Integration of Range Images from Multiple Viewpoints into a Particle Database

by

Paul Michael Linhardt

Bachelor of Arts
University of California at Berkeley
1981

Submitted to the Media Arts and Science Section
in Partial Fulfillment of the Requirements of the Degree of

Master of Science in Visual Studies

at the Massachusetts Institute of Technology

February 1989

Signature of the Author _____

Paul Michael Linhardt
Media Arts and Sciences Section
September 20, 1988

Certified by _____

Walter Bender
Principal Research Scientist, Media Laboratory
Thesis Supervisor

Accepted by _____

Stephen A. Benton
Chairman
Departmental Committee on Graduate Students

Integration of Range Images from Multiple Viewpoints into a Particle Database

by

Paul Michael Linhardt

Submitted to the Media Arts and Science Section on September 20, 1988 in partial fulfillment of the requirements of the degree of Master of Science in Visual Studies

Abstract

Movies and films will one day be stored as 3D graphics databases of photographic quality with which one can manipulate the viewpoint, focus, lighting and arrangement of objects in a scene. As a step towards this goal, this thesis examines how particle databases can be built to represent natural objects. Range camera technology already makes it possible to take a 3D image of a scene, but range images have no information about occluded and back faced regions of objects in the scene. To construct a complete representation of an object, it is necessary to combine data taken from several views of the object. The combining of views is significantly complicated by the distortion and noise inherent in real world data. In this thesis, a scalespace correspondence algorithm is proposed to match points in different views and move them towards each other. This algorithm is enhanced with a relaxation technique which guarantees that the points in the views move towards each other with a certain degree of uniformity. The algorithm is evaluated and problems such as occlusions are discussed.

Thesis Supervisor: Walter Bender
Title: Principal Research Scientist, Media Laboratory

The work reported herein was supported by IBM and CPW.

Contents

List of Figures

Chapter 1

Introduction

1.1 Movies of the Future

One of the principal themes of the Electronic Publishing Group at MIT's Media Laboratory is the interpretation of film and video as digital data streams [LB87]. Such a representation would allow the manipulation of film based on its content. Current techniques for manipulations of film and video, such as chroma keying and digital effects wipes, only manipulate 2D images without any knowledge of the context or content of the film or video. The recent process of Colorization(R) is the first commercial process to have a more substantial understanding of the contextual composition of the image[Mar84].

If a three dimensional database of a film were constructed, it would be possible to manipulate the objects in a movie. A number of tools currently available to computer

graphics animators would become available to all filmmakers. A three dimensional movie database would give filmmakers the freedom to change camera viewpoints, camera focus, and lighting in post production instead of limiting these choices to production decisions. It would also be possible to integrate scenes and independent components so that location sets, actors, and props could be filmed independently and combined in post-production. The same technology which will give new degrees of freedom in production, will eventually give this same freedom to manipulate video to consumers. Two paradigms, frequently discussed within the group, give a taste of the potential of spatially organized, interactive video. The first is one of a TV viewer watching a football game while interactively adjusting his viewing position with a joystick at his whim. The second is of a furniture shopper rearranging the furniture in a video picture of his living room to "try out" new furniture he is interested in purchasing.

Research under the aegis of the Movies of the Future project is currently examining how movies can be represented as three dimensional databases [Bov88] [Bov89]. Technology previously used for military and robotic applications has been adapted to this end. In particular, both a laser-scan depth camera and a depth from focus camera are used to create a 3D database of a scene.

1.2 Problem Description

A range camera produces two two-dimensional pixel arrays: one containing intensity values and one containing depth values for each visible point in the 3D scene. A range

image only provides depth information about surfaces which can be seen by the camera. Back surfaces and occluded areas of a scene are not represented in the range image. Using computer graphic techniques it is possible to render a range image from a point of view different than the original viewpoint of the camera. However, there will be "holes" in the image in those areas that were occluded in the original image, but visible from the new viewpoint.

In order to generate an object from an arbitrary viewpoint, it is necessary to have a complete model containing all surfaces of the object which are likely to be viewed. Such a model can be constructed by integrating several range images of an object taken from different viewpoints.

The task involved in combining two overlapping views of an object is deciding which portions of the views are redundant and which contain new data. This can be achieved by establishing a correspondence between pairs of pixels in the two views that represent the same position in the 3D scene.

If range images were "perfect" representations of real world scenes then the correspondence problem would be considerably easier. As it is, noise, arithmetic error and camera distortion (inherent in real world input), create imprecisions in the data which make it difficult to match points in multiple views. An essential consideration in the design of a correspondence algorithm is how well it is able to compensate for noise and distortion present in the data.

1.3 Literature Search

3D Model Building

The issue of combining range images into 3D models has been examined several times in the past. Common to all approaches is the notion of searching for corresponding features. The approaches vary in the data representation they use, the features and criteria used to determine a match, and the search algorithm used to find the match. Many of the following approaches are several years old and pre-date recent advances in storage capability and computing power. It is interesting to note that in most cases, more attention is focused on organizing the data than on the actual process of combining views. While this is partly a reflection of limited computing resources of those times, it is also an indication that the representation chosen can greatly simplify the matching process. My own approach differs considerably from the others in that my representation preserves the facsimile quality of the range data rather than synthesizing and reducing it.

Polygonal Approach

Bhanu uses a three-point seed algorithm to convert the range data into a polygon database [Bha81] [Bha82] [Bha84]. Groups of three nearby points are selected and planar surfaces are grown outward to include points roughly on these planes. The database is examined for points that are in several planes and this information is used to determine the adjacency of surfaces. The circumferences of the planar regions are

found and a polygonal approximation of the boundaries of the faces is established by noting the points of high curvature. Once the range data is polygonalized, a process called hierarchical stochastic labeling is used to repeatedly add views to a model made up of one or more views. Each face of a new view is assigned a label and a probability vector indicating the likelihood that it will match each of the faces in the model. These probabilities are based on comparing several features of polygon faces: (a) the area and perimeter, (b) length of maximum, minimum and average radius vectors from centroid of face, (c) number of vertices in polygon, (d) angle between maximum and minimum radius vectors, and (e) ratio of $area/perimeter^2$. The matching criterion is the weighted sum of the difference of each of these features. Before comparing any two faces, the faces are first aligned by the following transformations: (1) scaling until the areas of the polygons are the same, (2) translating the centroids into alignment, (3) orienting the polygons so that they are on the same plane, and (4) rotating them to obtain the maximum area of intercept. A compatibility function is applied which increases the likelihood of a match between any two faces when their neighboring faces are also likely to match. At each level of the hierarchy more and more neighboring faces are taken into consideration and the probabilities are adjusted accordingly. If the probability that a face from a new view matches none of the faces in the model is greater than that of any particular face in the model then the new face is added to the model.

A particularly desirable feature of Bhanu's algorithm is that it works even when the viewpoint of the view is not known. Another desirable feature is its notion of

probabilistic matching. Bhanu's thesis includes a detailed discussion on the problems of occlusion. The approach is well thought out and integrated into a large body of other work on 3D modeling done at INRIA [JF81] [JDB82] [ODF84] [Hen82] [HB82]. The success of the algorithm is a direct result of the simplification of the database into a manageable number of polygonal faces (e.g. 20 polygons) which unfortunately limits the resolution of the results. As more polygons are added the probability vectors get bigger and the compatibility functions become more complicated. It would be interesting to adapt this approach to a scalespace approach where each successive level of scalespace would have more polygons and information from the previous level could be used to limit the size of the probability vectors.

Surface Approaches

Potmesil [Pot82] [Pot83] represents the range data as a hierarchy of surface segments used for matching and merging. The basic surface element is a bicubic parametric patch. Surfaces are represented by networks of adjacent patches that maintain positional and derivative continuity. These surfaces are converted into a quadtree hierarchical structure by recursively merging four adjacent patches into a new coarser patch at the next level of the tree. Each quadtree node contains a bicubic segment, a bounding volume and an average surface normal vector. Matches are determined by minimizing the distance of shape features of the segments. The matching criteria evaluate differences in position, orientation and curvature of evaluation points representative of the two patches. A technique known as the state-space search method with an evaluation

function is used to intelligently guide the search for matching segments thus reducing the quantity of segments examined to a manageable number. The experimental results on a balsa wood car model show that the algorithm is quite competent on simple convex objects. Potmesil does not address the issue of occlusion.

Dane [Dan82] [DB82] organizes depth data into planar and quadratic surface primitives. Points are grouped into planes and surfaces. A least squared fit on the data points is used to determine quadratic coefficients. A region growing technique is used to expand and combine adjacent surfaces. An edge graph, containing information about the intersection of the surfaces and their adjacencies, is constructed. Cells are constructed with observed properties (number of data points in local area, average and deviation of z-values, average and deviation of x component of normal, and average and deviation of y component of normal) and derived properties (local curvature in x-z plane, local curvature in y-z plane, surface orientation continuity, and surface depth continuity). Surface primitives are grouped into hypersurfaces and classified by type (ellipsoid, hyperboloid, elliptic, paraboloid, etc.). For combining views, surface segments in the new view are compared to all other segments for the best match. Each match is evaluated by some error criteria and if the best match is below a threshold it is accepted; otherwise, the segment is considered to be new data. The match is also examined in the context of the edge graph to see if the adjacencies implicit in the match were consistent with other matches. Many portions of Dane's algorithm were not implemented and he was hindered by working on a 32K machine. Much of his

contribution was in the theoretical evaluation of the problem. As Dane used synthetic data because he did not have access to real world data, his work does not substantially address the critical issue of corresponding distorted data.

Related Fields

A number of fields in computer science use correspondence techniques or address issues that are similar in nature. Ullman [Ull86] and Bajcsy [BB82] examine the process of corresponding similar, but different images. Besl and Jain surveyed relevant techniques in object recognition and model building in 2D intensity and 3D range images[BJ85]. Many researchers in the area of computer vision have addressed the problem of model reconstruction from 2D intensity images. This problem is considerably more difficult than constructing objects from range images because much less shape data is available, but because of the attention given to this problem, many applicable techniques many be found in the literature[BJ85]. Object recognition in both 2D intensity images and 3D ranges images is another fruitful area of computer vision [HB84] [SK85] [TK84]. To recognize an object in an image it is necessary to construct a model of the object and then match surfaces in the image to the model. Since these operations are both necessary for combining views, there is a great deal of overlap in research in the areas of model construction and object recognition. In addition, research in object recognition frequently addresses the issue of occlusion since 3D models must be matched against partially occluded 2D views. In motion detection, a common approach is to

correspond succeeding images which are similar, but slightly different [Hee87] [Hil84] [HG81]. Similarly, shape from motion constructs a model from similar frames in an animated sequence [Ull79].

Media Laboratory Research

V. Michael Bove, Jr. of the Movies of the Future Project is conducting research into the problem of constructing 3D models from range images[Bov89]. Bove's research involves 3D model building through the use of depth from focus and 3D motion estimation. He obtains range images from a depth from focus camera which determines depth information by comparing two images taken from the same camera orientation but with different lens apertures[Bov88]. From a sequence of range images of an object in motion, Bove uses motion estimation to build up a more complete object model through time. Also, structure from motion computed by the motion estimator can be used to refine the depth from focus range values.

While both Bove's and my approach deal with the problem of constructing a 3D model from incomplete data in range images, in his approach the missing data is found in successive frames of a sequence whereas in mine it is provided by multiple cameras at different locations. The two techniques are complementary and in the context of movie production a combination of both may eventually be applied. Because the disparity between frames in a motion sequence is generally small, it is easier to construct 3D models from motion estimation than from combining views; but, in the general case, it

is not possible to guarantee that all object surfaces will become visible in the sequence.

1.4 Statement of Approach

To solve the problem of combining views I have adopted a scalespace approach inspired by the Marr/Poggio stereo pair algorithm [MP79] [Gri81], Laplacian Pyramid coding [OABB85], and the Multigrid methods [Ter84]. The essential concept is that views are aligned first by their large scale features and then successively by their more detailed features.

In this approach, each view[1] is converted to a particle database. Using the known camera geometry and the range information, the perspective is removed and the views are rotated into rough alignment, but distortions are not corrected. To add a new view to the particle database representing the sum of several views, both the new view's particle database and the combined particle database are orthographically projected into image arrays. These similar orthographic views are then filtered with a series of Gaussian filters. Starting with the lowest frequency images, a correspondence algorithm is used to match large patches in both images. The matching criterion used to evaluate each match is based on the squared difference in x and y position, the mean squared difference in z position and the mean squared difference in intensity of the patches being matched. By searching a constrained region in the other image, the correspondence

[1]The term "view" refers to a set containing a reflectivity image, a range image and a text file containing information about the camera geometry of the view. If other attributes, such as surface normals, are used then the term "view" will also include an array with an associated attribute value for each pixel position.

algorithm finds the best match for each patch (unless no good match is found). All patches that are matched are assigned an offset vector which moves that patch towards the position of the matching patch in the other image. A relaxation algorithm is used to correct errors and assign values to unmatched patches, so that all neighboring patches tend to move in the same direction. This process is repeated on each successive level of scalespace with higher frequency images and smaller patches. The offset vectors of the previous level are used to find the best regions to search for matches at the current level. The final pass is at the pixel level; the final offset vectors are added to the (x,y,z) positions of the associated particles and the two databases are merged.

1.5 Thesis Organization

Chapter one contains an overview and the problem description. Chapter two focuses on data related issues including data acquisition and database organization. Chapter three, which contains the crux of the theory, discusses scalespace, correspondence and relaxation. Chapter four presents photographs of experimental results and evaluates them. Chapter five makes suggestions for further research including the problem of occlusion.

Not all of the algorithms described in the following chapters have been fully implemented and many refinements are possible on those that have. In particular, the scalespace algorithm for merging views has been implemented but no attempt has been made to combine more than two views into a particle database.

Chapter 2

Data Acquisition and Representation

Before processing 3D image data, it is necessary to have a good understanding of the characteristics of the input data and select a representation suitable for processing. Input data is obtained using a range camera to generate a pair of image arrays containing intensity and depth information. The key to interpreting this data is carefully calibrating the camera and recording pertinent scene measurements. Even with these precautions, distortions enter into the data.

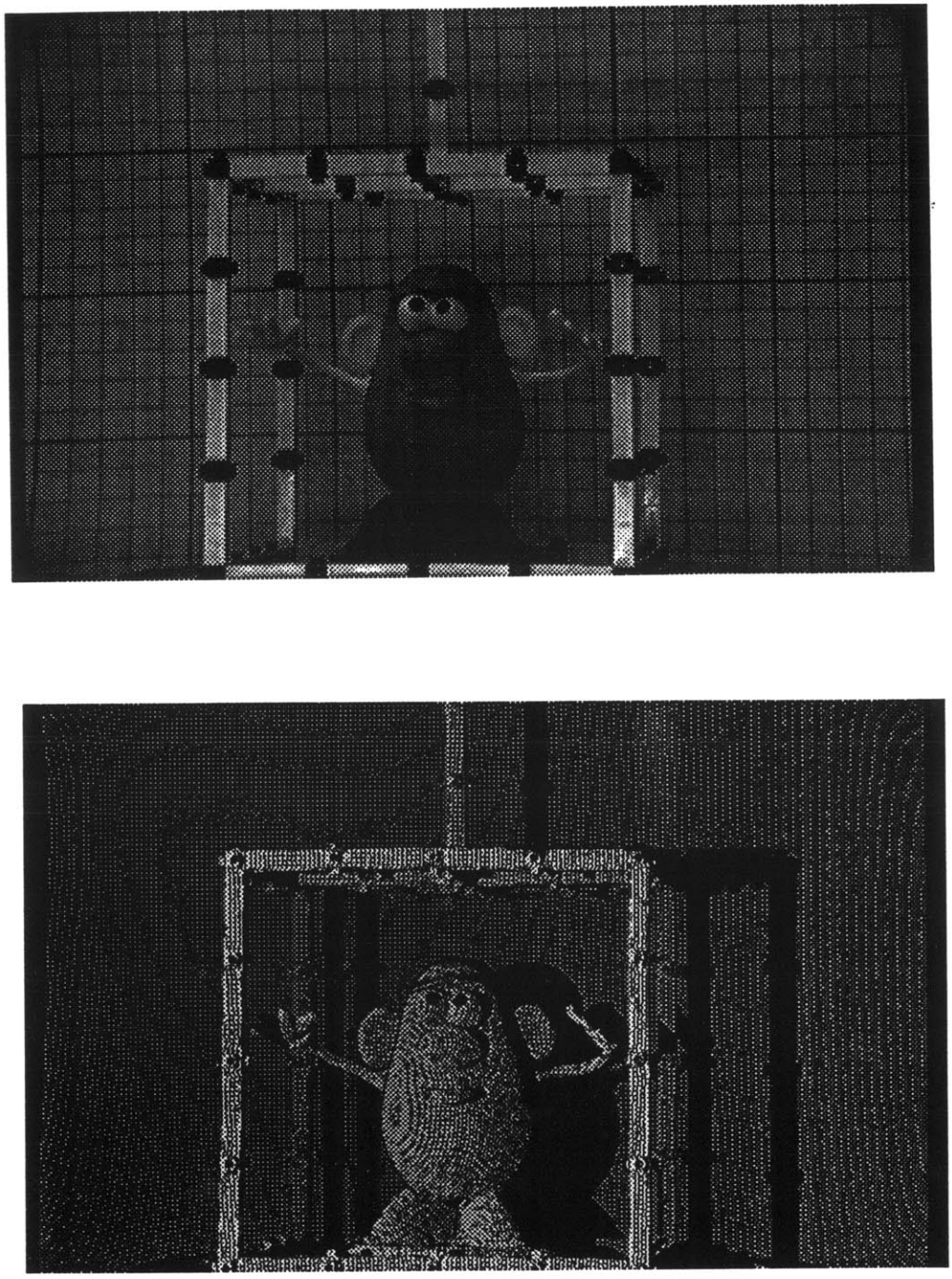


Figure 2-1: A sample of a reflectivity and range image pair.

2.1 Range Cameras

Range Data

A range camera produces 3D facsimile databases of real world scenes. When digitizing a 2D image from a camera, an image is divided into an array of pixels and each pixel is assigned an intensity value. The output of a range camera is two such arrays, one containing intensity values and one containing depth (z) values. The horizontal (x) and vertical (y) values are not explicitly included in the data, but are implicit in the scan conversion of the image.

Figure 2-1 presents a reflectivity and range image of a scene containing Mr. Potato Head inside a cube shaped cage made out of Construx. This pair is an example of the basic unit of information referred to as a "view". It is the output of the a range camera and the input to the correspondence algorithm.

Range images are similar to contour maps. The brighter a point in the image, the closer it is to the camera. Where no depth value is available for a pixel, the pixel is arbitrarily assigned the value zero.

The Laser Range Camera

Input data for this project was generated by a laser-scan depth camera [Jar83]. The laser-scan depth camera uses the disparity between the position of the camera and the position of the laser to triangulate the distance to points in the image. The laser-scan

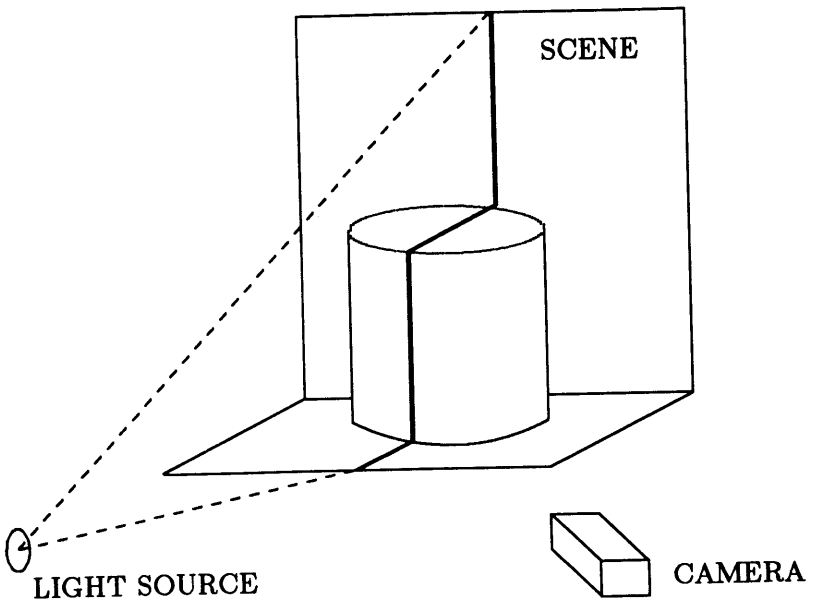


Figure 2-2: Diagram of a laser-scan depth camera (from [Bov89])

depth camera combines a laser, a cylindrical lens and a computer-driven rotating mirror to scan a narrow vertical stripe of light across a scene. The stripe is seen by a video camera placed at a small horizontal distance from the light source, with the result that on each scan line the beam will appear to be shifted horizontally as a function of the depth of the object it hits at that point. At each position of the stripe, the computer calculates the depth of each point on the stripe as a function of the offset caused by the horizontal parallax[Bov89].

Other Sources of Range Data

The algorithms described in Chapter 3 for combining range images are designed to be independent of the data acquisition method. While I specifically used a laser range

camera to acquire input data, the data could be obtained through a variety of other available techniques. Jarvis presents an excellent survey of these techniques in [Jar83]. There are a large number of algorithms that determine depth from stereopsis, the process of comparing two images from different viewpoints [MP76] [MP79] [Gri81] [Pra85]. Depth from motion, the process of determining the shape of an object from a series of images of the object in motion, is another technique modeled after the human visual system [Ull79]. Depth from shading is a technique in which shape is determined from variations in intensity in the image [Pen87a]. In depth from focus, a camera takes two images from the same position, one with a long depth of field and one with a short depth of field. The difference between how much the image is blurred in the two images can be used to calculate the depth at each point [Pen87a] [Bov88].

Each of these techniques has advantages and disadvantages. Some (such as depth from shading) are better for determining relative rather than absolute depth. As with the human visual system, there is a great advantage to combining some of these techniques to obtain more information with greater reliability. Bove's approach, for instance, combines both depth from focus and depth from motion information [Bov89].

Laser-range data was used in this project because it has the best accuracy and resolution of the choices currently available. The resolution of the depth is limited by the minimum amount of parallax of the laser beam that can be observed (one pixel's width). However, the unit of depth represented by the displacement of the beam by a single pixel's width varies in relation to the distance between the camera and the laser.

It follows that it is possible to detect very small variations in depth by separating the camera and the laser by the appropriate distance. Resolution in depth should be on the same order of magnitude as the horizontal and vertical resolution of the images (e.g. 256 depth steps for a 768 x 480 pixel image). In practice, it is convenient to select a camera geometry that maps the range of depth represented in the scene onto an 8 bit range.

One disadvantage of the laser-scan depth camera is the double occlusions inherent in this technique. Not only do objects occlude other objects from the camera's point of view, but also from the laser's viewpoint. In addition, the greater the distance between the camera and the laser, the greater the occlusions (the price of increasing the resolution). While the problem of double occlusions also exists in stereo depth perception, other methods such as depth from focus are free of this defect.

The laser-scan depth camera is an example of an active technique because an agent (the laser beam) is entered into the scene to obtain the depth information. The other techniques mentioned above (all inspired by the human visual system) are passive because they operate on the images without affecting the scene. While laser-scan depth data is appropriate to developing the techniques of combining views, the technique is not appropriate for filming movies. Besides the inconvenience of having the active agent enter into the scene, laser-scanning an object is a time consuming process[1]. The other passive techniques could be applied to source material that was recorded at realtime

[1] Typically it takes 10 minutes to scan in a view.

and then later analyzed for depth information. It is anticipated that the accuracy of passive techniques will improve considerably and that the issues discussed in this thesis will eventually be applied to passive ranging technology.

2.2 Range Image Distortion

In theory, it should be possible to align several views of an object on a turntable simply by rotating the points in each view in the opposite direction of the rotation of the turntable. If range images were as perfect as synthetic data then combining views would entail no more than simple rotations. In practice, a large number of real world distortions enter into the range data. The most obvious distortion is created by the perspective transformation of the scene onto the image sensor. Parts of the object close to the camera appear bigger in the image and, conversely, those far away appear smaller (as in the cage in photo 2-1). If the exact camera geometry is known then removing perspective is trivial [FD82] [Smi83]. However, the measurements necessary for recovering the camera geometry are subject to human error. In practice it is possible to remove most, but not all, of the distortion caused by perspective. Removing perspective using an incorrect camera geometry will result in new distortions in the data. Tsai [Tsa86] and Mansbach [Man86] itemize a number of other errors that are common to range camera images. These include: radial lens distortion (aka barrel distortion), CCD scanning timing errors, nodal separation, raster rotation, raster translation, camera noise and quantization errors. Other errors, specific to the laser-scan depth camera,

include: non-linearity in the laser motion, inaccuracy of the laser motion, non-linearity in the x/y scanning of the camera, distortions in the cylindrical lens of the laser, non-uniformity of the laser beam, and distortions in the rotating mirror. This list is not complete and is meant to impress the reader with the quantity of errors that can enter into data scanned from a real world scene.

Some errors, such as barrel distortion, can be corrected if one has a firm understanding of the mathematical cause of the error. Other errors, such as camera noise, are random in nature. Many distortions occur primarily at the edges of images and do not significantly effect objects scanned in the middle of the screen (e.g. barrel distortion). Using mathematical models to remove distortions is only efficient for the most significant among them. Based on experience with experimental data, I consider 5% uncorrected distortion to be acceptable. In practice, the only distortion I find necessary to correct is perspective distortion.

If a correspondence is known between a set of real world coordinates and a set of range image coordinates, interpolation can be used to correct the distortion in all other image points. If a calibration object covering the screen, such as the cage, is used in an image, the distortions in the image could be corrected and then the cage could be replaced by another object in the scene. Tsai proposes some complex models for removing distortions along these lines [Tsa86]. This type of correction shows no real understanding of the cause of the distortions and hence is more effective when the set of corresponding worldspace/image coordinates is large. The more measurements

necessary to satisfactorily implement the scheme, the less generally useful it is.

2.3 Lighting

Lighting poses a number of problems. Since intensity is the principal attribute used to match features, it is desirable that the intensity of each point on the surface of the object is the same as seen from all views. Ideally, all views would be illuminated by the same perfectly diffuse, ambient light. Specular reflections in the image should be avoided because they are more dependent on the position of the light source and the camera than on the surface characteristics of the object. Light from directional light sources create some problems when the object, rather than the camera, is moved. If the object is rotated on a turntable, but the camera and lighting source remain invariant, then the shadows and intensities of points on the object will vary considerably. Such conditions make matching featured based on intensity difficult. If the light sources are rotated with the object (or the camera and laser are rotated around a stationary object with fixed light sources) then the intensities of each object point will remain constant, improving the reliability of the matching algorithm.

