PREMIO: An Overview

Octavia I. Camps, Linda G. Shapiro, and Robert M. Haralick

Intelligent Systems Laboratory
Electrical Engineering Department, FT-10
University of Washington
Seattle, WA 98195, U.S.A.

Abstract

A model-based vision system attempts to find a correspondence between features of an object model and features detected in an image. Most feature-based matching schemes assume that all the features that are potentially visible in a view of an object will appear with equal probability. The resultant matching algorithms have to allow for "errors" without really understanding what they mean. **PREMIO** is an object recognition/localization system under construction at the University of Washington that attempts to model some of the physical processes that can cause these "errors". PREMIO combines techniques of analytic graphics and computer vision to predict how features of the object will appear in images under various assumptions of lighting, viewpoint, sensor, and image processing operators. These analytic predictions are used in a probabilistic matching algorithm to guide the search and to greatly reduce the search space. In this paper, which is a discussion of work in progress, we describe the PREMIO System.

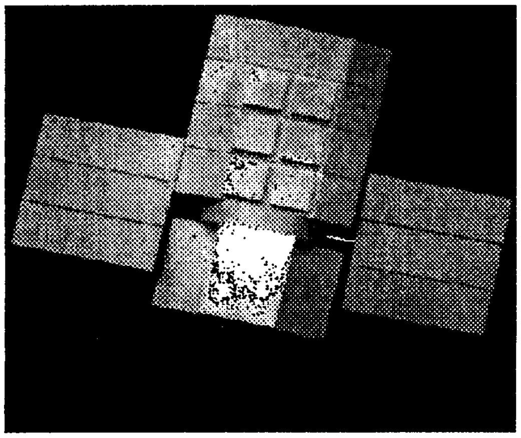
1 Introduction

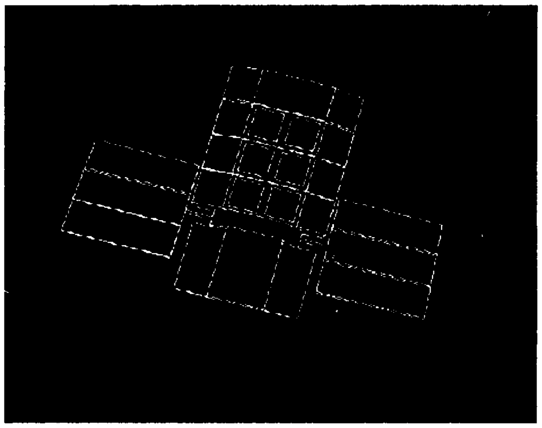
The design of a *model-based vision system* able to recognize and locate an object in an image is an arduous process that involves trial and error experiments and requires a great deal of expertise from the designer. The automation of the design process is highly desirable; it would produce more effective procedures in less time, reducing the software cost of vision systems and expanding their use. Although previous work on automating the design of vision systems have had some success [6, 4, 16, 12, 17], there is still much work to be done on the object recognition and pose estimation problems. We believe that most of the limitations of the previous systems can be removed by the use of a more realistic model of the world. Hence, a better way of representing the interactions between the object representation schemes and the light sources and sensor properties must be found.

An example of the difficulties that a working vision system must address is illustrated in figure 1. Figure 1 (a) shows a grayscale image of a scaled-model of the satellite "Solarmax". Figure 1 (b) shows a naive prediction, which does not take into account the lighting and sensor characteristics of the edges that would be detected by an edge detector applied on the image of figure 1 (a). Figure 1 (c) shows the actual output of an edge detector where several of the predicted edges are fragmented or missing altogether. Knowledge of the degree to which each edge boundary might break up under different lighting and viewing conditions is essential. This knowledge ensures that the inductive matching phase does not have incorrect expectations that cause the search to look for something that does not exist and that the deductive hypothesis verification phase can employ a proper statistical test in which assumptions about what ***should*** be there match the reality of what ***is*** there.

2 PREMIO: A Model–Based Vision System

Most feature-based matching schemes assume that all the features that are potentially visible in a view of an object will appear with equal probability. The resultant matching algorithms have to allow for "errors" without really understanding what they mean. **PREMIO (PREdiction in Matching Images to Objects)** is an object recognition/localization system under construction at the University of Washington [8] that attempts to model some of the physical processes that can cause these "errors". PREMIO uses CAD models of 3D objects and knowledge of surface reflectance properties, light sources, sensors characteristics, and the performance of feature detectors to build a model called the *Vision Model*. The Vision Model is used to generate a model called the *Prediction Model* that is used to automatically generate vision algorithms. The system is illustrated in Figure 2. PREMIO's Vision Model is a more complete model of the world than the ones presented in the literature. It not only describes the object, light sources and camera geometries, but it also models their interactions. The Vision Model has five components: (1) a 3D topological model of the possible objects, describing their geometric properties and the topological relations between their faces, edges, and vertices; (2) a surface physical model, formed by a general model of the light reflection of surfaces and the physical characteristics describing their materials; (3) a light source and sensor geometrical model, representing their configuration in space; (4) a light source


(a) Solarmax grayscale image.


(b) Edge prediction without taking lighting and sensor into account.

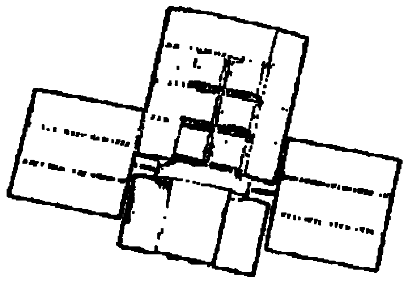

(c) Output of an edge operator.

Figure 1: Problems in Feature Prediction.

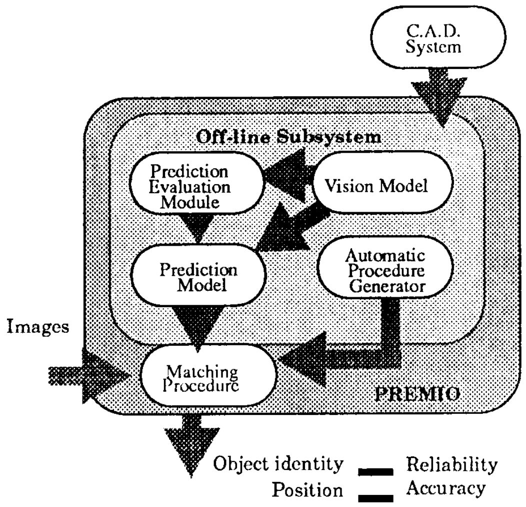


Figure 2: PREMIO: A Model-Based Vision System

and sensor physical model, describing their physical characteristics, and (5) a detector model describing the performance of the feature detectors available to the system.

The system has two major subsystems: an offline subsystem and a online subsystem. The offline subsystem, in turn, has three modules: a Vision Model generator, a feature predictor, and an automatic procedure generator. The Vision Model generator transforms the CAD models of the objects into their topological models and incorporates them into the Vision Model. The feature predictor uses the Vision Model to predict and evaluate the features that can be expected to be detected in an image of an object. The output of the Prediction Module is organized as the Prediction Model. The automatic procedure generator takes as its input the Prediction Model and generates the matching procedure to be used. The online subsystem consists of the matching procedure generated by the offline subsystem. It uses the Vision Model, the Feature Prediction Model, and the input images, first, to hypothesize the occurrence of an object and estimate the reliability of the hypotheses, and second, to determine the object position relative to the camera and estimate the accuracy of the calculated pose. We discuss each in turn.

3 The Vision Model

The Vision Model in a machine vision system is a *representation* of the world where the system works. A representation is a set of conventions about how to describe entities. Finding an appropriate representation is a major part of any problem-solving effort, and in particular in the design of a machine vision system.

The entities that must be described by our representation of the world are the objects to be imaged and the characteristics that these images will have. These characteristics depend on: the geometry of the object; the physical characteristics of the object surfaces; the position of the object with respect to the sensors; the light sources and other objects; the characteristics of the light sources and the sensors, and ultimately, the characteristics of the device that "observes" the image.

3.1 Object Models

PREMIO assumes that it has available PADL2 CAD models of all the possible objects to be imaged. PADL2 is a constructive solid geometry (CSG) modeler designed by H. B. Voelcker and A. G. Requicha at the University of Rochester. Its primitives are spheres, cylinders, cones, rectangular parallelepipeds, wedges and tori.

PREMIO's object model is a hierarchical, relational model similar to the one proposed in [27]. The object model is called a ***topological object model*** because it not only represents the geometry of the objects but also the relations among their faces, edges, and vertices.

The model has six levels. A world level that is concerned with the arrangement of the different objects in the world. An object level that is concerned with the arrangement of the different faces, edges and vertices that form the objects. A face level that describes a face in terms of its surfaces and its boundaries. A surface level that specifies the elemental pieces that form those surfaces and the arcs that form the boundaries. Finally, a 1D piece level that specifies the elemental pieces that form the arcs.

