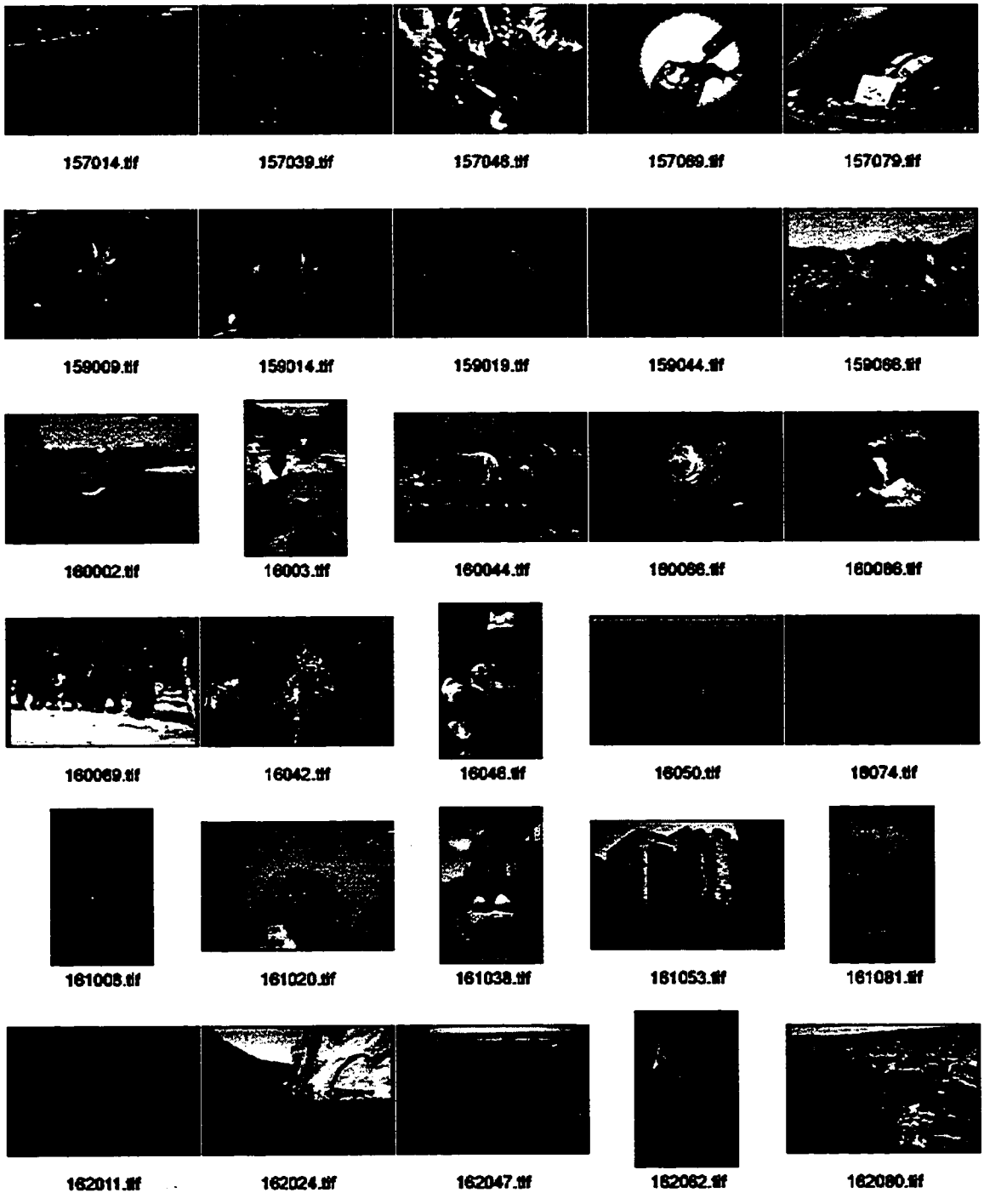


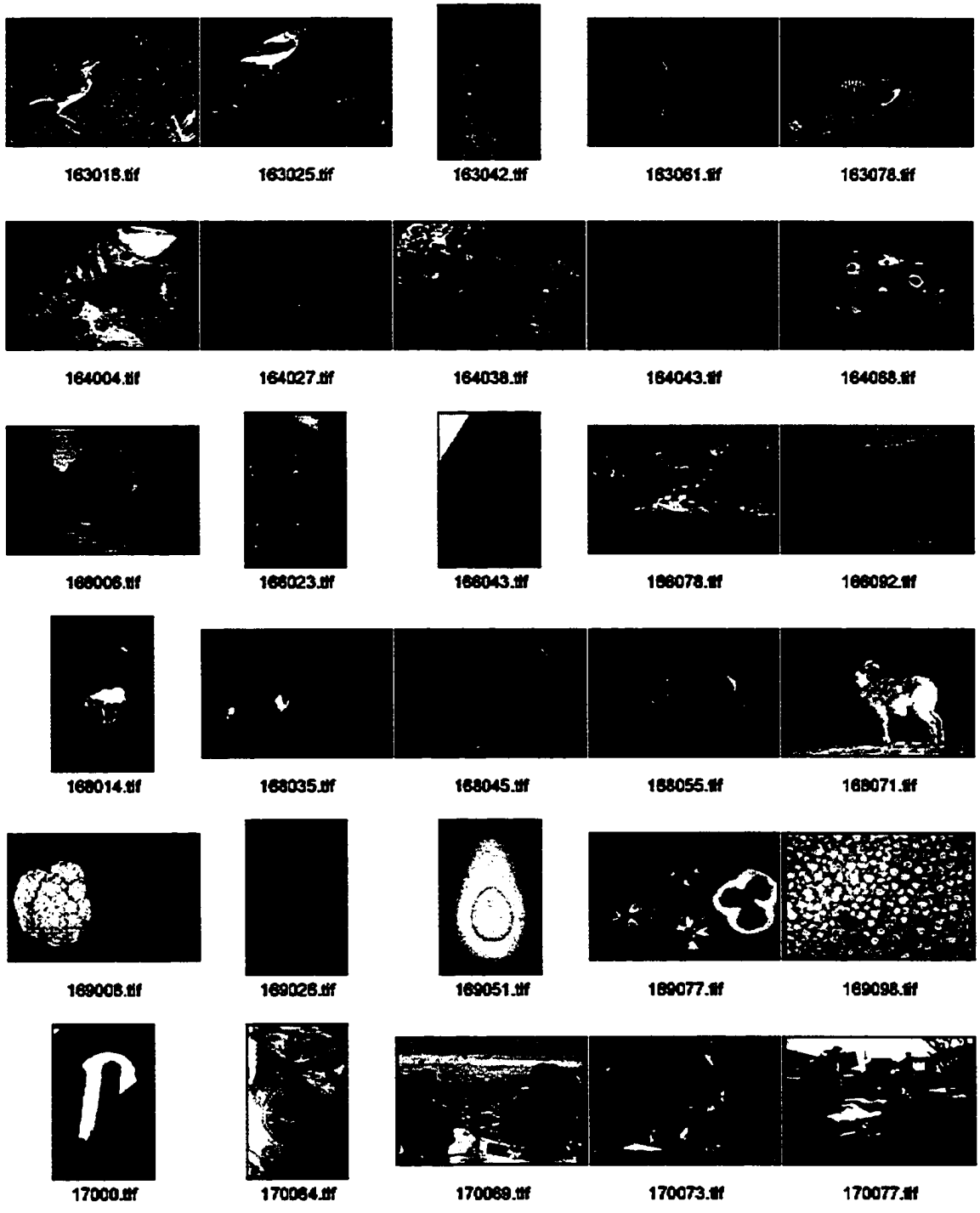












170082.tif 17021.tif 17036.tif 17062.tif 17093.tif

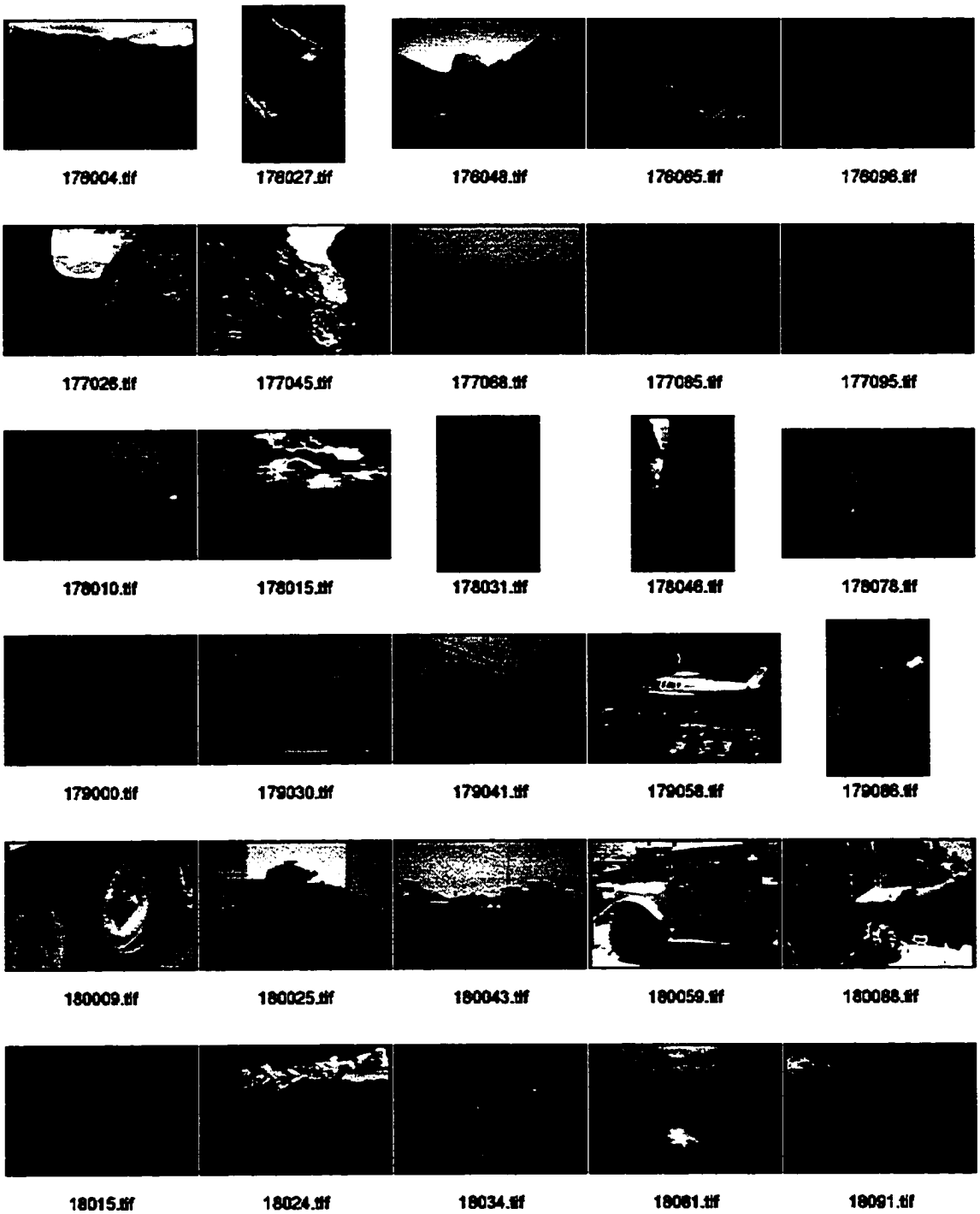
171001.tif 171004.tif 171013.tif 171025.tif 171089.tif