2.4 Camera Setup

Camera Calibration

The calibration process is used to align the coordinate systems used by the camera, the laser and the object space. To obtain additional views, either the camera or the object must be rotated[2]. It is essential that the relationship between the camera, object, laser and axes of rotation is known. A convenient approach is to place the object on a turntable whose axis of rotation is parallel to the vertical axis of both the camera and laser. Implementation details may require other alignments between the camera, laser and features of the scene. For instance, one requirement specific to my implementation is that the line between the camera and the right edge of the image must be perpendicular to the back wall of the scene. While it is not difficult to calibrate the camera so that it is possible to interpret the range data reasonably well, human error is always present in the calibration process.

Measurements

In order to recover real world coordinates from the range image, it is necessary to record the camera geometry at the time of scanning the object. A text file containing vital measurements of the geometry of the view is associated with every pair of range and reflectivity files for this purpose. An example of the text file is presented in appendix A.

[2]Or alternatively, several cameras may be placed around a scene taking into consideration similar alignment issues.

Since a camera creates a perspective image of a scene, the viewing angle of the camera is essential to recovering the 3D position of the points in the image[3]. It is also necessary to have at least one measurement in real world coordinates (e.g. the distance from the camera to the back wall in centimeters). The width and height in centimeters of the back wall of the image can be calculated from this number and the viewing angle. It is advantageous to map the 256 depth steps onto a small range of centimeters containing the object. A bias and gain for converting depth steps to centimeters can be computed from any two points in the scene for which both the z-value and the distance from the camera are known (e.g. the back wall and the front-most point of the object). Information about the display is also required: screen height and width in pixels, number of depth steps, screen aspect ratio and pixel aspect ratio. The more assumptions that can be made about the camera geometry, the less measurements required, but also the less general the camera configuration.

Several views of each object are taken by rotating the object on a turntable using the same camera geometry. To rotate the views into alignment it is necessary to record the x and z coordinates of the vertical axis that goes through the center of the turntable. The axis of the turntable is assumed to be aligned with the y-axis of worldspace[4]. Lastly, one must record how many degrees each object has been rotated in each view.

[3]The horizontal viewing angle is sufficient since the vertical viewing angle can be deduced from the aspect ratio.

[4]The term "worldspace" refers to rectangular coordinate system based in the real world whose origin is at the position of the camera and whose units are measured in centimeters.

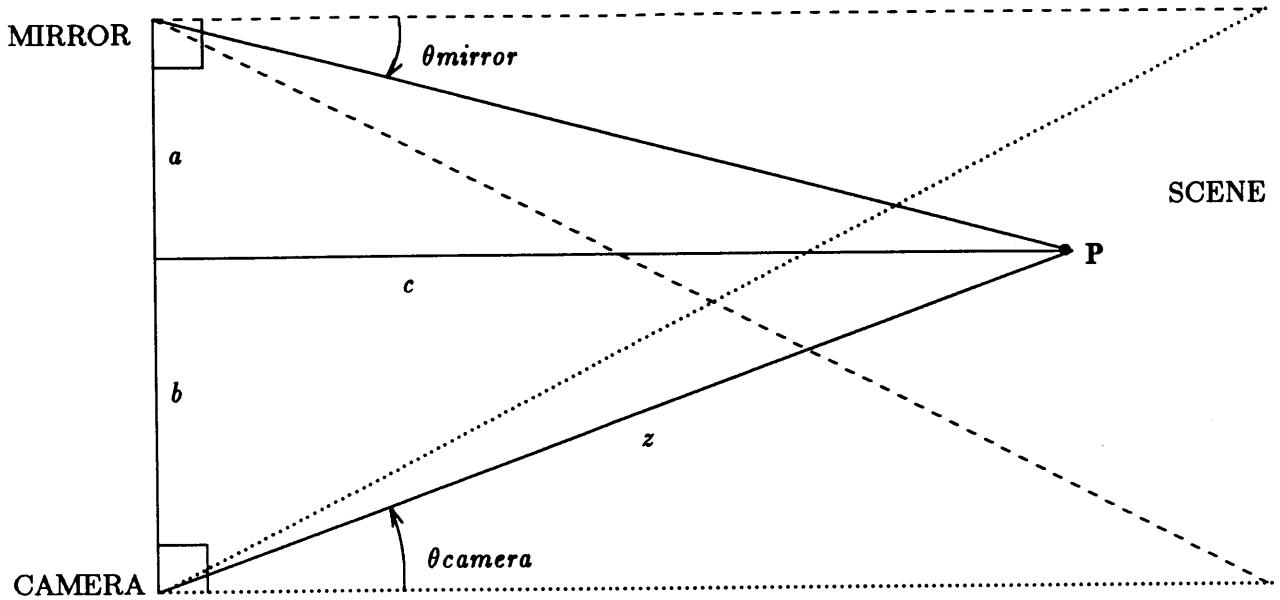


Figure 2-3: Diagram of alignment of turntable, camera and laser (from [Bov89])

Object Selection

The selection of objects for the correspondence test reflects the limitations of the laser-range camera and the correspondence algorithm. The object should not be too small so that the coarse size of the laser beam would be sufficient. Thin stringy textures such as hair cannot be accurately detected by the coarse beam. Objects with many small holes and tiny self-occlusions are problematic because the laser cannot get into all the little nooks and crannies. Dark colors like black and dark blue should not be used because they do not reflect the infrared laser sufficiently for the camera detection. On shiny objects, particularly metals, the beam bounces off the material confusing the program detecting the parallax.

2.5 Particle Databases

The range camera data structure consisting of two 2D arrays containing intensity and range data is not sufficient for constructing complete 3D models of objects and scenes. In order to "plug holes" caused by occlusions it is necessary to select a data representation that can store the information obtained from several different views. The 2D array format is not appropriate because each x and y coordinate pair can only have one z-value in a range image. Any data representation used for this problem must be able to represent objects such as a cube where an entire surface is parallel to the z-axis and many points share the same x and y coordinates.

Several types of representations are possible. Voxels (volume elements) are the three dimensional analogy to pixels (picture elements). In a voxel representation, space is divided up into rectangular, elemental volume elements where the total number of elements is the width times the height times the depth. The data size of the range/reflectivity images pairs I use are 2 x 768 x 512 bytes. An equivalent voxel space would contain 2 x 768 x 512 x 256 bytes. Clearly, this representation is expensive particularly since a great percentage of the voxel space will be empty. There are some similar representations such as oct-trees which are useful for compressing voxel representations, but since we are more interested in the surface of the volume rather than its interior this representation is not very efficient. Some work has been done on this problem using cubic patches [Pot82] and polygon representations [JDB82] [SBD86]. B-splines and superquadrics [Pen87b] are other possible candidates.

The representation format that I adopted was a particle database [Ree83]. A particle database is a list of particles. A particle contains an x coordinate, a y coordinate, a z coordinate, an intensity value and possibly a number of other attributes such as a surface normal and color characteristics[5]. It may also be useful to include a pointer to a file containing information about the camera geometry of the original view of the particle. It is a straight forward matter to render a particle database using the perspective transformation [Smi83]. Lighting models [CT82] can be applied (if surface normal information is available) as well as a great number of other common computer graphic techniques [FD82].

The advantage of the particle database is that it maintains the detailed information of the original range images. Other models such as polygon sets are generally used to reduce the size of the data which tends to reduce the photographic quality of the data. Unlike the voxel and image representations, the particle database does not take up storage space for portions of the scene that contain no relevant information (empty cells). There is, however, a high storage cost for the x and y coordinates which have become explicit rather than implicit as they were in image arrays. The exact size of a particle database depends on the scene or object in question and the attributes associated with the particles. In practice, a view occupies 2 x 768 x 512 bytes and the equivalent particle database costs 8 x P bytes where P is the number of particles and each particle contains four 2-byte values (x, y, z and intensity). Thus if the object

[5]Reeves discusses particle systems where particles also contain dynamic attributes which vary such as velocity and life span, but the particles I use do not, at present, incorporate these kinds of attributes.

covers 25% of the screen, the particle database requires the same amount of space as the view.

A particle database may be over-specified or under-specified. If there is too much redundant data (after adding several views) then the worldspace can be quantized and adjacent particles can be combined. If, on the other hand, some surfaces are too sparsely represented, there are techniques available for generating additional particles by taking advantage of the smoothness constraint [Gri81].

Chapter 3

The Scalespace Correspondence Algorithm

Overview

This chapter discusses the scalespace correspondence algorithm for combining views into a particle database. Correspondence is a technique for matching points in two images that are similar, but slightly different. It is applied to the problem of combining views in order to reduce distortions and align the points in the different views. This alignment is essential for determining exactly where to add the particles from the new view into the database. Scalespace is a term for describing a space in which operations can be performed successively on versions of images at varying resolutions. The essence of the scalespace correspondence approach is that large scale features in the views are

matched first and then this information is used to locate the matches in smaller scale features of the views.

The operations comprising the algorithm are filtering, searching and comparing. These operations are computationally expensive and require much memory. The correspondence algorithm would be most effective if it were applied to a three dimensional voxel representation of the scene. However, in order to reduce computational and memory requirements to an acceptable level, I have chosen to perform the correspondence operations on two-dimensional projections of the 3D data. This effectively reduces the technique from an order N^3 to an order N^2 algorithm, at the cost of some side-effects concerning occlusion. Implications of working in 2D rather than 3D will be discussed in chapter 4.

When adding a new view to a particle database containing one or more views, the first step is to project the database into a 2D plane along the same viewing axis as that of the new view. This produces two views rendered from same viewpoint which can then be corresponded. These views are filtered with a series of low pass filters to build a scalespace. Beginning with the lowest resolution images, the patches in the views are compared and corresponded. Matching criteria, based on position and intensity of patches, are used to evaluate the potential correspondences. A search algorithm is applied to determine where to look for the best match for a patch. Once correspondences are established between patches in the two views, offset vectors are assigned to move the patches towards each other and into alignment. After all patches

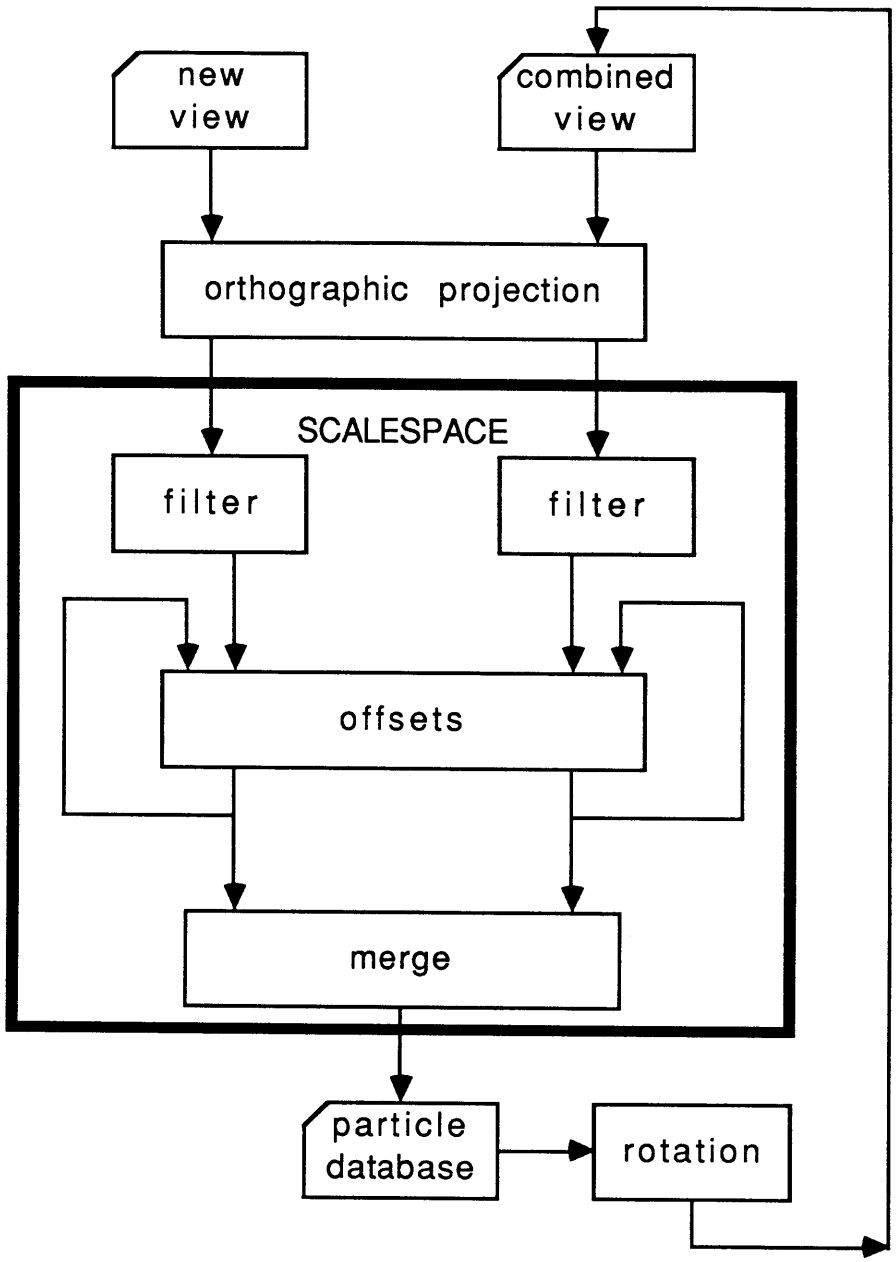


Figure 3-1: Block Diagram of Correspondence Algorithm

have been assigned offset vectors, a relaxation algorithm is applied to the vectors so that all vectors are similar to the vectors for neighboring patches. Relaxation ensures local smoothness, corrects for matching errors and assigns offset vectors to unmatched patches. The procedure for determining offset vectors for patches is repeated at each level of scalespace with smaller patches. The offset vectors from the previous level are used to determine where the search algorithm looks for matches at the current level. In this way, matching information is passed from level to level. At the final level of scalespace, the correspondence is performed on the original views with the patch size being a single pixel. The output of the correspondence algorithm is an offset vector for each pixel in each of the two views. The new view and the particle database are merged by converting the new view into a particle database, adding the associated offset vectors to particles in both databases and then appending the new view's database to the old one. This process is repeated for each new view added to the database.

3.1 Scalespace

Overview

A scalespace approach is a technique where an operation is performed on several different versions of an image at different levels of detail. Once a task is completed on a low resolution image, the results of the analysis can be used to evaluated a higher resolution image at the next level of scalespace. Complex processes can be performed

with greater efficiency and reliability in scalespace than when applied directly to high resolution images [OABB85] [Gri81].

For the problem of combining range images, different views of an object can be rotated into the same orientation and then matched in scalespace. If the original views overlapped considerably then the rotated views will be very similar. As in the case of the stereo pairs, large scale features in low resolution images will be matched first and the results used to successively match more detailed features at subsequent levels of scalespace. At the end of the process, the correspondence at the pixel level makes it possible to correct distortions and merge the two views into a coherent model.

View Preparation

Consider the problem of adding a new view to a particle database that already contains one or more views[1]. Before filtering, several preprocessing steps must be applied in order to make the views as similar as possible for comparison. First, the background behind the object in the new view must be removed. This can be done by depth keying, a process where all pixels with a depth value less than a certain threshold are separated from the object.

The next step is to remove the perspective on the new view and convert it into a particle database. The new particle database is orthographically projected into the image arrays (using the original axis of projection). The same orthographic projection

[1]The following discussion assumes the views are of an object rotated on a turntable and scanned by a stationary camera, but is generally applicable to other configurations.

is applied to the combined particle database using the new view's axis of projection[2].

When projecting a particle database of an object into an image array, not all the pixels in the array are affected. This is because of sparseness of data, occluded regions and the eliminated background. In order to filter an image, the image must be continuous in x and y; thus, every pixel location must have a defined value. Sparseness occurs when surfaces originally pointing away from the camera are rotated to face towards the camera position. The few points needed to represent a surface from the side view are not sufficient to cover its area as seen from the front. Perspective removal also causes sparseness because surfaces that were far away from the camera are expanded to cover larger areas of the screen. A number of techniques can be used to fill sparse surfaces. I used an iterative median fill program which assigned the median value of its defined neighbors to undefined pixels with enough surrounding neighbors. All remaining pixels can be assigned an arbitrary background value (such as an intensity and depth from the original background). Occluded regions are very difficult to identify and pose many problems in filtering and matching. In some cases they are filled, in others, they are assigned the background value.

The image arrays for each attribute used by the matching criteria are prepared for filtering in the same fashion. Depth can be treated as an attribute of a two dimensional array in the same way as intensity. In a two dimensional range image, each pixel can have any value for the depth attribute of the two dimensional space. Like intensity,

[2]Only particles that are visible from the new viewpoint should be projected. This implies that the surface normals should be kept to determine which particles face towards the viewpoint. This implementation ignores the problem of back-faced particles and projects all particles.

depth information is locally smooth, but can provide both high and low frequency information which makes it a good candidate for scalespace matching. Depth is largely independent of intensity so it makes a good complementary criterion for matching. Depth data has advantages over surface normal information in that it is more reliable and more compact. Unit surface normals can be derived from range data and require two coordinates.

The result of this preparation is two similar orthographic views of an object rendered from the same viewpoint.

Filtering and Subsampling

The next step is to filter the orthographic views with a series of Gaussian filters ($\pi/4$, $\pi/16$, and $\pi/64$). This effectively generates a series of images at full resolution, 1/4 resolution, 1/16 resolution and 1/64 resolution. The views are divided into square patches which will be matched and assigned offset vectors. The size of the patches depends on the filter used. The $\pi/64$, the $\pi/16$, and the $\pi/4$ filtered images are broken up into 64x64, 16x16 and 4x4 pixel patches respectively. Since the filters remove high frequency information from the images, the patch size is directly tied to the effective resolution of the filtered image. For example, in the case of a $\pi/64$ filter, the effective resolution of the filtered image has been reduced by a factor of 64 in both directions so the patch size is 64 x 64 pixels. At the last level of scalespace, the patch size is 1x1 and the original views are corresponded on a pixel by pixel basis.

Starting with the $\pi/64$ filtered views, a search algorithm is applied to find the best match for each patch in the other view. Once a match is found, an offset vector is assigned to adjust the x, y and z position of all points in that patch. This process is repeated for the $\pi/16$ filtered views, then the $\pi/4$ filtered views, and then finally the original views, each time using successively smaller patches. At each level of scalespace, the offset vectors of the patches at the previous level are used to guide the search for the best match of patches at the current level.

From Nyquist's theorem it is clear that the essential information in a patch of a filtered image can be represented by a single pixel. It is possible to subsample the views according to the filters used and compare pixels rather than patches in the correspondence algorithm. Subsampling can simplify many computations. However, it poses a problem for the search algorithm comparing the patches. Specifically, it is not sufficient to compare patches in one view only to patches that are along patch boundaries in the other view[3]. As the views are roughly aligned by the initial rotation, it is necessary to be able to move large patches small distances. Thus, on subsampled views, it must be possible to compare and displace pixels on a sub-pixel scale [JJ81]. This can be achieved by interpolating sub-pixel values. While my implementation compares patches of full-size filtered views, it is equivalent to comparing pixels of subsampled filtered views.

[3]e.g. In a $\pi/64$ filtered view, patch boundaries occur at pixel rows or columns that are multiples of 64.

Filtering Problem

The experimental results indicate some problems with this filtering scheme (examples are provided in chapter 4). When a particle database is projected into an image array, not all pixels in the array are assigned values. In order to filter an image, all pixels must have some value. Assume for the moment that these unused pixels are assigned the arbitrary value zero. When an image is filtered, neighboring pixels are averaged. Either sparsity of data or occlusions can create undesirable effects in filtered images. Normalization of the filtered images will compensate for the effects of sparsity. The filtering problems caused by occlusions cannot be easily resolved.

3.2 Implementation

This section describes the implementation of the correspondence algorithm which is applied successively at each level of scalespace. The input to this algorithm is the filtered views associated with the current level and the offset vectors from the previous level. These two elements combined give the state of the views after corrections for distortions were made at the previous scale. The output of the correspondence algorithm is the offset vectors for the current level. These offset vectors are used as input to the next level; at the final level, the offset vectors are added to the particles from the original views to correct distortions and align the two views.

Two things characterize any correspondence algorithm: the matching criteria and the search algorithm. The matching criteria are the evaluations used to determine how

well a patch from one view matches a patch from the other. The search algorithm determines which patches are tested for a match. Once a correspondence is found for a patch, the patch is assigned an offset vector which is used to move the patch towards the position of its corresponding patch. After all patches are assigned offset vectors, a process called relaxation averages adjacent offset vectors thus guaranteeing a consistency of displacement.

Matching Criteria

There are many different attributes which can be used for corresponding patches in two views. In a 2D intensity image it is difficult to distinguish between contour edges and sudden changes of intensity on flat surfaces. Zero-crossings of the 2nd derivative of intensity are often used to extract intensity and shape features from intensity images[MH80] [Mar82]. Since the depth information provided by a range image is much more substantive than the contour information available from zero-crossings of intensity, it is appropriate to use a matching criterion that takes advantage of both the surface and intensity information available.

A combination of positional and intensity features derived from the original data can be used. For example: average intensity and/or depth in a patch, deviation of intensity and/or depth, average and/or deviation of surface normals. Similar functions of other attributes such as color can be used if that attribute data is available. The more criteria used, the greater the reliability of the matches. At the same time, reliability

must be balanced with simplicity and computational efficiency. The matching criterion is a critical function, extensively applied during the correspondence algorithm.

Criteria Equation

I have adopted a scheme using (1) displacement in x and y, (2) mean squared error in z and (3) mean squared error in intensity. Each time I compare a patch from one view to a patch in the other view I calculate an error value for that comparison[4]. The smaller the error value the better the match. The error value is the weighted sum of three criteria errors: (1) the square of the sum of the difference in x and the difference in y of the position of the two patches:

$$Error_{pos} = (dx^2 + dy^2) * Weight_{pos} \tag{3.1}$$

(2) the sum of the square of the difference between z values in the same pixel location in both patches divided by the number of pixels that are active in both patches:

$$Error_z = \frac{\sum(range1[x][y] - range2[x][y])^2 * Weight_z}{density} \tag{3.2}$$

[4]These comparisons are based on the positions of the patches after taking into account the offset vectors from the previous level of scalespace. For simplicity, the displacement of the patches is left out of the equations.

and (3) the sum of the square of the difference between intensity values in the same pixel location in both patches divided by the number of pixels active in both patches:

$$Error_i = \frac{\sum(refl1[x][y] - refl2[x][y])^2 * Weight_i}{density} \tag{3.3}$$

Thus the final error value is the sum of these criteria:

$$Error = Error_{pos} + Error_z + Error_i \tag{3.4}$$

As a weight is associated with each of the three criteria, it is easy to change the emphasis on the individual attributes. Different objects have different properties and distinguishing features. A white plaster bust (photo 4-4) has little intensity information, but many surface features. A world globe (photo 4-2) has a uniform surface and contains distinguishing features in the printing on the surface. If prior knowledge can be assumed about the types of objects that will be in a scene, it is possible to emphasize the distinctive attributes of those types of objects.

Threshold

Using the matching criterion to compare any two patches produces an error value which is the measure of the dissimilarity between those two patches. The smaller the error value, the greater the similarity between the patches. When searching for a match, a patch is compared to number of patches in the other view and the comparison resulting

in the smallest error value is the best match. This error value must be less than a threshold or else the match is rejected and the original patch is considered to have no match. The value of the threshold must reflect the maximum amount of distortion expected in the original views. A common sense notion of what distortion is acceptable should be the basis of the threshold value. A different threshold value may be used for each level of scalespace.

Density

The density of the patches must be taken into account during the matching process. In this discussion, density refers to the ratio of non-zero pixels to the total number of possible pixels in the patch. It is not necessary that both patches have the same density of pixels because density is not a surface characteristic but rather depends on the camera geometry. Nonetheless, enough points must exist in both patches to make the comparison valid. The individual density of each patch is not important; what is important is that a large number of pixels exist in the same pixel locations in both patches since the error criterion is based on the difference of pixel values. If a small percentage of the pixel positions are represented in both patches then the patches cannot be considered to match because of insufficient information[5]. The relationship between density and the matching criterion is worthy of further consideration.

[5]My implementation rejects comparisons with less than 25% density. The choice of 25% is based on experimental results and is otherwise arbitrary.

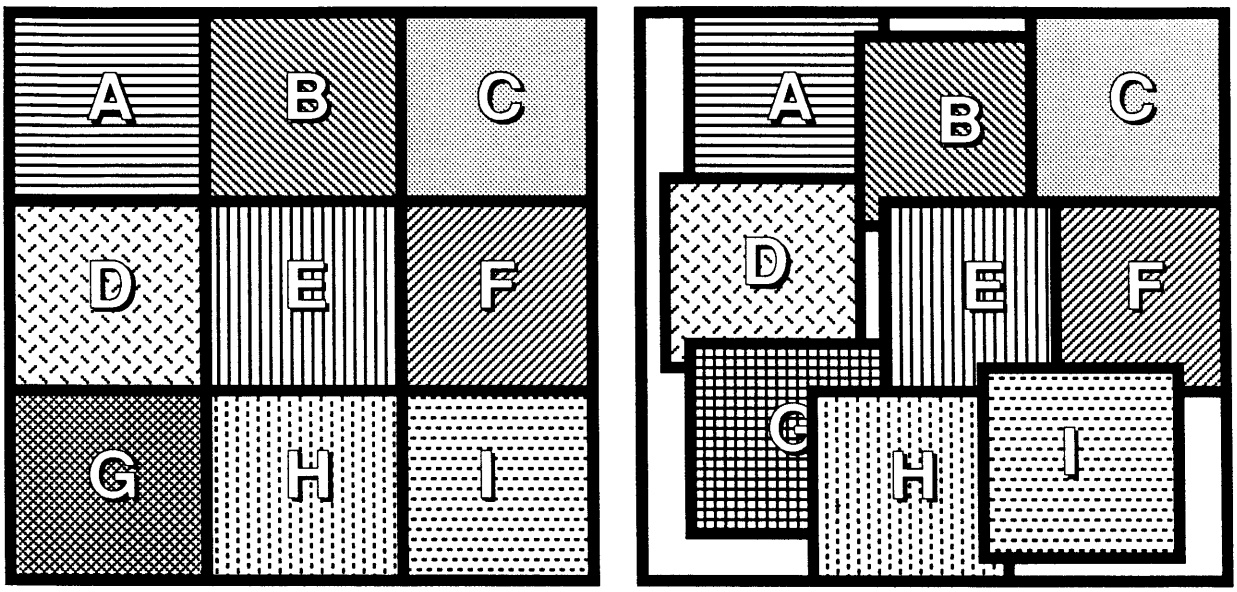


Figure 3-2: Overlapping patches

Figure 3.2 shows that when the patches in view A are matched to patches in view B, some pixels in view B are included in several overlapping patches while others are not included in any patches. This demonstrates why the correspondence algorithm must be applied twice (once in each direction) in order to determine offsets for all the pixels in both views.