The type of surfaces that a PADL2 model can have are the surfaces of its primitives: planes, spheres, cylinders, cones and tori. To represent these surfaces PREMIO uses the same representation that PADL2 uses: an implicit mathematical expression that represents the corresponding primitive in a "natural" coordinate system that makes this expression as simple as possible. An object modeled with PADL2 can only present boundary arcs that result from the intersection of its primitives. To represent these curves we also followed PADL2 choice: a parametric expression for each coordinate, with a range interval for each parameter, that represents the curve in a "natural" coordinate system that makes these expressions as simple as possible.

To create the topological object model from the PADL2 model, PREMIO uses the boundary file routines provided by PADL2. These routines give access to all the information concerning the face surfaces and the boundary arcs of the objects, but do not provide a direct way to extract the boundary, edge and vertex information that we need. To find the boundaries of a face, its arcs must be grouped together to form closed loops. This can be done using the algorithm developed by Welch [31] to find closed loops in an undirected graph. At the same time the edge and the vertex information can be updated. The edge relation provides a way to relate two faces that have an arc in common, while the vertex relation relates all the arcs that have a vertex in common. These two relations are very useful in the prediction of image features.

3.2 Object Surface Model

Given the physical properties of a material, it is possible to predict the properties of images of this material. Different materials reflect the light in different ways, producing different intensity values in the image. A reflection model of a surface is a series of equations designed to predict the intensity values of points in a scene. Given the light sources, the surface, and the position of the observer, the model describes the intensity and spectral composition of the reflected light reaching the observer. The intensity of the reflected light depends on the intensities, sizes, and positions of the light sources and on the reflecting ability and surface properties of the material. The spectral composition of the reflected light depends on the spectral composition of the light sources and on the wavelength-selective reflection of the surface.

The light reflected by a small region of a material can be broken down into three components: ambient, diffuse and specular. The ambient component describes the amount of light reaching the surface by reflection or scattering of the light sources or other background illuminators. Usually, ambient light can be assumed to be equal for all points on the surface and is reflected equally in all directions.

Most real surfaces are neither ideal specular (mirror-like) reflectors nor ideal diffuse (Lambertian) reflectors. Buchanan [7] has evaluated several reflectance models and concluded that Cook and Torrance's model [10] is the most accurate when the incident light is completely unpolarized. However, in general light is partially polarized. Yi [32] derived an extension of Cook's model for polarized light. PREMIO uses this model.

3.3 Light Sources and Sensors Models

Image formation occurs when a sensor registers radiation that has interacted with physical objects. Hence, it is important to include the light sources and sensor models in our vision model. A light source model must describe its position in space, its size and shape, and its wavelength components. A sensor model must describe its position in space, its response to the radiation input, and its resolution. In the offline system of PREMIO, the sensor and the light source positions are known. The sensor and light sources are placed on the surface of a sphere centered at the origin of the object coordinate system, called the reference sphere. The points on the reference sphere constitute a continuous viewing space. The viewing space is sampled [32] in a way such that the distance between any two neighboring points in the discrete viewing space is approximately the same.

The image intensity of a given point P in a given surface is given by [32]:

$$I = \int CSQ(\lambda) d\omega \vec{N} \cdot \vec{L} (R_{\parallel}(\lambda) J^i_{\parallel}(\lambda) + R_{\perp}(\lambda) J^i_{\perp}(\lambda)) d\lambda \tag{1}$$

where $\vec{N}$ is the unit normal to the given surface at P, $\vec{L}$ is the unit vector in the direction of the light source from P, C is the lens collection factor, S is the sensor responsivity, Q is the spectral distribution of the illumination source, ω is solid angle, $J^i_{\parallel}$ and $J^i_{\perp}$ are the illumination intensities of the parallel and perpendicular polarized incident light, and $R_{\parallel}$ and $R_{\perp}$ are the bi-directional functions for the parallel and perpendicular polarized incident light.

The lens collection factor, C, is given by [32]:

$$C = \frac{\pi}{4}\left(\frac{a}{f}\right)^2 \cos^4 \alpha \ . \qquad (2)$$

where f is the focal distance of the lens, a is the diameter of the lens, and α is the angle between the ray from the object patch to the center of the lens. The sensor responsivity S, is in general a function of the wavelength of the incident light. However, for monochromatic sensors it can be approximated to one, regardless of the wavelength of the incident light.

4 Feature Prediction Module

Given a vision model representing the world, the goal of the prediction module is threefold: (1) it has to predict the features that will appear on an image taken from the object from a given viewpoint and under given lighting conditions; (2) it has to evaluate the usefulness of the predicted features, and (3) it has to organize the data produced by (1) and (2) in a efficient and convenient way for later use. Our approach to this is analytic.

4.1 Predicting Features

There are two different approaches to the use of CAD-Vision models for feature prediction: synthetic-image-based prediction and model-based feature prediction.

Synthetic-image-based feature prediction consists of generating synthetic images and extracting their features by applying the same process that will be applied to the real images. Amanatides [1] recently surveyed different techniques used in realistic image generation. A particularly powerful technique used to achieve realism is ray casting: cast a ray from the center of projection through each picture element and identify the visible surface as the surface that intersects the ray closer to the center of projection. Bhanu et al [3] used ray casting to simulate range images for their vision model.

Model-based feature prediction uses models of the object, of the light sources and of the reflectance properties of the materials together with the laws of physics to analytically predict those features that will appear in the image for a given view without actually generating the gray-tone images. Instead, only data structures are generated. This is a more difficult approach, but it provides a more computationally efficient framework suitable for deductive and inductive reasoning. This is the approach used by PREMIO.

4.2 Model-Based Feature Prediction

The model-based feature prediction task can be divided into three steps: The first step is to find the edges that would appear in the image, taking into account only the object geometry and the viewing specifications. The result is similar to a wireframe rendering of the object, with the hidden lines and surfaces removed. The second step is to use the material reflectance properties and the lighting knowledge to find the contrast values along the edges in a perspective projective image, and to predict any edge that may appear due to highlighted or shaded regions on the image. The third and last step is to interpret and group the predicted edges into more complex features such as triplets, corners, forks, holes, etc.

4.2.1 Wireframe Prediction

The problem of determining which parts of an object should appear and which parts should be omitted is a well-known problem in computer graphics. A complete survey of algorithms to solve the "Hidden-Line, Hidden-Surface" problem can be found in [29]. A particularly efficient way of solving this problem is using an analytical approach, by projecting the object surface and boundary equations onto the image plane and determining whether the resulting edges are visible or not. This approach obtains the edges as a whole, as opposed to the ray casting approach, which finds the edges pixel by pixel. The aim of the solution is to compute "exactly" what the image should be; it will be correct even if enlarged many times, while ray casting solutions are calculated for a given resolution. Hence this is the preferred method for our application.

In order to analytically predict a wireframe we need to introduce the following definitions:

Def. 4.1 A *boundary* is a closed curve formed by points on the object where the surface normal is discontinuous.

Def. 4.2 A *limb* is a curve formed by points on the surface of the object where the line of sight is tangent to the surface, i.e. perpendicular to the surface normal.

Def. 4.3 A *contour* is the projection of a limb or a boundary onto the image plane.

Def. 4.4 A *T-junction* is a point where two contours intersect.

Def. 4.5 A *cusp point* is a limb point where the line of sight is aligned with the limb tangent.

The edges in an image are a subset of the set of contours. A piece of a contour will not appear in the image if its corresponding boundary or limb is part of a surface that is partially or totally occluded by another surface closer to the point of view. Since the visibility of a contour only changes at a cusp point or a T-junction point, it follows that to find the edges on the image the following steps have to be taken: (1) find all the limbs and cusp points, (2) project the boundaries and limbs to find the contours and all the T-junctions and (3) determine the visibility of the contours by finding the object surface closest to the point of view at each T-junction and cusp point.

Finding Limbs and Cusp Points

To find the analytical expressions for the limbs and cusp points, PREMIO uses an approach similar to the one used in [22], but designed for PADL2-modelable objects instead of generalized cylinders.

Let P_0 with object coordinates (X_0, Y_0, Z_0) be the projection center and let P with object coordinates (X, Y, Z) be a point on a limb on the surface S defined by the implicit equation $f(X, Y, Z) = 0$. Then, the vector of sight $\vec{v}$ from P_0 to P is given by:

$$\vec{v} = (X - X_0, Y - Y_0, Z - Z_0) \tag{3}$$

and the normal $\vec{N}$ to the surface S is given by:

$$\vec{N} = \left(\frac{\partial f}{\partial X}, \frac{\partial f}{\partial Y}, \frac{\partial f}{\partial Z}\right) \tag{4}$$

In order for P to belong to the limb curve, P must be on the surface S and the line of sight must be perpendicular to the normal $\vec{N}$ at P. Hence the limb equations are given by:

$$\left\{\begin{array}{rcl} \vec{v} \cdot \vec{N} & = & 0 \\ f(X, Y, Z) & = & 0 \end{array}\right. \tag{5}$$

Once the limb equations are solved, a limb can be expressed in a parametrized form:

$$\left\{\begin{array}{rcl} X & = & X(t) \\ Y & = & Y(t) \quad t_{min} \leq t \leq t_{max} \\ Z & = & Z(t) \end{array}\right. \tag{6}$$

Then, the tangent vector $\vec{T}$ to the limb is given by:

$$\vec{T} = \left(\frac{\partial X}{\partial t}, \frac{\partial Y}{\partial t}, \frac{\partial Z}{\partial t}\right) \tag{7}$$

Since a cusp point C is a limb point where the line of sight is aligned with the limb tangent, its coordinates must satisfy the following equations:

$$\left\{\begin{array}{rcl} \vec{T} \times \vec{v} & = & 0 \\ X & = & X(t) \\ Y & = & Y(t) \quad t_{min} \leq t \leq t_{max} \\ Z & = & Z(t) \end{array}\right. \tag{8}$$

This procedure is performed in $O(s)$ time where s is the number of curved surfaces of the object.