172004.tif 172025.tif 172042.tif 172076.tif 172092.tif

173007.tif 173017.tif 173062.tif 173071.tif 173084.tif

174005.tif 174008.tif 174014.tif 174041.tif 174095.tif

175003.tif 175018.tif 175028.tif 175036.tif 175073.tif



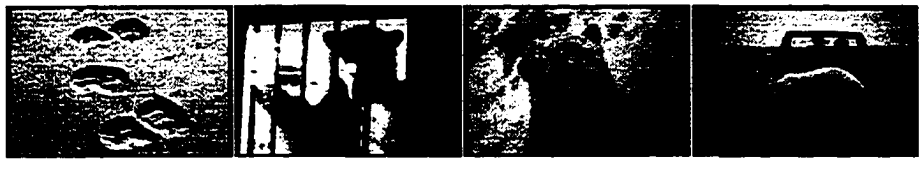

181004.tif


181024.tif

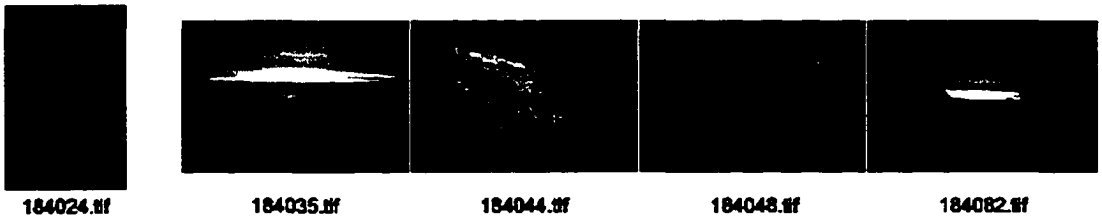

181039.tif

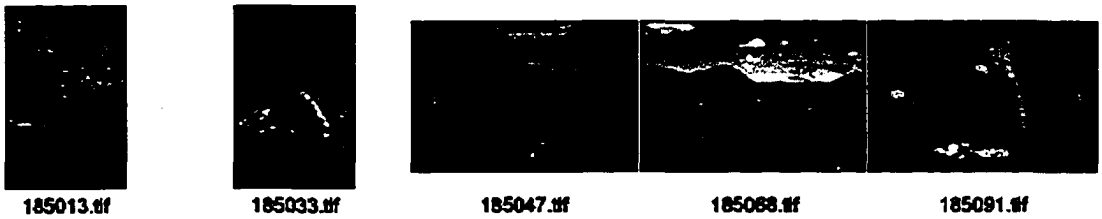