Search Algorithm

Description

The search algorithm is responsible for finding the best match for each patch in the other view. To find a match for a patch, the search algorithm applies the matching criterion to a number of patches in the other view and selects the patch with the smallest error value.

The first view is divided into patches along patch boundaries. For each of these patches, a rectangular window of the other view is searched to find the best match. It is important to note that matching patches found in the second view are generally not aligned on patch boundaries (see figure 3.2). This is because the distortions in the views are small so the matching patches are generally not more than a few pixels away from the position of the original patch. The matching patch in the second view can be located at any pixel location.

One important feature of the search algorithm is that information from the search done at the last level of scalespace is use to guide the search at the current level. The output of the correspondence algorithm is an offset vector for each patch. The search algorithm determines correspondences between patches. Offset vectors are assigned to each patch to move patches towards their corresponding patch, thus reducing the distortions in the views. The offset vectors from the previous level of scalespace must be added to the views before applying the search algorithm to the current level. This creates a cumulative effect of first correcting big distortions and then correcting suc-

cessively smaller distortions. In terms of the search algorithm, this cumulative effect means that at successive levels of scalespace, matching patches will tend to move closer and closer to their correct position.

The correspondence algorithm does not establish a one-to-one relationship between patches in the two views. To correct the distortions in both views, it is necessary to apply the correspondence algorithm twice: first corresponding the first view to the second and then corresponding the second view to the first.

Computational Characteristics

Searching for matching patches is the primary activity of the process of combining views. For this reason, it is important to optimize the search task by intelligently guiding the search.

Consider the cost of matching one view to another at an arbitrary level of scalespace (the level determines the patch size). This cost can be represented by the following equation:

$$< MatchingCost >=< NumberPatches > * < TestCost > * < TestsPerMatch > \tag{3.5}$$

Let pH and pW be the patch height and patch width then:

$$< NumberPatches >= H * W/pH * pW \tag{3.6}$$

Let k1 be the overhead associated with a comparison regardless of the patch size and k2 be the overhead associated with pixel comparison in the patch[6] Then:

$$< TestCost >= k1 + k2 * pH * pW \tag{3.7}$$

So:

$$< MatchingCost >= (H * W/pH * pW) * (k1 + k2 * pH * pW) * < TestsPerMatch > \tag{3.8}$$

If the patch size is big, this number k1 is insignificant:

$$< BigPatchMatchingCost >= H * W * k2 * < TestsPerMatch > \tag{3.9}$$

On the last pass of scalespace the patch size is a single pixel (pH=pW=1) so:

$$< PixelMatchingCost >= H * W * (k1 + k2) * < TestsPerMatch > \tag{3.10}$$

It is clear from these equations that real savings in the computation can be achieved if an intelligent search technique is used that minimizes the number of tests done per match.

[6]K1 is essentially the cost of computing $Error_{pos}$ plus the cost of the divisions used in $Error_i$ and $Error_z$. K2 is the cost of doing the operations inside the summations for $Error_i$ and $Error_z$.

Optimization Considerations

One way to ensure that the best match is found is to test each patch against all other patches in the other view. This is both computationally intensive and unnecessary. Since our initial assumption is that the two views are roughly aligned to begin with, the search need only examine patches within the region containing all possible acceptable distortions.

The size of the search region can be calculated from our assumptions about the distortions and the pixel size of the object. For example, if we assume that there is no more than a 5% distortion in the data and that the object is 400 pixels wide, the search window only needs to be 20 pixels wide, so the matching patch is +/-10 pixels in either direction.

Since the threshold of the matching criterion is also set based on our assumptions about how much distortion is acceptable there is a relationship between the threshold and the search window size. Specifically, the search window should not include patches which would automatically be rejected because their x-y position makes their error value greater than the threshold.

For the particular turntable/camera geometry that I have been using, it is advantageous to have the search window much smaller vertically than it is wide horizontally. For simplicity, I have implemented a rectangular search window although there is no reason why the window could not be circular or elliptical. Algorithms that emulate human stereo matching allow an epipolar constraint on the matching[MP79]. This

constraint states that since in human vision the two eyes are vertically aligned, points in the stereo pair will be vertically aligned as well so it is sufficient to search only for horizontal disparity. If a similar restraint is put on the range camera geometry, namely that the camera is set at the same height for all views, then it is possible to reduce the search window vertically to a few pixels of height while maintaining a broad width. This constraint is useful for constructing models of an object rotated on a turntable, but an unreasonable constraint in the case where several cameras are set up at different locations throughout a scene.

An essential feature of the scalespace approach is that the largest scale distortions are corrected on the lowest frequency views and that successively smaller distortions are corrected on successively higher frequency views. The views are coarsely aligned after the first pass of scalespace and at each successive pass they are more finely aligned. This means that the search window size can be reduced significantly after each pass. If the search window size is 20 pixels in width for the $\pi/64$ filtered views then it can be 20/4, 20/16 and 20/64 pixels wide for the $\pi/16$ filtered, $\pi/4$ filtered and original views respectively. Slightly wider windows should be used to allow for errors at the previous level.

Bi-directional Matching

The search algorithm finds the best match for the patches in the first view and then, independently, finds the best match for the patches in the second view. It would be convenient to be able to establish a one-to-one mapping between the patches in two

views because this would halve the number of calculations. However, this is not possible given the assumption that it must be possible to match a large patch with a patch in the other view that is only a single pixel away.

The domain of the best match mapping is the set of all patches on patch boundaries in a given view. The range is the set of all patches on any pixel location. The size of the domain is:

$$N(domain) = (W * H)/(pW * pH) \tag{3.11}$$

Where W x H are the dimensions of the image array and pW x pH are the dimensions of a patch. The size of the range is:

$$N(range) = W * H \tag{3.12}$$

Since the range is much larger than the domain, the mapping cannot be a one-to-one, onto mapping and thus does not describe an inverse mapping.

The domain is the set of patches on patch boundaries in the view because each pixel in the view must belong to a single patch. The ultimate goal of the scalespace correspondence algorithm is to determine a single offset vector for each pixel of the two views. The basic assumption of the scalespace approach is that there is a relationship between pixels in the original view and the patches in the filtered views. Offset vectors assigned to patches are used as initial approximations to the sub-patches (and ultimately pixels) at succeeding levels of scalespace. It makes no sense to have patches

on pixel boundaries in the domain because that would mean that several offset vectors would apply to the same pixel in the overlapping patches.

By the nature of the mapping, patches in the range will overlap because they are on pixel boundaries. It is also probable that many pixels in the second view will not be in any of the patches matched with the first view (see figure 3.2).

Calculating Offset Vectors

Displacement

The purpose of corresponding one view to another is to determine the direction each patch should be displaced so that the first view approaches the second. To do this, I constructed a set of image arrays for each view containing offset values for the x, y, and z coordinates for each pixel in the view. When a correspondence is made between a patch in one view and a position in the other view, I compute the offset vector necessary to move the patch towards that location. The x and y offsets are straightforward computations. For z, I use the difference between the average z-values of both patches to calculate the z-offset[7].

Instead of displacing a patch the total distance of the offset vector, I only displace it half the distance. Since the best match function is not reflexive, there is no guarantee that the patch from the other view will move to the same halfway position. However, as both views will move half way towards the other, the particles from the two views

[7]Displacing a patch based on its average z value has significant side-effects which are discussed in section? 4.6.1.

will have a tendency to meet in the middle[8].

Relaxation

The smoothness constraint can be applied to assign offsets to unmatched patches as well as correct the offsets of poorly matched patches. The smoothness constraint is used in many correspondence algorithms[Gri81] and states that surfaces tend to be locally smooth. It is generally true that if two pixels in a view are adjacent then their associated particles in worldspace are adjacent. As distortions are continuous in nature, there is a local smoothness in the distortions[9]. It follows that the offset vector correcting for distortion at a pixel should be very similar to the offset vectors for neighboring pixels. Relaxation can be used to correct offset vectors of bad matches as well as generate offset vectors for unmatched patches. It achieves this by averaging the offset values of neighboring pixels.

Some correspondence algorithms incorporate the smoothness constraint in the matching criteria (so that a patch is more likely to match another patch if neighboring patches tend to match the other patch's neighbors). In my implementation, the smoothness constraint is only applied after the matching has been completed to correct errors that occurred during the matching process and determine offset vectors for unmatched patches.

By filtering the offset tables with a low pass filter, I obtain offset vectors that are

[8]Chapter five discusses a method for weighting the movement towards the more reliable particles.
[9]Random noise is the one distortion that for which the smoothness constraint does not apply.

considerably more uniform but still compensate for the smooth fluctuations inherent in the original distortion. This process will also determine new values for patches which never had any offset vector (no-matches). The relaxed offset tables must be normalized so that unmatched patches are not included in the averaging process.

An improvement on this scheme takes into account the reliability of each correspondence. Each time two patches are compared, the matching criterion assigns an error value to the comparison. When searching for a match for a given patch, the search algorithm looks for the patch which returns the smallest error value when compared to the original patch. This error value must be less than a threshold or else the match is rejected and the patch is considered to have no match. It is possible to take advantage of the reliability information available on each match. During the correspondence process, I set up a correspondence confidence table which contains one byte values representing the confidence I have in the match for each patch. These values are determined by the ratio of the error value to the threshold using the following equation:

$$confidence[x][y] = (threshold - error)/threshold * 255 \tag{3.13}$$

Hence, the smaller the error the bigger the confidence value. Patches where the error is greater than the threshold (no matches) are automatically assigned a confidence value of 0. The confidence values are multiplied into the weighted averaging scheme (and the

results are normalized):

$$offset[x][y] = \frac{\sum(offset[x-i][y-j] * filter[i][j] * conf[x-i][y-j])}{\sum(filter[i][j] * conf[x-i][y-j])} \qquad (3.14)$$

The result of this improvement is that matches of questionable integrity have less of a corrupting effect on neighboring patches but still provide some useful information.

Edges between one surface that partially occludes another are an exception to the smoothness constraint. For instance, in trying to correct for a slight rotation in an object, it is possible that offset vectors for front surfaces will move particles along the positive x-axis and vectors for back surfaces will move particles along the negative x-axis. If the object is self-occluding (e.g. cage in photo 2-1) then two adjacent pixels can legitimately have offset vectors that point in opposite directions. This is a consequence of working in two-space instead of three-space. The general principle of local smoothness is valid and if the relaxation is performed in three-space with a 3D filter there is no problem with edges. Another observation is that, even in two-space, local smoothness is a far more frequent occurrence then the occasional discontinuity caused by an edge. Thus, a 2D relaxation will potentially correct more errors than it will generate and remains a useful tool for improving the reliability of the offset vectors. Since depth discontinuities are easy to detect in the range image (see Dane on edge graphs [Dan82]), it should be possible to modify the relaxation algorithm to take into account these discontinuities.

There is a relationship between the filter size and the matching criteria threshold.

The threshold determines the density of correspondence matches. Obviously, if a 5x5 filter is used then the relaxation will not be effective if there isn't at least one match in most 5x5 windows of the view. Nonetheless, it is better to have a small amount of reliable matches than a large amount of unreliable matches.

Additional techniques such as elasticity may be applicable to the task of relaxation [TWK87] [Gri81].

Merging Data

Once the final offset vectors are determined (at the end of the scalespace correspondence process) then the particle database of the new view and the cumulative particle database can be merged. The offset tables contain x, y, and z offsets for the pixels in the 2D orthographic projections. The problem remains of determining which particles in 3D space are linked to which offsets in the particle databases.

Converting the pixel locations back into worldspace locations will result in some quantization errors that will complicate the process of determining which offset vectors are associated with which particles. When the orthographic views were generated, it would be possible to generate tables with pointers for each pixel location back into the particle database file, but such pointer files would be huge.

The solution I adopted is to take each particle in the particle databases, use the camera geometry to re-calculate its pixel location, use that location to look up its associated offset vector, add the vector to the particle and save it in the new merged

particle database. This process uses the same equations as were used to originally do the orthographic projection of the particle database into a view. The process is performed on both the cumulative particle database and the particle database of the new view, one after the other. The particles can later be sorted by y, x and z position if that is useful.

The displacement process only applies to the points visible in the orthographic views! Sometimes two points which were adjacent to each other in the perspective view are mapped to the same z-location in the orthographic projection (hence only one of them appears in the orthographic view and has an offset vector). Also, back-faced and otherwise occluded particles do not appear at all in the views. When comparing the computed range value of a particle to the corresponding range value of the associated pixel in the view, it is possible to determine if the particle is (1) visible in the view, (2) occluded, but on the same visible surface as the pixel or (3) occluded and on a separate occluded surface of the object. If the particle is occluded but on the same surface as the visible pixel, the smoothness constraint allows us to add the same offset vector that we are adding to its neighbor.

If the range values are very different then the point is on an occluded surface, substantially different than the surface for which the offset vector was computed. No direct information about how to displace these types of points can be obtained from the offset tables. For instance, a distortion in the form of a slight rotation may result in the same offset vectors as a slight translation for the visible point at that z-value, but would

have entirely different consequences for a point on the other side of the object. This problem is a consequence of using a 2D rather than 3D correspondence algorithm. In the absence of any reliable displacement information, the particles should be left unchanged to be aligned later when adding views in which they are visible. It is probably possible to use the offset vectors for the coarse scale views to detect global rotations and other simple transformations that would also apply to the occluded particles.

This system will result in the cumulative particle database ending up N times bigger than the particle database for a single view, where N is the number of views. If the database is too big, it can be reduced by quantizing the worldspace and eliminating particles that are too close to other particles.

3.3 Combining Views

This section addresses the issues of how many views are necessary to construct a particle database and in what order they should be added.

Any algorithm that combines the views in an intelligent fashion will have to match features in the different views. The more redundancy there is in the set of views, the easier it will be for the algorithm to match the views and the more reliable the results. The goal is to have a large amount of shared surfaces between adjacent views, but still have a small amount of views each with a significant amount of new surfaces. Redundant surfaces in the views are also useful because when merged they yield a higher density than the density of the surfaces in the individual views.

In the case of the scalespace correspondence algorithm, the greater the angle between two views the harder it is to match them. The larger the angle, the larger the "holes" in the views caused by back-faced surfaces being rotated into view. The larger the "holes", the more difficulty the scalespace algorithm has aligning the objects at coarse levels. The algorithm was adapted from human stereo vision where the angle between the two viewpoints is small. The limit to its usefulness when applied to larger disparities is not known.

Quantity of Views

The number of views necessary to construct a "complete" 3D model of an object depends on the complexity of the object and the intended use of the model. There are two main issues: one is the visibility of the object's surfaces and the other is the density of data available on those surfaces. Surfaces of an object are invisible to a view if they are facing away from the camera (back surfaces) or are occluded by other surfaces. The more a surface faces towards the camera in the original view, the higher the density of particles.

In terms of visible surfaces, a completely convex object such as a beach ball is entirely visible in two or three views[10]. Other objects have so many self-occlusions that there is no upper-bound to the number of views necessary to see all surfaces of

[10]Because of the double occlusions inherent in laser-scan range images, each view provides a little less than 180 degrees so three views are necessary with this technology. To simplify the discussion, assume 180 degrees of data is available in a view unless otherwise noted.

an arbitrarily complex object (e.g. pineapple leaves in photo 4-3). It is not necessary to scan views for all surfaces of an object. Only those surfaces that will potentially be rendered need to be in the database. For example, there is no reason to scan surfaces on the inside of a soda can, if it is certain that the can will never be rendered from a viewpoint in which the inside is visible.

In order to discuss the density of a view we need a more precise definition for density. Assume that the surface of an object is perfectly represented in a voxel space with the same resolution as a view[11]. We can see all the voxels in any surface patch if we are viewing the patch head on. Specifically, the view normal must be coincident with the surface normal of the patch in order for 100% of the patch's voxels to be visible. If we view a surface patch from an angle then the percentage of visible voxels is a cosine function of the angle between the view normal and the its surface normal. Density is the percentage of visible voxels to total voxels. It can also be thought of as the probability that a voxel on the surface can be seen from a viewpoint.

In terms of density, we must evaluate the minimum density we are willing to accept should the model be rendered from any conceivable viewpoint. Suppose we had a cylindrical vase on a turntable and we wish to know how many views we need to scan to be able to render the vase at any turntable position with guaranteed minimum density of 50%. Since we know that the cos(60) is 0.5, surface patches with normals +/-60 degrees from the view normal have densities greater than or equal to 50%.

[11]The maximum of the x, y and z resolutions of the view.

Thus, scanning three views at 0, 120 and 240 degrees is sufficient to ensure that every surface patch on the (side of the) vase has at least 50% density in one of the views. This simplified example only considers one degree of freedom of rotation whereas two degrees of rotation are possible in the general case.

Perspective removal also causes a loss of density in a view. However, since the depth of field of an object is typically small compared to the distance between object and camera, the loss of density is negligible for 3D model-building of objects. For 3D moviemaking, the effects of perspective on density are a concern in scenes where a large depth of field is required.

Differences of resolution along the horizontal, vertical and depth axes of a range image will affect the density. For example, if the horizontal resolution of a view is 20 pixels/cm and the resolution in depth is 5 pixels/cm then the more depth a surface has, the less density it has by virtue of the fact that it has less resolution. Hence, the density is affected by both lack of resolution and lack of visibility. It is preferable to have an equal resolution for all axes of a range image.

Adding views will result in a higher density than either of the individual views. Some of the data will be redundant in the views and some will be new data. If View A is rotated α degrees into alignment with View B which has been rotated β degrees then the minimum density of the combined views is:

$$max(\cos(\alpha), \cos(\beta)) <= density_{AB} <= \cos(\alpha) + \cos(\beta) \tag{3.15}$$

In other words, the combined density of several views is greater than the density of each of the component views individually and less than the sum of the densities of the component views.

There is a relationship between density and effective resolution. In a rendered image it is always possible to fill missing pixels but the effective resolution remains unchanged. Consider a picture of a soda can with writing on its side that is visible, but not legible. If the can is rotated 90 degrees and the missing pixels are filled in, the writing would still be illegible. If the density of several views combined is greater than 100% then the resolution of the database is greater than the resolution of the individual views[Sue86]. Effective resolution is the product of the resolution of the view times the density.

In my work, I scanned 8 views of an object on a turntable at 45 degree intervals of horizontal rotation. To simplify the problem, I ignored the top and bottom views (which could not be easily obtained on a turntable). This gives a minimum horizontal density of 92% ($\cos(45/2) = 0.9238$) but does not take into account surfaces with a vertical component in their surface normals. Another reasonable approach would be to take six views: front, back, left, right, top and bottom. On a convex beach ball, all regions on the surface appear at least three times. This means we are guaranteed a minimum density of 70% ($\cos(90/2) = \sqrt{2}/2 = 0.707$). In a real world environment, the camera placement may be restricted, and may not be symmetrically disposed around the scene.

Order of Adding Views

There are many different possible orders in which to add several views to a particle database. One possible way to start the process is to add two opposite views (0 and 180 degrees) to begin with a relatively complete model. When the algorithm adds a new view, such as a 90 degree view, to the particle database, it projects the particle database onto an orthographic 90 degree view. Because the particle database has already been seeded with complementary views, this appears to have resolved the problem of large unknown regions in the particle database caused by occluded back surfaces. In trying to directly add a 0 and 90 degree view, one encounters difficulty corresponding the two images because there are large portions of each view that do not exist in the other view. One could add the views in a pattern of opposite pairs (0, 180, 90, 270, 45, 225, 135, 315) putting the emphasis on filling in occluded regions as quickly as possible and then going back to fill in detail.

The weakness of this approach is that it begins by adding two opposite views because there is no contradictory information in the two views instead of trying to add together views that contain as much information confirming the correctness of the alignment as possible. Matching opposite views is essentially a problem of matching silhouettes. On a concave object, it can be impossible to correctly align opposite views in terms of depth. For instance, combining the two views of the opposite ends of a barbell may not give a very accurate representation of the length of the barbell and gives no information about the bar itself. Silhouette matching is a particular problem with

laser-range images which do not represent a full 180 degrees of an object and which tend to have a jagged edge on the side of the object opposite the laser.

The approach I have adopted is to add views that overlap as much as possible. Thus I start with the 0 degree view and then add views by 45 degree increments until I've worked my way 360 degrees around the object. The advantage is that the first two views already have a great deal in common and thus have the greatest basis for meaningful alignment. For example, on a convex beach ball, when adding a 45 degree view to a 0 degree view, 75% of the surface area in the second view will already be represented and 25% will be a new surface[12]. The density of particles in the overlapping surfaces for the different views will vary. On the beachball, the first view will give 180 degrees of information and each of the next four views will each add 45 degrees more of information. The last three views will only give more detail about surfaces that are already in the database.

A criticism of this scheme is that it might cause an error-of-closure problem. Specifically, an error might propagate when adding views so that the views that were being added to the database were being displaced more and more away from the orientation of the first view. However, this is less of a problem than it first appears because the last three (out of eight) views overlap with the first view. This means that the correspondence algorithm will be allowed to smooth over the closure error over several

[12]In this simplified example, I assume that a view provides 180 degrees of information, ignoring the fact that a laser-range camera provides information on a slightly small range because of the double occlusions. Also, the beach ball represents an object with no occlusions. A more complex object with self-occlusions will have more new surfaces in each new view.

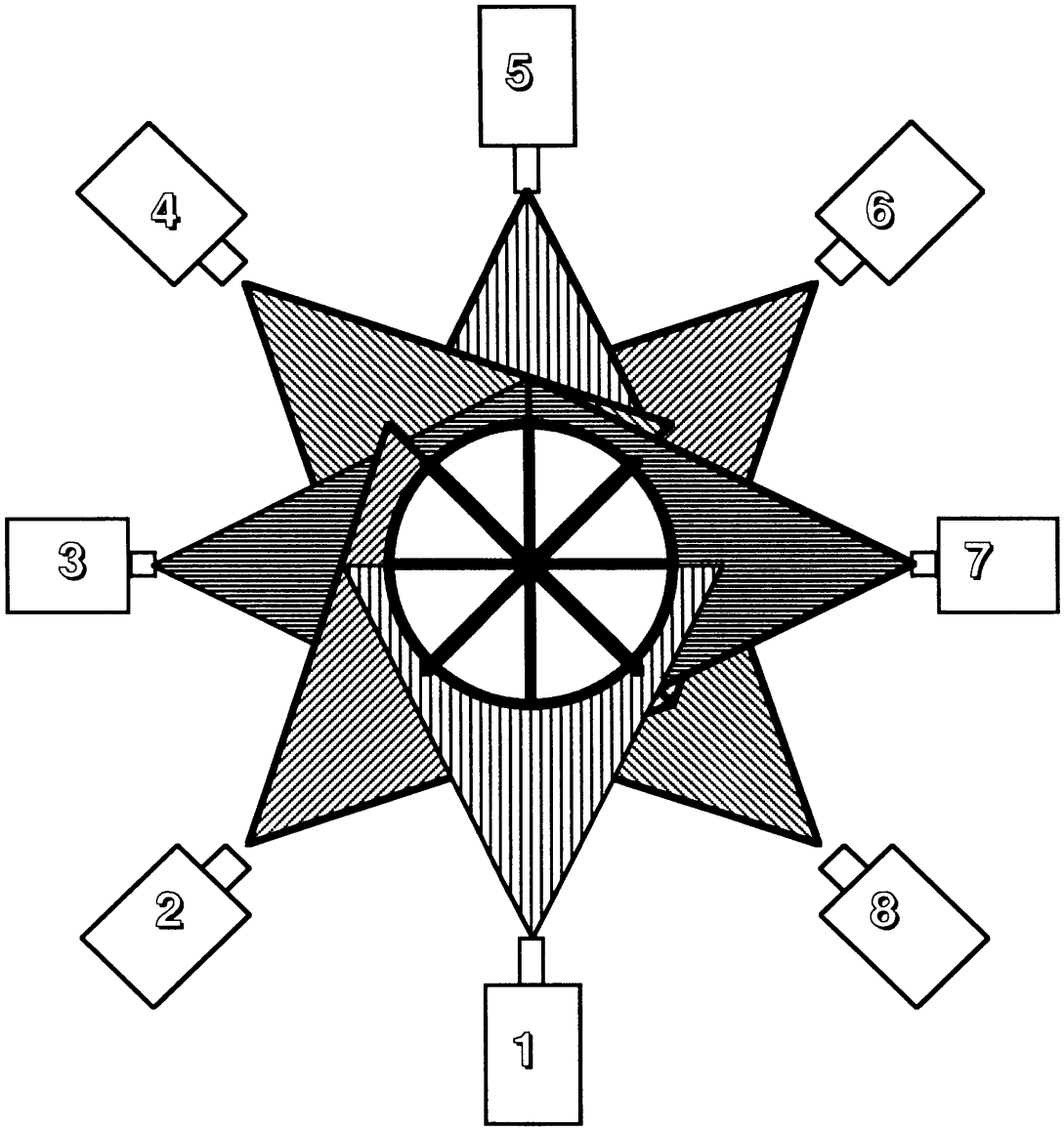


Figure 3-3: Diagram of overlapping of views on a beach ball
Each section of the beach ball can be seen in four different views.

passes. In the correspondence algorithm, each time a new view is added to the particle database, both the database and the new view adapt themselves to be more like the other. Hence adding three views to the wrap-around region will have a big influence on correcting any alignment mismatch. The best way to correct for the error-of-closure problem is to add all views at each level of scalespace before going on to the next level, instead of adding views one at a time.

Chapter 4

Experimental Results

4.1 Input Objects

Photos 2-1 and 4-1 through 4-5 show the input data used for these experiments. Each object was selected because it demonstrates one or several interesting properties of real world objects that must be taken into consideration by the scalespace correspondence algorithm.