Finding the contours and T-junctions

To find the contours, the limbs and boundaries of the object are projected onto the image plane; to find the T-junctions the intersections of the contours are found. The intersection detection problem for n planar objects has been extensively studied and it can be solved in $O(n \log n + s)$ time [23], where s is the number of intersections. In our case, the objects are the set of contours. For PADL2 primitives, the limb curves are either circles or straight lines, while the boundaries can be either straight lines, conics or more complex curves. Since the perspective projection of a straight line is another straight line, and the perspective projection of a conic is another conic, we can find a closed form solution for the intersections between the contours that result from projecting straight lines and conics. To find other type of T-junctions, a numerical approach must be used.

Determining Visibility

The next step is to determine the edges and surfaces that are hidden by occlusion. Appel [2], Loutrel [19], and Galimberti and Montanari [11] have presented similar algorithms for analytical hidden line removal for line drawings. They define the ***quantitative invisibility*** of a point as the number of relevant faces that lie between the point and the camera. Then, the problem of hidden line removal reduces to computing the quantitative invisibility of every point on each relevant edge. The computational effort involved in this task is dramatically reduced by the fact that an object's visibility in the image can change only at a T-junction or at a cusp point. At such points, the quantitative invisibility increases or decreases by 1. This change can be determined by casting a ray through the point and ordering the corresponding object surfaces in a "toothpick" manner along the ray. Hence, if the invisibility of an initial vertex is known, the visibility of each segment can be calculated by summing the quantitative invisibility changes.

The quantitative invisibility of the initial vertex is determined by doing an exhaustive search of all relevant object faces in order to count how many faces hide the vertex. An object face is considered relevant if it "faces" the camera, i.e. its outside surface normal points towards the camera. A face hides a vertex if the line of sight to the vertex intersects the face surface and if the intersection point is inside the boundary of the face. To propagate the quantitative invisibility from one edge to another edge starting at its ending vertex, a correction must be applied to the quantitative invisibility of the starting point of the new edge. The complication arises from the fact that faces that intersect at the considered vertex may hide edges emanating from the vertex. This correction factor involves only those faces that intersect at the vertex. For an object with e edges, f faces, and with an average of 3 faces meeting at each vertex, the computational time needed to remove its hidden lines using this algorithm is $O(f + 2 \times 3 \times e)$.

4.2.2 Using Material and Lighting Knowledge

Boundaries of objects show up as intensity discontinuities in an image. A line segment that is potentially visible in a set of views of an object may appear as a whole, disappear entirely, or break up into small segments under various lighting assumptions depending upon the contrast along the edges and the detector

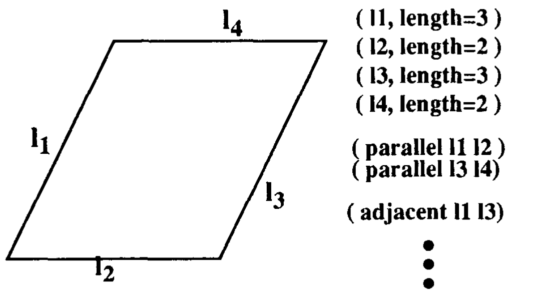


Figure 3: Feature and relationships example

characteristics. Hence, to complete the prediction process PREMIO needs to calculate the intensity values along the predicted wireframe.

The contrast at an edge point is computed as the difference in the intensity of the reflected light from two small neighboring patches at each side of the edge. These intensities, in turn, are obtained by using ray casting and the surface reflection model at a finite number of points along the edges. To represent the contrast along the edge, a contrast graph is fitted with piece-wise continuous polynomials using a regression analysis technique [32].

4.2.3 Interpreting and Grouping Features

A ***feature*** is an entity that describes a part of an object or an image. Simple features such as edges can be interpreted by themselves, or can be grouped to be considered as higher-level features. Matching ***perceptual groupings*** of features was suggested first by Lowe [20]. Henikoff and Shapiro [15] have found useful for object matching arrangements of triplets of line segments called ***interesting patterns***. Other useful high-level features are junctions, and closed loops [21].

In PREMIO a feature is an abstract concept; it can be a point, an edge, a triplet, a hole, a junction, or a higher-level combination of any of these. A feature has a ***type*** that identifies it, a vector of ***attributes*** that represent its global properties, and a real number between 0 and 1 called its ***strength***. The strength is a measurement of the confidence of the feature being of a particular type.

A feature participates in spatial relationships with other features. Each such relationship is represented by a ***relational tuple***, which consists of a ***type*** specifying the relationship and a vector of ***related features*** that participate in that relationship. Associated with every relational tuple of features there is a real number between 0 and 1 called the ***strength*** of the relational tuple. The strength is a measurement of the confidence of the feature vector satisfying the specified relationship.

As a simple example, consider the parallelogram shown in figure 3. It can be described in terms of its four sides and the relationships among them. In this case, the features are the four sides of the parallelogram l_1, l_2, l_3 and l_4. Each side has associated the attribute ***length*** and it is related to the other three features by the relationships ***adjacent*** and ***parallel***.

4.3 Evaluating Predicted Features

After a feature is predicted its potential utility must be evaluated. PREMIO uses the concepts of detectability, reliability and accuracy of a feature. The *detectability* of a feature is defined as the probability of finding the feature using a given detector on an image taken with a given sensor. Therefore, its value depends not only on the feature, but also on the sensor and detector models. The *reliability* of a feature is the probability of correctly matching the detected feature to the corresponding one in the model. Two features that look very similar to each other, should not be considered as very reliable since each of them can be mistakenly identified as the other. In general, the reliability is closely related to the distinguishability power of the feature; i.e. a unique feature immediately matches the model, and therefore is highly reliable. The feature *accuracy* is a measure of the error or uncertainty propagated from the detected feature to a geometric property of the object. This means that if, for example, we detect a straight line in the image, we want to bound the error of its location and orientation. Hence, the accuracy is calculated taking into account the sensor and detector models.

4.4 Output of the Predictor Module

For a given object and a given configuration of light sources, and sensors, the output of the predictor module is a hierarchical relational data structure similar to the one defined in section 3. This structure is called a ***prediction*** of the object. Each prediction contains a set of features, their attribute values such as detectability, reliability, and accuracy, and their originating three-dimensional features. The prediction has at least five levels: the image level, an object level, one or more feature levels, an arc level and a 1D piece level. The image level at the top of the hierarchy is concerned with the imaging conditions that generated the prediction, the general object position, and the background information. The object level is concerned with the different features that will appear on the image and their interrelationships. The feature levels describe the features in terms of simpler features, down to the arc level. The arc level describes the arcs in terms of 1D pieces. Finally, the 1D-piece level specifies the elemental pieces that form the arcs.

5 Using Prediction in Matching

The predictions that PREMIO produces are powerful new tools in recognizing and determining the pose of a 3D object. In order to take advantage of these tools, we have developed an entirely new matching algorithm, a branch-and-bound search that explicitly takes advantage of the probabilities obtained during the prediction stage to guide the search and prune the tree. The matching algorithm represents a large theoretical effort that is actually independent of the PREMIO system, and it is fully described in [9]. The algorithm has been implemented as a C program and tested independently on data specifically generated to fit the abstract paradigm for the probabilistic search.

The matching algorithm can be thought of in two ways, as a relational matching algorithm and as a constrained branch-and-bound search. The theory behind branch-and-bound search is well known [18]. Relational matching has been expressed in several different formalisms. Early papers concentrated on graph or subgraph isomorphisms [30]. This led to many algorithms for discrete relaxation and the introduction of probabilistic relaxation [24]. The exact matching problem was generalized to the consistent labeling problem [14] and to the inexact matching problem [26]. This was extended further to the problem of determining the relational distance between two structural descriptions [28, 25]. Some recent related work includes structural stereopsis using information theory [5] . The present algorithm differs from all of these in its attempt to provide a solid theoretical probabilistic framework for the matching problem and the search.

5.1 Definitions and Notation

Models and images are represented by their features, the relationships among them, and the measurements associated with them. As in the consistent labeling formalism [14], we will call the image features ***units*** and the model features ***labels***. The matching algorithm must determine the correspondences between the units and the labels. Formally, a ***model*** M is a quadruple $M = (L, R, f_L, g_R)$ where L is the set of model features or labels, R is a set of relational tuples of labels, f_L is the attribute-value mapping that associates a value with each attribute of a label of L, and g_R is the strength mapping that associates a strength with each relational tuple of R. Similarly, an ***image*** I is a quadruple $I = (U, S, f_U, g_S)$ where U is the set of image features or units, S is a set of relational tuples of units, f_U is the attribute-value mapping associated with U, and g_S is the strength mapping associated with S.