181050.tif


181066.tif

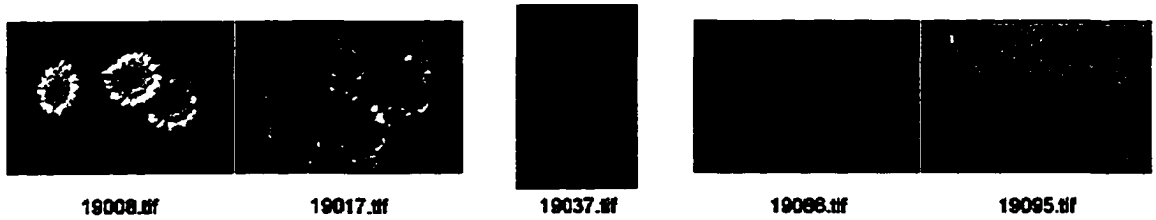

183005.tif 183010.tif 183020.tif 183046.tif

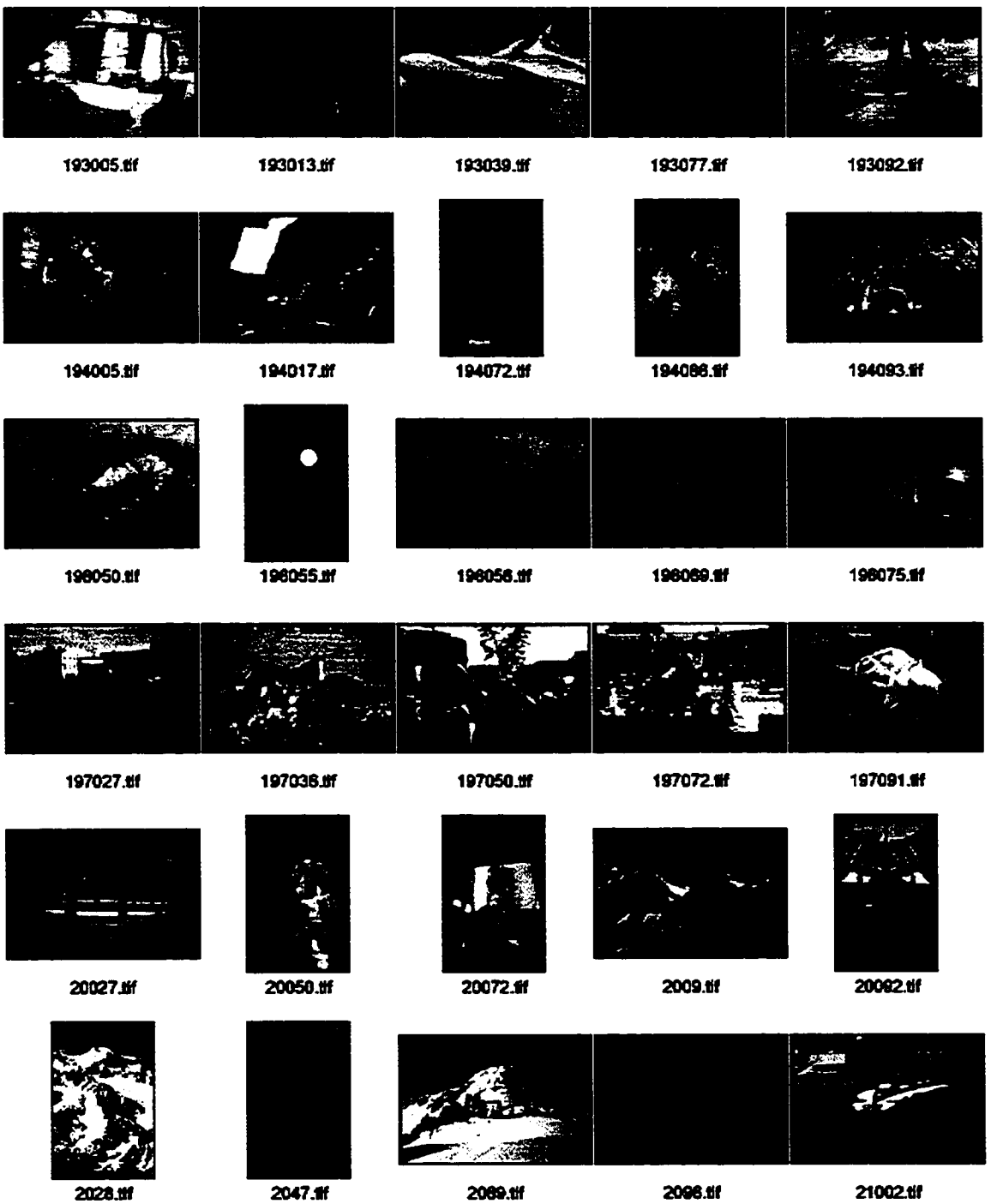

183063.tif

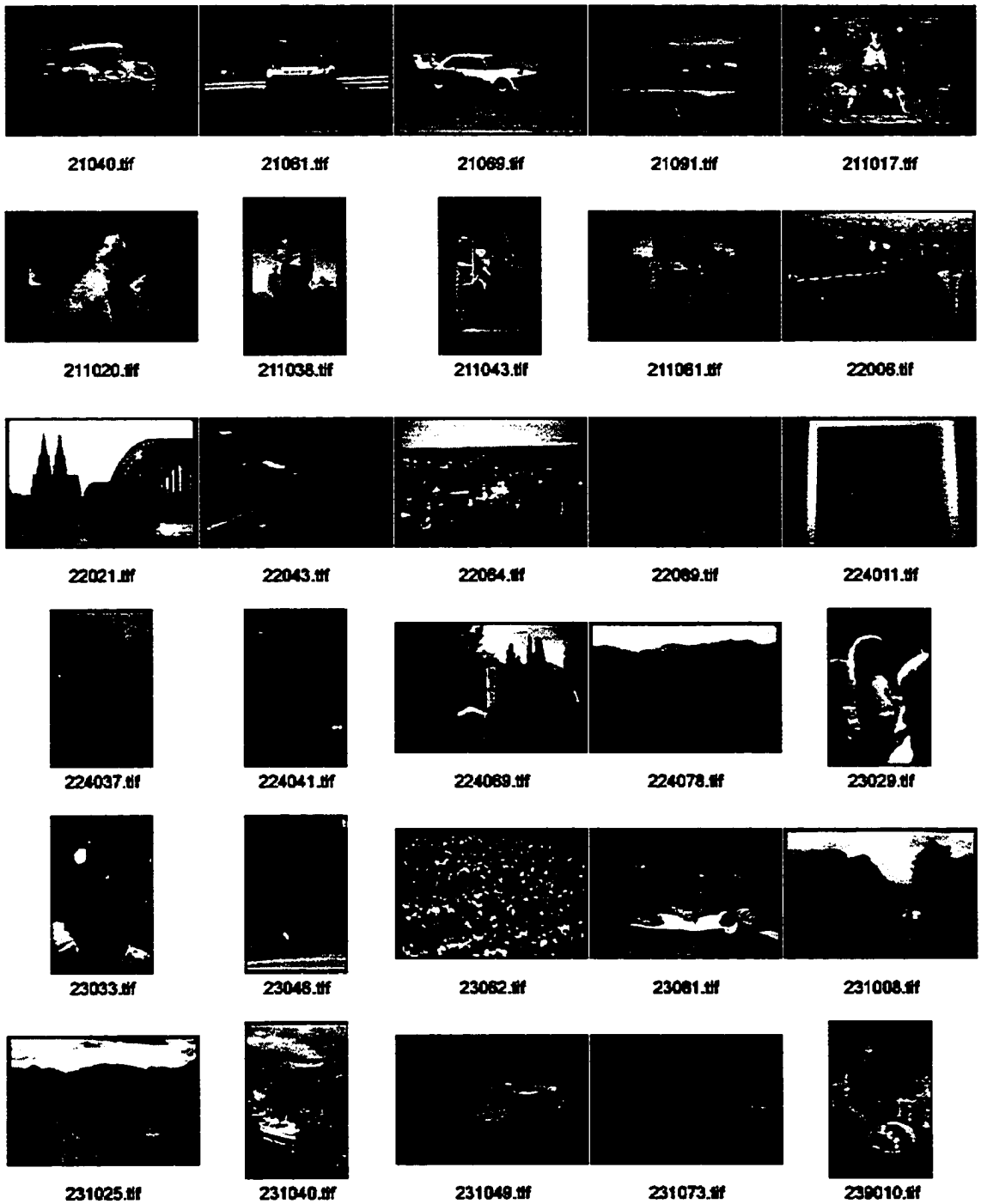

184024.tif 184035.tif 184044.tif 184048.tif 184082.tif

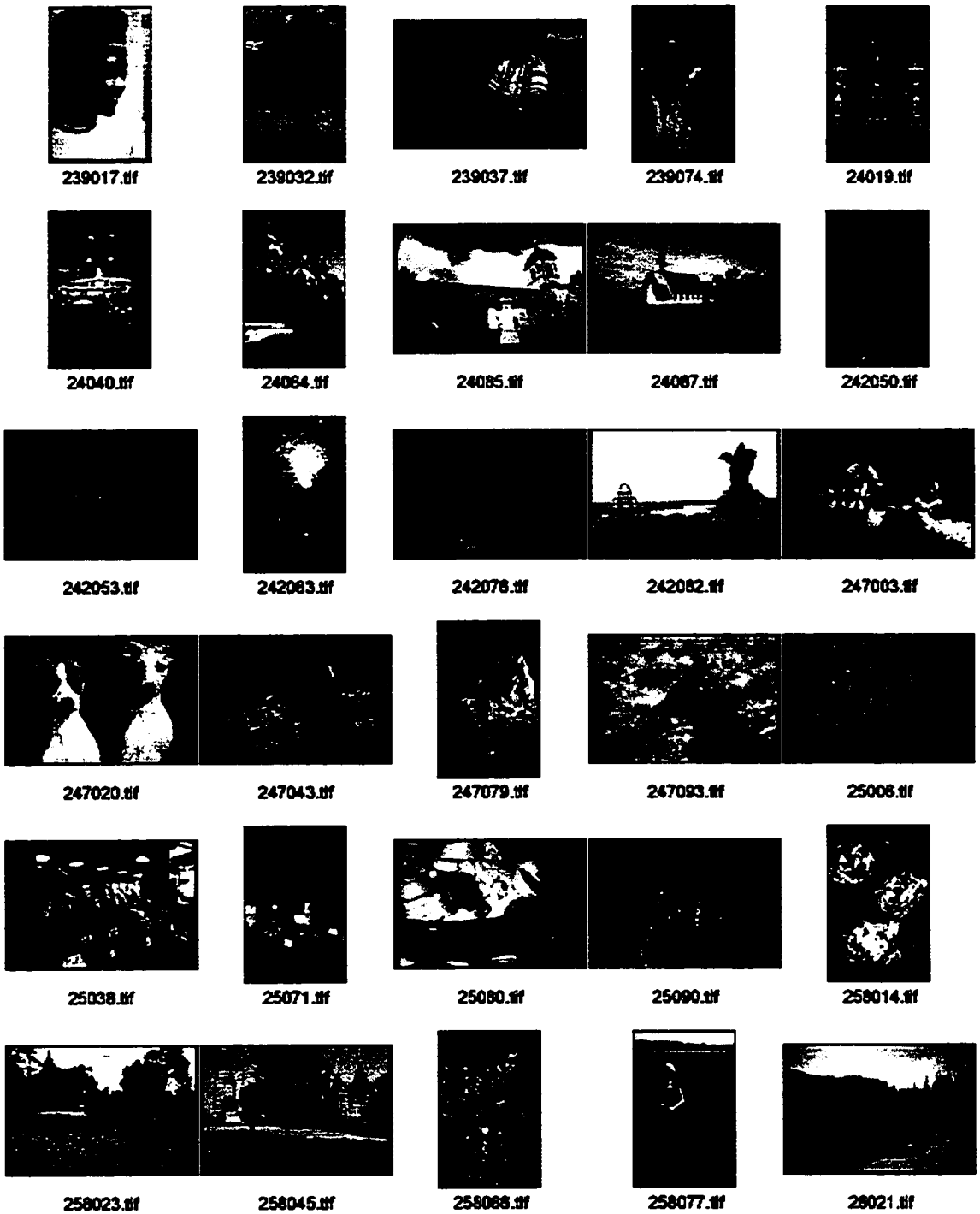

185013.tif 185033.tif 185047.tif 185068.tif 185091.tif