Scalespace Features and Textures

Ideally, the scalespace algorithm works best if the object has features to match at all levels of scalespace. Features useful for matching can exist in any of the different attribute arrays. For some objects, coarse scale features will be better represented in a different attribute array from the one best presenting the fine scale features. The

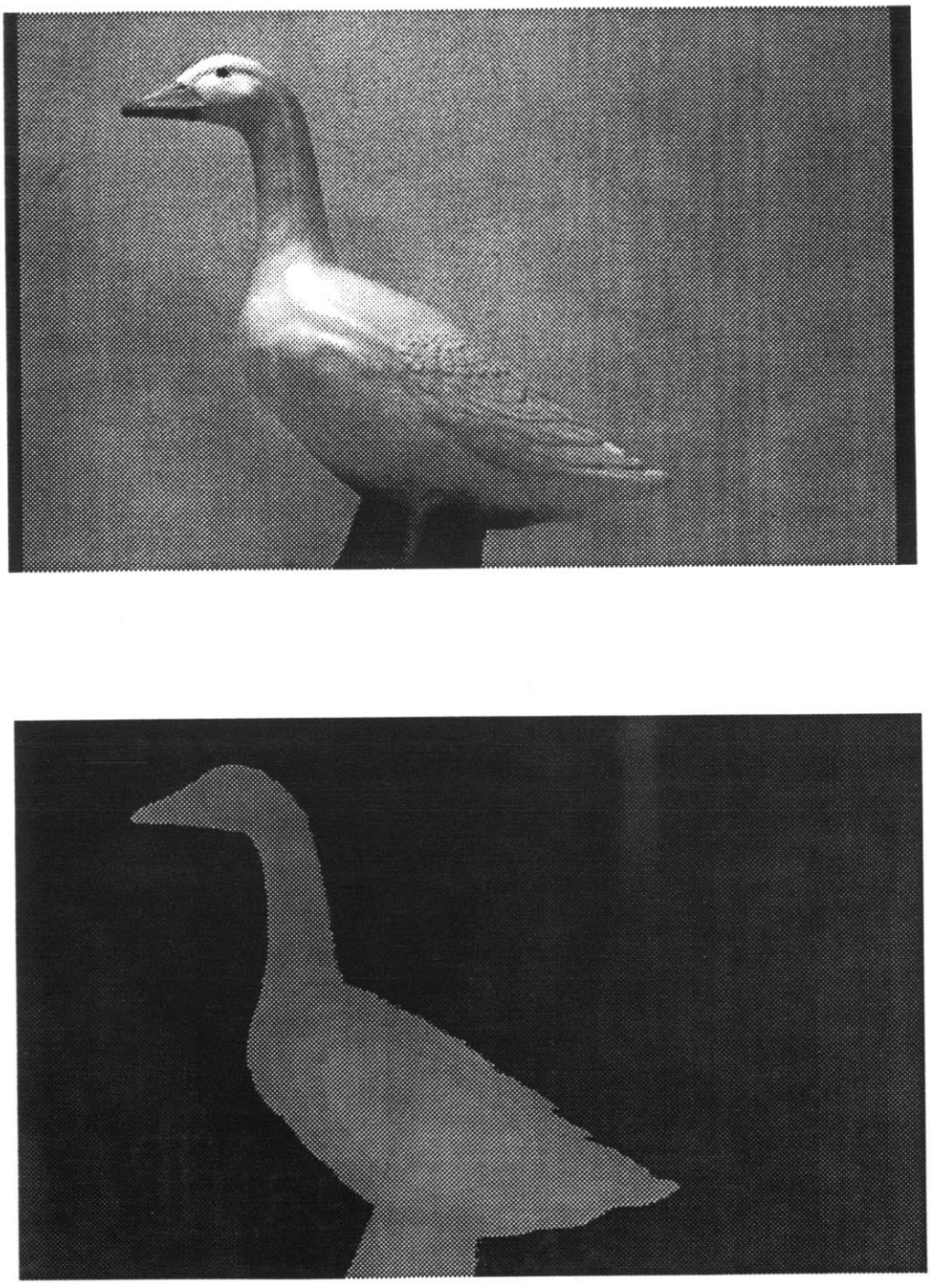


Figure 4-1: Goose reflectivity and range image

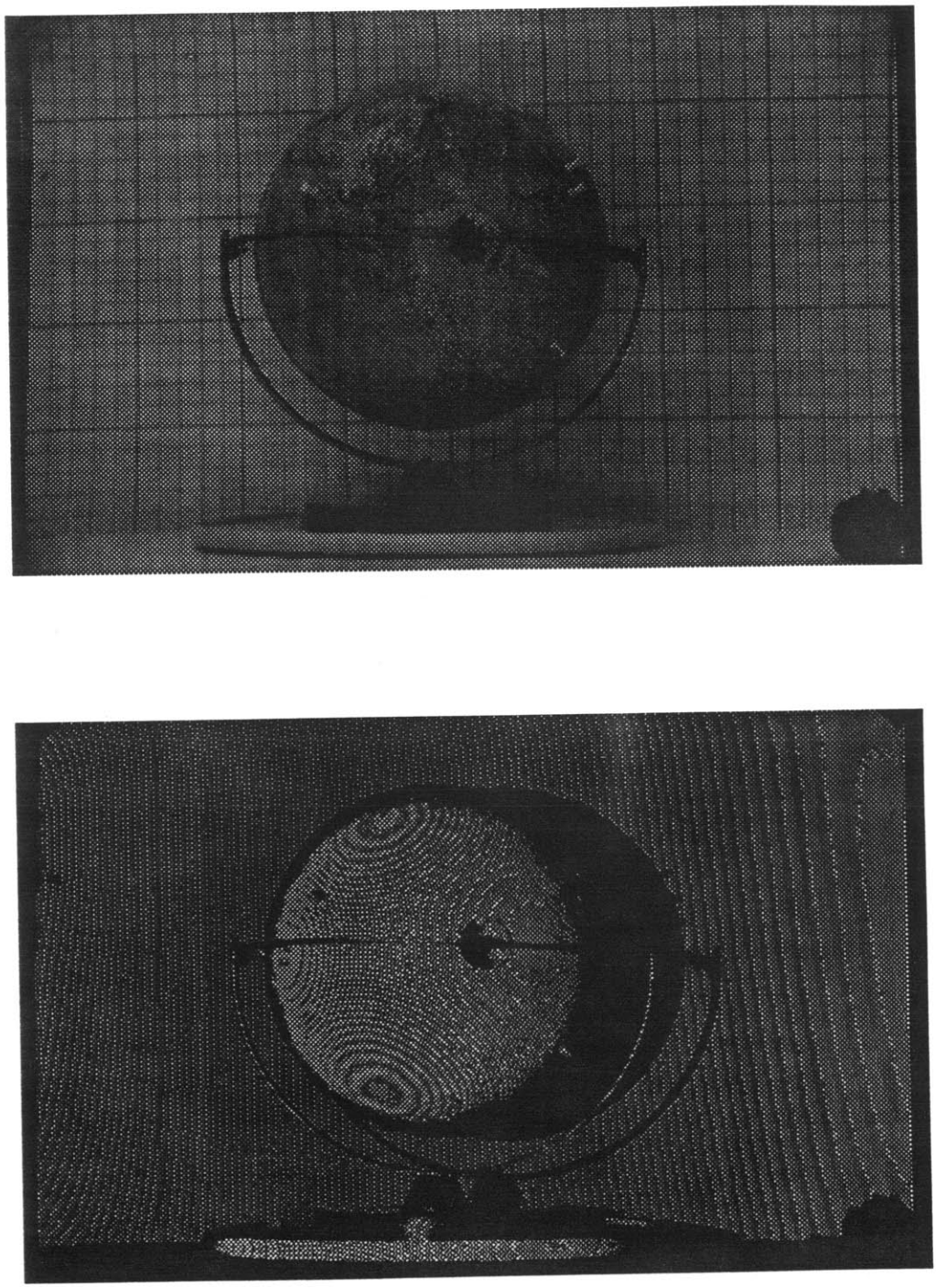


Figure 4-2: Globe reflectivity and range image

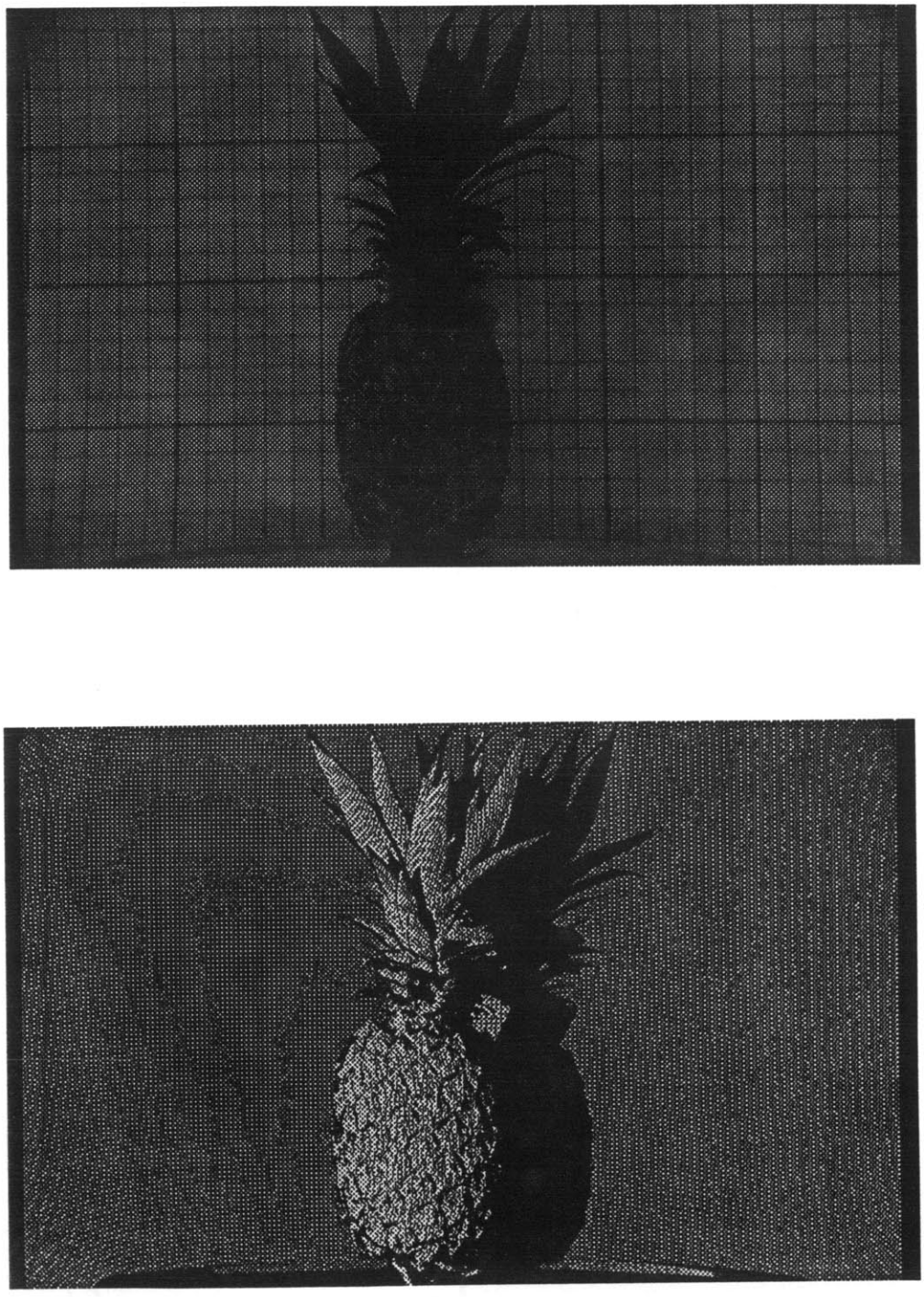
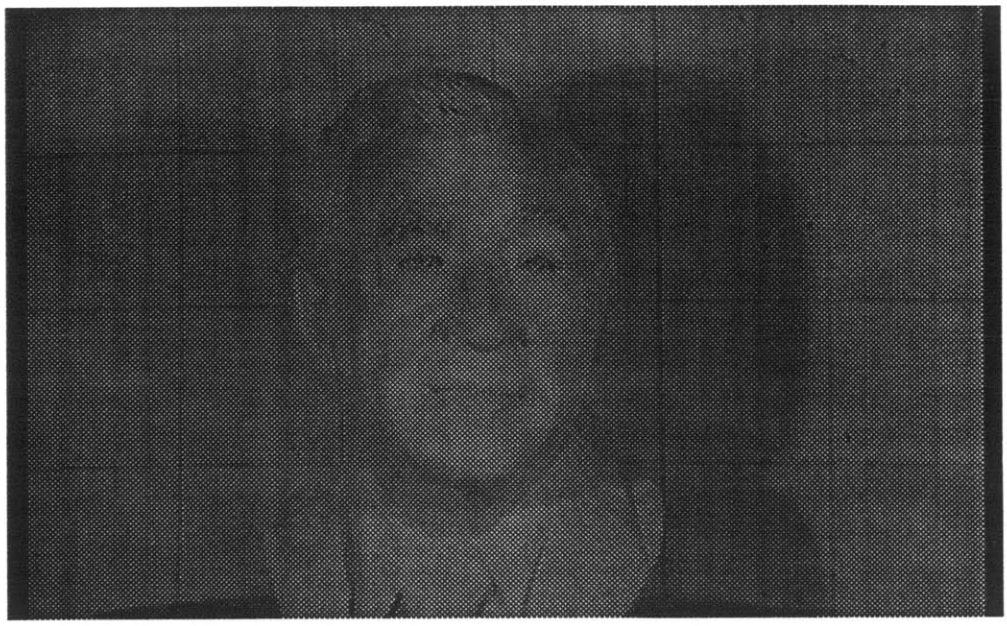


Figure 4-3: Pineapple reflectivity and range image



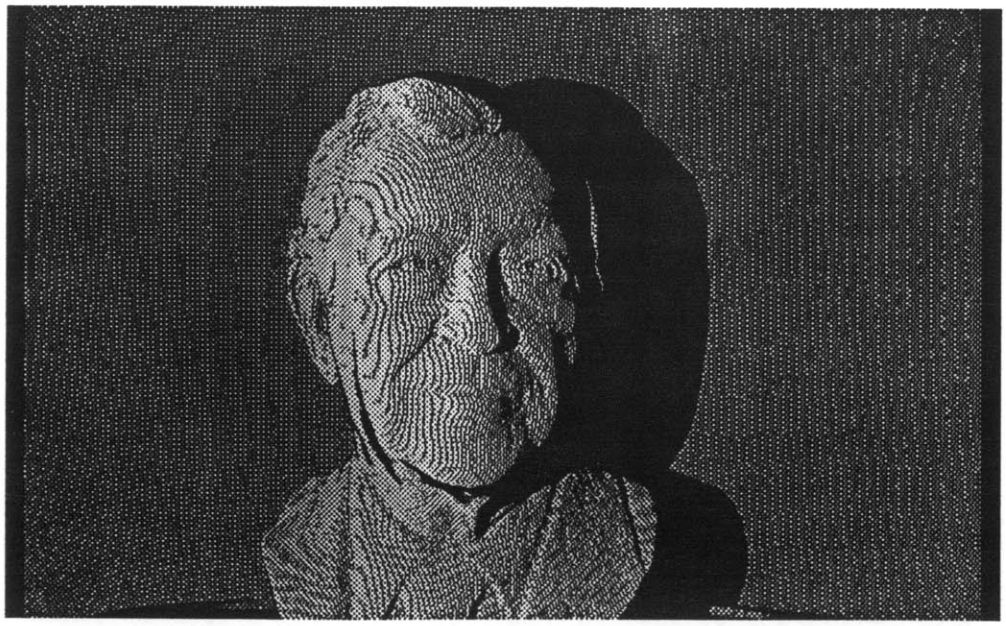


Figure 4-4: Wiesner Bust reflectivity and range image

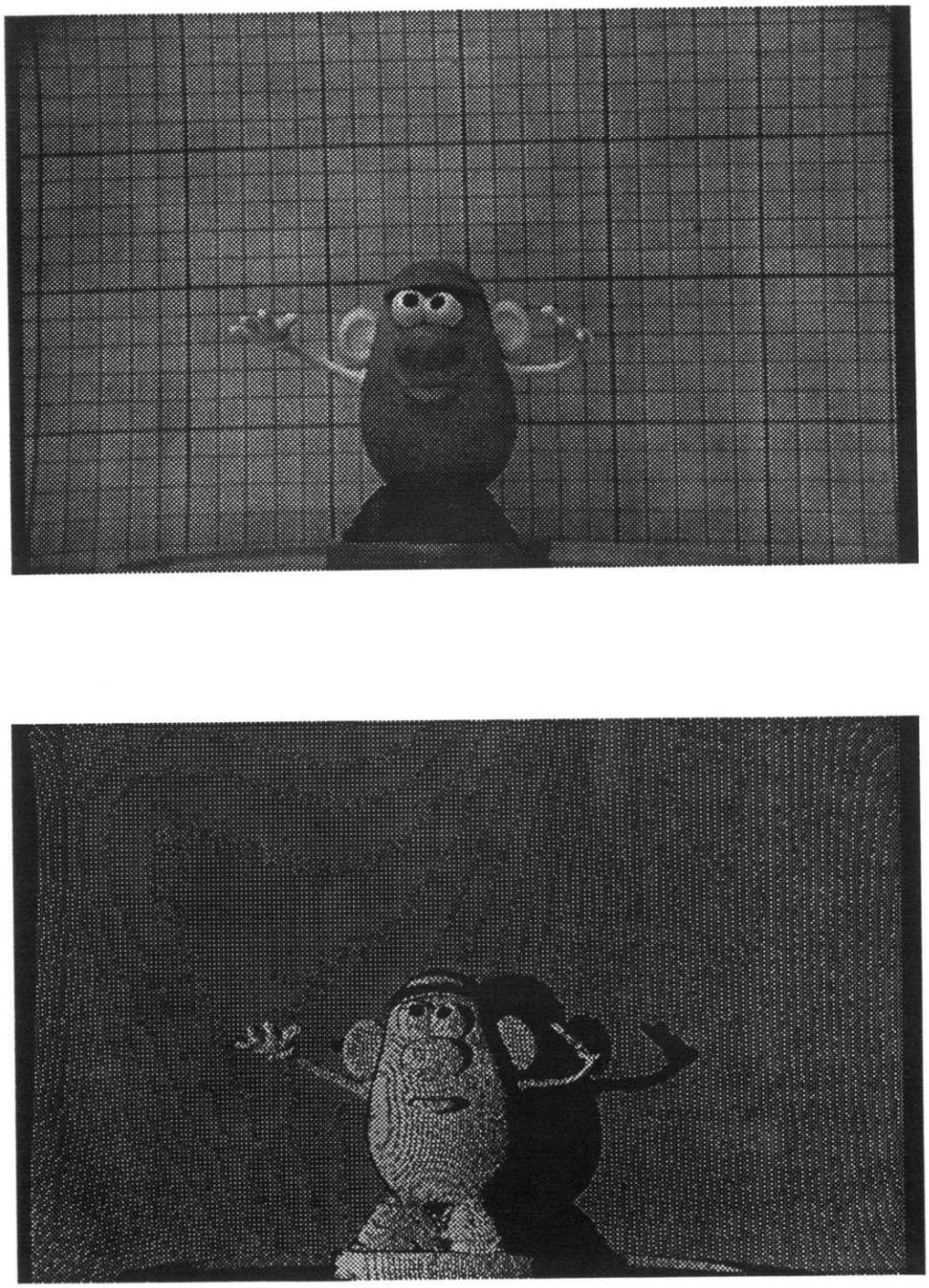


Figure 4-5: Mr. Potato Head reflectivity and range image

present implementation of the algorithm works best at matching coarse and mid-range details. It is expected that significant improvements in matching fine detail can be achieved.

Mr. Potato Head in photo 4-5 and Mr. P. with cage in photo 2-1 are examples of objects with primarily coarse and mid-range features. Coarse scale features include the shape and intensity of the potato and cage. Mid-range features include the eyes, ears, nose and hat of Mr. P. There are few fine details other than the fingers on the hands and the ridges in the boots. The parts of the body tend to be uniform. There is a high correlation between the features in the intensity and range images. Color would be an appropriate feature for matching mid-range features for these objects.

The pineapple in photo 4-3 is an example of an object with many different levels of texture in both reflectivity and range images. Because of repeating patterns on body of the fruit, coarse and mid-range alignment must be achieved before fine details can be matched reliably.

The goose (photo 4-1) and the bust of Jerome Wiesner (photo 4-4) are examples where there is considerable detail in the range image, but little reliable information in the reflectivity images. For the globe (photo 4-2), the opposite is true.

Lighting

The goose (photos 4-1 and 4-6) and the Wiesner bust are examples where lighting is particularly detrimental to the correspondence process. The actual intensity of both

of these objects is almost entirely uniform. Shading creates features in the reflectivity images such as the feathers on the goose and lines in the bust around the nose and mouth. When the object is rotated on a turntable respective to the light source then the position of the shadows on the object shift. While deep shadows (e.g. in the nostrils and naso-labial-furrows) may be useful for coarse scale matching, they are misleading at a finer scale. Specular highlights, apparent in the globe and goose pose similar problems.

Occlusions

Almost all of the range images show some occlusions. There are two types of occlusions: points not visible to the camera and points not visible to the laser. Pixels representing occluded regions are arbitrarily assigned the value 0 and appear as "shadows" in the range images. Areas occluded from the cameras viewpoint only become visible as "shadows" when perspective is removed or when the object is rotated (e.g. the double occlusion caused by the cage in photo 4-22b). Because most of the experimental objects are roughly convex, they do not contain many large scale occlusions. Large occlusions of the back wall are visible in all range images, but pose no problem to the matching task. The cage in photo 2-1 is the most interesting of the objects for studying occlusions. The top bars of the cage clearly occlude each other from the camera's viewpoint. Laser occlusions can be seen at the top right corner of the cage. Small laser occlusions also appear on Mr. P.'s left ear, right hand and hat. The leaves of the pineapple contain

numerous small occlusions. The Wiesner bust reveals that the features of the face make many tiny occlusions.

In the range image of the goose, no "shadows" appear because a simple technique was used for filling the occlusion. The neighboring surface furthest away from camera was assumed to be occluded and extended to fill the missing region. This assumption is generally valid, but poses significant problems when erroneous.

Some black regions of the range images are not caused by occlusions, but rather by some defects of the laser-scan depth camera. For instance, no depth information is available for Mr. P.'s pupils because the infrared beam of the laser did not reflect off the black surface. The laser scanner also had trouble along the edges of the globe where the beam tended to bounce off the shiny surface. Notice that the left edge of the silhouette in the range image of the goose is more jagged than the right edge. This is a result of the camera geometry and is a common feature of all laser-scan range images.

Camera Geometry and Distortions

The cage made of Construx (photo 2-1) is a good subject for studying the camera geometry and perspective of the system. Because of its Euclidean form, it is useful for evaluating how the correspondence algorithm handles distortions.

The globe was meant to be a study in reconstruction of an object based on rotations around two non-perpendicular axes. All the other objects were only rotated around the vertical axis of the turntable. The globe had it's own second axis which was used

to take views of the top and bottom of the object. No experimentation was done with this data.

4.2 Multiple Views

Original Views

The photo 4-6 show the reflectivity image of 8 views of the goose. There is a view for each 45 degree rotation along the vertical axis of the turntable. Range images also exist for each view. Eight views were also taken for the majority of the experimental objects.

Rotated Views

The photos in 4-7 show the reflectivity images of 8 views of Mr. P. in the cage. In this case, each of the views, spread at 45 degree increments, have been rotated back into alignment with the 0 degree view. The distortions in the cage reveal the difficulty in reconstructing the precise geometry of the scene. The rotated 90 degree view in photo 4-7c reveals the low density of particles in surfaces pointing away from the camera as well as the missing back-faced surface. The 180 degree view in photo 4-7e is concave as we see the particles from their back side. In the 45, 135, 225 and 315 degree views, there is a vertical black strip on Mr. P. in the region occluded by a bar of the cage.