The relational matching problem is a special case of the pattern complex recognition problem [13]. An image is an observation of a particular model. Let $M = (L, R, f_L, g_R)$ be the model, and $I = (U, S, f_U, g_S)$ be the observed image. Not all the labels in L participate in the observation, only a subset of labels $H \subseteq L$ is actually observed. Furthermore, only the relational tuples of labels representing relationships among labels in H can be observed, and only a subset of them are actually observed. The set U consists of the unrecognized units. Some of the units observed in U come from labels in H; others are unrelated and can be thought of as clutter objects.

The relational matching problem is to find an unknown one-to-one correspondence $h: L \longrightarrow U$ between a subset of L, H, and a subset of U, associating some labels of L with some units of U. The mapping h is called the ***observation mapping***, and it must satisfy that the number of labels associated with units and the number of relations preserved in the observation are maximized. Notice that the matching process consists not only of finding the model M, but also of finding the correspondence h and its domain H, which are the explanation of why the model M is the most likely model. In general we seek to maximize the ***a posteriori*** probability $P\Big(M, h \Big| I\Big)$. That is, we want to maximize the probability of the model being M and the observation mapping being h, given that the image I is observed.

The relational matching problem requires a search procedure that can identify the model M and the mapping h such that $P\Big(M, h \Big| I\Big)$ is maximized. If the ***relational matching cost*** of an observation mapping h, $C(M, h, I)$ is defined by,

$$C\Big(M, h, I\Big) = -\log P\Big(M, h, I\Big) \ , \tag{9}$$

then maximizing $P\Big(M, h, I\Big)$ is equivalent to minimize the relational cost of h.

The relational cost C can be broken down into five terms, each one representing a different aspect of the cost of the mapping [9]:

$$C\Big(M, h, I\Big) = C_M + C_U + C_S + C_{f_U} + C_{g_S} \ , \tag{10}$$

where

$$\begin{aligned} C_M &= -\log P\Big(M\Big) \ , \\ C_U &= -\log P\Big(U, h \Big| M\Big) \ , \\ C_{f_U} &= -\log P\Big(f_U \Big| U, M, h\Big) \ , \\ C_S &= -\log P\Big(S \Big| U, M, h\Big) \ , \\ C_{g_S} &= -\log P\Big(g_S \Big| U, f_U, M, h\Big) \ . \end{aligned}$$

The cost C_M is the ***model cost***. This is the cost associated with the model being considered, and it penalizes the selection of models whose prior probability of occurring, $P(M)$, is low.

The costs C_U and C_{f_U} are the ***label-unit assignment costs***, and they evaluate how well the labels and units match through the mapping h. C_U is the part of the cost that penalizes for the differences of sizes between the set of observed features U and the set of features of the model L. C_{f_U} is the part of the cost that penalizes the "differences" between labels and their correspondent units. The costs C_U and C_{f_U} are given by [9]:

$$\begin{aligned} C_U &= -\log k_f - N_f \log k_f q_f \ , \\ C_{f_U} &= -\log \Big(P\Big(\rho(f_U \circ h, f_{L|_H}) \Big| M\Big)\Big) \ , \end{aligned} \tag{11}$$

where $N_f = \#L + \#U - 2\#H$, $k_f > 0$ and $0 < q_f < 1$ are constants and are determined for each model from the predictions using regression analysis techniques, ρ is a suitable metric function, $f_U \circ h$ is the composition of f_U with h, and $f_{L|_H}$ represents the attribute-value mapping f_L restricted to the labels in the domain H.

The costs C_S and C_{g_S} are the *relational structural costs* and they evaluate how well the relationships among the labels are preserved by the mapping h. C_S is the part of the cost that accounts for the differences between the set of observed relationships S and the set of relationships of the model R. C_{g_S} is the part of the cost that penalizes the "differences" between the relational tuples of labels and their correspondent relational tuples of units. The costs C_S and C_{g_S} are given by [9]:

$$\begin{aligned} C_S &= -\log k_r - N_r \log q_r \ , \\ C_{g_S} &= -\log\left(P\left(\rho(h \circ g_S, g_R)\middle| M\right)\right) \ , \end{aligned} \quad (12)$$

where $N_r = \#(R - S \circ h^{-1}) + \#(S - R \circ h)$, $k_r > 0$ and $0 < q_r < 1$ are constants and are determined for each model from the predictions using regression analysis, $S \circ h^{-1}$ is the composition of S with the inverse mapping of h, h^{-1}, $R \circ h$ is the composition of R with h, ρ is a suitable metric function, and $g_S \circ h$ is the composition of g_S with h.

5.2 Partial Matching

Finding the full mapping h would require a full tree search. But, only a few correspondences between units and labels are needed to hypothesize a match between an object and a model and to estimate the object's pose. The number of correspondences needed is determined by the number of degrees of freedom that the matched features fix. Instead of finding the entire mapping h, we would like to find a partial match m that is a *restriction* of h, in the following sense:

Def. 5.1 Given two one-to-one mappings h and m, such that Dom(m) $\subseteq$ Dom(h), and $m(l) = h(l)$ for all $l \in$ Dom(m), we say that the function h is an *extension* of the function m, and that the function m is a *restriction* of the function h. The *order* of the extension h with respect to m is the difference between the cardinalities of the sets Dom(h) and Dom(m).

Let $m: L \rightarrow U$ be a partial mapping assigning some labels to some units. The mapping m partitions the sets of features L and U into the set of *used* features in the match and the set of *residual* features – i.e., those not used in the match. Figure 4 gives a diagram of the sets L and U showing the partitions induced by a partial match m.

Let L^u be the set of used labels, L^r the set of residual labels, U^u the set of used units, and U^r the set of residual units induced by the partial mapping m. Consider the set $E_j = \{ext_j(m)\}$, of all the possible extensions of m of order j that assign some labels to some units. The maximum possible order of an extension of m is given by: $J = \min\{\#L^r, \#U^r\}$. The set

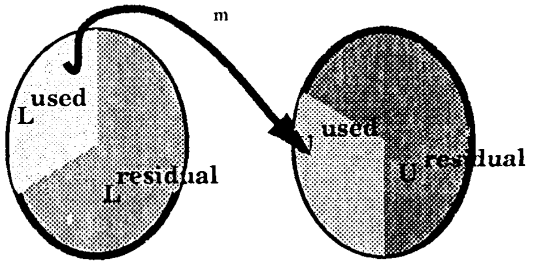


Figure 4: Partition of the sets of features induced by a partial match.

$E = \{ext(m)\}$ of all possible extensions of m can be expressed as the union of all the extensions of different orders: $E = \bigcup_{0 \le j \le J} E_j$, and its cardinal is given by:

$$\#E = \sum_{j=0}^{j=J} \binom{\#L^r}{j} \cdot \binom{\#U^r}{j} \cdot j! \ .$$

The probability that the "true" observation mapping h is an extension of a partial mapping m – that is the probability that the observation mapping h that maximizes the probability $P\left(M, h, I\right)$ belongs to the set $E = \{ext(m)\}$ of all possible extensions of the partial mapping m is given by [9]:

$$P\left(M, (m, L^u), h \in E, I\right) = \frac{P_M \cdot P_S \cdot P_{f_U} \cdot P_{g_S}}{{P_U}^2} \quad (13)$$

where,

$$\begin{aligned} P_M &= P\left(M\right) \\ P_U &= k_f \sum_{j=0}^{j=J} \#E_j {q_f}^{N_{f_j}} \\ P_S &= k_f k_r \sum_{j=0}^{j=J} {q_f}^{N_{f_j}} \sum_{h_i \in E_j} {q_r}^{N_{r_i}} \\ P_{f_U} &= k_f \sum_{j=0}^{j=J} {q_f}^{N_{f_j}} \sum_{h_i \in E_j} P\left(\rho(f_U \circ h_i, f_{L|_{H_i}})\middle| M\right) \\ P_{g_S} &= k_f \sum_{j=0}^{j=J} {q_f}^{N_{f_j}} \sum_{h_i \in E_j} P\left(\rho(h_i \circ g_S, g_R)\middle| M\right) \\ N_{f_j} &= \#L^r + \#U^r - 2j \\ N_{r_i} &= \#(R - S \circ h_i^{-1}) + \#(S - R \circ h_i) \ . \end{aligned}$$

Although the terms P_S, P_{f_U}, and P_{g_S} cannot be calculated unless all the possible extensions $h_i \in E$

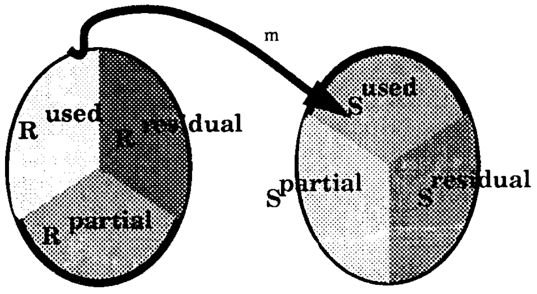


Figure 5: Partition of the sets of relational tuples induced by a partial match.