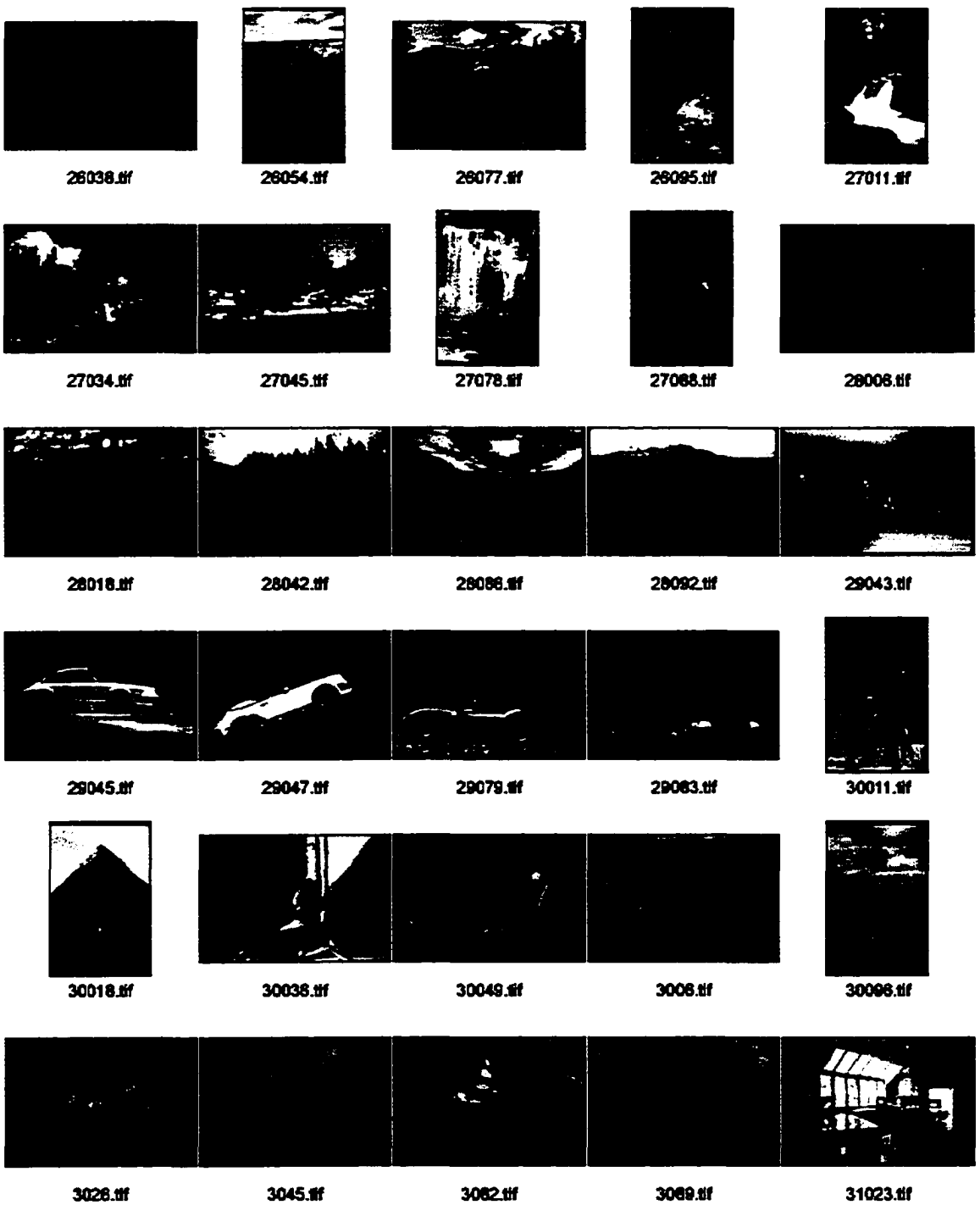

189000.tif 189017.tif 189033.tif 189053.tif 189068.tif

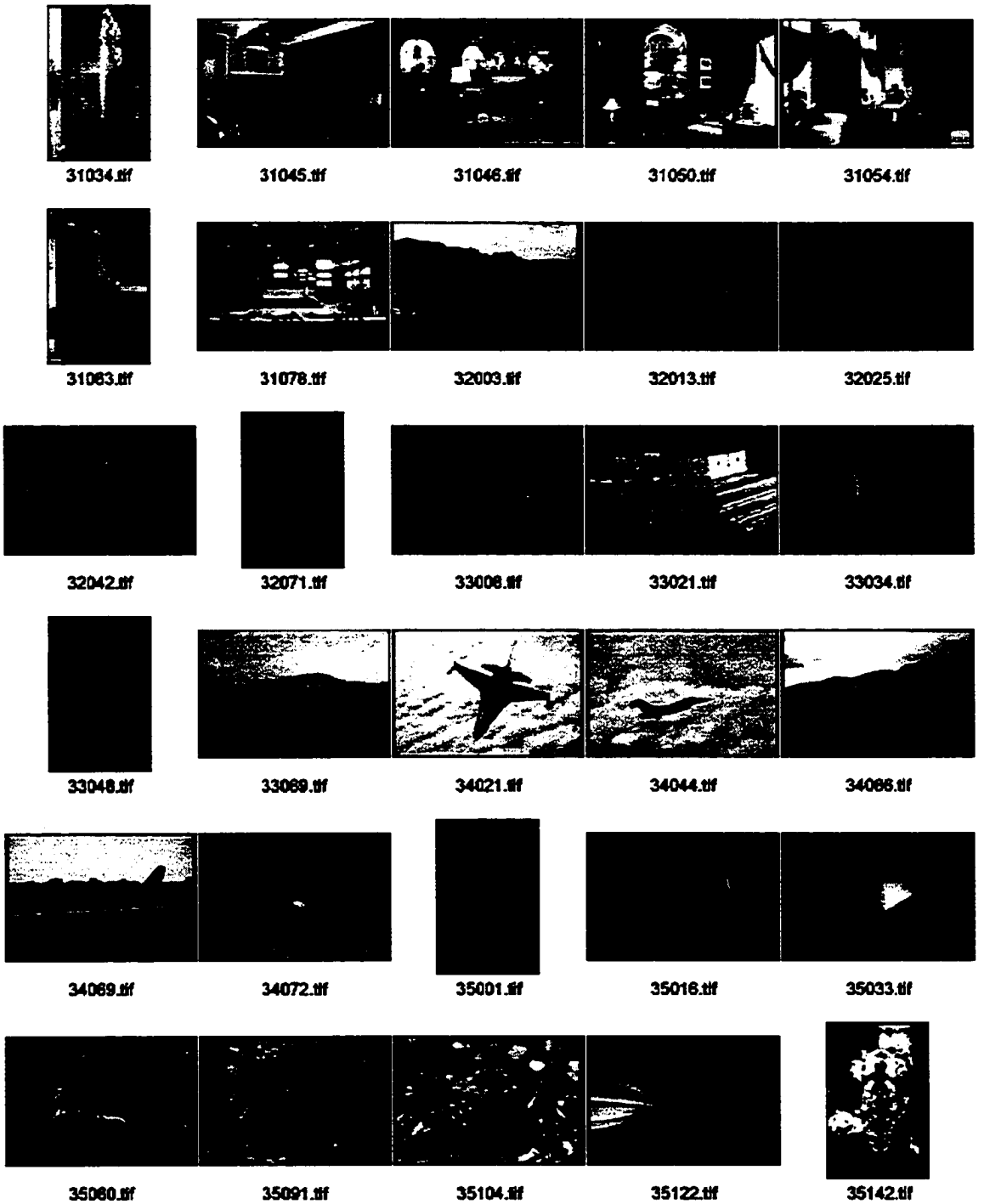

19008.tif 19017.tif 19037.tif 19066.tif 19095.tif

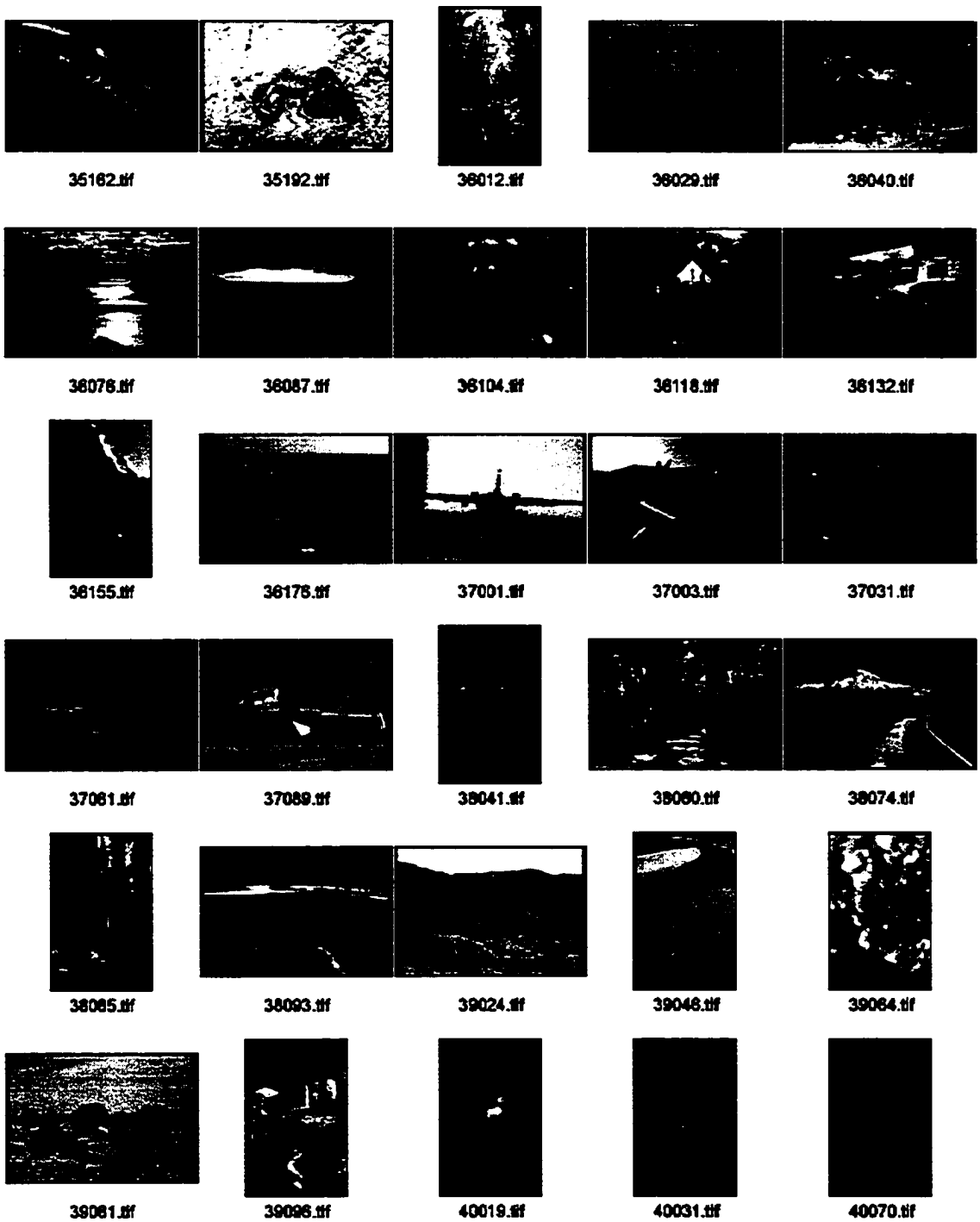


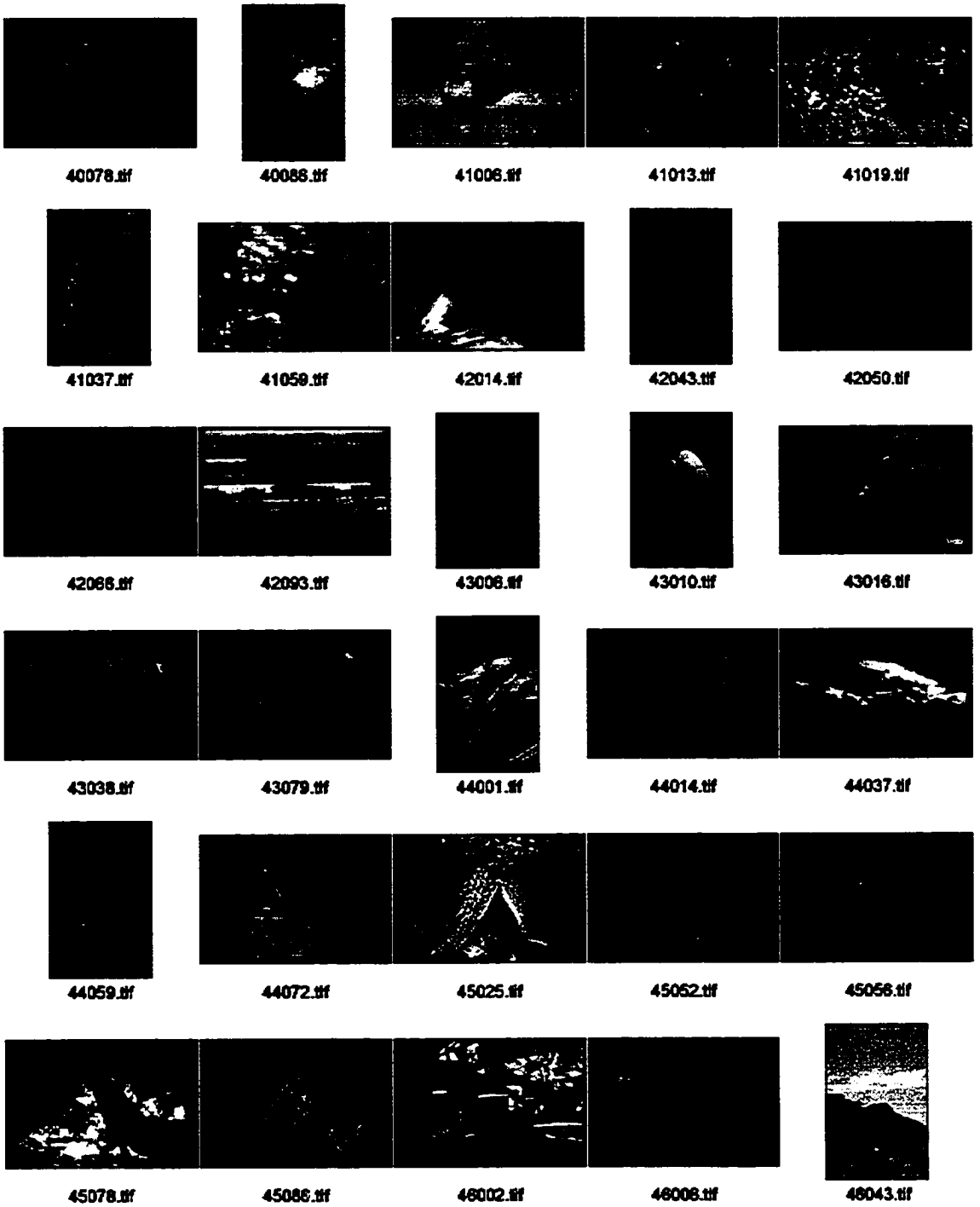


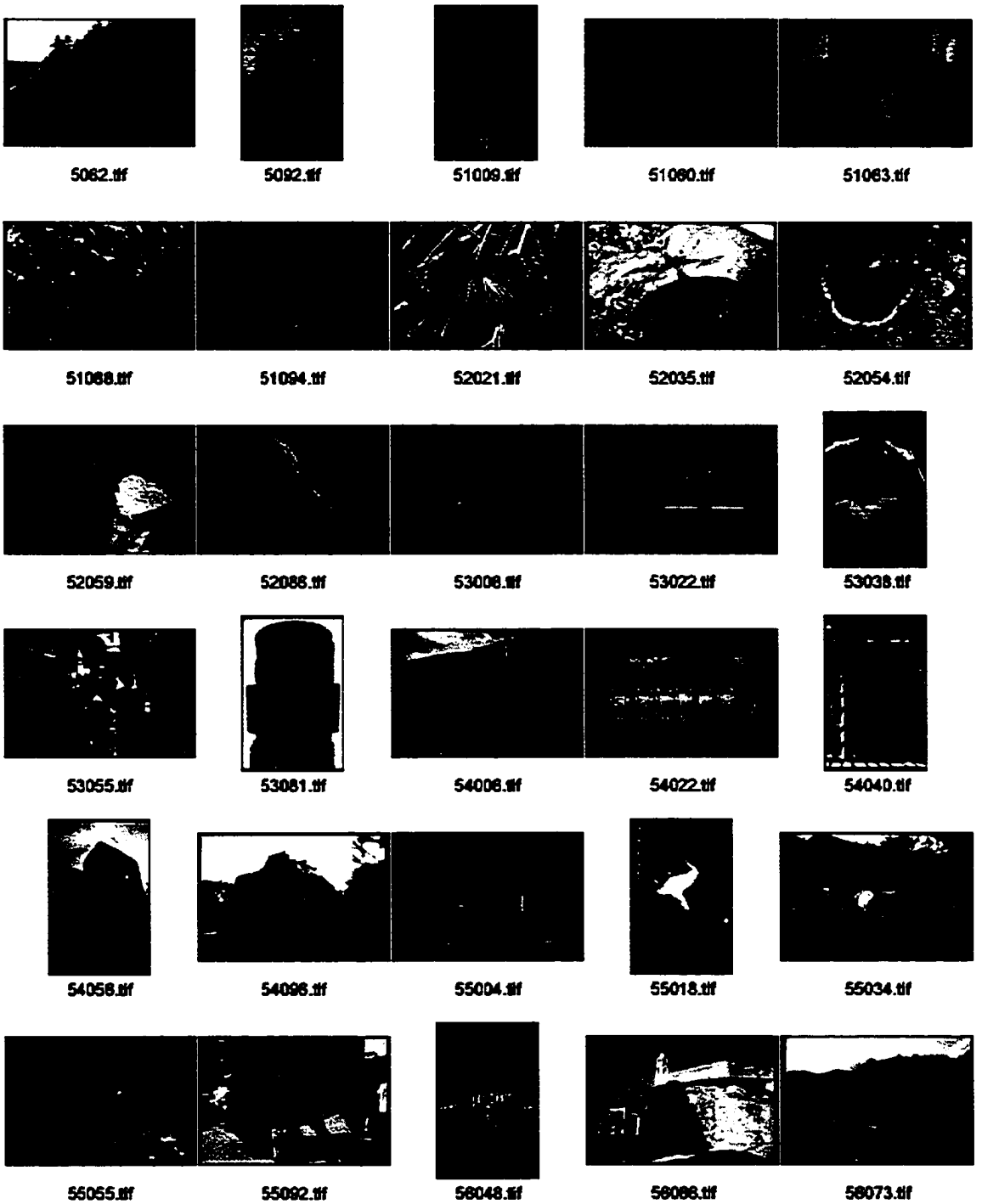


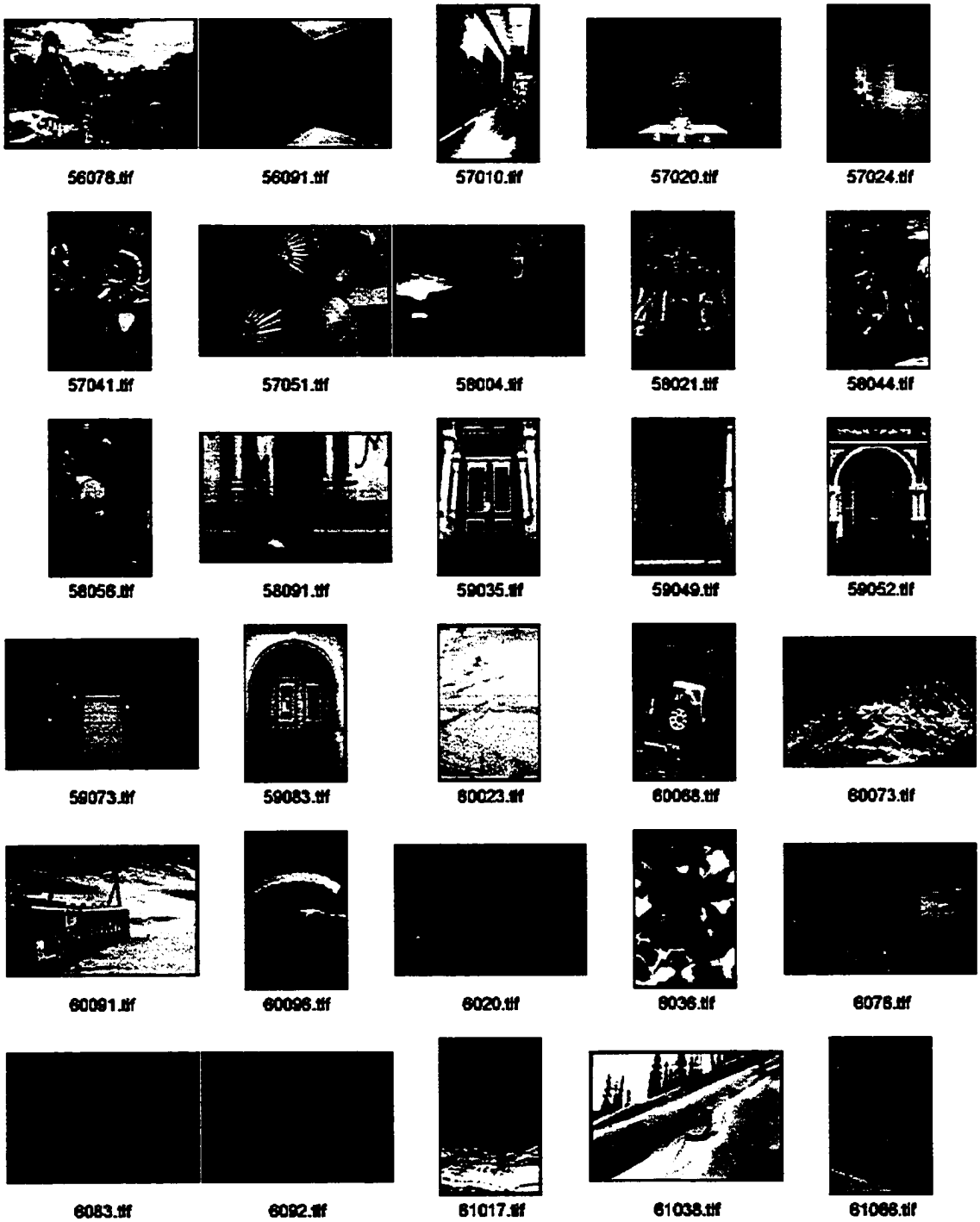