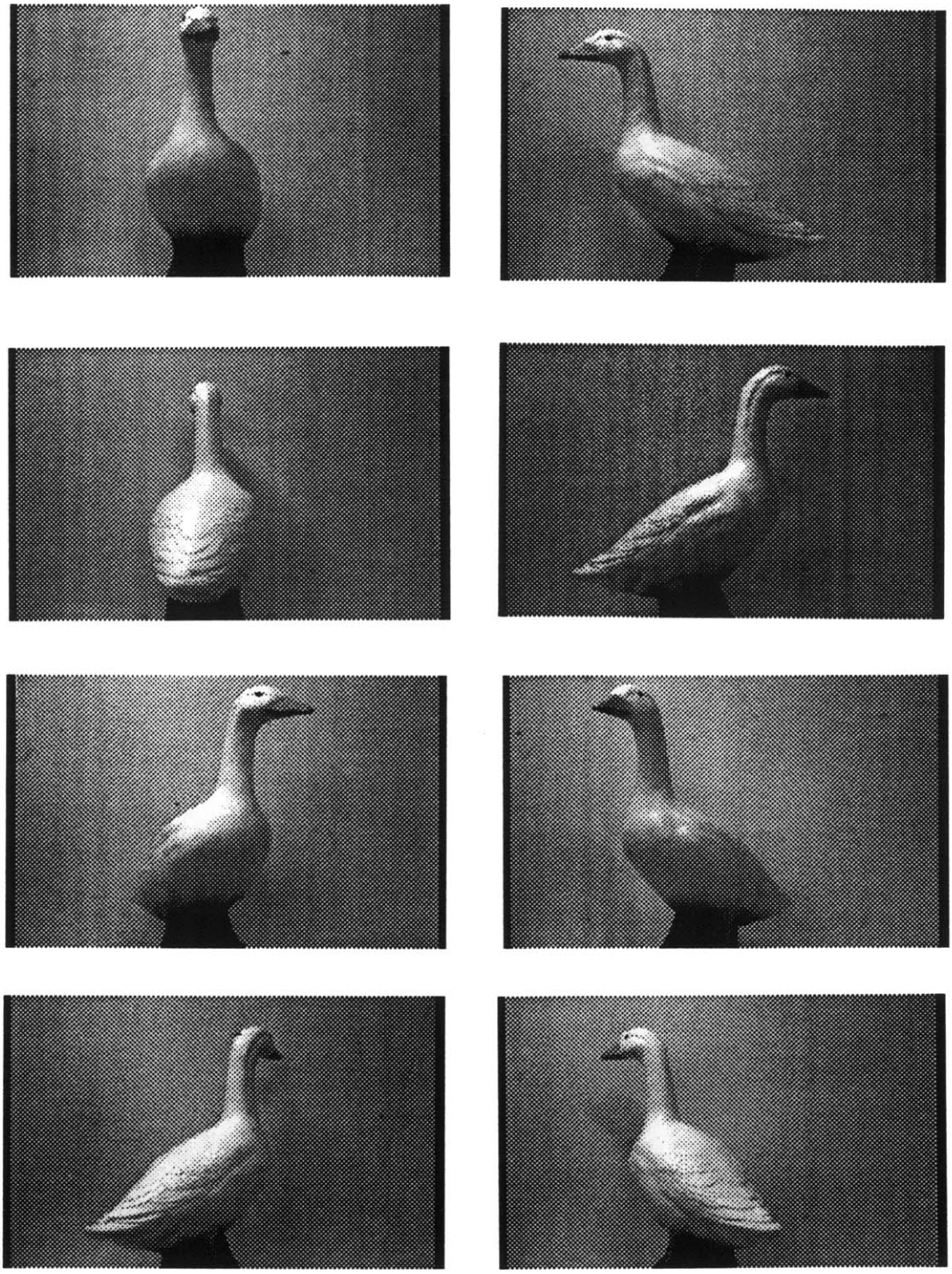
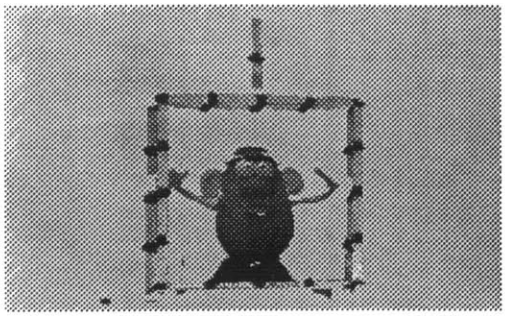
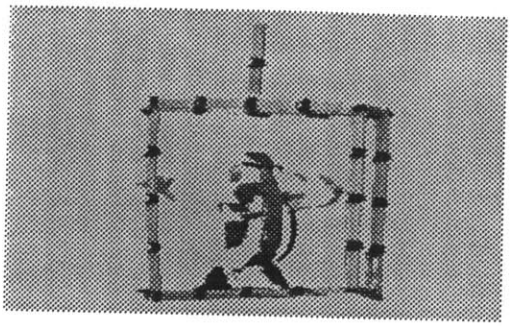


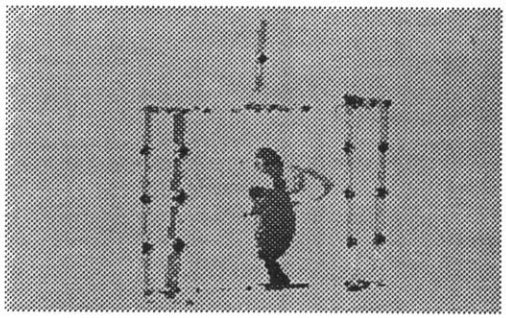
Figure 4-6: Eight views of the goose taken at 45 degree increments



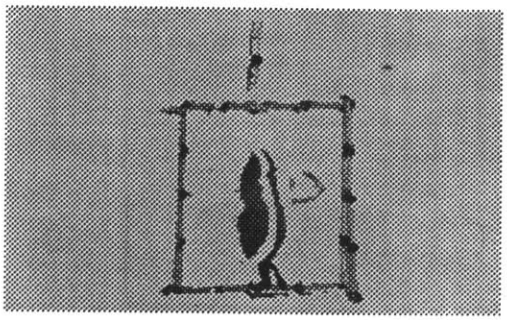
(a) view 0



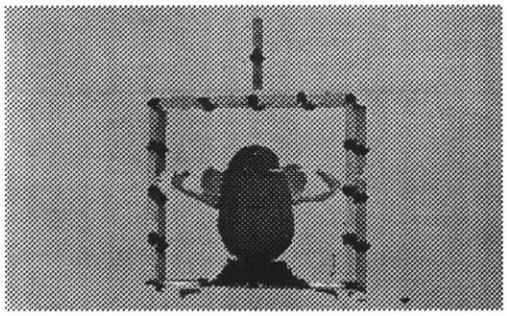
(b) view 45



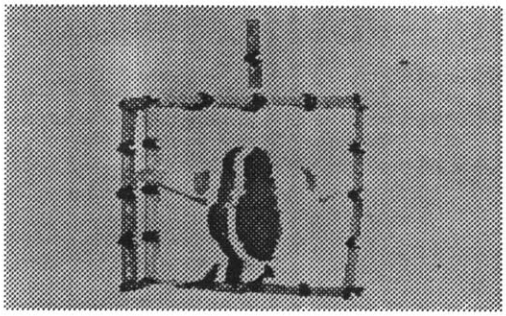
(c) view 90



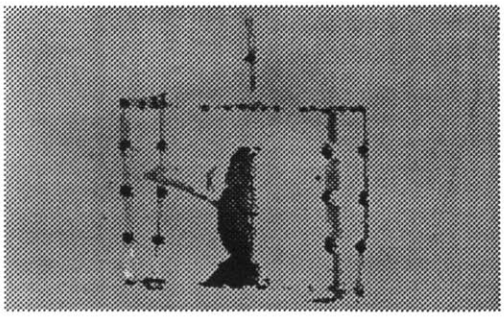
(d) view 135



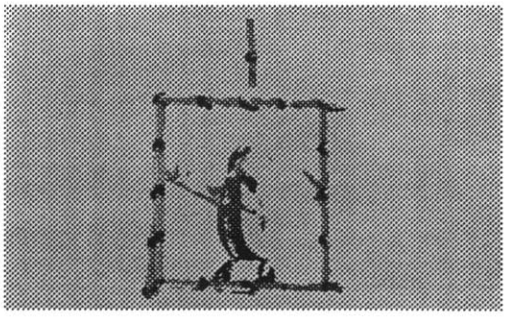
(e) view 180



(f) view 225



(g) view 270



(h) view 315

Figure 4-7: Eight views of the cage rotated back into alignment with view 0

4.3 Perspective

Juxtaposition Without Perspective Removal

Best Case

The photos in 4-8 show how juxtaposition of views can be accomplished without perspective removal. Photos 4-8a and 4-8b are the range images for the left and right side of the goose respectively. In photo 4-8c the right side had been flipped and the two slides have been "pasted" together. The alignment of the contours is good although there is some mismatching around the neck and beak. Photo 4-8d shows the same data as photo 4-8c rotated 90 degrees where the back side of the goose is visible. The bodies are well aligned. There is some problem around the neck. The data is sparse around the seam of the goose lamp which was not very visible from either the right or left view.

Worst Case

The photos in 4-9 show why juxtaposition of range images does not work without perspective removal when there is a big depth of field in the object. Photos 4-9a and 4-9b show the range image of the front and back view of the goose respectively. Notice the neck stretches towards the camera in 4-9a and away from the camera in 4-9b. In photo 4-9c, the back image has been flipped and juxtaposed to the front image so that both sides are viewed from the front view. The bodies are relatively aligned, but there are major problems in the neck and the heads are not at the same height. Photo 4-9d

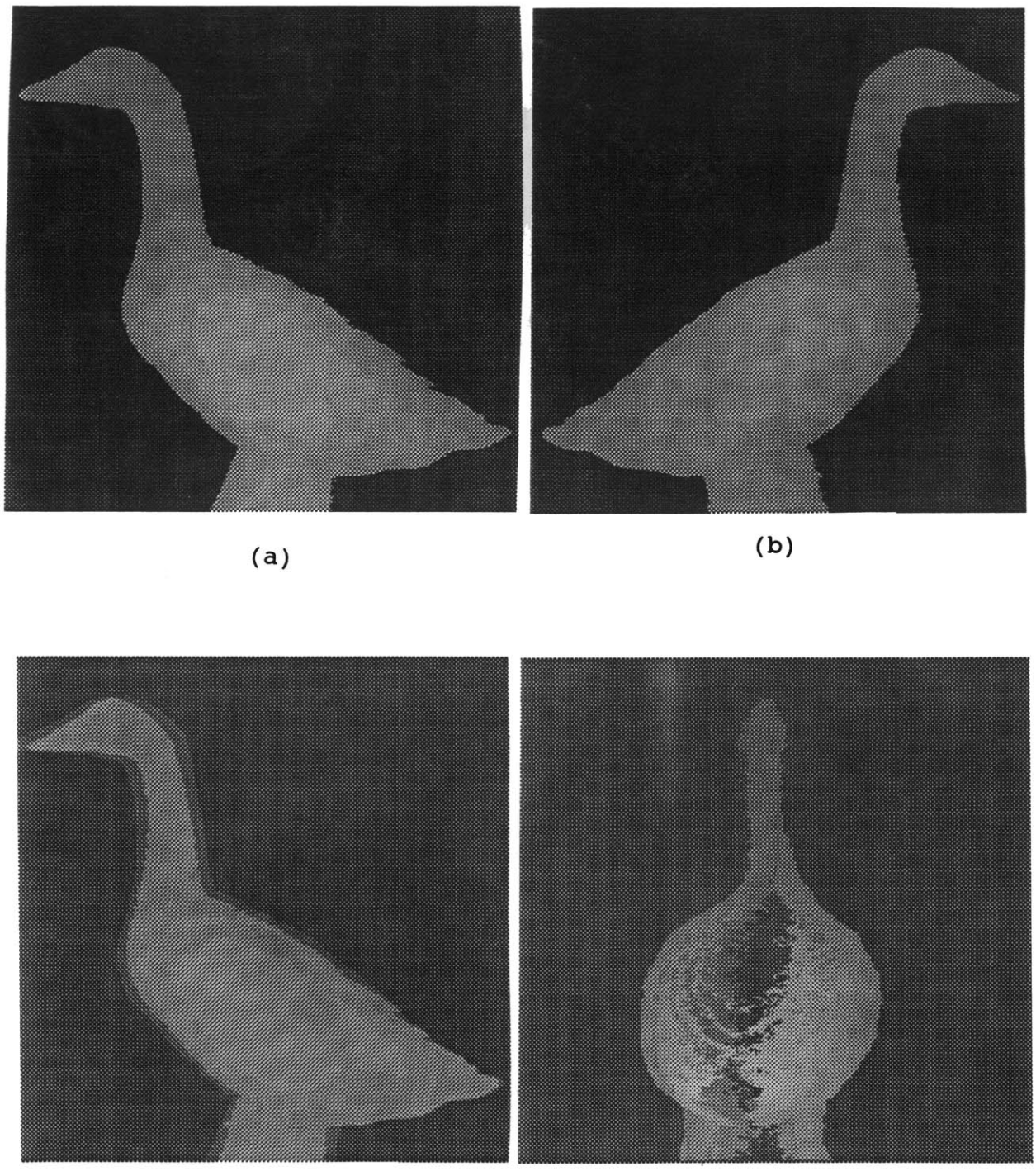


Figure 4-8: Left and right half of the goose juxtaposed and rotated

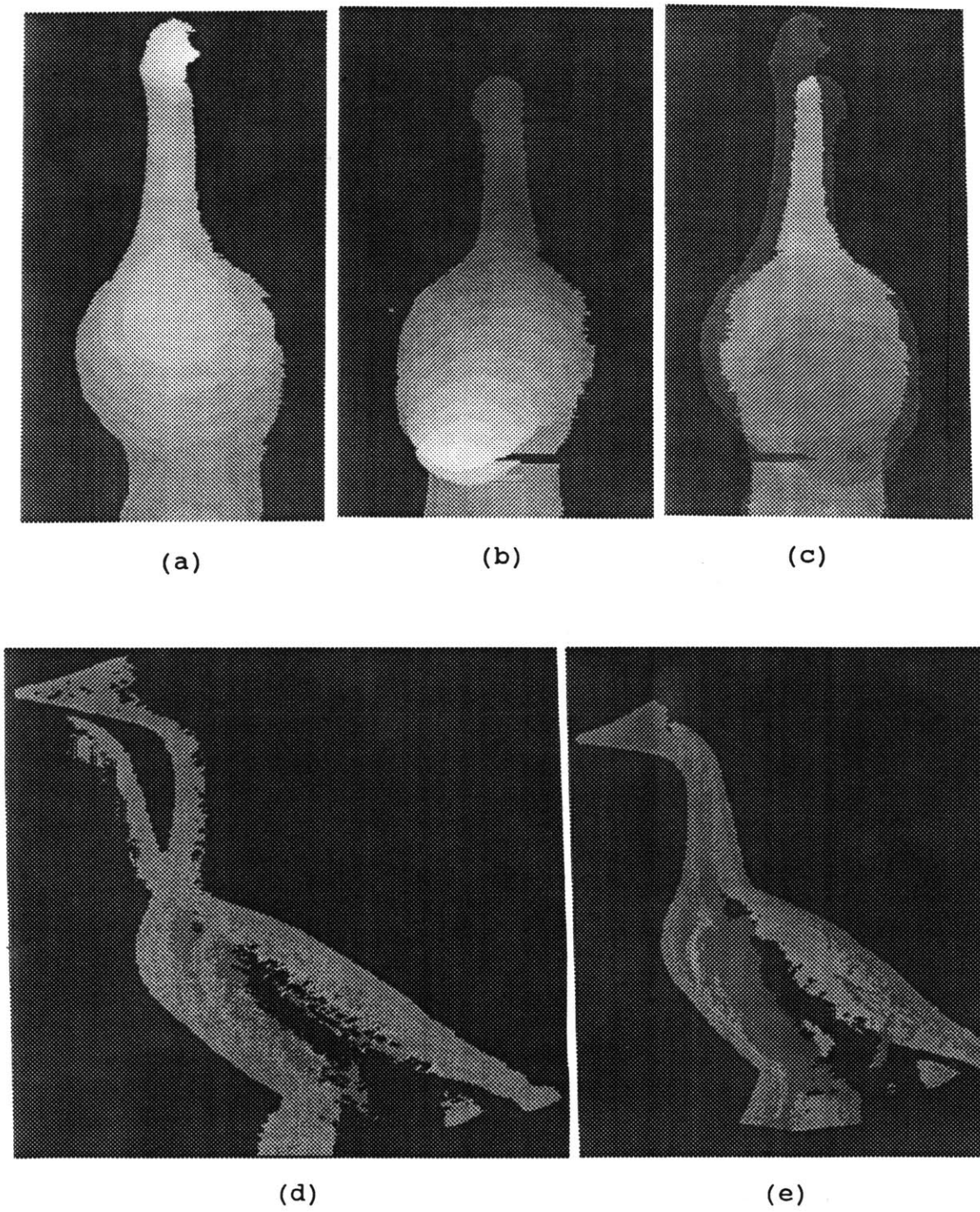


Figure 4-9: Front and back half of the goose juxtaposed and rotated

shows the same data as in photo4-9c rotated 90 degrees. The bodies align, but the necks and heads of the two views are grossly mis-aligned due to perspective. Photo 4-9e is the front and back views combined after perspective removal has been performed. The alignment is much better, even though the front and back of the goose's head are still slightly mismatched.

Perspective Removal

The photos in 4-10 show the effects of perspective removal on the images. Photo 4-10a is the range image (with background removed) of Mr. P. and the cage with perspective. Photo 4-10b shows the same image with perspective removed. In this image, the front and back faces of the squares are roughly the same size. Notice perspective removal caused "holes" in the cage in portions that were occluded by Mr. P.'s hands in the perspective image, but are now visible. Photo 4-10c shows the orthographic view seen in photo 4-10b rotated 90 degrees. The distortion in the image is evident in the bars of the cage. Photo 4-10d shows an orthographic projection of the view scanned by the camera when the object was rotated 90 degrees on the turntable. Photo 4-10e shows the rotated 0 degree view added to the 90 degree view. Mr. P.'s body is roughly aligned. The bars of the cage are not well aligned. Perspective removal is a necessary, but not sufficient, step towards eliminating distortion.

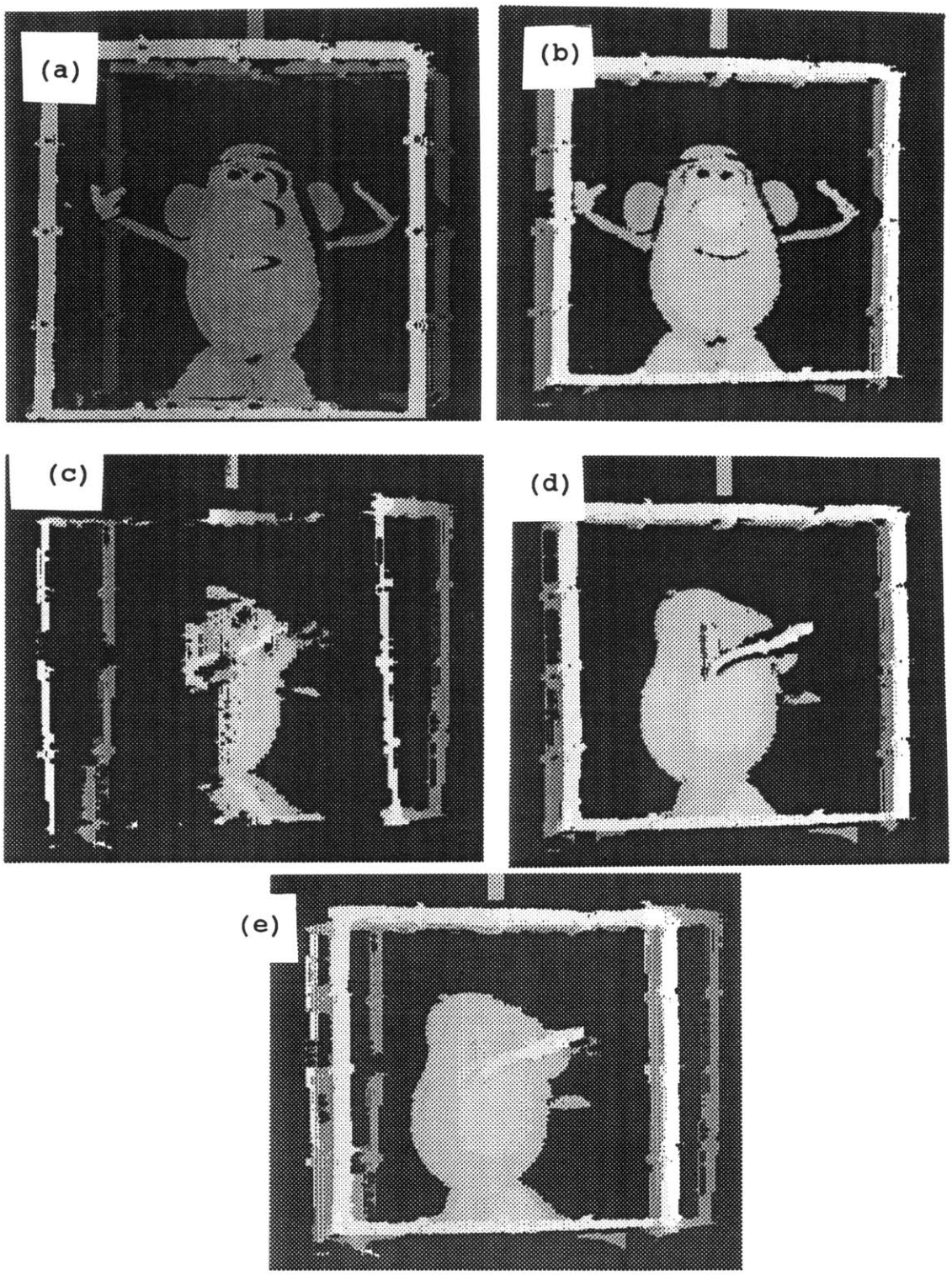


Figure 4-10: Juxtaposition two views of the cage with perspective removal

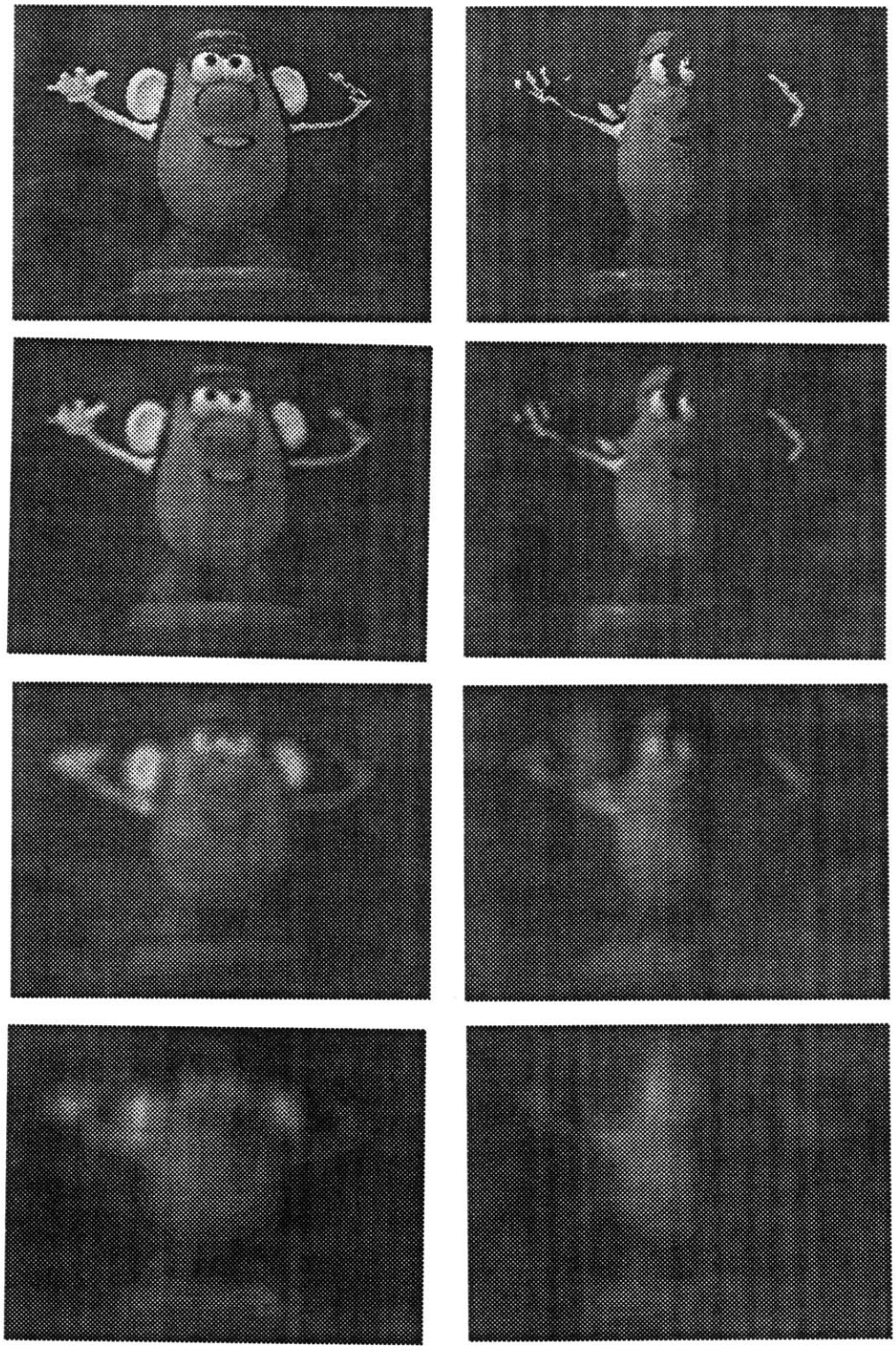
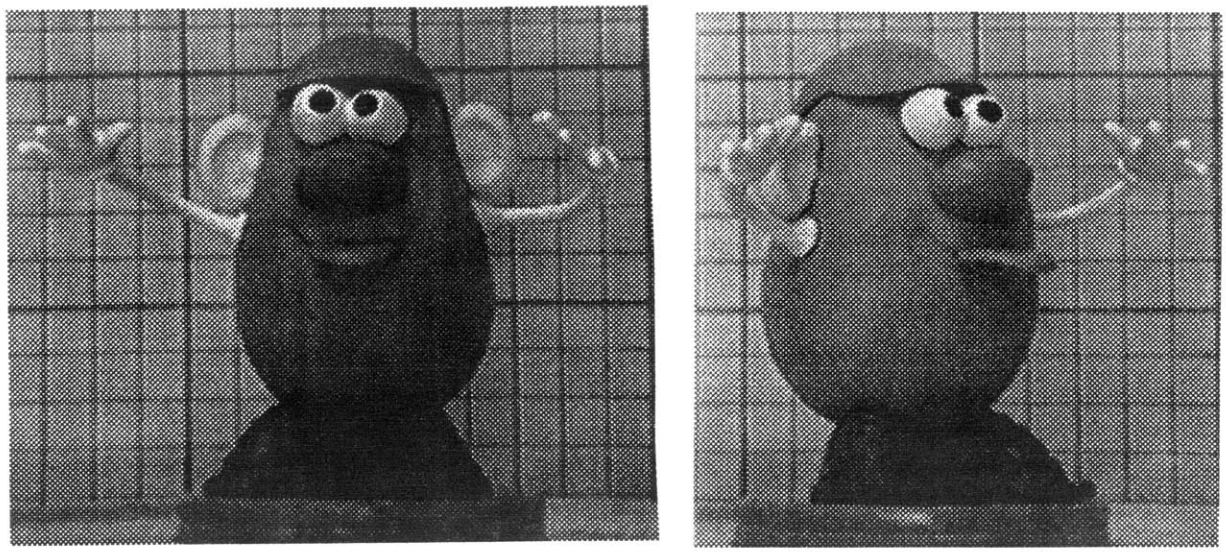


Figure 4-11: Scalespace views of view 0 and the rotated view 45

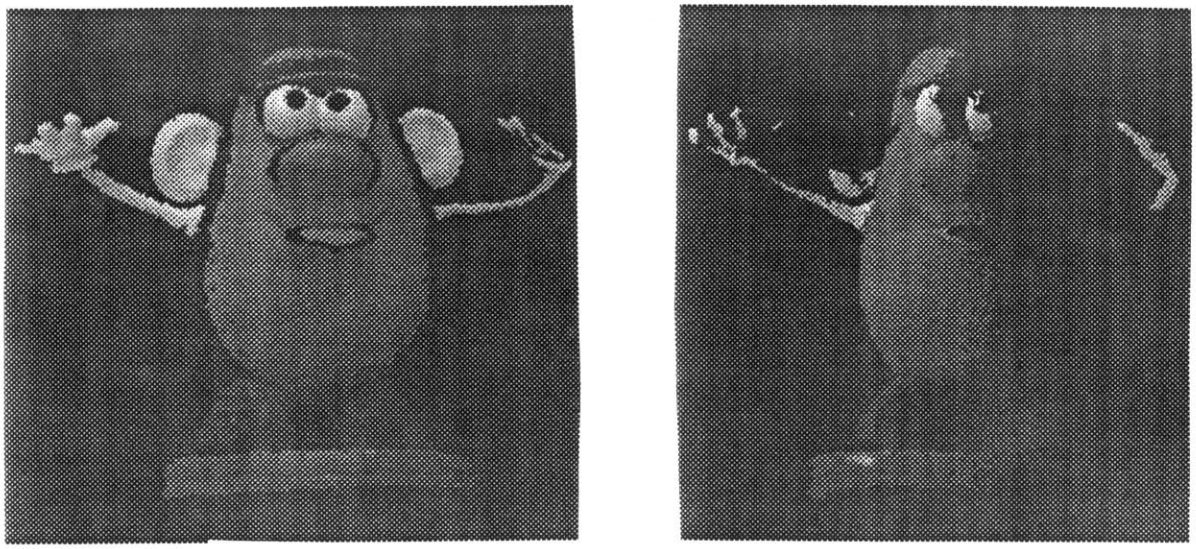


(a) (b)

Figure 4-12: Input for the correspondence is a 0 and 45 degree view

4.4 Scalespace

The photos in 4-11 show the four levels of scalespace for an image of Mr. P. Photo 4-11a is the original reflectivity image for Mr. P. Photos 4-11b, 4-11c and 4-11d are the reflectivity images filtered with $\pi/4$, $\pi/16$, and $\pi/64$ filters respectively. Photos 4-11e through 4-11h show the scalespace reflectivity images of 45 degree view of Mr. P. rotated into alignment with the 0 degree view prior to filtering.