are considered, they can be *upper bounded* by values depending only on m and not on its extensions [9]. These upper bounds can be found by noticing that the partial mapping m induces a partition of the sets of relational tuples S and R into three types of sets: the set of *used* relational tuples, the set of *partially used* relational tuples, and the sets of *residual* relational tuples, depending on whether all, some, or none of the features in the feature vector of the tuple have been associated a correspondent through the mapping m. Figure 5 gives a diagram of the sets R and S showing the partitions induced by a partial match m. Then, it can be shown [9] that

$$P\Big(M, m, h \in E, I\Big) \leq P_{max} \tag{14}$$

where

$$P_{max} = P_M \cdot k_f k_r \sum_{j=0}^{j=J} \#E_j q_f^{N_{fj}} q_r^{N_{rmaxj}} . \tag{15}$$

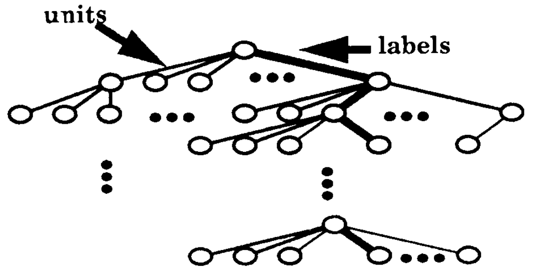
where $N_{rmaxj} = \#R + \#S - R_{maxj} - S_{maxj}$, and R_{maxj} and S_{maxj} are the total number of relational tuples of labels and units with at most j labels or units without a correspondent in the mapping m.

5.2.1 Matching by Tree Search

The matching process can be thought of as a state space search through the space of all possible interpretations Σ. The state space Σ is called the *matching space* and it is defined as follows:

Def. 5.2 The *matching space*, Σ, is the state space of all possible interpretations, in which each state σ is defined by an observation mapping h_σ with degree of match $k_\sigma = \#\text{Dom}(h_\sigma)$.

The search through the state space Σ can be achieved by doing an ordered search on a tree $\mathcal{T}$ such as the one shown in figure 6. Each node in $\mathcal{T}$ represents a unit and each of its branches represents an



Search Space: All possible interpretations
Search State: A path in the tree

Figure 6: Search tree $\mathcal{T}$.

assignment of the unit to a label. A search state σ in Σ is represented by a path $\mathcal{P}$ in the tree $\mathcal{T}$. In the rest of th paper, the terms "path" and "partial mapping" will be used interchangeably.

A path $\mathcal{P}$ defines an observation mapping $m_\mathcal{P}$, and it has an associated cost $C_\mathcal{P} = C(m_\mathcal{P}, M, I)$ defined in equation (9). The matching process consists of finding the path $\mathcal{P}^*$ such that its associated observation mapping $m_{\mathcal{P}^*}$ has the least cost.

A match can be found by using the well known *branch-and-bound* tree search technique. In the standard branch and bound approach during search there are many incomplete paths contending for further consideration. The one with the least cost is extended one level, creating as many new incomplete paths as there are branches. This procedure is repeated until the tree is exhausted.

5.2.2 Improved Branch–and–Bound Search

Branch and bound search can be improved greatly if the path to be extended is selected such that a *lower bound* estimate of its the total cost is minimal. Those branches that have an estimated total cost greater than the maximum cost allowed can be pruned.

Let m be a partial mapping and m_1 be an extension of m. The relational matching cost of m_1 is given by $C_{m_1} = -\log P\Big(M, m_1, I\Big)$. An underestimate of C_{m_1} is found by finding an upper bound of $P\Big(M, m_1, I\Big)$. Let h^* be the true observation mapping. Since $h^* \in E$ is one of a set of disjoint events, the probability $P\Big(M, m, h^* \in E, I\Big)$ can be expressed as the sum of the probabilities of these events:

$$P\Big(M, m, h^* \in E, I\Big) = \sum_{h_i \in E} P\Big(M, h_i, I\Big) .$$

Hence, for an extension m_1 we have

$$P\Big(M, m_1, I\Big) \leq P\Big(M, m, h^* \in E, I\Big) \leq P_{max} . \tag{16}$$

```
Step 1: Initialization.
Form a queue $Q_{\mathcal{P}}$ of partial matches, and let $\mathcal{P}_0$ be the
initial partial match.
Step 2: Iterate over current paths.
Until $Q_{\mathcal{P}}$ is empty, do
Begin
    $\mathcal{P}$ := FRONT($Q_{\mathcal{P}}$)
    $m$ := partial mapping associated with $\mathcal{P}$
    $C_m$ := relational cost of $m$
    Step 2.1: Test if $\mathcal{P}$ can be extended.
    If the path $\mathcal{P}$ can be extended,
        Begin
            Step 2.1.1: Select next label.
            Look for two tuples, one from $R$ and one
            from $S$ whose components are not all
            matched, that are compatible. Two relational
            tuples are compatible if they have the same
            number of features and they agree on the
            features that have been already matched.
            The relational tuples that are partially
            matched should be checked before than those
            that are not.
            Step 2.1.2 Extend the path
            For each $u \in U^r$, do
            Begin
                $h_1$ := path $m$ extended with the pair
                    $(l, u)$.
                $\mathcal{P}'$ := path associated with the
                    mapping $h_1$.
                $C_{h_1}$ := relational cost of $h_1$.
                $\Gamma_{h_1}$ := underestimate of the
                    cost of the extensions of $h_1$.
                Step 2.1.2.1 Compare with $\epsilon$.
                If $\Gamma_{h_1} \leq \epsilon$
                Begin
                    Step 2.1.2.1.1 Finished?
                    If FP($\mathcal{P}'$) = 6 and $C_{h_1} \leq \epsilon$
                    Begin
                        $\mathcal{P}'$ is a satisfactory match.
                        Exit .
                    End if.
                    Step 2.1.2.1.2 Add $\mathcal{P}'$.
                    BACK($Q_{\mathcal{P}}$) := $\mathcal{P}'$.
                End if.
            End for.
            Step 2.1.3 Resort the queue.
            Sort $Q_{\mathcal{P}}$ by underestimated cost.
        End if.
    End until.
Step 3: End of Algorithm
Announce failure.
```

Figure 7: Matching Algorithm

The matching algorithm is given in figure 7. The algorithm is being independently tested using controlled experiments designed under a rigorous experimental protocol [9]. So far, it has been tested on more than 4000 runs for models with five and seven labels, and with ordered binary and ternary relational tuples. Figure 8 is a plot of the ratio of the number of paths pruned to the total number of paths opened during the search. The graph shows that the use of the underestimate bound of the cost results in a high pruning ratio (from 30% to nearly 80% of the tree), and hence greatly reduces the computational time.

6 Summary and Future Work

Our research consists of two parallel activities: theoretical developments, and test of the resulting theory by

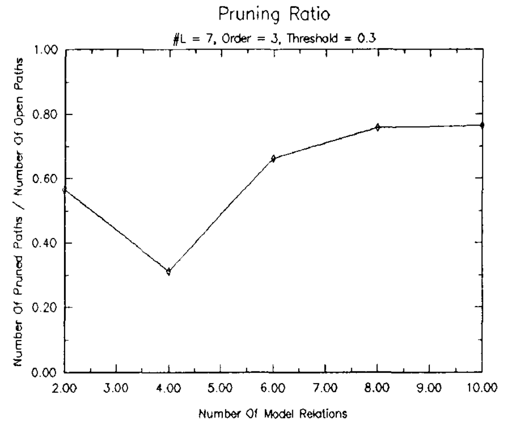


Figure 8: Pruning Ratio

the implementation of the **PREMIO** system, which is implemented as a set of routines with over 39000 lines of C language on a SUN system running Unix. **PREMIO** has two major subsystems: an offline subsystem and a online subsystem. The offline subsystem, in turn, has three modules: a vision model generator, a feature predictor, and an automatic procedure generator. We have proposed the use of a complete model of the world, the Vision Model, that incorporates PADL2 CAD models, surface reflectance properties, light sources, sensors, and processing models to symbolically predict the features that will appear on the images of the objects being modeled. The predictions are organized in a Prediction Model that produces the knowledge base of the probabilistic matching algorithm.

For the vision model, we have implemented a hierarchical, topological representation of the objects in the world, using as input their PADL2 CAD models. The prediction module can now predict line segments feature for objects with planar surfaces. The probabilistic matching algorithm is being tested independently using artificial data generated under a rigorous experimental protocol. The results obtained so far are promising in that a large percentage of the tree being searched is pruned by the matching procedure proposed. The remaining work is to integrate the parts of the system and test it on real image data. On the basis of our results so far, we expect PREMIO, when fully integrated, to solve many of the difficulties that most CAD-based vision systems encounter.

References

[1] J. Amanatides. Realism in computer graphics: A survey. *IEEE Computer Graphics and Applications*, 7:44–56, January 1987.

[2] A. Appel. The notion of quantitative invisibility. In *Proc. ACM National Conference*, pages 387–393, 1967.