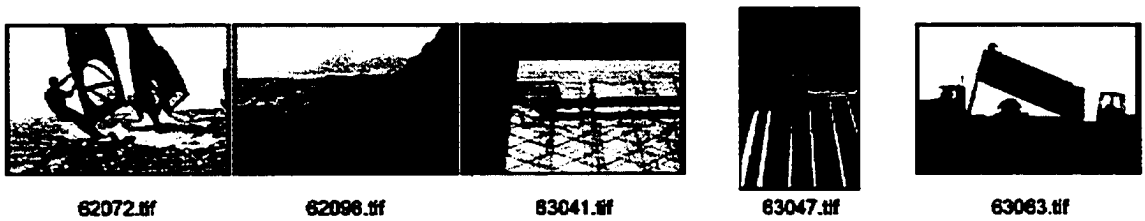








61072.tif 61075.tif 62006.tif 62025.tif 62050.tif



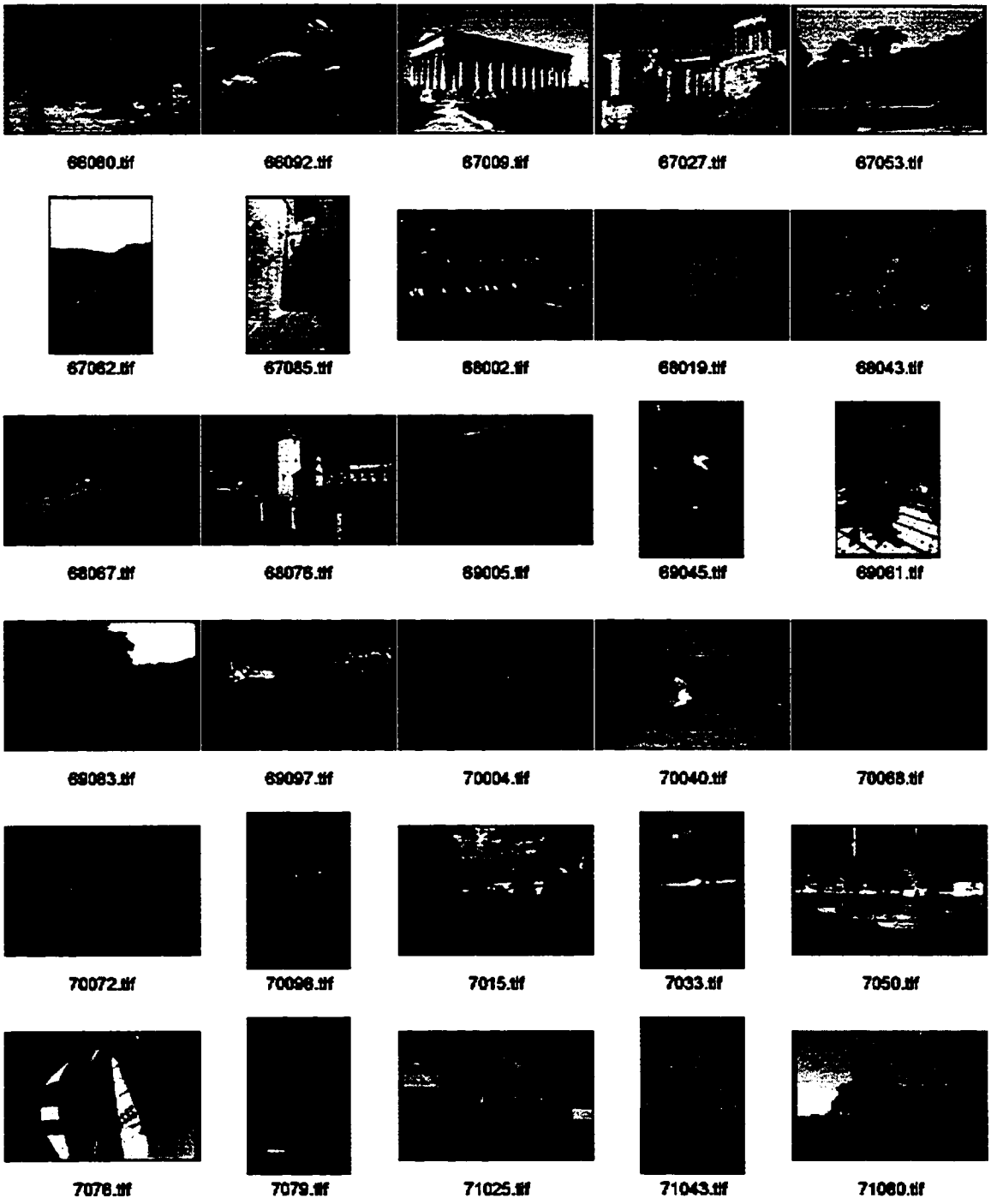
62072.tif 62096.tif 63041.tif 63047.tif 63063.tif

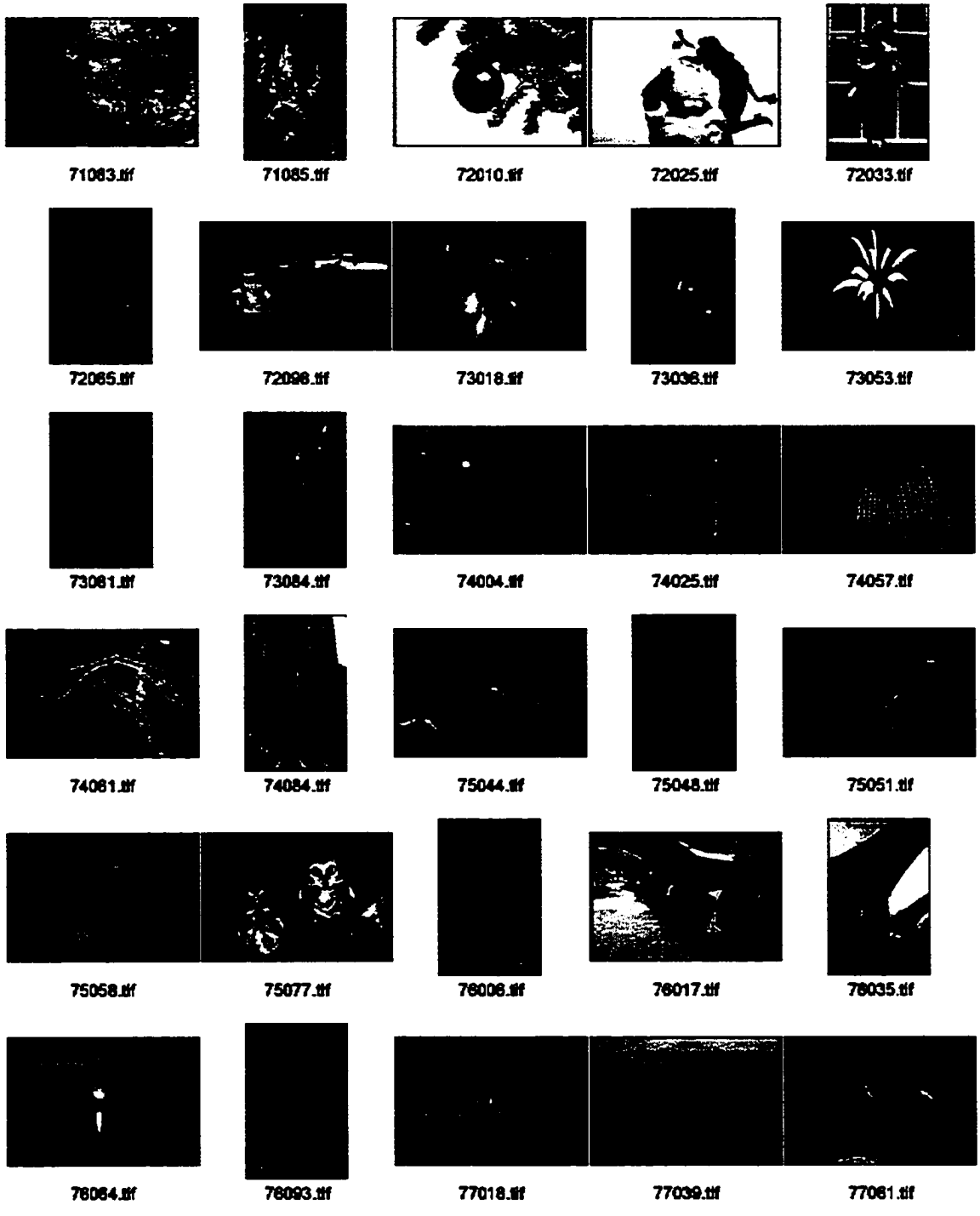
63080.tif 63096.tif 64009.tif 64023.tif 64043.tif

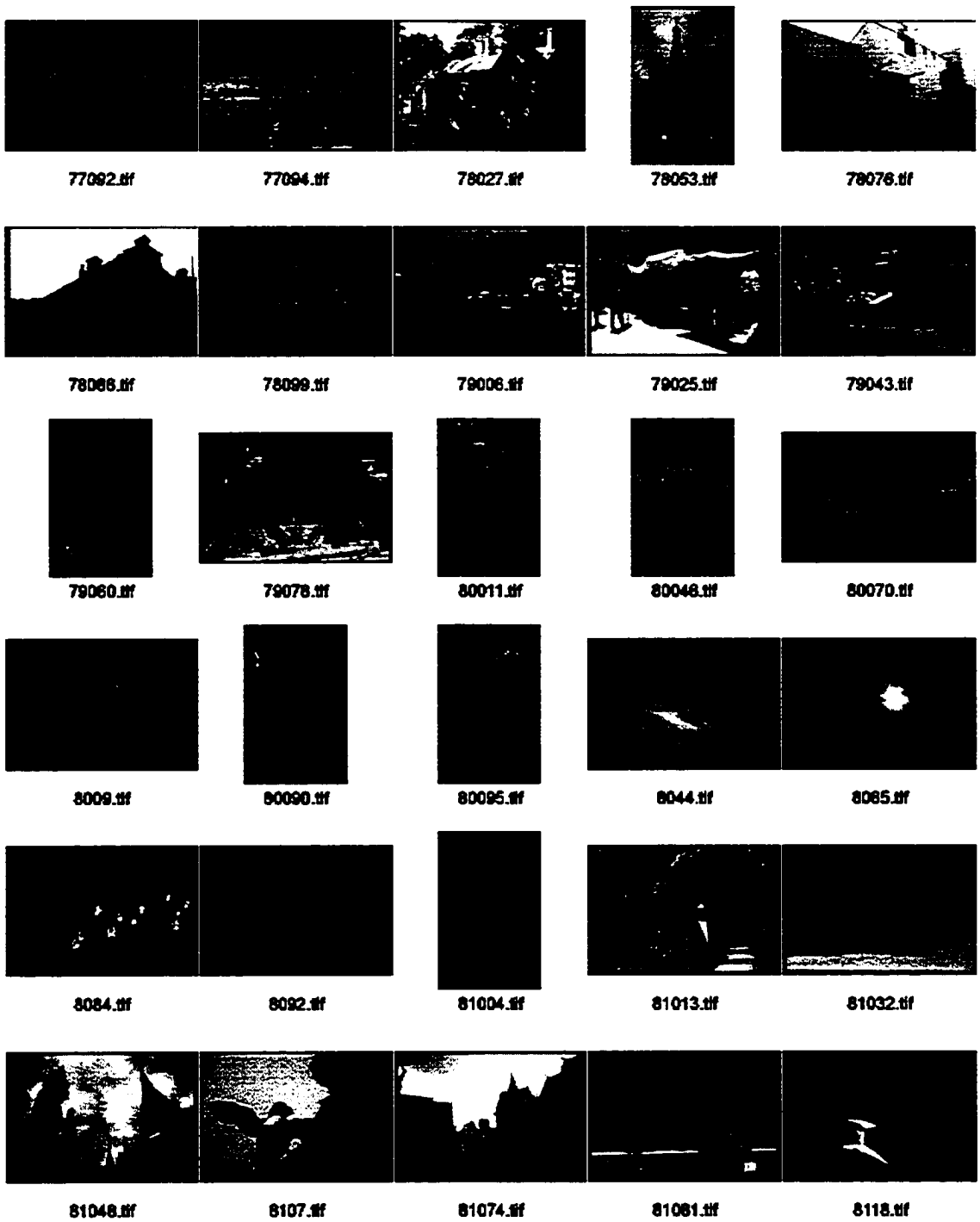
64057.tif 64083.tif 65006.tif 65025.tif 65042.tif