(a) (b)

Figure 4-13: The view 45 is rotated into alignment with view 0

4.5 Correspondence Examples

Results

Photos 4-12a and 4-12b show a 0 degree and a 45 degree view of Mr. Potato Head respectively. Photos 4-13a and 4-13b show the same views with perspective removed and the 45 degree view rotated into alignment with the zero degree view.

Photo 4-14a shows the 0 and 45 degree view after they have been corresponded (from the 0 degree viewpoint) and merged. Photo 4-14b shows the merged particle database projected from a 45 degree viewpoint.

Photo 4-15a shows the same particle database projected from a 22 degree viewpoint. This demonstrates how a complete new view can be constructed from two different

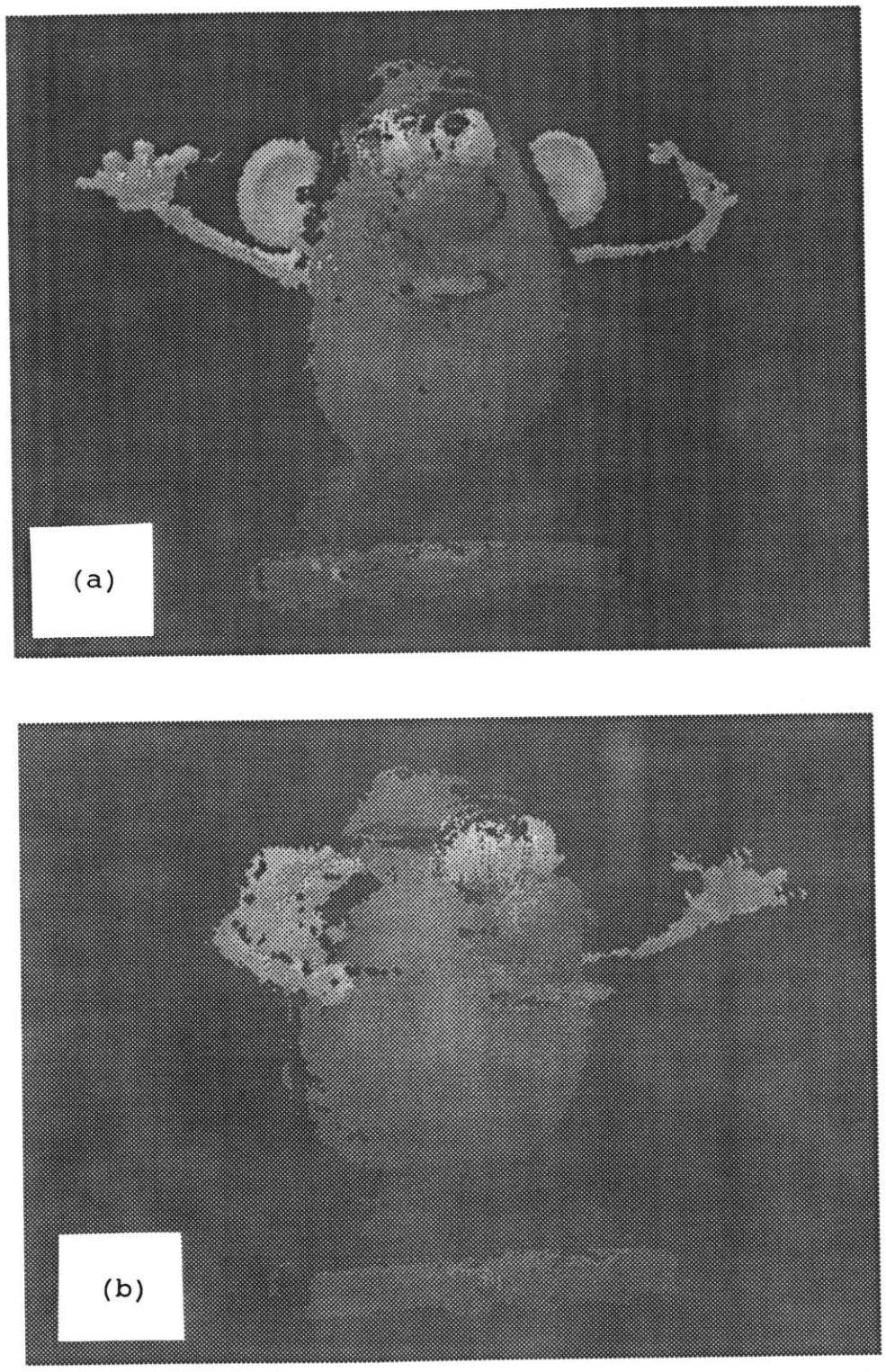


Figure 4-14: Output of the correspondence projected as 0 and 45 degree views

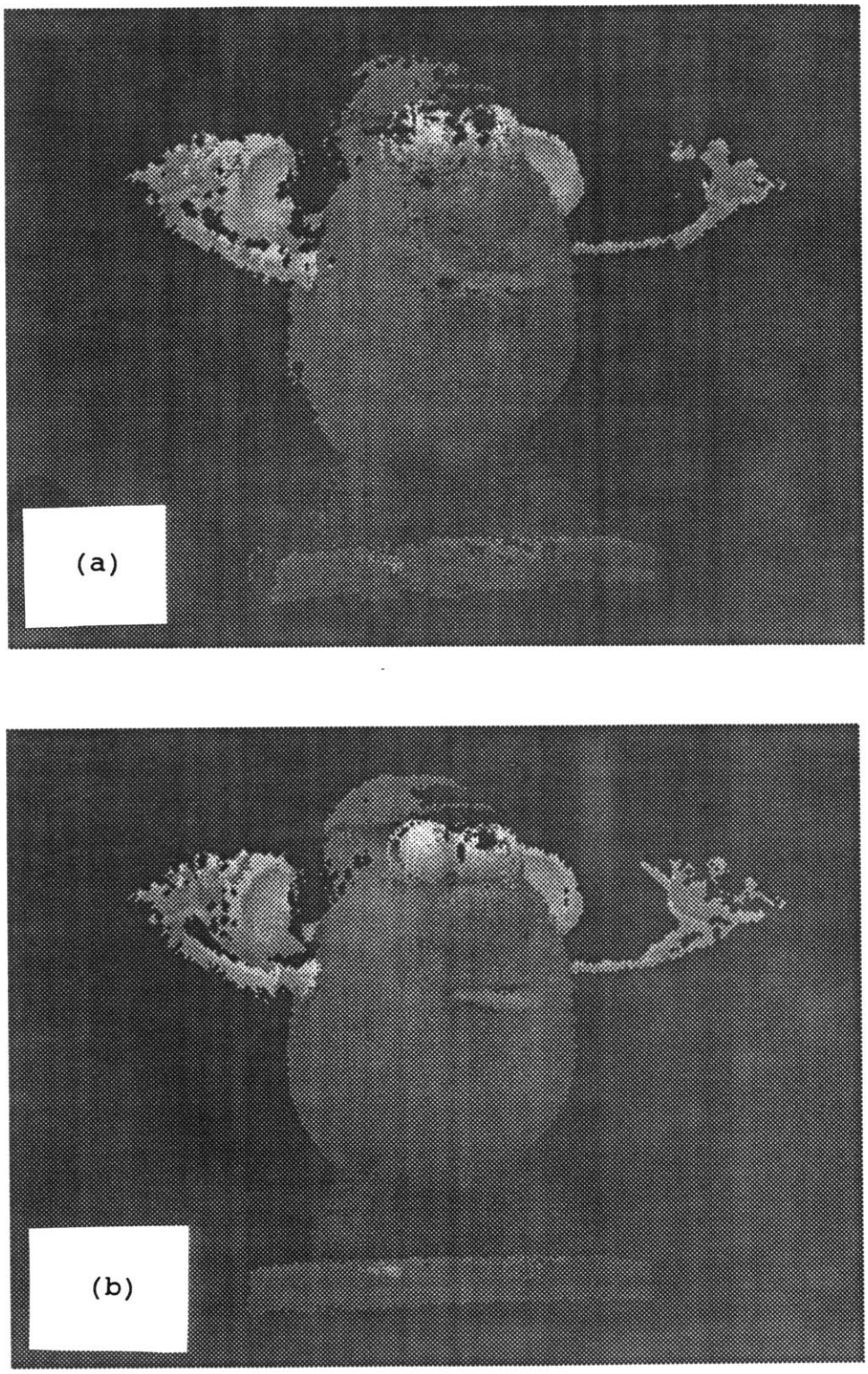


Figure 4-15: Comparing correspondence to juxtaposition on a 22 degree view

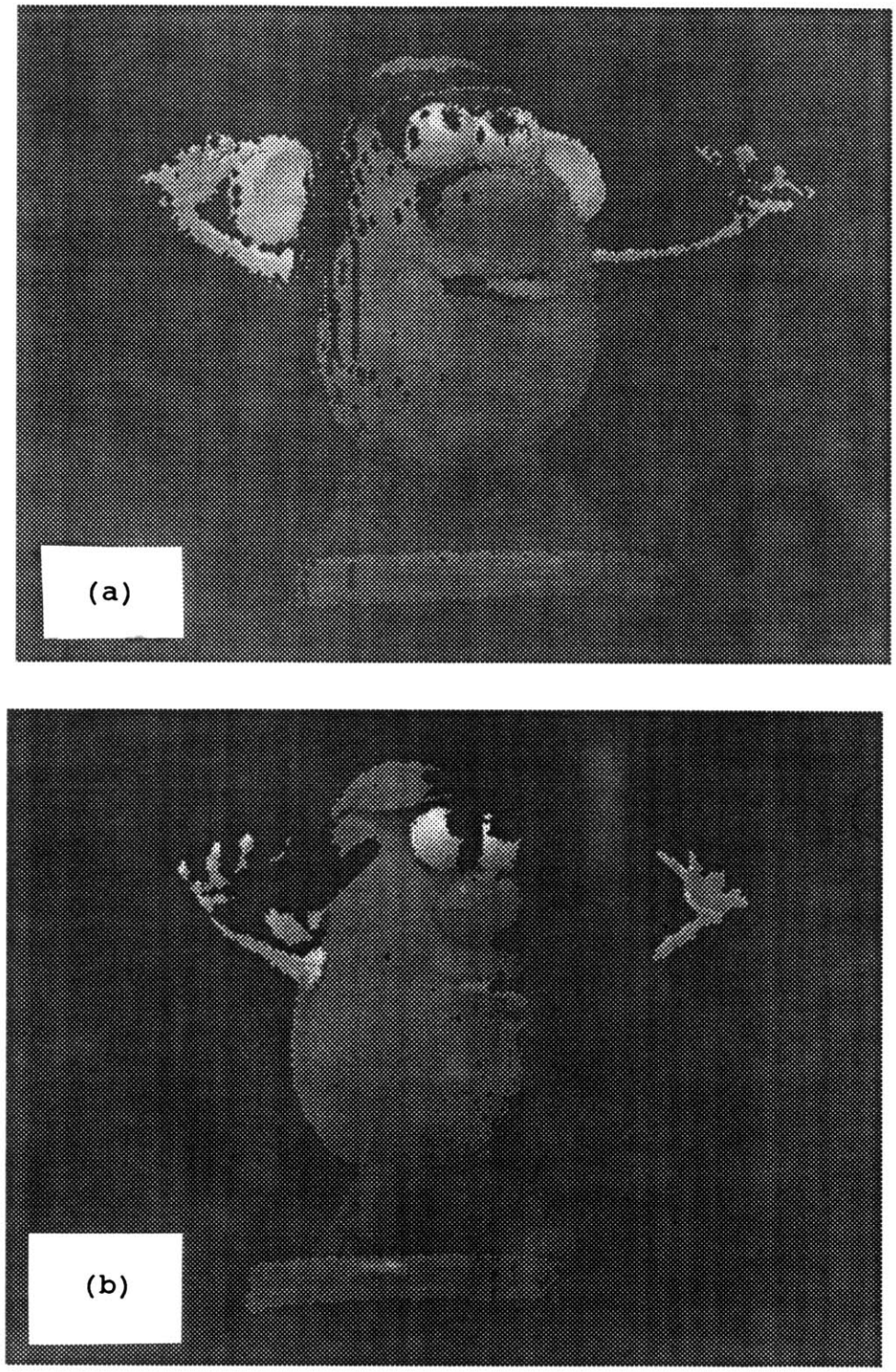
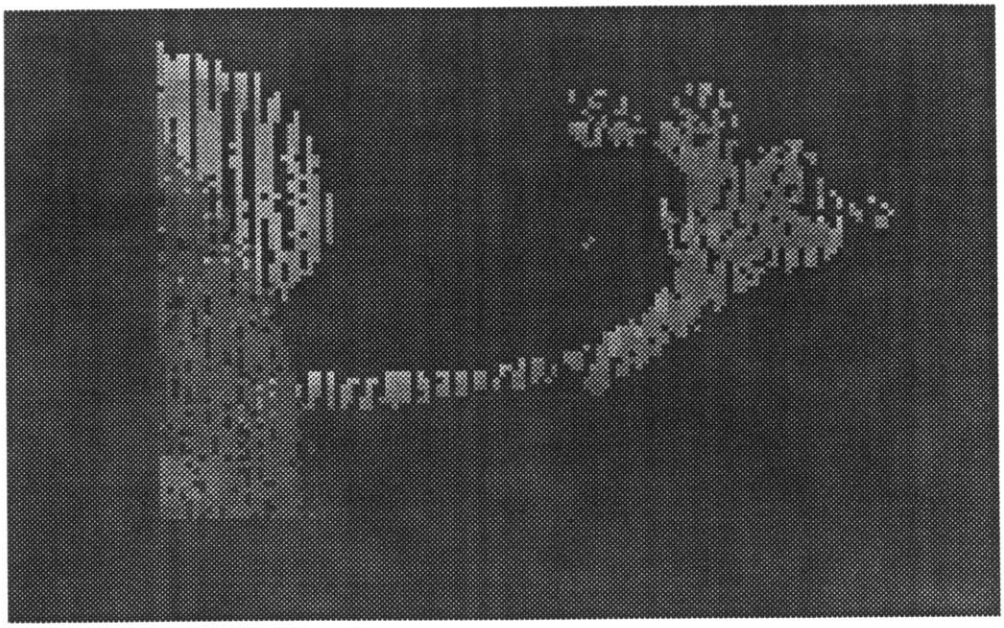
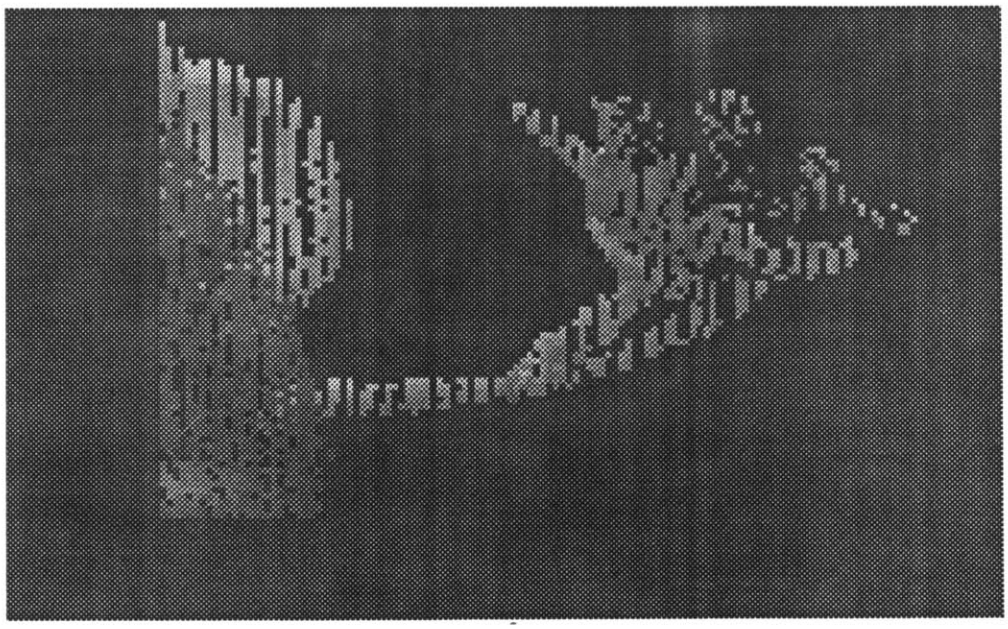


Figure 4-16: Separate 22 degree projections of the 0 and 45 degree views



(a)



(b)

Figure 4-17: Close-ups on alignment of the hand by correspondence and juxtaposition

views.

Photo 4-15a can be compared to photo 4-15b which shows the juxtaposition of a 0 degree and 45 degree view projected from a 22 degree viewpoint. Photo 4-16a shows the 0 degree view rotated to a 22 degree position and photo 4-16b shows the 45 degree view rotated to the same position. Photo 4-15b is the sum of 4-16a and 4-16b. Comparing 4-16a and 4-16b, the considerable distortion in the images is evident.

Photo 4-17a and 4-17b are close-ups of Mr.P.'s left hand from photo 4-15a and 4-15b respectively. These photographs show that the position of the two hands is very poorly aligned in the juxtaposition (4-17b) and that the correspondence algorithm was able to correct the distortion (4-17a). There is even some success in aligning the detail of the fingers on the hand.

However, some areas of the juxtaposed image (4-15b) are clearer than the corresponded image (4-15a). This is particularly true in areas where there were occlusions in one of the original views. In photo 4-16b, the right hand is occluding the right ear. There is also an occlusion near the ear in photo 4-16a. There is a problem with the eyes caused by the lack of range data for the pupils. In photo 4-16b, there is no distinction between the pupils and the background because half of the whites of the eyes are occluded.

Photo 4-18 shows a 22 degree projection of a particle database constructed from the 0 and 45 degree views of the pineapple. The texture of the fruit is reasonably well preserved. Although the leaves are not particularly well matched, this is not perceived

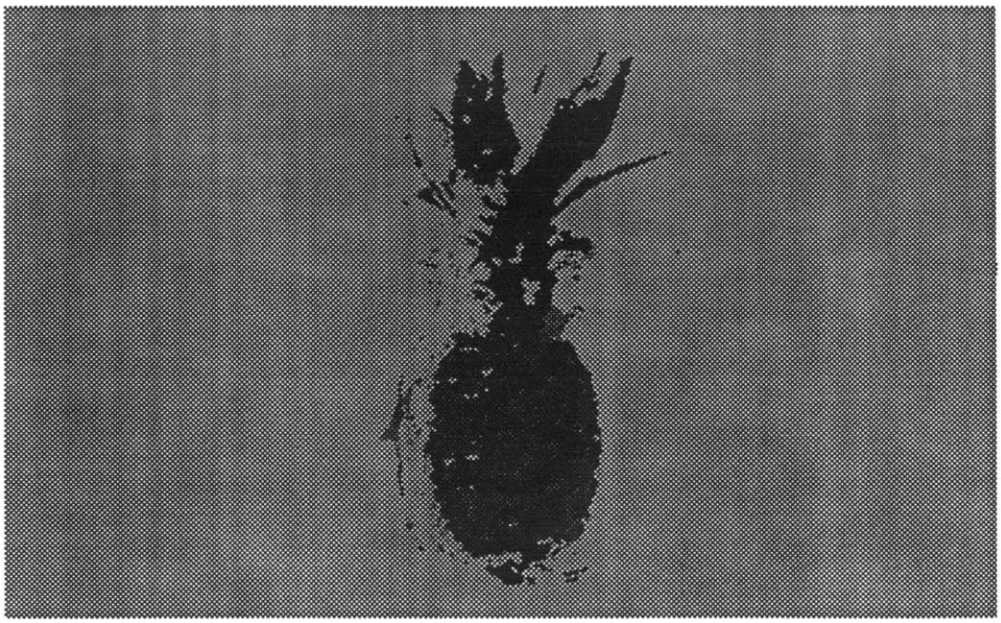


Figure 4-18: Correspondence of 0 and 45 degree view (projected at 22 degrees)

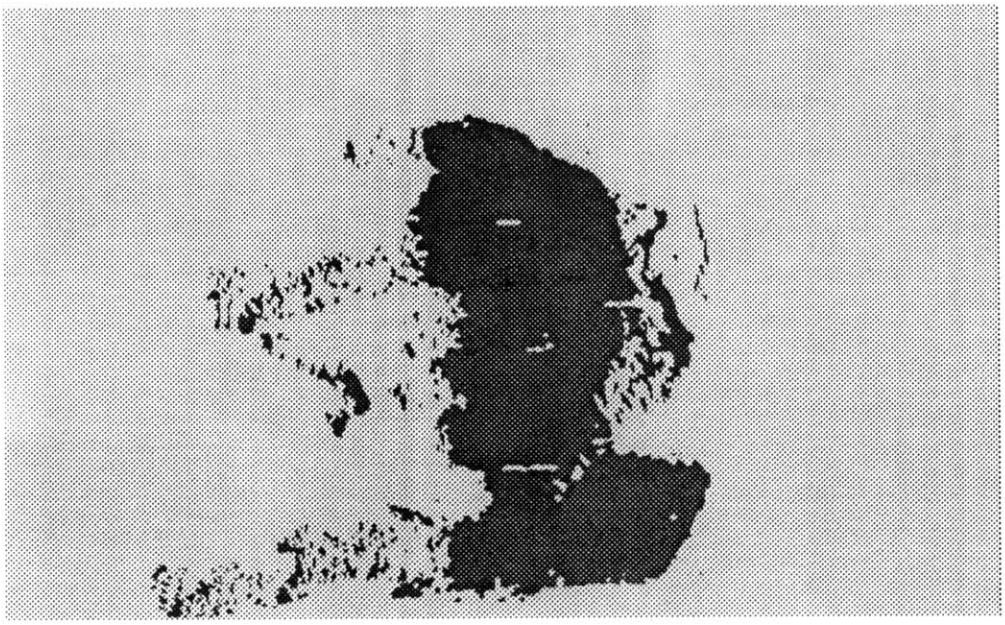


Figure 4-19: Correspondence of 0 and 315 degree views (projected at 337 degrees)

as a defect because the pattern of leaves was originally relatively random.

Photo 4-19 shows a 337 degree projection of a particle database constructed from the 0 and 315 degree views of the Wiesner bust. Without lighting on the bust, it is difficult to evaluate the results.

The photos in 4-20 show that adjusting the threshold of the matching criterion can significantly affect the results. In photo 4-20a, the threshold is set much too high and matches are moving too far away. More experimentation with the weights could resolve some of the lack of clarity in photos 4-15a, 4-18 and 4-19.