[3] B. Bhanu, T.Henderson, and S.Thomas. 3-D model building using CAGD techniques. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 234–239, June 1985.

[4] R.C. Bolles and R.A. Cain. Recognizing and locating partially visible objects: The local-feature focus method. *Int. J. Robot. Res.*, 1(3):57–82, Fall 1982.

[5] K. L. Boyer and A. C. Kak. Structural stereopsis for 3–d vision. *IEEE Transactions on Systems, Man and Cybernetics*, PAMI–10(2):144–166, March 1988.

[6] R.A. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17(1-3):285–348, 1981.

[7] C.G. Buchanan. Determining surface orientation from specular highlights. Master's thesis, Dep. Comp. Sc., Univ. of Toronto, Toronto, Ontario, Canada, 1986.

[8] Octavia I. Camps, Linda G. Shapiro, and Robert M. Haralick. PREMIO: The Use of Prediction in a CAD-Model-Based Vision System. Technical Report EE-ISL-89-01, Department of Electrical Engineering, University of Washington, 1989.

[9] Octavia I. Camps, Linda G. Shapiro, and Robert M. Haralick. A probabilistic matching algorithm for object recognition. Technical Report EE-ISL-90-08, Department of Electrical Engineering, University of Washington, 1990.

[10] R.L. Cook and K.E. Torrance. A reflectance model for computer graphics. *ACM Trans. on Graphics*, 1(1):7–24, January 1982.

[11] R. Galimberti and U. Montanari. An algorithm for hidden-line elimination. *Comm. ACM*, 12(4):206–211, April 1969.

[12] C. Goad. Special purpose automatic programming for 3D model-based vision. In *Proc. of the Image Understanding Workshop*, pages 94–104, June 1983.

[13] R.M. Haralick. The pattern complex. In Roger Mohr, Theo Pavlidis, and Alberto Sanfeliu, editors, *Structural Pattern Analysis*, pages 57–66. World Scientific Public. Co, 1989.

[14] R.M. Haralick and L. G. Shapiro. The consistent labeling problem: part i. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI–1(2):173–184, April 1979.

[15] J. Henikoff and L. Shapiro. Interesting patterns for model-based matching. In *ICCV*, 1990.

[16] P. Horaud and R.C. Bolles. 3DPO: A system for matching 3-D objects in range data. In A.P. Pentland, editor, *From Pixels to Predicates*, pages 359–370. Ablex Publishing Corporation, Norwood, New Jersey, 1986.

[17] K. Ikeuchi. Generating an interpretation tree from a CAD model for 3D-Object recognition in bin-picking tasks. *Int. J. Comp. Vision*, 1(2):145–165, 1987.

[18] Richard E. Korf. Search: A survey of recent results. In Howard E. Shrobe and The American Association for Artificial Intelligence, editors, *Exploring Artificial Intelligence*, chapter 6, pages 197–237. Morgan Kaufmann Publishers, Inc., 1988.

[19] P. P. Loutrel. A solution to the hidden-line problem for computer-drawn polyhedra. *IEEE Trans. on Computers*, 19(3):205–210, March 1970.

[20] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[21] H. Lu and L. G. Shapiro. Model-based vision using relational summaries. In *SPIE Conference on Applications of Artificial Intelligence VII*, March 1989.

[22] J. Ponce and D. Chelberg. Finding the limbs and cusps of generalized cylinders. *Int. J. Comp. Vision*, April 1987.

[23] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag New York Inc., 1985.

[24] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *IEEE Trans. Syst. Man Cybern.*, SMC-06, June 1976.

[25] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, MAn and Cybernetics*, SMC–13(13):353–362, May 1983.

[26] L.G. Shapiro and R.M. Haralick. Structural descriptions and inexact matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI–3(5):504–519, September 1981.

[27] L.G. Shapiro and R.M. Haralick. A hierarchical relational model for automated inspection tasks. In *Proc. 1st IEEE Int. Conf. on Robotics*, Atlanta, March 1984.

[28] L.G. Shapiro and R.M. Haralick. A metric for comparing relational descriptions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI–7, 1985.

[29] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker. A characterization of ten hidden-surface algorithms. *Computing Surveys*, 6(1), March 1974.

[30] J. R. Ullman. An algorithm for subgraph homomorphisms. *J. Assoc. Comput. Mach.*, 23:31–42, January 1976.

[31] J. T. Jr. Welch. A mechanical analysis of the cyclic structure of undirected linear graphs. *Journal of the Association for Computing Machinery*, 3(2):205–210, April 1966.

[32] Seungku Yi. *Illumination Control Expert for Machine Vision: A Goal Driven Approach.* PhD thesis, Department of Electrical Engineering, University of Washington, Seattle, Washington, 1990.

Automatic Camera and Light-Source Placement Using CAD Models

Cregg K. Cowan
Advanced Automation Technology Center
Information, Telecommunications, and Automation Division
SRI International, Menlo Park, California 94025

Abstract

This paper describes model-based methods to determine the region where a camera and light source should be placed. We present new techniques that determine the region of light placements that guarantee specified object edges will be detected. These methods use geometric models of the objects and the characteristics of the camera, lens, digitizer, and edge detector to determine the region of acceptable camera and light-source locations. Other recent work in automatic camera and light-source placement is also described.

1 Introduction

One of the main limitations in applying machine vision to industrial automation is the high cost of designing, installing, and debugging each system. Developing a successful vision system for a given task is as much art as science, because it often depends on poorly understood "rules of thumb." Furthermore, in current practice it is all too common that a system must be redesigned due to initial failure on installation.

To address these concerns, a few researchers have recently begun investigating the relationship between a vision task and acceptable sensor configurations for executing it. The ultimate goal of this line of research is the ability use models of the objects and a description of the vision task to generate a complete vision system, including placement of the required sensors and light sources and selection of appropriate image analysis procedures (e.g., object recognition programs). Such a capability will form the core of future systems for vision system design. For example, a vision system engineer could interactively evaluate many alternative designs, each one of which is guaranteed to satisfy the task requirements. Such operation is in contrast to the current trial-and-error practice where task requirements are sometimes overlooked, requiring costly system redesign. A second potential application of this research is to control sensors on an autonomous vehicle—e.g., to determine the appropriate sensor and light-source locations for performing a visual inspection in the presence of environmental obstacles that were not known a priori (and therefore could not have been accounted for in any preplanned strategy).

Given computer-aided design (CAD) models of a set of objects, a camera model and viewpoint, and a light-source model and location, it is possible to calculate the expected appearance of the objects. This is precisely the approach taken in graphical simulation. In contrast, we are developing techniques that use the desired characteristics of the resulting image and a *partially* specified sensor system to solve for the remaining sensor parameters. In particular, this paper describes methods for using models of polyhedral objects in conjunction with models of the camera, light-source, and surface reflectance to determine the three-dimensional (3-D) regions where a camera and light source can be placed to satisfy the requirements of a vision task. The camera and light-source regions are calculated by finding their bounding surfaces, thereby describing their 3-D shape. The remainder of this section describes important considerations for CAD-based sensor placement and summarizes prior work in the field. Section 2 describes methods for placing a given camera for a vision task. Section 3 describes a method that integrates camera and light-source placement and presents a method for selecting the camera lens aperture setting.

Limitations and Special Considerations

An important consideration in this research area is the accuracy of model-based vision system configurations. In particular, no model-based technique can be expected to evaluate configurations to arbitrarily fine accuracy because geometric models are not perfectly accurate. Imperfect models are even more of a problem for predicting radiant flux from light sources, surface reflectance, and camera sensitivity. Therefore, precise and time-consuming calculations for sensor placement are less important than fast, conservative approximations.[1] For example, it would not be appropriate to use computationally intensive graphical rendering techniques, such as ray tracing, for exhaustive examination of all possible light-source and camera locations.

When selecting an "optimum" camera or light-source location, many criteria may need to be considered, such as convenient mounting, traffic patterns, and access for maintenance. It would be extremely difficult to construct a system that accounted for all such criteria. Consequently, we prefer techniques that calculate entire regions where the camera and light source may be placed, from which one or two are automatically or interactively selected.

Related Work

A system for determining unoccluded camera locations for a visual task was developed by Sakane et al. [1, 2]. Their technique is to choose the radius of a sphere centered on the object that is to be observed and to tesselate the sphere such that the centroids of the regions on the spherical surface represent all possible camera locations. For each location, graphical projection is used to determine if the object is occluded. Groups of adjacent tessels are combined into regions. The center of the largest region is chosen as the desired viewpoint.

A method to synthesize the entire 3-D region of acceptable camera locations for a visual task were described by Cowan et al. [3, 4, 5]. The technique first generates a 3-D region satisfying each of the visual requirements: spatial resolution, field of view, focus, and visibility. A special-purpose representation is then used for fast intersection of these regions to determine the space of possible viewpoints for the visual task. Improved methods for calculating the visibility region using hierarchical object models were developed by Tarabanis and Tsai [6]. These authors also developed methods to directly calculate the optical settings for satisfying the spatial resolution for a vision task [7].

Methods to automatically determine the location for a light source based on the camera dynamic range were developed by Cowan and Bergman [8]. These techniques are described in Section 3 of this paper.