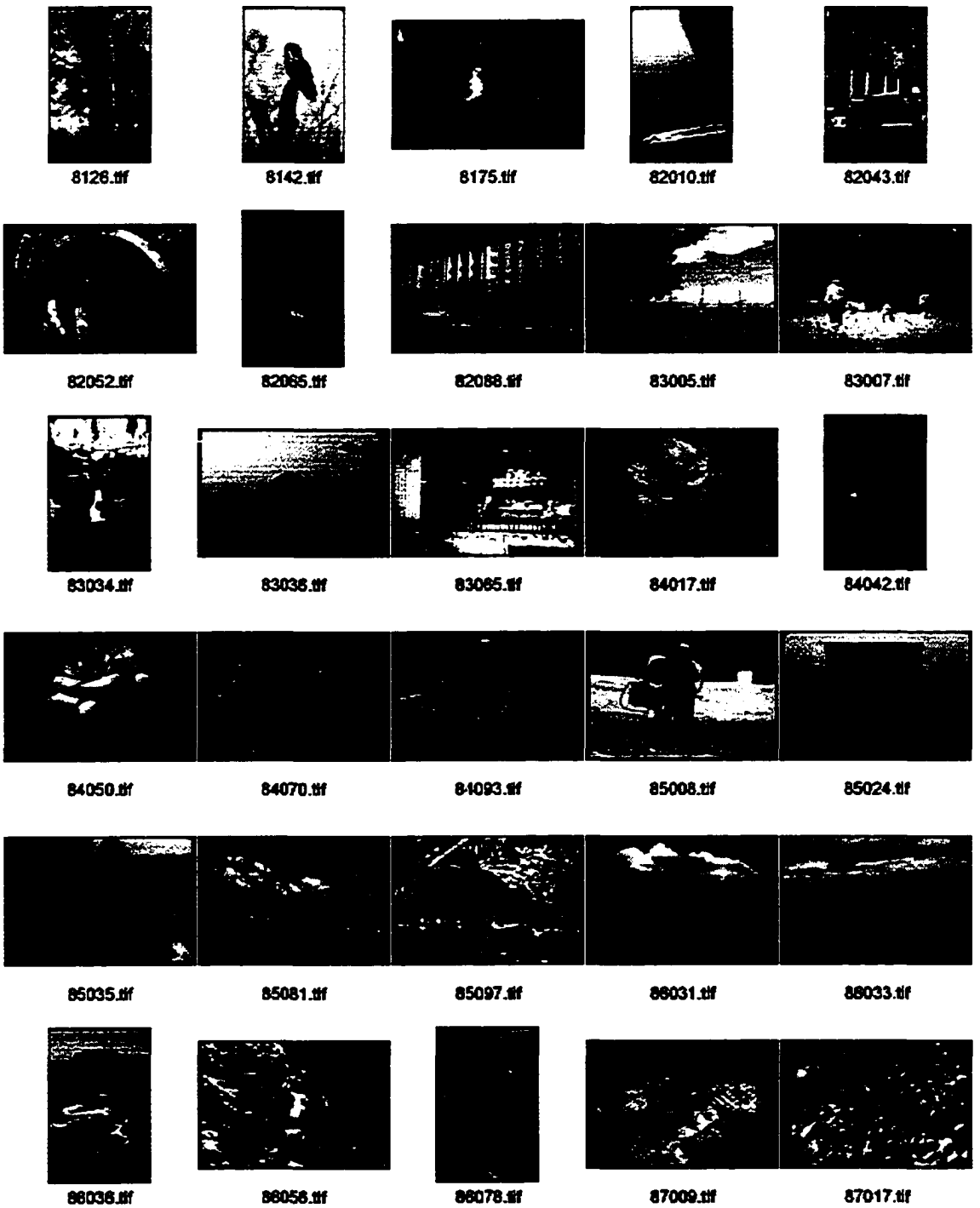
65061.tif 65083.tif 65107.tif 65124.tif 65148.tif

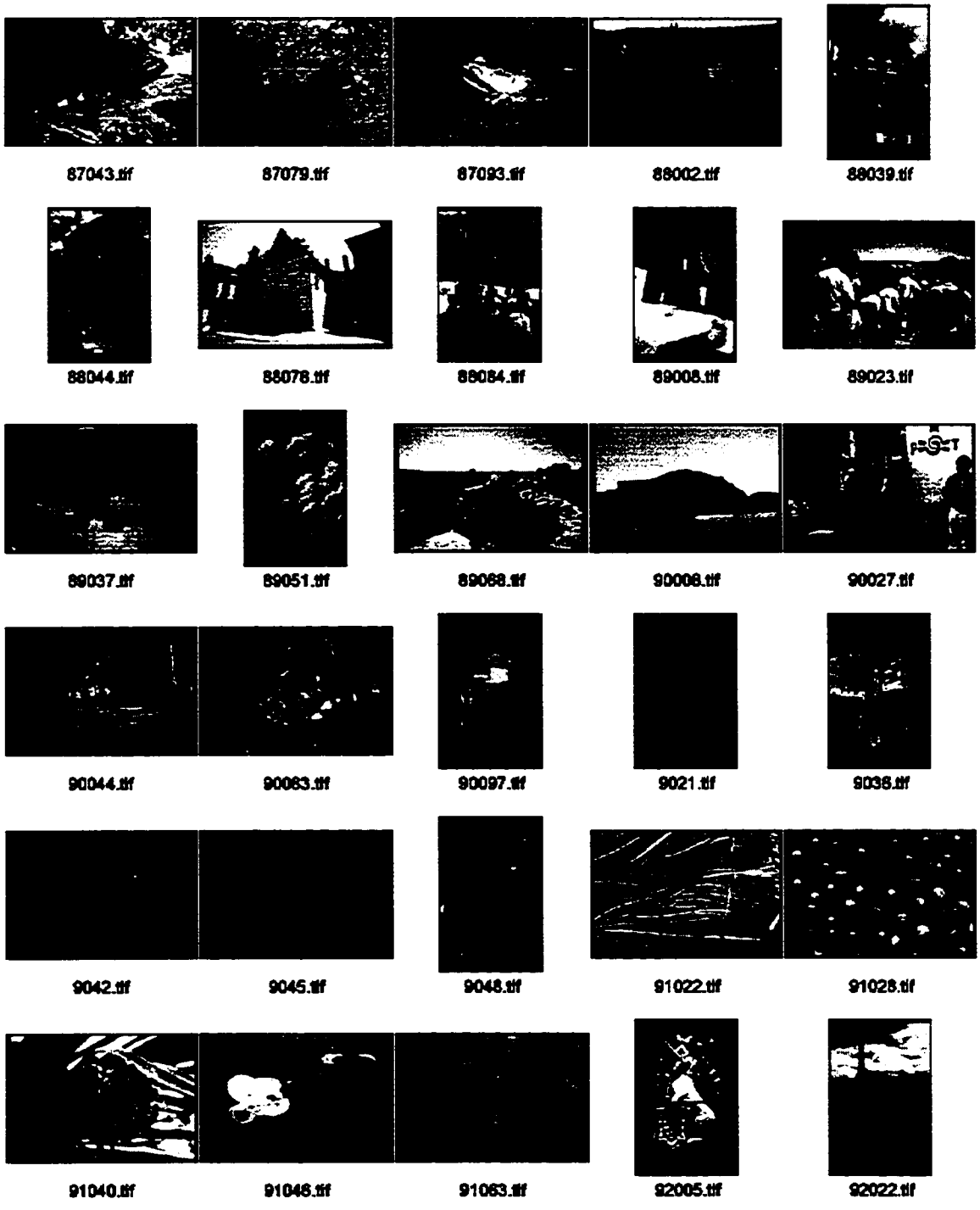
65164.tif 35193.tif 66016.tif 66026.tif 66037.tif

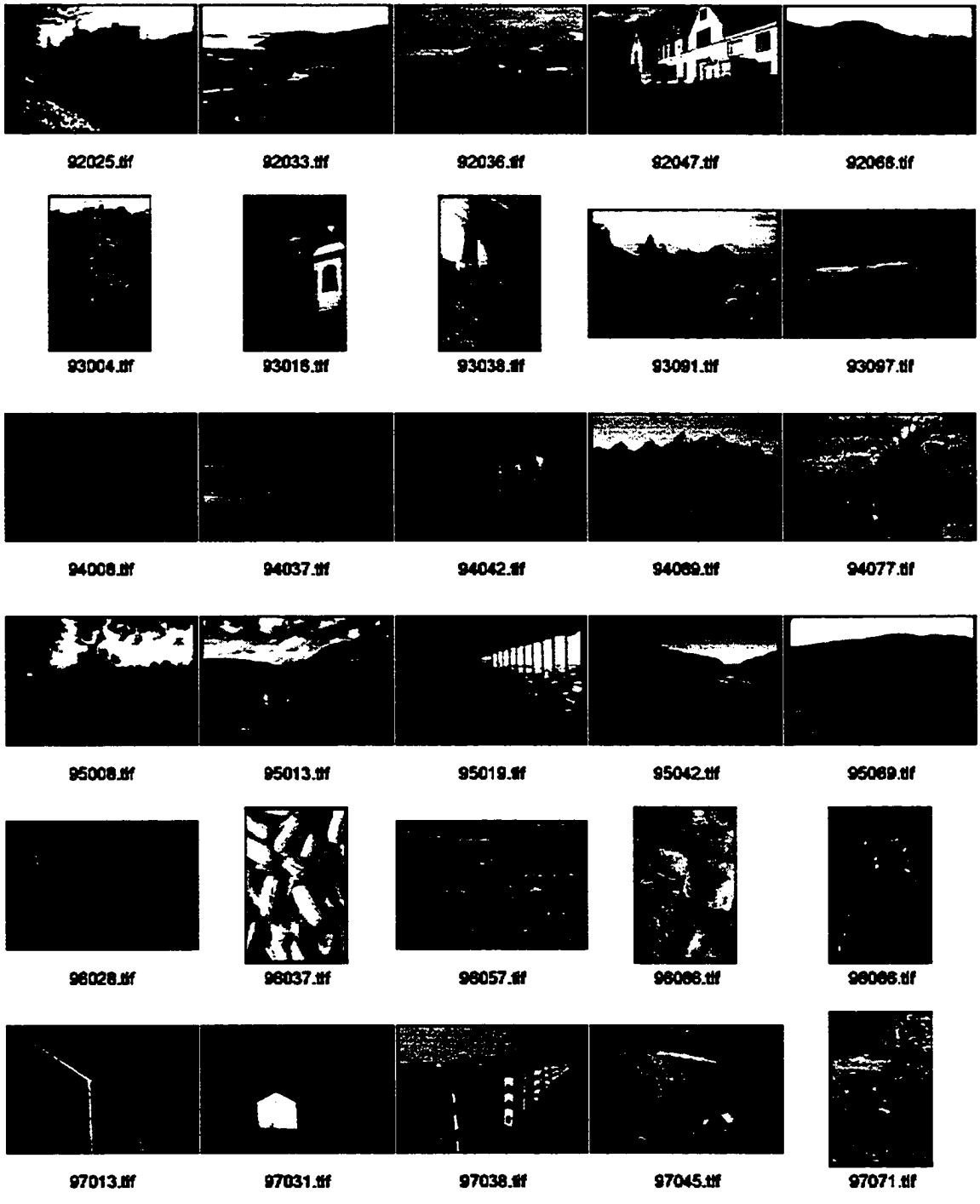












REFERENCES

[1] Augmented Reality for Development, Production and Servicing, Germany. http://www.arvika.de.