Displacement and Relaxation

The photos 4-21 and 4-22 show how the patches are displaced after each level of scalespace. Each of the photos is divided into four quadrants. The upper left quadrant is the 0 degree view of Mr. Potato Head rotated 45 degrees and filtered. The upper right quadrant is the filtered 45 degree view of Mr. P. Underneath each view is the same view after offsets have been added to the patches. Figure 4-23 shows the offset tables for the first level of scalespace (with 64x64 pixel patches) both before and after relaxation. Because the frame of the cage is wider in view 0, the x offset table prescribes that patches be displaced towards the center of the image. There is not much displacement occurring along the y axis. In the z offset tables, a mismatch causes one patch to move in the opposite direction of the others. After relaxation, it conforms to the general movement of its neighbors. The confidence table indicates that the offset

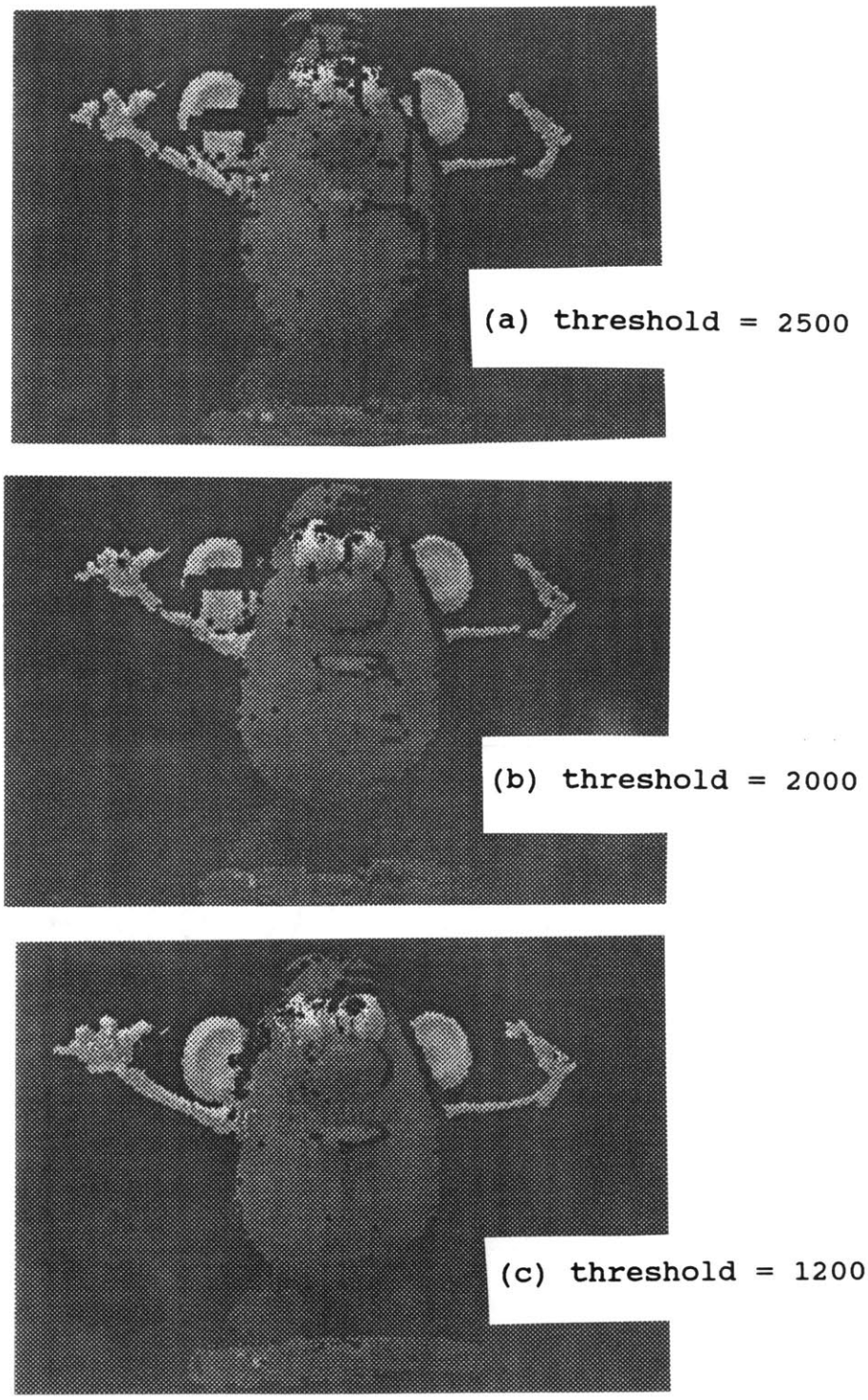


Figure 4-20: Effects of changing threshold of matching criteria

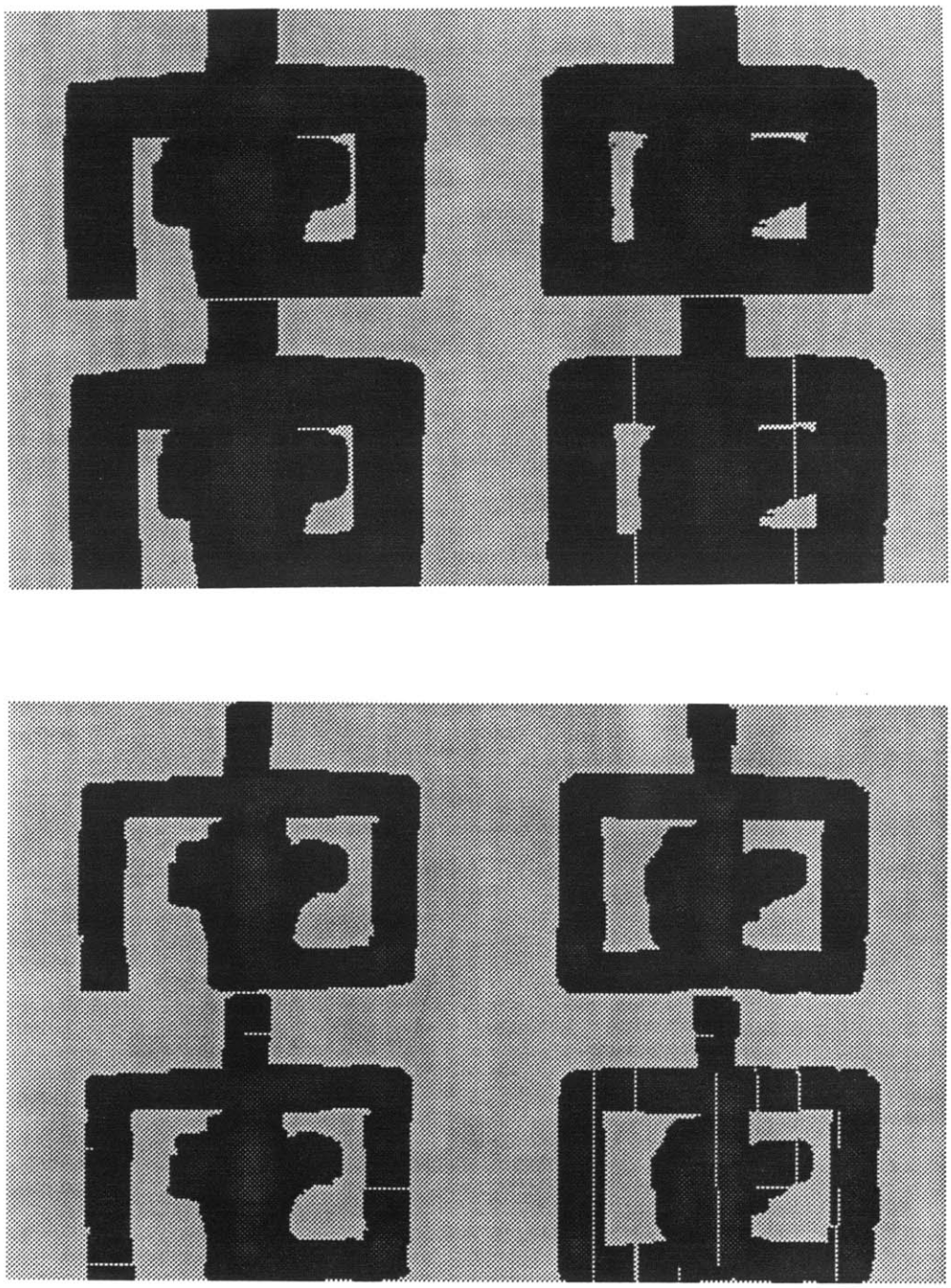


Figure 4-21: $\pi/64$ and $\pi/16$ filtered images before and after offsets

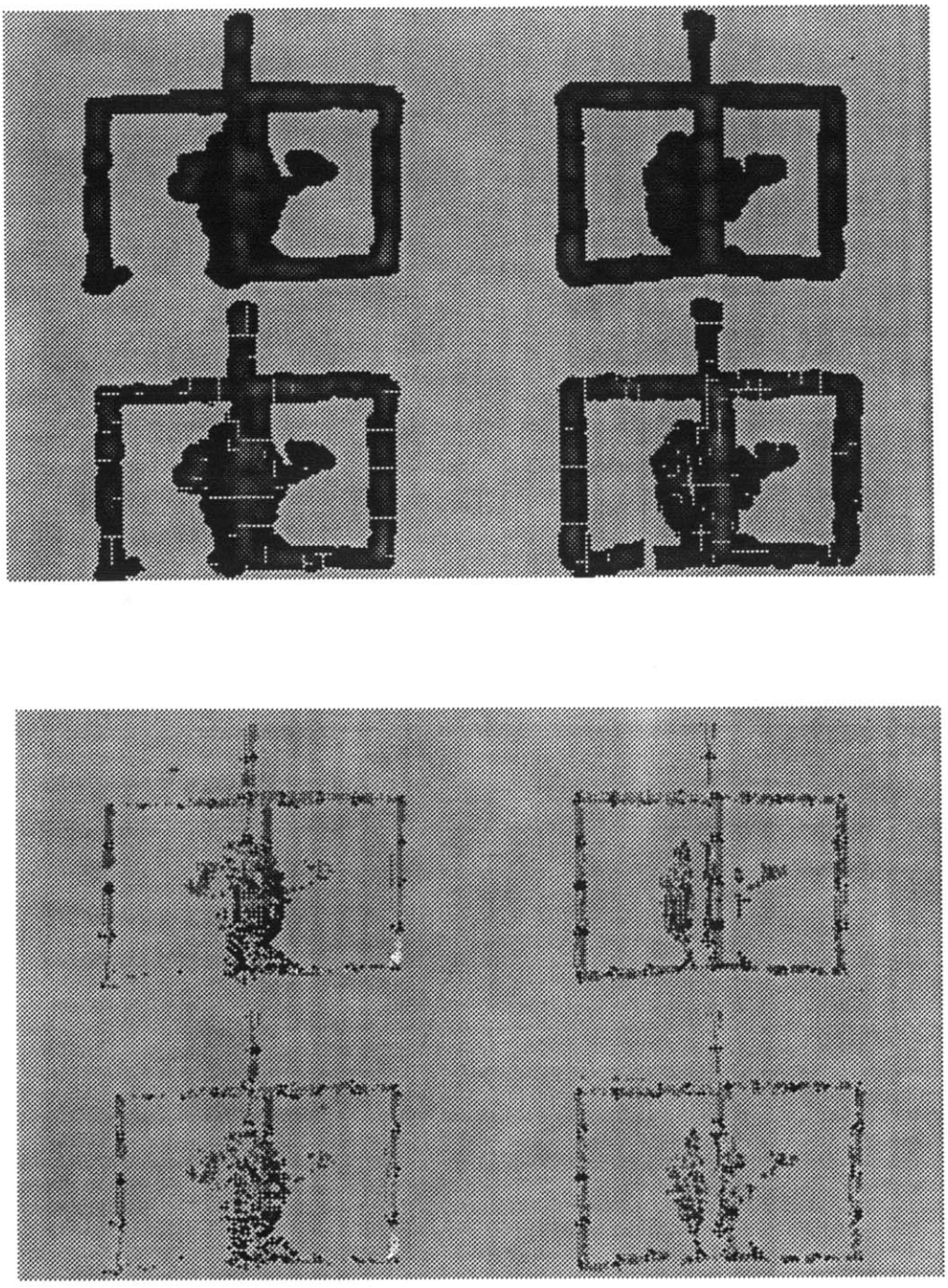


Figure 4-22: $\pi/4$ and original image before and after offsets

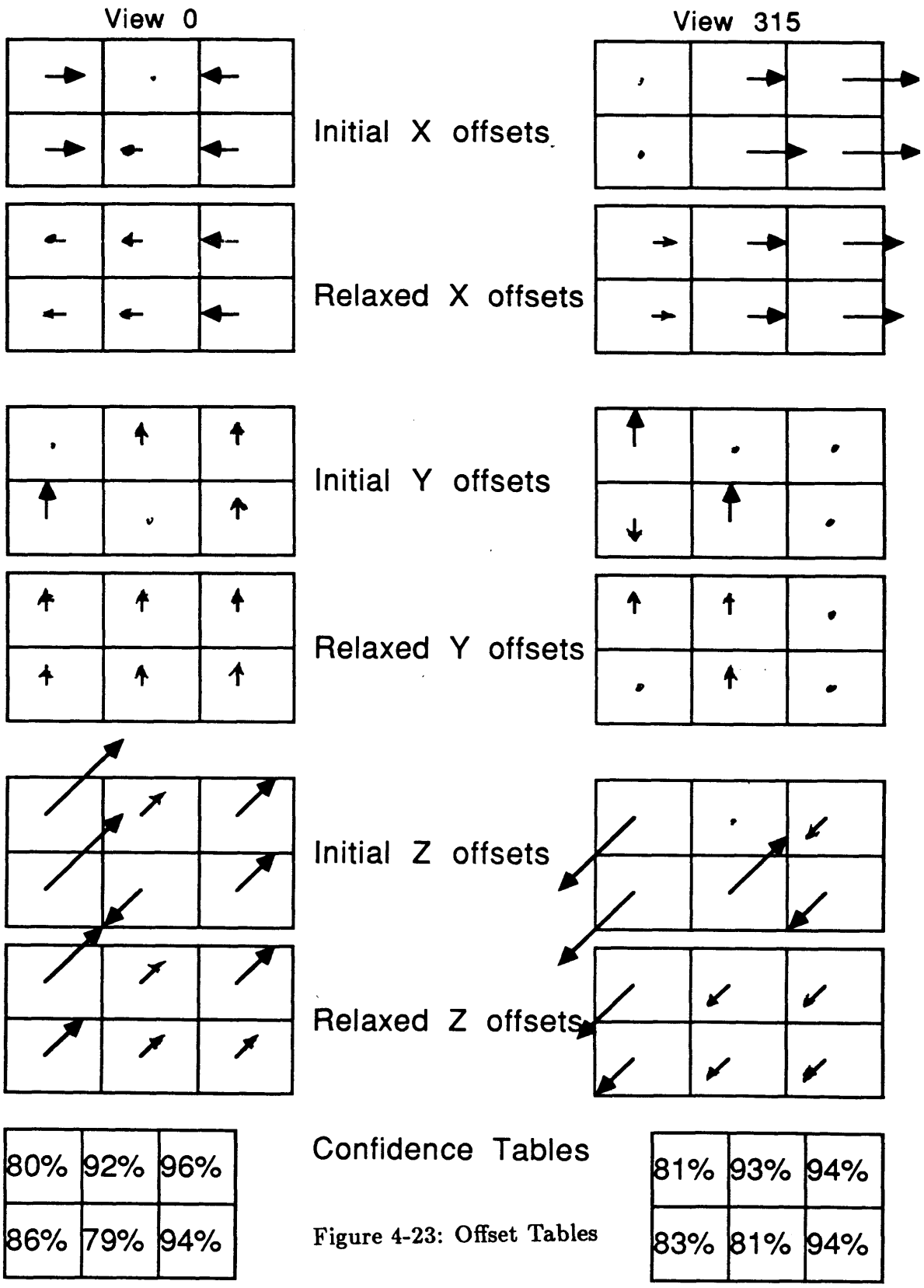


Figure 4-23: Offset Tables

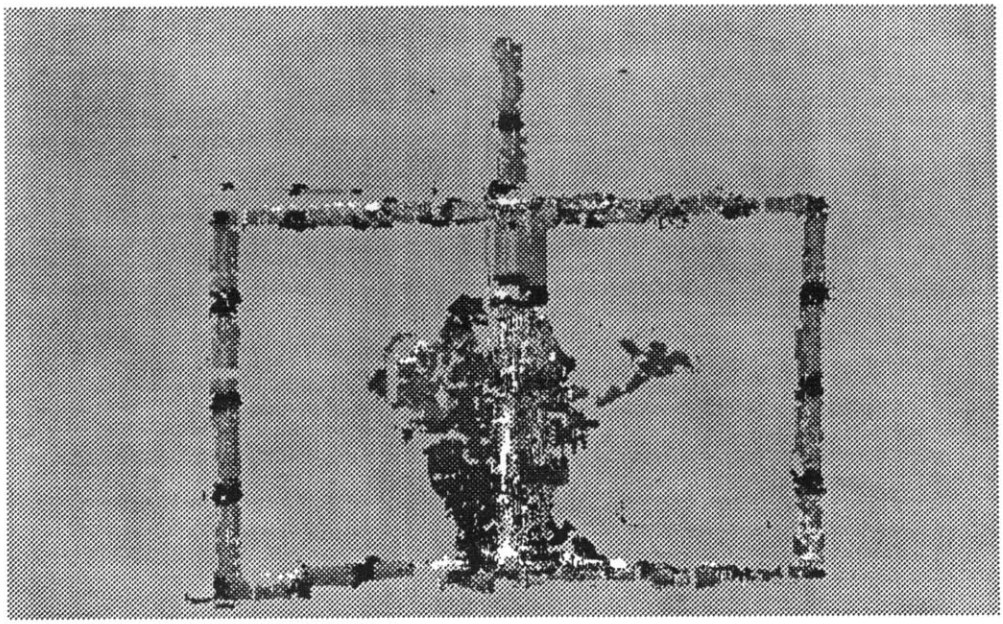


Figure 4-24: Corresponded 0 degree and 45 degree view

vector originally prescribed for this patch was less reliable than those of its neighbors.

Implications of 2D Processing

Photos 4-24, 4-25, 4-26, 4-27, and 4-28 demonstrate a problem that occurs when trying to match 2D patches containing discontinuous surfaces and occlusions. Photo 4-24 shows the projection of a particle database constructed by corresponding a 45 and 0 degree. The 0 degree view was rotated 45 degrees and projected onto a 2D plane for the correspondence (as in photos 4-21 and 4-22). Photo 4-25 shows the juxtaposition of the 45 degree view and the rotated 0 degree view with no correspondence. A comparison of photos 4-24 and 4-25 shows clearly that the correspondence algorithm was very successful in aligning the bars of the cage and removing distortions.

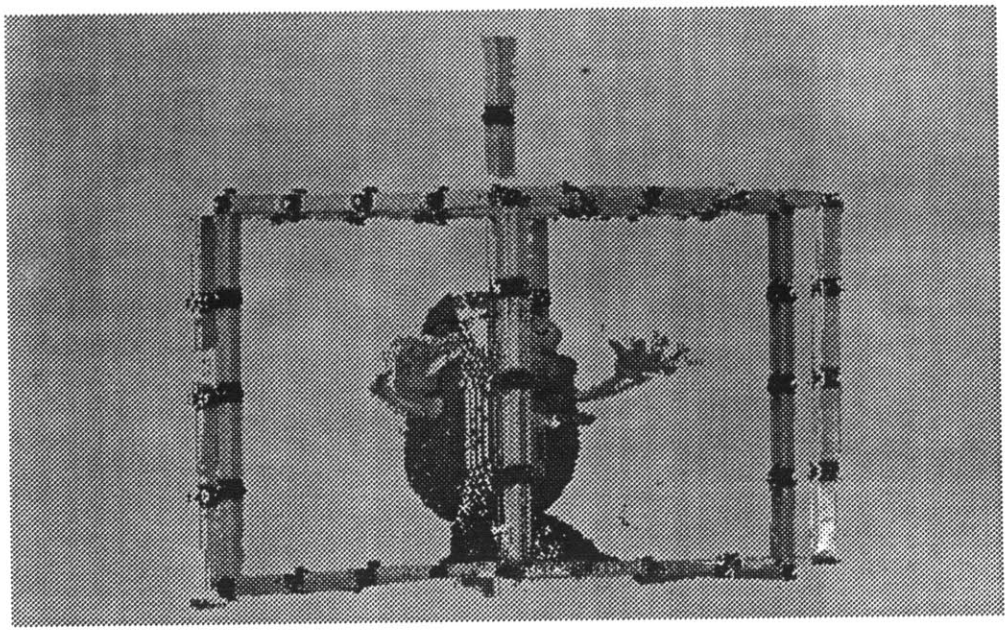


Figure 4-25: Juxtaposed 0 degree and 45 degree view

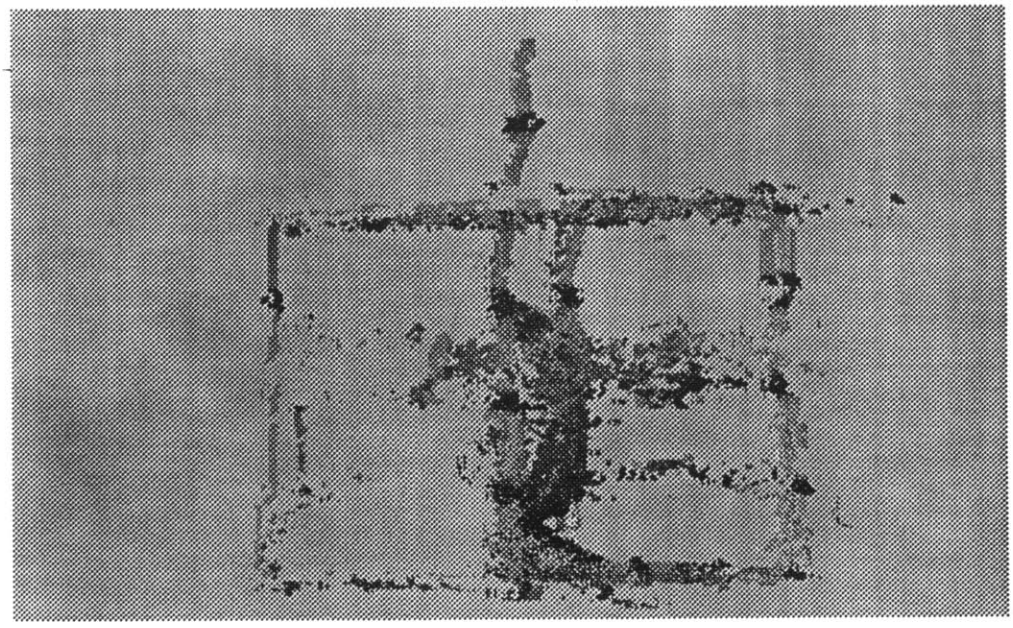


Figure 4-26: Corresponded 0 degree and 45 degree view rotated 90 degrees

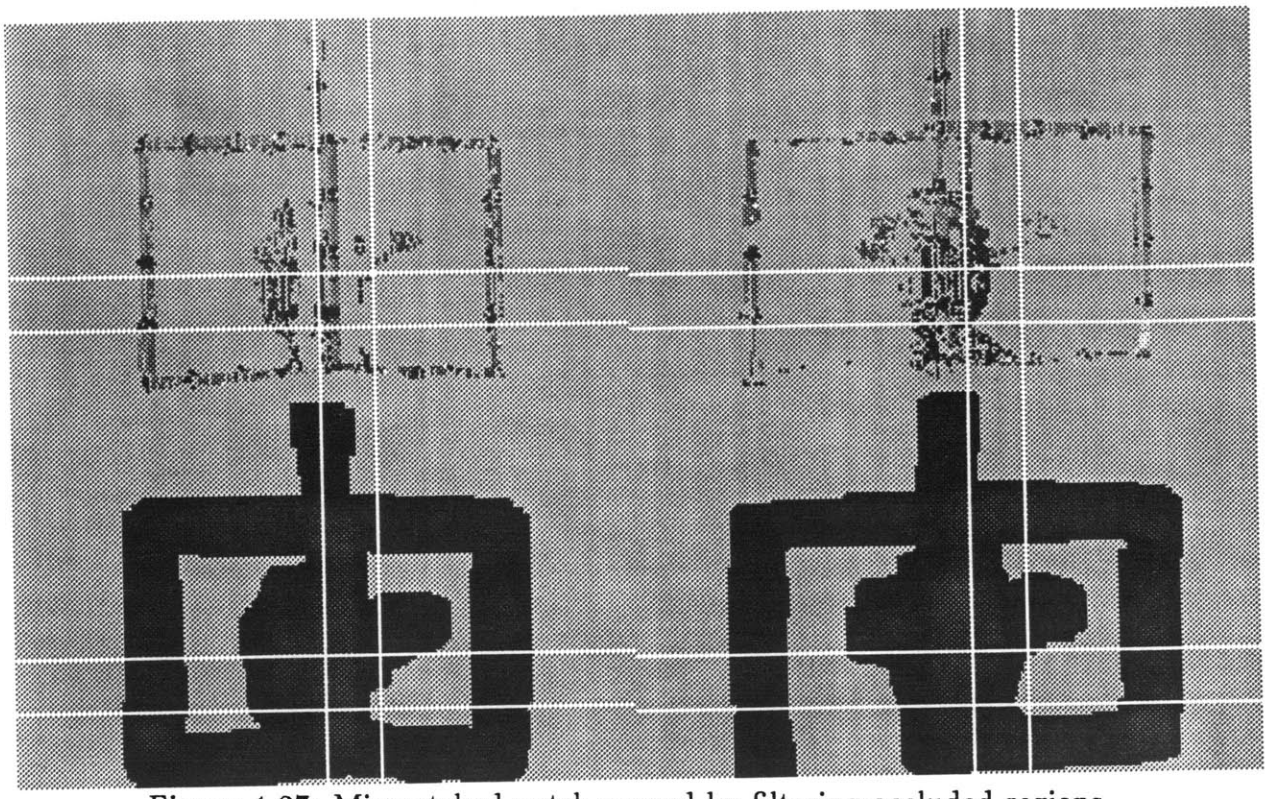
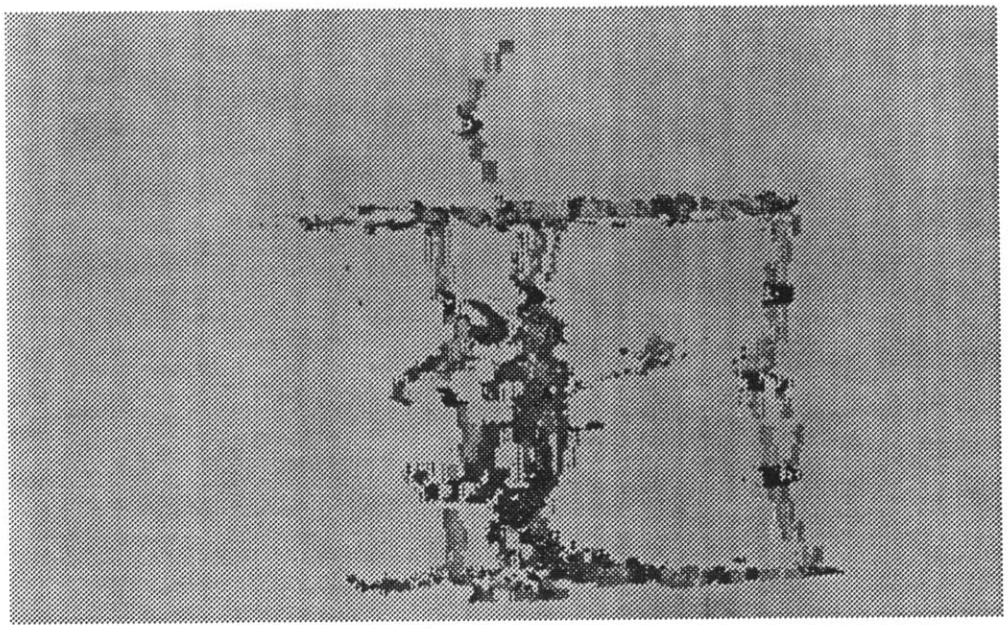


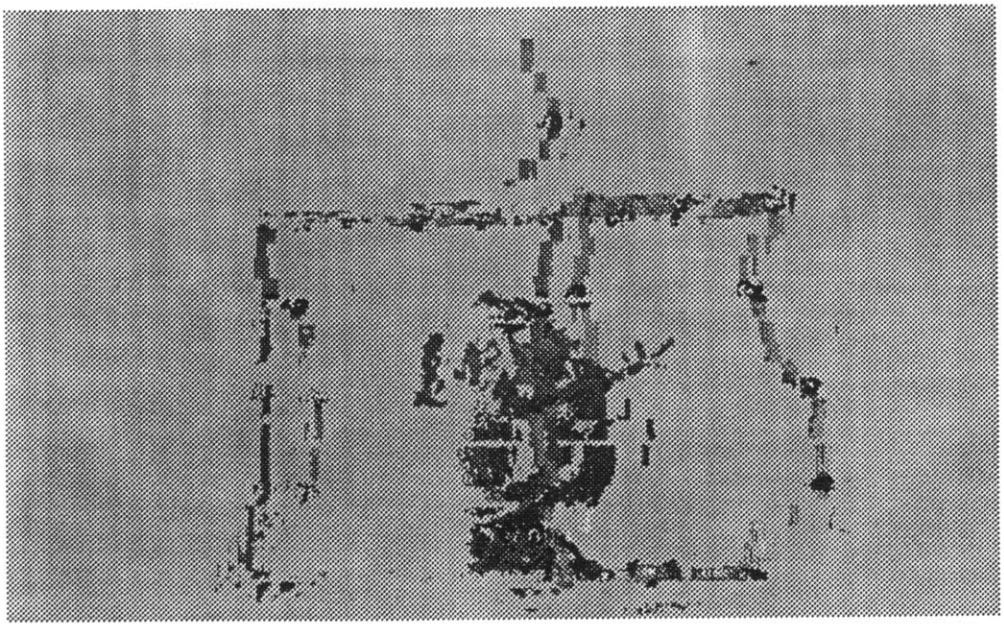
Figure 4-27: Mismatched patch caused by filtering occluded regions

Photo 4-26 shows the same particle database shown in photo 4-24 rotated 90 degrees. There are a large number of misplaced particles floating in between the front bar of the cage and the body of Mr. Potato Head. These particles are on the same vertical plane as the bar and were occluded by it in photo 4-24. The correspondence algorithm applied to 2D views has problems correcting distortions in z, despite its success in correcting distortions in x and y.