Sakane et al. developed a method for selecting each location for a mobile light source in a photometric stereo system [9]. The technique constrained both the camera and light source to enumerated locations on a sphere. Pairs of light-source and camera locations were evaluated to determine the best combination.

The ICE system was developed by Yi et al. [10] to determine an optimal sensor and light-source configuration for a vision task. Their approach is to constrain the sensor and light source to a sphere and formulate the vision task as an optimization problem. Mathematical programming techniques are used to maximize the length of an edge that appears in the resulting image.

[1] Conservative approximations guarantee that any calculated solution is valid. Some valid solutions may exist outside the region of calculated solutions.

2 Model-Based Camera Placement

In this section we summarize our method to determine the region of acceptable camera locations using CAD models of the objects. For a given set of object surfaces, called the *target* surfaces, the 3-D viewpoint regions that satisfy the resolution, field of view, focus, and visibility (nonocclusion) requirements are determined. The 3-D region for each requirement is found by calculating the locus of viewpoints where the achieved performance equals the requirement. This locus is a surface forming the boundary of the region of acceptable viewpoints. The intersection of these regions is the space where the camera can be placed to simultaneously satisfy all requirements.

Spatial Resolution

Given a minimum required spatial resolution on each target surface, the camera must be placed within a region defined by the intersection of spheres. For each sphere the diameter depends on the required resolution for the surface, and its center lies along the normal vector located at each vertex of the surface's convex hull [5]. For example, the viewpoint region that satisfies the resolution constraint for a square surface is simply the intersection of the resolution spheres for the square's four vertices. The region that simultaneously satisfies the resolution requirement for multiple surfaces is the intersection of the resolution regions for each surface.

Field of View

Since cameras have a limited field of view, there is a surface of minimum distance, called the *field-of-view constraint* surface, outside of which the camera's field of view is sufficient to contain all of the target surfaces. For the simple case of an orthogonal view of a fixed planar surface, we determine the minimum camera distance by observing that the image pixel array and the minimum rectangle that bounds the surface form similar triangles with respect to the lens center. If we ignore perspective effects, which is appropriate to do when using a telephoto lens, the minimum camera distance is also valid for viewing directions that are not orthogonal to the surface.

For target surfaces whose orientation is not fixed, we calculate their minimum circumscribing sphere and then find the field-of-view constraint such that the entire sphere lies in the field of view. This guarantees that the target surfaces will lie in the camera field of view irrespective of their orientation.

Focus

For any camera and lens setting there is only one distance, called the *focus distance*, at which a point is perfectly focused on the image plane. In addition, we consider points within a range of distances to be in acceptable focus if the diameter of their blur circle is less than the minimum pixel dimension [5]. This range of distances is known as the depth of field (DOF). The locus of viewpoints for which the extent of a target surface $\mathcal{S}$ equals the DOF forms a constraint surface in three dimensions—any viewpoint that lies farther than the constraint surface also has sufficient DOF to contain all of $\mathcal{S}$. We use a simple iterative procedure to calculate the constraint surface such that all of the target surfaces are in focus.

Visibility

The visibility constraint is the requirement that all target surfaces must be completely visible from the selected sensor location. The 3-D region of viewpoints where an object, $\mathcal{O}$, occludes a portion of a target surface, $\mathcal{S}$, is bounded by a set of planes generated by combining the edges of $\mathcal{O}$ with the vertices of $\mathcal{S}$. For multiple occluding objects, we find the occluded viewpoint region for each of these objects and then take the union of the resulting occluded regions to obtain the entire set of occluded viewpoints with respect to target surface $\mathcal{S}$. The set of viewpoints that satisfies the visibility constraint for observing $\mathcal{S}$ is the complement of the set of the occluded viewpoints.

Camera Placement Example

As an example task, consider the dimensional inspection of the 3 slots (10mm × 30mm) in the base of the bracket shown in Figure 1(a) to a minimum spatial resolution of 1.5 pixels per millimeter. In this case, the target "surfaces" are in fact rectangular holes in the sheet metal. We compute the viewpoint regions that satisfy the resolution, focus, field-of-view, and visibility constraints for the three slots. At run time a binary vision module provides the bracket location, and the system then calculates the viewpoint region that satisfies all of the constraints. The image obtained from one such viewpoint is shown in Figure 1(k). The polyhedral appearance of the regions is due to display quantization.

3 Model-Based Light-Source Placement

In this section we summarize our method for integrated camera and light-source placement, then present new light-placement techniques that guarantee that specified object edges will be detected by the image processing system.

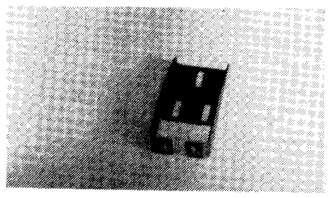
3.1 Lens Aperture Selection

The methods described in the previous section used a given lens aperture (for calculating the focus requirement). In practice, however, we must relate the lens aperture to the required illumination. We have therefore developed a four-step method for selecting an acceptable lens aperture.

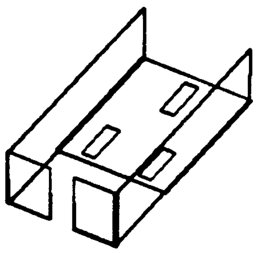
1. Calculate the viewpoint region that satisfies the resolution, field-of-view, and visibility requirements, as described in the previous section.

2. Find the maximum aperture diameter, a_{max}, for which the focus constraint intersects region R, as shown in Figure 2. This may be found by examining the quantized polar representation of region R, determining the largest aperture for each ray through R, and keeping the global maximum. Since a_{max} is the largest usable aperture, it is related to the minimum scene brightness, or radiance.

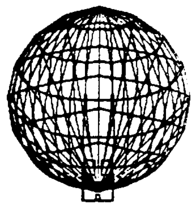
3. Determine the diffraction-limited aperture, a_d, that occurs when the angular separation of the two first minima of the diffraction pattern corresponds to the pixel size. That is, for pixel size, p,



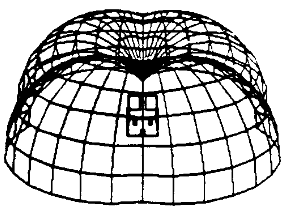
(a) Bracket photograph.



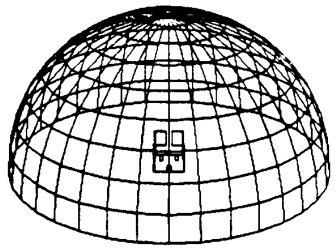
(b) Bracket model.



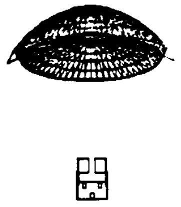
(c) Resolution constraint region.



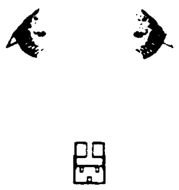
(d) Focus constraint region.



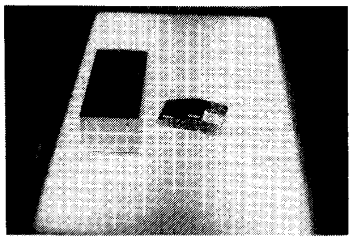
(e) Field-of-View
constraint region.



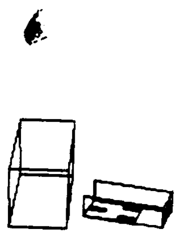
(f) Viewpoints satisfying
constraints (c), (d), and (e).



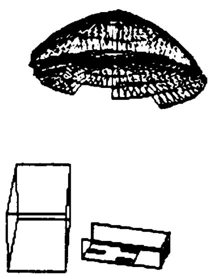
(g) Viewpoints from (f)
occluded by sides of bracket.



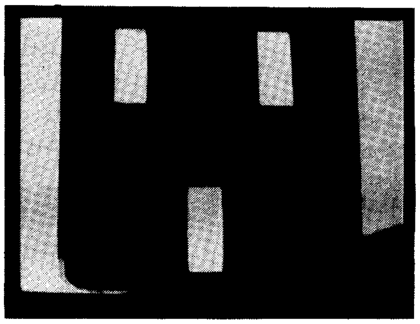
(h) Bracket placed beside
obstacle in world.



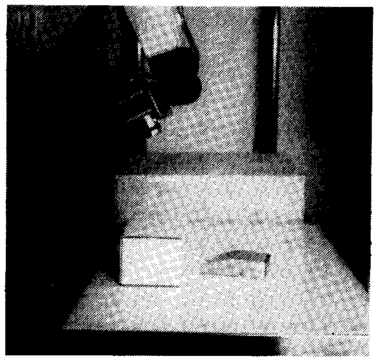
(i) World model and viewpoints
occluded by obstacle.



(j) World viewpoints that
satisfy all constraints.



(k) Sample image obtained
(looking over obstacle).



(l) Bracket, obstacle, and
camera for sample image.

Figure 1: Viewpoint Constraints for Inspecting the Dimensions of Three Slots of a Bracket.