[2] Adelson, E. H., and J. R. Bergen. The Plenoptic Function and Elements of Early Vision. *Computational Models of Visual Processing* (1991).

[3] Aiger, Dror, and Daniel Cohen-Or. Mosaicing Ultrasonic Volumes for Visual Simulation. *IEEE Computer Graphics and Applications 20*, 2 (March/April 2000), 53–61.

[4] Anandan, P. A Computational Framework and an Algorithm for the Measurement of Visual Motion. *Intl. J. Computer Vision 2* (1989), 283–310.

[5] Anuta, P. E., M. Svedlow, and C. D. McGillem. Image Registration: Similarity Measure and Processing Method Comparisons. *IEEE Trans. on Aerospace and Electronic Systems 14*, 1 (1978), 141–149.

[6] Appel, Mirko, and Nassir Navab. 3D Reconstruction from Co-Registered Orthographic and Perspective Images: Theoretical Framework and Applications.

[7] Appel, Mirko, and Nassir Navab. Registration of Technical Drawings and Calibrated Images for Industrial Augmented Reality. *Machine Vision and Applications 13* (2002), 111–118. Springer-Verlag.

[8] Arribas, P. C., and F. M. Maciá. FPGA Implementation of a Log-polar Algorithm for Real Time Applications. *Conf. on Design of Circuits and Integrated Systems* (1997).

[9] Azuma, R. T. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments* (August 1997), 355–385.

[10] Azuma, R. T., Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications 21*, 6 (Nov/Dec 2001), 34–47.

[11] Barnea, D. I., and H. F. Silverman. A Class of Algorithms for Fast Digital Image Registration. *IEEE Trans. on Computers C-21*, 2 (February 1972), 179–186.

[12] Bascle, B., A. Blake, and A. Zisserman. Motion Deblurring and Super-Resolution from an Image Sequence. II:573–582.

[13] Bederson, B. B., R. S. Wallace, and E. Schwartz. A Miniaturized Space-Variant Active Vision System: Cortex-I. *Machine Vision and Applications 8* (1995), 101–109.

[14] Bergen, James R., and E.H. Adelson. Hierarchical, Computationally Efficient Motion Estimation Algorithm. *J. Opt. Soc. Am. 4*, 35 (1987).

[15] Bergen, James R., P. Anandan, Keith J. Hanna, and Rajesh Hingorani. Hierarchical Model-Based Motion Estimation. *Proc. Euro. Conf. Computer Vision (ECCV)* (1992), 237–252.

[16] Bernardino, Alexandre, and Jos Santos-Victor. A Binocular Stereo Algorithm for Log-Polar Foveated Systems. *Proc. Second Intl. Workshop on Biologically Motivated Computer Vision* (2002), 127–136.

[17] Beyer, W. H. *CRC Standard Mathematical Tables*, 28th ed. CRC Press, Boca Raton, FL, 1987.

[18] Birkfellner, Wolfgang, Michael Figl, Klaus Huber, and Franz Watzinger. A Head-Mounted Operating Binocular for augmented Reality Visualization in Medicine-Design and Initial Evaluation. *IEEE Transaction on Medical Imaging 21*, 8 (August 2002), 991–997.

[19] Blasco, F. J., F. Pardo, and J. A. Boluda. A FPGA Based PCI Bus Interface for a Realtime Log-Polar Image Processing System. *Conf. on Design of Circuits and Integrated Systems* (November 1999).

[20] Brown, Lisa G. A Survey of Image Registration Techniques. *ACM Computing Surveys 24*, 4 (December 1992), 325–376.

[21] Burt, Peter J., and Edward H. Adelson. A Multiresolution Spline with Application to Image Mosaics. *ACM Trans. Graphics 2*, 4 (October 1983), 217–236.

[22] Burt, P. J., and R. J. Kolczynski. Enhanced Image Capture Through Fusion. *Intl. Conf. Computer Vision* (1993), 173–182.

[23] Capel, D., and A. Zisserman. Super-Resolution from Multiple Views Using Learnt Image Models. *IEEE Conf. Computer Vision and Pattern Recognition* (2001), II:627–634.

[24] Capurro, C., F. Panerai, and G. Sandini. Dynamic Vergence using Log-polar Images. *International Journal of Computer Vision 24* (1997), 79–94.

[25] Casasent, D., and D. Psaltis. Position, Rotation, and Scale-Invariant Optical Correlation. *Applied Optics 15* (1976), 1793–1799.

[26] Castro, E. De, and C. Morandi. Registration of Translated and Rotated Images Using Finite Fourier Transforms. *IEEE Trans. Pattern Analysis and Machine Intelligence 9*, 3 (September 1987), 700–703.

[27] Caudell, T. P., and D. W. Mizell. Augmented Reality: An Application of Heads-up Display Technology to Manual Manufactuing Processes. *Proc. Hawaii Intl. Conf. on System Sciences II* (January 1992), 659–669.

[28] Chang, S. H., F. H. Chen, W. H. Hsu, and G. Z. Wu. Fast Algorithm for Pointer Pattern Matching; Invariant to Translations, Rotations, and Scale Changes. *Pattern Recognition 30(2)* (February 1997), 311–320.

[29] Chen, Q. S., M. Defrise, and F. Deconinck. Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence 16(12)* (December 1994), 1156–1168.

[30] Chen, S. E. Quicktime VR: An Image-Based Approach to Virtual Environment Navigation. *Computer Graphics (Proc. SIGGRAPH '95)* (1995), 29–38.

[31] Collignon, A., F. Maes, D. Delaere, D. Vandermeulen, P. Seutens, and G. Marchal. Automated multimodality image registration using information theory. *Information Processing in Medical Imaging: Proc. 14th Intl. Conf.* (1995), 263–274.

[32] Cover, T., and J. Thomas. *Element of Information Theory.* Wiley Interscience, New York, 1991.

[33] Davis, James. Mosaics of Scenes with Moving Objects. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (1998).

[34] Debevec, P. E., C. J. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. *Computer Graphics (Proc. SIGGRAPH '96)* (1996).

[35] der Spiegel, J. Van, G. Kreider, C. Claeys, I. Debusschere, G. Sandini, P. Dario, F. Fantini, P. Bellutti, and G. Soncini. A Foveated Retina-Like Sensor Based on CCD Technology. In *Analog VLSI Implementation of Neural Systems*. Academic, Boston, MA, 1989, pp. 189–210.

[36] Dow, B. M., A. Z. Snyder, R. G. Vautin, and R. Bauer. Magnification Factor and Receptive Field Size in Foveal Striate Cortex of the Monkey. 1981, pp. 213–228.

[37] Duda, Richard O., and Peter E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Comm. ACM 15*, 1 (January 1972), 11–15.

[38] Dufournaud, Yves, Cordelia Schmid, and Radu Horaud. Matching Images with Different Resolutions. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (June 2000), 612–618.

[39] Dufournaud, Yves, Cordelia Schmid, and Radu Horaud. Image Matching with Scale Adjustment. Tech. Rep. RR 4458, INRIA Rhône-Alpes, Montbonnot Saint-Martin, May 2002.

[40] Dyer, C. R. Volumetric Scene Reconstruction from Multiple Views. In *Foundations of Image Understanding*, L. S. Davis, Ed. Kluwer, 2001, pp. 469–489.

[41] Elad, M., and A. Feuer. Super-Resolution Reconstruction of Image Sequences. *IEEE Trans. Pattern Analysis and Machine Intelligence 21*, 9 (September 1999), 817–834.

[42] Essen, D. C. Van, W. T. Newsome, and J. H. Maunsell. The Visual Field Representation in Striate Cortex of the Macaque Monkey: Asymmetries, Anisotropies, and Individual Variability. *Vision Research 24*, 5 (1984), 429–48.

[43] Etienne-Cummings, R., J. Van der Spiegel, P. Mueller, and M. Z. Zhang. A Foveated Silicon Retina for Two-Dimensional Tracking. *IEEE Trans. Circuits and Systems II 47* (June 2000), 504–517.

[44] Ferrari, F., J. Nielsen, and G. Sandini. Space Variant Imaging. *Sensor Review 15*, 2 (1995), 17–20.

[45] Ferrari, V., T. Tuytelaars, and L. Van Gool. Wide-baseline Multiple-view Correspondences. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2003).

[46] Fischler, M. A., and R. C. Bolles. Random Sample Consensus: a Paradigm for Model Fitting with Application to Image Analysis and Automated Cartographys. *Communication Association and Computing Machine 24*, 6 (1981), 381–395.

[47] Ford, Adrian, and Alan Roberts. Colour Space Conversions. *Computer Vision and Image Understanding* (August 1998).

[48] Giesler, Björn, Rene Graf, Rüdiger Dillmann, and Carl F.R. Weiman. Fast Mapping using the Log-Hough Transformation. *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS98* (1998).

[49] Gortler, S. J., R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. *Computer Graphics (Proc. SIGGRAPH '96)* (1996).

[50] Haralick, R. M., and L. Shapiro. *Computer and Robot Vision (Vols. 1 and 2)*. Addison Wesley, 1993.

[51] Hardie, R. C., K. J. Barnard, and E. E. Armstrong. Joint MAP Registration and High Resolution Image Estimation Using a Sequence of Undersampled Images. *Image Processing 6*, 12 (December 1997), 1621–1633.

[52] Harris, C., and M. Stephens. A Combined Corner and Edge Detector. *Proc. 4th Alvey Vision Conf.* (1988), 189–192.

[53] Hough, P.V.C. Methods and Means for Recognizing Complex Patterns. *U.S. Patent 3,069,654* (December 1962).

[54] Hubel, D. H., and T. N. Wiesel. Sequence Regularity and Geometry of Orientation Columns in the Monkey Striate Cortex. *Journal of Comparative Neurology* (1974), 158:267293.

[55] Irani, Michal, and P. Anandan. Video Indexing Based on Mosaic Representations. *Proc. IEEE 86*, 5 (May 1998), 237–252.

[56] Irani, M., and S. Peleg. Improving resolution by Image Registration. *CVGIP: Graph. Models Image Process 53* (March 1991), 231–239.

[57] Junck, L., J. G. Moen, G. D. Hutchins, M. B. Brown, and D.E. Kuhl. Correlation Methos for the Centering, Rotation, and Alignment of Functional Brain Images. *Journal of nuclear medicine*, 31 (1990), 1220–1276.

[58] Keller, Y., A. Averbuch, and M. Israeli. Pseudo-Polar Based Estimation of Large Translations Rotations and Scalings in Images. *to appear in IEEE Trans. Image Processing* (2003).

[59] Khamene, Ali, Frank Wacker, Sebastian Vogt, Fred Azar, Michael Wendt, Frank Sauer, and Jonathan Lewin. An Augmented Reality System for MRI-Guided Needle Biopsies. *MMVR03* (January 2003), 151–157.

[60] Klinker, G., O. Creighton, A. Dutoit, R. Kobylinski, C. Vilsmeier, and B. Brgge. Augmented Maintenance of Powerplants: A Prototyping Case Study of a Mobile AR System.

[61] Koenderink, J.J., and A.J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics. 55* (1987), 367–375.

[62] Krüger, V., and G. Sommer. Gabor wavelet networks for object representation. *Proc. of the Int. Dagstuhl 2000 Workshop* (2000).

[63] Kuglin, C. D., and D. C. Hines. The Phase Correlation Image Aligment Method. *Proc. Intl. Conf. on Cybernetics and Society* (1975), 163–165.