Photo 4-27 elucidates the causes of the distortions along the z-axis. The upper left corner of photo 4-27 is the 45 view of Mr. P. and the cage. The upper right corner is the 0 degree view rotated 45 degrees. In the lower left and right corners are the same views filtered with a $\pi/16$ filter. A 16x16 patch in the 45 degree view is marked as is its corresponding patch in the 0 degree view. The patch in the 45 degree view contains



(a) view 45



(b) view 0

Figure 4-28: View 0 and View 45 after adding $\pi/4$ offsets (rotated 90 degrees)

a portion of the front bar of the cage, a region occluded by the bar to the right, and a portion of Mr. Potato Head. The patch in the 0 degree view is 8 pixels to the right and contains only part of the potato and the background.

Two problems have occurred: (1) a bad match has been made between the views, and (2) a bad displacement z-offset has been assigned to the patch in the 45 degree view.

A bad match was made because the patch in 45 degree view contained an occluded region that was filled with the arbitrary value 0. In the filtered view, the patch contains weighted averages calculated from three elements (bar, potato, and occluded zero values). The most similar patch in the 0 degree filtered view contains the weighted averages of only the potato and the background (also arbitrarily assigned the value 0). The occluded region has unduly influenced the weighted average of the patch.

This example demonstrates why filtered views should be normalized. If no information is known about a pixel, it should not influence the averaging of neighboring pixels.

In this instance, the problem could be corrected by normalizing the filtered view. However, the example points out a larger problem. Because of occlusions, and the very nature of the task of combining views, there will always be elements in one image that are not in another. When filtering 2D images, it is always a possibility that patches that should correspond will not because they contain a different number of surfaces. For instance, notice in the zero degree view, the back bar of the cage is visible, but in

the same location in the 45 degree view it is not visible. Some points from the back bar are even visible through the potato because of its sparseness.

The second problem, the assignment of bad z-offset values, is related to the first. Z-offsets are calculated by taking the difference of the average z values of the matching patches. Again, if the matching patches have a different number of depth elements the z-offset will not be relevant.

Photos 4-28a and 4-28b show the 45 and 0 degree views respectively after offsets have been added at an intermediate level of scalespace. The offsets were calculated by comparing $\pi/4$ filtered images of views (both in the 45 degree orientation). After the offsets were added, both views were rotated an additional 90 degrees so that the effects of the z-offsets appear as horizontal displacements (i.e. x and z have been swapped).

Notice the front bar curves backward in the 45 degree view and forward in the 0 degree view. In the 45 degree view, a patch might contain three different surfaces: the front bar, the potato and the back bar. The average depth is in the neighborhood of the depth of the potato. The 0 degree view only contains two surfaces in that area: the front bar and the potato. Its average depth is about half way between the two surfaces, well in front of the potato. The bowing of the bars is caused by the attempt to align the two average depths. It is the 2D filtering of patches with missing occluded surfaces that causes the scattering of points seen in photo 4-26.

Searching and filtering should be done in three dimensions in order to correctly resolve these problems. By using 2D filters for the scalespace algorithm, surfaces which

are near each other in x and y, but potentially far in z are averaged together. Z-offsets, obtained from 2D correspondence of filtered range images, are not reliable.

Nonetheless, two techniques could be used to compensate for the problem. The first is to use the correspondence algorithm to compute only the x and y offsets and then rotate the views 90 degrees and do the correspondence again to compute the z and y offsets. The y offsets should be virtually identical for both passes of the algorithm. This requires at least twice as many computations. Since half of the new view will probably be occluded when rotated 90 degrees, it will be necessary to rotate the views another 270 degree and correspond them a third time.

The second approach is to separate the surfaces in a patch by depth. Once a match is determined in the 2D view, a histogram could be done on the depth values of the patch. Discontinuities in depth distribution would be noted and the points would be grouped into surfaces at different depths. The surfaces in the two matching patches could be matched up and different depth z-offsets could be assigned to each set of points independently. Both solutions have complications and are not as affective as a 3D approach.

4.6 Evaluation

Experimental results confirm that scalespace is a useful tool for combining range images from multiple viewpoints. The power of the algorithm is its ability to match features at different levels of detail. The results are promising and minor optimizations of the

algorithm should produce considerably clear images.

The approach taken attempted to computationally reduce the task from an order N^3 to an order N^2 problem by filtering and corresponding 2D projections of the 3D particle databases. The 2D approach brought up a number of problems associated with occlusions which would not exist if the algorithm were performed in three dimensions. Experimental results reveal that the 2D approach is not sufficient and further research with this algorithm should be done using 3D filtering and corresponding. Other than the computational and memory burdens, no difficulties adapting the scalespace correspondence algorithm into 3D processing are anticipated.

The method of preparing scalespace versions of the particle database must be improved. The current method of filtering the views assumes that the data is continuous in x and y. Filtering in three dimensions will resolve some of the problems discussed in the previous section, but does not affect the problem of variable density of data. Some form of normalization is necessary.

The choice of 45 degree increments between views may have been too ambitious for a scalespace approach which depends on similarities between images. Experimentation on combining views with many different amounts of disparity would be useful for evaluating the scalespace technique.

Chapter 5

Future Directions

5.1 Unimplemented Features of the Scalespace Approach

Occlusion

The fact that the issue of occlusion comes up again and again shows how central it is to the task of 3D model building. Most significant matching problems described in chapters 3 and 4 are directly attributable to occlusions. A comprehensive theory for handling occlusions is much needed.

Chapter 3 discusses one method for avoiding occlusion problems by seeding the database with two opposite views. This would initiate the database with a relatively complete model which could be projected from any angle with minimal occlusions.

Such a scheme would only resolve problems caused by occluded back surfaces of convex objects. It is important to realize that in most real world objects there are a great many large and small occlusions.

In my preliminary study of the problem, I proposed using the shadow volumes technique to locate occluded regions in the particle database. Shadow volumes are used in computer graphics to determine areas not lit by a light source [Cro77] [Ber86]. This technique can be used to give an image realistic quality, because dust in the air can be seen in well lit areas but not in shadows[Max86]. The same mathematical model can be used to determine volumes which cannot be seen from a given camera viewpoint. By creating a shadow volume for each view and then clipping the volumes against each other it should be possible to determine which areas in the particle database are not visible from any of the member views.

For each view added to the particle database, a volume equivalent to the area that would be in a shadow if the light source were at the camera could be created. A polygonal front to the surface of the shadow volume could be constructed by connecting adjacent points in the original range image into triangles[1]. Projecting the contour of the surface against the back wall would give the back surface of the volume. The volume could be completed by connecting the two surfaces along the lines of projection. By rotating the shadow volumes into the same worldspace and clipping them against each other, a union of the volumes can be obtained. This is the area that is not visible in any

[1]For a lower resolution polygonal surface, a algorithm like the 3 point seed algorithm which Bhanu used must be applied [Bha81] [HB82]).

of the views. In the scalespace correspondence algorithm, when the particle database is projected into a 2D frame buffer for comparison with a new view, the (clippped) shadow volume can also be projected into a 2D frame buffer. This 2D representation of the shadow volume can be referred to as the occlusion mask. When patches are compared, only those portions of the patches not masked out by the occlusion mask need be considered for the matching criteria.

Although interesting, this scheme is fatally flawed. Consider the final shadow volume after all views are added. In theory, the surface of this volume would be identical to the surface of the object. After all the views have been added, the only areas that remain occluded are those in the interior of the object! The problem with the shadow volume theory is that it ignores the original problem of distortion. The clipping of the volumes against each other assumes that the camera geometry has been "perfectly" recovered and that the volumes can be rotated into alignment in the same space. This theory may be appropriate for finding large occluded regions but is not sufficient to deal with small occlusions like those in the leaves of the pineapple.

It may indeed be useful to locate large occluded regions, but shadow volumes are not necessary for this task. One can generate a 2D occlusion mask of all the points that can be seen in one view but not the other by simply doing a pixel by pixel comparison of the two images. Then throw away all the pixels in the occlusion mask that are alone and only keep the pixels that are surrounded by a large number of neighboring points. The scattered points are caused by distortions and are eliminated in this fashion; the

points grouped together are caused by occlusions. It is not possible to tell the difference between small occlusions and distortions.

The occlusion problem is complex and these suggestions only reflect a perfunctory examination. Literature on object recognition in complex scenes may provide insight. Symmetry-seeking models are sometimes used to fill missing portions of the object[SK85] [TWK87].

Expanding the Matching Criteria

Improving the precision of the correspondence between the views can be achieved by adding attributes to the matching criterion. Diversity in features is a good quality. Surface normals would be a good complement to the intensity and positional data already used to authenticate a match. Surface normal information can be derived from the range images[2] but is less reliable than the intensity and range data. For each point in the particle database, x and y coordinates of the unit surface normal would have to be stored (z can be deduced from x and y), so the addition of this attribute would be costly in terms of storage[3]. Color is another likely candidate. Each attributes could be added to improve the effectiveness of the matching criteria and would result in a corresponding

[2]Bove extracts surface normals from range images by differentiating dz/dx and dz/dy for each point. To calculate the horizontal (x) coordinate of the surface normal of each point, a window of 50 surrounding points on the same horizontal line is examined. Sudden changes in depth (z) values of adjacent horizontal points in the window represent an edge in the object. The window is truncated to exclude surrounding edges from the surface normal calculation, and thus preventing legitimate edges from being smoothed away. The points in the remaining window can be averaged to obtain the change in z per unit x. In this way, the surface normals will be able to represent even very slight gradients. The same windowing technique is applied to vertical lines to determine the vertical (y) coordinates of the surface normals.

[3]Although a particle does not typically have a surface, the term "surface normal" is used here as a synonym for "orientation".

increase in computation and storage needs. Another interesting possibility would be to use Laplacian rather than Gaussian filters for extracting scalespace features.

Reliability

Another unimplemented concept is that each particle could have a reliability associated with it. If a particle has a high reliability then it is probably in the correct position in space and should have little tendency to move. Conversely, particles with a low reliability should have much flexibility to move to better positions. When matching patches, patches with a low average reliability should tend to move towards patches with a high average reliability. The difficulty with this concept is deciding what criteria to use for measuring reliability. When examining a range image of an object, the tendency is to say that points on the front of the object (i.e. facing the camera) have a higher reliability than those on the side. At first glance, it seems that the surface normals could be used to measure reliability. At the same time, one would think that a view projected from a particle database combining several views would be more reliable than the projection of a single view. When a new view is added to a particle database representing many views, the particle database is projected along the viewing axis of the new view for comparison. Which is more reliable: the new view which is ideally seen from the viewpoint selected or the projected view which represents several views combined together and then rotated?

Density is a good measure of reliability and that reliability should reflect patches

rather than individual particles. Front surfaces are more reliable because there are more visible points but it isn't necessarily true that the few points visible on a side surface are less reliable than any individual point from the front surface. It also stands to reason that a combined particle database is more reliable in those areas which have lots of particles and is not particularly reliable in those regions with few particles. So when two patches are matched, the algorithm should count the points in the (unfiltered) patches and move them towards each other so that the less dense patch tends to move more towards the more dense one.

Parallel Algorithms

Because of the modularity of the overall approach taken here many components of this system can be replaced by more efficient algorithms performing the same tasks without invalidating the whole of the approach.

An obvious example would be the implementation of a parallel algorithm for searching for the best match of a patch. The current algorithm that performs the best match search is the computational bottleneck of the procedure for combining views. If the algorithm were implemented on a machine with parallel processors, the search for the best match of all patches could be done at the same time instead of sequentially.

Some parallel algorithms that do stereo pair matching have a feedback loop where processors communicate to neighboring processors which way their patch is planning on moving [Pra85] [MP76]. Similar in structure to a neural network, there is a collective

correction of the distortions in the images. In this case, the relaxation is incorporated into the matching process rather than applied afterwards. Typically, parallel correspondence algorithms are contrasted to the scalespace approach because scalespace implies a sequential processing of images where information from each level must be garnered before the next level begins. Nonetheless, as suggested above, parallel processing can be used to accelerate the processing of each level. At the same time, it is worth examining non-scalespace parallel correspondence algorithms to see if they can be adapted to this problem.

Elasticity

Doubtlessly there can be improvements made on the relaxation algorithm. It would be interesting to know if elasticity models [TWK87] [AW87] can be fruitfully applied to the relaxation problem. The implementation of the necessary partial differential equations would most likely make the process more computationally costly than would justify its use. Nonetheless, the solution is intellectually elegant and its further study could provide much insight into the problem.

5.2 Extensions

It would be interesting to apply the scalespace correspondence algorithm on range data obtained from a depth from focus camera. This should pose no particular problems and the matching will be considerably quicker on the lower resolution images it provides.

Another worthy experiment would be to test the algorithm on data acquired through several cameras placed at different locations rather than one camera with an object on a turntable. While different distortion characteristics of the cameras may pose some problems, the uniform lighting will enhance the matching process.

The question of a calibration object for aligning the coordinate systems of the different cameras is of interest. It may be possible to recover the camera geometry without recourse to direct measurements of the scene if it is known, for example, that the wire-frame cube is one meter long on all sides. By tracing the vanishing point of the parallel lines in the cube[Bar82], it should be possible to determine the orientation of the cube. The accuracy of results will determine if this technique is tenable.

The human visual system uses a combination of many different techniques for 3D model building. A powerful line of research would be to combine several techniques for model building. For example, combining the structure from motion obtained from Bove's research [Bov89] and the structure from multiple viewpoints would greatly enhance the quality of input used by my algorithm and thus reduce the problems of occlusion. A simple integration of the two techniques could be to use particle databases generated by the structure from motion algorithm as input (instead of views) to the combining algorithm. More sophisticated collaborations where two or more algorithms worked in parallel and gave each other feedback is a fascinating area of further study.

5.3 Beyond Scalespace

A fundamental concept of Movies of the Future Project is that movies should be stored in a representation describing the spatial composition of an scene rather than simply as a sequence of 2D intensity maps. If these scenes can be decomposed into objects many powerful applications will be possible.

To do this it will be necessary to develop techniques for separating objects from the scene and organizing them hierarchically in the database. Extracting objects from a scene can be a simple procedure provided that it is easy to distinguish them from surrounding objects. It is possible to extract objects based on their position; for instance, a lamp could be extracted from a particle database of a scene by clipping a rectangular volume out of the database. Color and intensity can also be used for separating objects from a scene. Zero and first degree discontinuities in the derivatives of range could be indications of boundaries between objects. Combinations of these criteria could be used. With some human intervention it is probably simple to extract objects from 3D scenes. An automatic process for extracting objects without human intervention requires some artificial intelligence and is a significant computer vision issue.

As for the hierarchy, the more sophisticated it is the more powerful the model. In a 3D movie, invariant background information need not be stored or transmitted for every frame of a sequence but may be represented only at one point in the database. Distinctions must be made between rigid and non-rigid objects. Objects may have characteristics like velocity. Ultimately, the total representation may entirely ignore

the concept of the the frame and may choose to represent scenes in terms of objects and motions.

While the football and shopping paradigms discussed in chapter 1 remain many years in the future, significant progress in both hardware and software is currently being achieved.

Appendix A

Sample Camera Geometry Data Structure

MrP.0 5/6/88

VIEWANGLE must equal atan(XLEN/ZLEN)
VIEWANGLE: 0.30893 rads = 17.70043 degrees (viewing angle of camera)
ZLEN: 188 cm (distance from camera to back wall)
XLEN: 60 cm (width of view window at z = ZLEN)

calculate conversion from z steps to cm by any 2 measured points
211 z-steps = 32.5 cm (front of cage upper left corner)
129 z-steps = 8.5 cm (back of cage upper left corner)
cm(z) = (z-zwall)*zstep
ZSTEP: 0.282352941 cm
ZWALL: 101 z

SCREENHEIGHT: 480
SCREENWIDTH: 768
SCREENDEPTH: 255
DEPTHCULL: 125 (clip background: all pts with depth¡125)
ASPECT_RATIO: 1.18

center of turntable
CENTERX: 361 pixels
CENTERY: 480 pixels (irrelevant)
CENTERZ: 173 depth steps
ROTATION: 0.0 degrees

Acknowledgments

Above all, I wish to express my appreciation to my advisor, Walter Bender, whose guidance in my research and studies made it possible to achieve everything I hoped to get out of my experience at the Media Lab and more. I particularly appreciated the freedom Walter gave me to select a passionate thesis topic and the fact that, even on the busiest demo days, he always found time to squeeze me in when I needed help.

Special thanks go to V. Michael Bove for his generosity in sharing the fruits of his research and insight...

And to B. Butera, whose joviality and bottomless well of ideas were a great source of inspiration. Bill was always there with a beacon to help me through the unchartered seas (which he had often sent me into the week before).

My thanks to my office mates, Pat, for his wonderfully brutal critique of my paper on English muffins, and Janet, for her free compile-time therapy sessions.

A tip of the hat to Lacsap, Joel and Janet for their helpful comments on my drafts and to Sphere for his help in the early stages of the project.

Thanks to Janette, Kenji, Dead, Steph, Wad, March and countless others who warmed my days here with a friendly smile.

I would like to extend my thanks to MIT Media Lab staff and community for this unique opportunity to learn and my family and friends for their loving support.

And lastly, this could not be complete without acknowledging Lacsap and the Chairman for the color they added to my stay at the Lab.

Bibliography

[AW87] Alan Barr Andrew Witkin, Kurt Fleischer. Energy constraints on parameterized models. In *SIGGRAPH–87 Conference Proceedings*, pages 225–232, ACM, 1987.

[Bar82] Stephen T. Barnard. *Interpreting Perspective Images.* Technical Report 271, SRI International, November 1982.

[BB82] Ruzena Bajcsy and Chaim Broit. Matching of deformed images. In *Proc. of 6th International Conference on Pattern Recognition*, pages 351–353, IAPR and IEEE, 1982.

[Ber86] Philippe Bergeron. A general version of Crow's shadow volumes. *Computer Graphics and Applications*, September 1986.

[Bha81] Bir Bhanu. *Shape Matching and Image Segmentation Using Stochastic Labeling.* PhD thesis, Image Processing Institute, University of Southern California, 1981.

[Bha82] Bir Bhanu. Surface representation and shape matching of 3-D objects. In *Proc. of the Pattern Recognition and Image Processing Conference*, pages 349–354, IEEE, 1982.

[Bha84] Bir Bhanu. Representation and shape matching of 3-D objects. *IEEE Trans. Pattern Anal. Machine Intell.*, 340–350, may 1984.

[BJ85] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *Computing Surveys*, 75–145, March 1985.

[Bov88] V. Michael Bove, Jr. Pictorial applications for range sensing cameras. In *Proc. Image Processing, Analysis, Measurement, and Quality*, pages 10–17, SPIE, January 1988.

[Bov89] V. Michael Bove, Jr. *Synthetic Movies Derived from Multi-Dimensional Image Sensors.* PhD thesis, MIT Media Laboratory, 1989. (in progress).

[Cro77] Franklin C. Crow. Shadow algorithms for computer graphics. *Computer Graphics*, 11(2), summer 1977.

[CT82] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. In *ACM Transactions on Graphics*, pages 7–24, ACM, january 1982.

[Dan82] Clayton Dane. *An object-centered three-dimensional model builder.* PhD thesis, University of Pennsylvania, 1982.

[DB82] Clayton Dane and Ruzena Bajcsy. An object-centered three-dimensional model builder. In *Proc. of 6th International Conference on Pattern Recognition*, pages 348–350, IAPR and IEEE, 1982.

[FD82] J. D. Foley and A. Van Dam. *Fundamentals of Interactive Computer Graphics.* Addison Wesley, 1982.

[Gri81] William Eric Leifur Grimson. *From Images to Surfaces.* MIT Press, 1981.

[HB82] T.C. Henderson and B. Bhanu. Three-point seed method for the extraction of planar faces from range data. In *Proc. of the Workshop on Industrial Applications of Machine Vision*, pages 181–186, IEEE, 1982.

[HB84] P. Horaud and R.C. Bolles. 3DPO's strategy for matching three-dimensional objects in range data. In *Proc. of the International Conference on Robotics*, pages 78–85, IEEE, 1984.

[Hee87] David Heeger. Model for the extraction of image flow. *J. Opt. Soc. Am.*, August 1987. General Robotics and Active Sensory Processing Laboratory, University of Pennsylvania.

[Hen82] T.C. Henderson. Efficient segmentation method for range data. In *Proc. of the Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision*, pages 46–47, SPJE, 1982.

[HG81] Bethold K. P. Horn and Brian G.Schunck. Determining optical flow. *Artif. Intell.*, 185–203, 1981.

[Hil84] Ellen Hildreth. The computation of the velocity field. In *Proc. R. Soc. Lond.*, pages 189–220, Royal Society of London, 1984.

[Jar83] R.A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2), March 1983.

[JDB82] J.D.Boissonnnat. Representation of object triangularting points in 3-D space. In *Proc. of 6th International Conference on Pattern Recognition*, pages 830–832, IAPR and IEEE, 1982.

[JF81] J.D.Boissonnnat and O.D. Faugeras. Triangulation of 3-D objects. In *Proc. of 7th International Joint Conference on Artificial Intelligence*, pages 658–660, IJCAI, 1981.

[JJ81] Jaswant R. Jain and Anil K. Jain. Displacement measurement and its application in interframe image coding. In *IEEE Transactions on Communications*, pages 1799–1808, IEEE, December 1981.

[LB87] A. Lippman and W. Bender. News and movies in the 50 megabit living room. In *Globecom Proceeding*, IEEE, November 1987.

[Man86] Peter Mansbach. Calibration of a camera and light source by fitting to a physical model. *Computer Vision, Graphics, and Image Processing*, 200–219, 1986.

[Mar82] David Marr. *Vision.* W. H. Freeman and Company, 1982.

[Mar84] Wilson Markle. The development and application of Colorization(R). *SMPTE Journal*, July 1984.

[Max86] Nelson L. Max. Atmospheric illumination and shadows. In *SIGGRAPH-86 Conference Proceedings*, pages 117–124, ACM, 1986.

[MH80] D. Marr and E. Hildreth. Theory of edge detection. In *Proc. R. Soc. Lond.*, pages 187–217, Royal Society of London, 1980.

[MP76] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:283–287, October 1976.

[MP79] D. Marr and T. Poggio. A computational theory of human stereo vision. In *Proc. R. Soc. Lond.*, pages 301–328, Royal Society of London, 1979.

[OABB85] J.M. Ogden, E.H. Adelson, J.R. Bergen, and P.J. Burt. Pyramid-based computer graphics. *RCA Engineer*, Sept/Oct 1985.

[ODF84] O.D.Faugeras. New steps toward a flexible 3-D vision system for robotics. In *Proc. of 7th International Conference on Pattern Recognition*, pages 796–805, IEEE, 1984.

[Pen87a] Alex Pentland. A new sense of depth of field. *IEEE Pattern Analysis and Machine Intelligence*, July 1987.

[Pen87b] Alex Pentland. Shape information from shading: a theory about human perception. In *Proceedings of the International Conference on Computer Vision*, IEEE, December 1987.

[Pot82] Michael Potmesil. *Generating a Three-Dimensional Surface Model of an Object from Multiple Projections.* PhD thesis, Image Processing Laboratory, Rensselaer Polytechnic Institue, October 1982.

[Pot83] Michael Potmesil. Generating models of solid objects by matching 3D surface segments. In *Proc. of 8th International Joint Conference on Artificial Intelligence*, pages 1089–1093, IJCAI, 1983.

[Pra85] Kvetoslav Prazdny. Dectection of binococular disparities. *Biological Cybernetics*, 52, 1985.

[Ree83] William T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *Computer Graphics*, 17(3), July 1983.

[SBD86] Francis J. M. Schmitt, Brian A. Barsky, and Wen-Hui Du. An adaptive subdivision method for surface-fitting from sampled data. In *SIGGRAPH-86 Conference Proceedings*, ACM, 1986.

[SK85] David R. Smith and Takeo Kanade. Autonomous scene description with range imagery. *Computer Vision, Graphics, and Image Processing*, 31, 1985.

[Smi83] Alvy Ray Smith. *The Viewing Transformation.* Technical Report, Lucasfilm Ltd, 1983.

[Sue86] Yashuhito Suenaga. Super-resolution: getting a sharp image from a set of multiple frames. 1986. MIT Media Laboratory (work in progress).

[Ter84] Demetri Terzopoulous. *Multigrid relaxation methods and the analysis of lightness, shading and flow.* Technical Report, MIT Artificial Intelligence Laboratory and Center for Biological Information Processing Whitaker College, October 1984. A.I. Memo No. 803.

[TK84] Fumiaki Tomita and Takeo Kanade. A 3D vision system: generating and matching shape descriptions in range images. In *Second International Symposium of Robotics Research*, pages 35–42, August 1984.

[Tsa86] Roger Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proc. IEEE Computer Vision and Pattern Recognition*, IEEE, 1986.

[TWK87] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Symmetry-seeking models for 3D object reconstruction. In *IEEE Proceedings*, pages 269–276, IEEE, 1987.

[Ull79] Shimon Ullman. *The Interpretation of Visual Motion.* MIT Press, 1979.

[Ull86] Shimon Ullman. *An Approach to Object Recognition: Aligning Pictorial Descriptions.* Technical Report 931, M.I.T. Artificial Intelligence Laboratory, December 1986.