[64] Laveau, S., and O. Faugeras. 3-D Scene Representation as a Collection of Images and Fundamental Matrices. *Proceedings of 12th International Conference on Pattern Recognition 1* (1994).

[65] Lee, Seungyong, George Wolberg, and Sung Yong Shin. Scattered Data Interpolation Using Multilevel B-splines. *IEEE Trans. on Visualization and Computer Graphics 3*, 3 (July-September 1997), 228–244.

[66] Lepetit, Vincent, and Marie-Odile Berger. A Semi-Automatic Method for Resolving Occlusion in Augmented Reality. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2000).

[67] Levoy, M., and P. Hanrahan. Light Field Rendering. *Computer Graphics (Proc. SIGGRAPH '96)* (1996).

[68] Lucchese, L., G. Cortelazzo, and M. Rizzato. A Phase Correlation Technique for Estimating Planar Rotations. *Proc. Intl. Workshop on Time-Varying Image Processing and Moving Object Recognition* (September 1996), 244–249.

[69] Lucchese, L., and G. M. Cortelazzo. Noise-robust Estimation of Planar Roto-Translations with High Precision. *Proc. of IEEE Intl. Conf. on Image Processing* (October 1997), 699–702.

[70] Lucchese, L., G. M. Cortelazzo, and C. Monti. High Resolution Estimation of Planar Rotations Based on Fourier Transform and Radial Projections. *Proc. ISCAS 97 2* (June 1997), 1181–1184.

[71] Mann, Steve, and James Fung. VideoOrbits on Eye Tap Devices for Deliberately Diminished Reality or Altering the Visual Perception of Rigid Planar Patches of a Real World Scene. *International Symposium on Mixed Reality (ISMR2001)* (March 2001).

[72] Mann, Steve, and Rosalind W. Picard. Video Orbits of the Projective Group: A Simple Approach to Featureless Estimation of Parameters. *IEEE Trans. Image Processing 6*, 9 (September 1997), 1281–1295.

[73] Manzotti, R., A. Gasteratos, G. Metta, and G. Sandini. Disparity Estimation on Log-Polar Images and Vergence Control. *Computer Vision and Image Understanding 83*, 2 (2001), 97–117.

[74] Marquardt, Donald W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Society for Industrial and Applied Mathematics 11*, 2 (1963), 431–441.

[75] Marshall, W. H., C. N. Woolsey, and Philip Bard. Observations on cortical somatic sensory mechanisms of cat and monkey. *Journal of Neurophysiology 4* (1941), 1–43.

[76] McMillan, L., and G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. *Computer Graphics (Proc. SIGGRAPH '95)* (August 1995).

[77] Mikolajczyk, K., and C. Schmid. Indexing Based on Scale Invariant Interest Points. *Intl. Conf. on Computer Vision* (July 2001), 525–531.

[78] Mourgues, F., F. Devernay, and É. Coste-Maniére. 3D Reconstruction of the Operating Field for Image Overlay in 3D-Endoscopic Surgery. *International Symposium on Augmented Reality (ISAR 2001)* (October 2001).

[79] Navab, N., B. Bascle, M. Appel, and E. Cubillo. Scene Augmentation Via the Fusion of Industrial Drawings and Uncalibrated Images With a View to Marker-less Calibration. *Proc. IEEE Intl. Workshop on Augmented Reality (IWAR 99),* (October 1999).

[80] Papadimitriou, D. V., and T. J. Dennis. Stereo Disparity Using Phase Correlation. *Electronics Letters 30*, 18 (September 1994), 1475–1477.

[81] Pardo, F., J. A. Boluda, I. Coma., and F. Micó. High Speed Log-Polar Time to Crash Calculation for Mobile Vehicles. *Image Processing and Communications 8*, 2 (2002), 23–32.

[82] Peleg, S., and J. Herman. Panoramic Mosaicing with VideoBrush. *DARPA Image Understanding Workshop* (May 1997), 261–264.

[83] Pluim, J. P. W., A. Maintz, and M. A. Viergever. Interpolation Artefacts in Mutual Information-Based Image Registration. *Computer Vision and Image Understanding 77* (2000), 211–232.

[84] Pratt, W. K. Correlation Techniques for Image Registration. *IEEE Trans. on Aerospace and Electronic Systems* (May 1974), 353–358.

[85] Press, W. H. *Numerical Recipes in C:The Art of Scientfic Computing.* Combridge University Press, New York, 1993.

[86] Radcliffe, T., R. Rajapaskshe, and S. Shalev. Pseudo-Correlation: A Fast, Robust, Absolute, Grey-Level Image Alignment algorithm. *Medical Physics 21*, 6 (1994), 761–769.

[87] Reddy, B. S., and B. N. Chatterji. An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration. *IEEE Trans. Pattern Analysis and Machine Intelligence 5*, 8 (August 1996), 1266–1270.

[88] Romeny, B. M., L. M. Florack, A. H. Salden, and M. A. Viergever. Higher Order Differential Structure of Images. *Image and Vision Computing 12*, 6 (1994), 317–325.

[89] Sandini, G., and M. Tistarelli. Vision and Space-Variant Sensing. *Human and Machine Perception, vol. 1 of Neural Networks for Perception* (1992), 398–425.

[90] Schaffalitzky, F., and A. Zisserman. Viewpoint Invariant Texture Matching and Wide Baseline Stereo. *Proc. ICCV* (July 2001).

[91] Schmid, C., R. Mohr, and C. Bauckhage. Evaluation of Interest Point Detectors. *Intl. Journal of Computer Vision. 37*, 2 (2000), 151–172.

[92] Schwartz, E. L. Spatial Mapping in Primate Sensory Projection: Analytic Structure and Relevance to Perception. *Biological Cybernetics 25* (1979), 181–194.

[93] Schwartz, E. L. Topographical Mapping in Primate Visual Cortex: History, Anatomy and Computation. In *In D.H. Kelly; editor; Visual Science and Engineering. Models and Applications*. Marcel Dekker, New York, 1994.

[94] Seitz, S. M., and C. R. Dyer. View Morphing. *Computer Graphics (Proc. SIGGRAPH '96)* (1996).

[95] Shekarforoush, H., M. Berthod, and J. Zerubia. Subpixel Image Registration by Estimating the Polyphase Decomposition of Cross Power Spectrum. *Computer Vision and Pattern Recognition* (1996), 532–537.

[96] Shekarforoush, H., and J. B. Zerubia. Extension of Phase Correlation to Subpixel Registration. *IEEE Trans. on Image Processing 11(3)* (2002).

[97] Shum, Heung-Yueng, and Richard Szeliski. Construction and Refinement of Panoramic Mosaics with Global and Local Alignment. *Proc. Intl. Conf. Computer Vision* (1998), 953–958.

[98] Stone, Harold, Michael Orchard, and E.C. Chang. Subpixel Registration of Images. *33rd Asilomar Conf. on Signals, Systems, and Computers* (October 1999).

[99] Stone, Harold, B. Tao, and Morgan McGuire. Analysis of image registration noise due to rotationally dependent aliasing. *Journal of Visual Communication and Image Representation 14* (2003), 114–135.

[100] Szeliski, Richard. Rapid Octree Construction from Image Sequences. *CVGIP: Image Understanding* (July 1993).

[101] Szeliski, Richard. Image Mosaicing for Tele-Reality Applications. *Proc. IEEE Workshop on Applications of Computer Vision* (1994), 230–236.

[102] Szeliski, Richard, and Heung-Yueng Shum. Video Mosaics for Virtual Environments. *IEEE Computer Graphics and Applications 16* (1996), 22–30.

[103] Szeliski, Richard, and Heung-Yueng Shum. Creating Full View Panoramic Image Mosaics and Environment Maps. *Computer Graphics (Proc. SIGGRAPH '97)* (1997), 251–258.

[104] Thévenaz, Philippe, Urs E. Ruttimann, and Michael Unser. A Pyramid Approach to Subpixel Registration Based on Intensity. *IEEE Trans. Image Processing 7*, 1 (1998), 27–41.

[105] Tootell, R. B., M. S. Silverman, E. Switkes, and R. de Valois. Deoxyglucose, Retinotopic Mapping and the Complex Log Model in Striate Cortex. *Science 218* (1982), 902–904.

[106] Tuytelaars, T., and L. Van Gool. Wide Baseline Stereo Matching Based on Local Affinely Invariant Regions. *Proc. of the 11th British Machine Vision Conference* (2000).

[107] Vincze, Markus, and Carl F. Weiman. General relationship for optimal tracking performance. *Proc. SPIE: Intelligent Robots and Computer Vision XV: Algorithms, Techniques,Active Vision, and Materials Handling 2904* (1996), 402–412.

[108] Viola, P., and W. M. Wells. Alignment by Maximization of Mutual Information. *Intl. Journal of Computer Vision 24(2)* (1995), 137–154.

[109] Wang, John, and Edward Adelson. Representing Moving Images with Layers. *IEEE Trans. on Image Processing Special Issue: Image Sequence Compression 3*, 5 (September 1994).

[110] Wang, L., S. Kang, R. Szeliski, and H. Shum. Optimal texture map reconstruction from multiple views. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (December 2001), 347–354.

[111] Weiman, Carl F.R. 3-D sensing with polar exponential sensor arrays. *Proc. SPIE: Digital and Optical Shape Representation and Pattern Recognition 938* (1988), 78–87.

[112] Weiman, Carl F.R. Polar exponential sensor arrays unify iconic and Hough space representation. *Proc. SPIE: Intelligent Robots and Computer Vision VIII: Algorithms and Techniques 1102* (1990), 832–842.

[113] Weiman, Carl F.R. Binocular Stereo via Log-Polar Retinas. *Proc. SPIE: Visual Information Processing IV 2488* (1995), 309–320.

[114] Weiman, Carl F.R., and G. Chaikin. Logarithmic spiral grids for image processing and display. *Computer Graphics and Image Processing 11* (1979), 197–226.

[115] Weinshall, D., and E. L. Schwartz. A New Method for Measuring the Visuotopic Map Function of Striate Cortex: Validation with Macaque Data and Possible Extension to Measurement of the Human Map. *Society of Nerouscience Abstracts* (1987), 1291.

[116] Wilson, S. W. On the Retino-Cortical Mapping. *International Journal of Man-Machine Studies 18* (1983), 361–389.

[117] Wodnicki, R., G. W. Roberts, and M. Levine. Design and Evaluation of a Log-Polar Image Sensor Fabricated using a Standard 1.2 um ASIC CMOS Process. *IEEE Journal of Solid-State Circuits 32*, 8 (1997), 1274–1277.

[118] Wolberg, George. *Digital Image Warping.* IEEE Computer Society Press, Los Alamitos, CA, 1990.

[119] Wolberg, George, and Siavash Zokai. Image Registration for Perspective Deformation Recovery. *Proc. SPIE Conf. on Automatic Target Recognition X 4050* (April 2000).

[120] Wolberg, George, and Siavash Zokai. Robust Image Registration Using Log-Polar Transform. *Proc. IEEE Intl. Conf. on Image Processing* (September 2000).

[121] Yeung, M. M., B. Yeo, S. Liou, and A. Bani-Hashemi. Three-dimensional Image Registration for Spiral (CT Angiography). *Computer Vision and Pattern Recognition* (1994), 423–429.

[122] Young, D. Straight Lines and Circles in the Log-Polar Image. *Proc. of the 11th British Machine Vision Conference* (September 2000), 426–435.

[123] Zhang, X., Y. Genc, and N. Navab. Taking AR into Large Scale Industrial Environments: Navigation and Information Access with Mobile Computers. *International Symposium on Augmented Reality (ISAR 2001)* (October 2001).

[124] Zokai, Siavash, Nassir Navab, Yakup Genc, and Julien Esteve. Multiview Paraperspective Projection Model for Diminished Reality. *IEEE Intl. Symposium on Mixed and Augmented Reality* (October 2003).

[125] Zomet, A., and S. Peleg. Efficient Super-resolution and Applications to Mosaics. *Intl. Conf. Pattern Recognition* (2000), Vol I: 579–583.