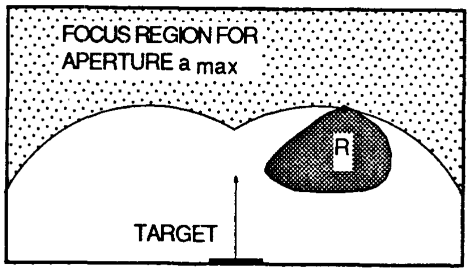


Figure 2: Choosing the Maximum Lens Aperture

and image plane distance, v, the following relation holds:

$$a_d = 2.44\lambda \frac{v}{p} \qquad . \qquad (1)$$

This criterion is illustrated in Figure 3. Note its magnitude is twice that of the Rayleigh criterion, which is also arbitrary.

Because the actual camera location is not yet known, the focus distance and hence the image plane distance are also not known. A conservative estimate for a_d is found by assuming that the camera is placed at the viewpoint in R closest to the target (thus maximizing v).

Also of interest is aperture a_R, the one for which the focus region contains all of R. If a_R is larger than a_d, then it is a good choice for the minimum, a_{min}, since smaller apertures do not result in additional freedom for locating the camera. Otherwise, set a_{min} to a_d.

4. Choose aperture a in the interval $[a_{min}, a_{max}]$. Since any value in the range will suffice, the actual aperture is usually chosen on the basis of other criteria. For example, larger apertures have less stringent light requirements, but smaller apertures have greater depth of field and typically reduce the effects of some lens aberrations, e.g., spherical aberrations. In our example cases, we choose the largest possible lens aperture because of the low power of the light source used.

After choosing an aperture, the final step in camera placement is to calculate the region satisfying the focus requirement for the chosen aperture and to intersect it with the other requirements to find the region of acceptable camera viewpoints.

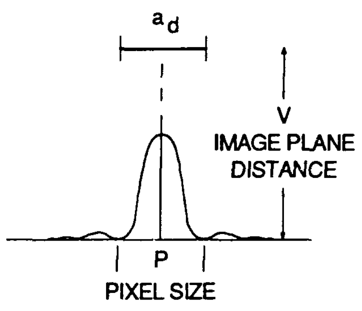


Figure 3: Image Diffraction of a Distant Point

3.2 Light Placement for Camera Dynamic Range

Our current approach is to use a single point light source and to model reflectance from each surface as having a diffuse component that follows Lambert's cosine law and a specular component forming a lobe within an angular tolerance, ψ, of the perfect specular angle. That is, for each surface a fraction, ρ_A, of the incident light is absorbed (heating the surface), a fraction, ρ_L, is reflected diffusely, and a fraction, ρ_S, is reflected specularly, where $\rho_A + \rho_L + \rho_S = 1$.

The 3-D region where a point light source can be placed to achieve adequate illumination depends on the dynamic range of the camera (the allowable interval of irradiance at the camera, $[E_{C,min}, E_{C,max}]$, in watts per square meter). The scene radiance (brightness), L_S, at every point is related to the image irradiance (for the projection of that point) as follows:

$$L_S = \frac{4\,E_C}{k\pi\,cos^4\alpha}\left(\frac{v}{a}\right)^2 \qquad , \qquad (2)$$

where k is the fraction of light transmitted by the lens, α is the angle from the camera's principal axis, and a and v are the lens aperture and image plane distance, respectively.

Diffuse Reflectance

For a Lambertian surface, the scene radiance at every point is proportional to the irradiance at that point [11]. For an infinitesimal patch of such a surface, it has been shown [8] that the distance of the light source, r, is related to the camera irradiance:

$$r = \sqrt{\frac{\Phi_{total}\,k\rho_L\,cos^4\alpha}{16\,\pi\,E_C}}\,\left(\frac{a}{v}\right)\sqrt{cos\,\theta} \qquad , \qquad (3)$$

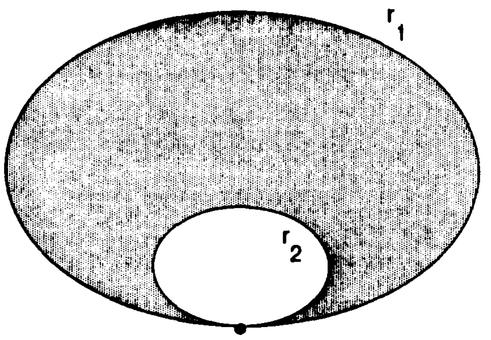


Figure 4: Region of Acceptable Light Locations for Illuminating a Single Point.

where Φ_{total} is the total flux of the point source (in watts) and θ is the incident angle.

Note that minimum and maximum light source distances at a given incidence angle are ***independent*** of the distance between the camera and the target surface. This is because the target surface area that each camera pixel "sees" increases with the square of the distance, cancelling the decrease in light intensity.

One may also observe that only 1 term in Equation 3 depends on the camera-to-target geometry—the off-axis angle, α. When using a long telephoto lens the off-axis angle is negligible. For such a case, the light-source distance must lie in the interval $[r_1, r_2]$:

$$r_1 = K_1\sqrt{\cos\theta} \quad , \quad (4)$$

$$r_2 = K_2\sqrt{\cos\theta} \quad , \quad (5)$$

where K_1 and K_2 are the same for all camera positions. That is, the camera may be positioned independently of the light source when a telephoto lens is used.

In summary, the 3-D region where a point light source may be placed so that the diffuse reflection from an infinitesimal surface patch is within the camera's dynamic range lies between 2 closed curves that are proportional to the square root of the incident angle, as shown in Figure 4.

For properly illuminating an entire surface, the light-source region, $R_{diffuse}$, is the intersection of the regions for all the infinitesimal patches on the surface. This region can be calculated by intersecting the interior regions of the r_1 curves for the vertices of the surface and subtracting the convex hull of the r_2 curves for the vertices — as long as the light source is placed within the intersection of the r_1 curves, all surface points will receive sufficient illumination; as long as the light source is outside of all the r_2 curves, no surface point will appear too bright.

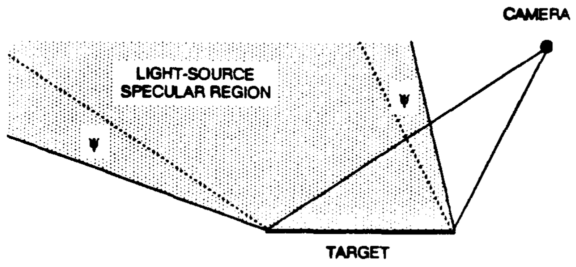


Figure 5: Light Location Region where Specular Reflection Is Seen by Camera.

Specular Reflectance

Although specular reflections can sometimes be used to advantage in machine vision tasks, we currently wish to avoid specular reflection into the camera. Therefore, we wish to calculate the light-source region, $R_{specular}$, where specular reflection is seen by the camera so that we may subtract it from $R_{diffuse}$. Consider the illumination of a perfect mirror surface with a point light source. The entire surface would appear dark from the camera viewpoint, except that for some light positions a single bright point will appear in the image. As previously described, we model the specular component of reflection as a cone whose axis is the perfect specular angle and has apex half-angle ψ. Figure 5 shows a cross-section of a specular surface, with region $R_{specular}$ shown as the shaded region. The dotted lines illustrate the region that would result if ψ were zero. Note that for any polygonal surface, region $R_{specular}$ can be found by connecting the specular light source directions of the polygon's vertices, including the angular tolerance ψ.

3.3 Experiment

In order to test these methods, we have developed a system having a camera on one robot arm and a small incandescent light source (approximating a point light source) on a second. The example object used to demonstrate these methods is shown in Figure 6. This object was chosen because it contains several gray levels and because the surface reflectance has a significant specular component. We model the top of the object as a set of 4 polygonal surfaces. Although the surfaces have different spectral properties (i.e., they are different colors), we currently treat them as grayscale only. Similarly, we do not currently take account of the spectral properties of the light source or of the camera. The surface reflectance properties have been measured as follows:

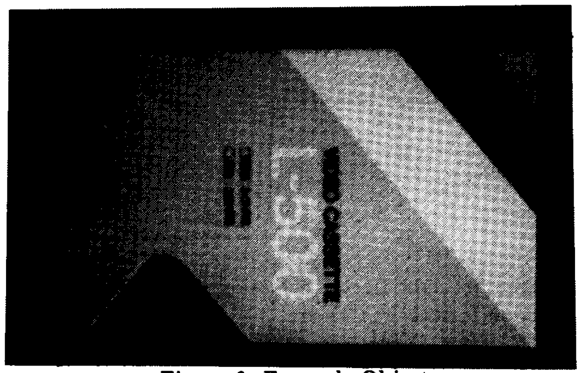


Figure 6: Example Object

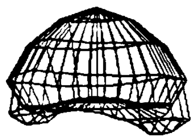


Figure 7: Camera Viewpoint Region Satisfying the Resolution, Field-of-View, and Focus Requirements.

	ρ_L	ψ
Light Gray	.704	15°
Medium Gray	.563	15°
Dark Gray	.469	15°

The camera viewpoint region shown in Figure 7 simultaneously satisfies the resolution, field-of-view, and focus requirements for the object. For this example, we simply choose the viewpoint at the centroid of the region and use it to determine the possible locations for placing the light.

As described in the previous section, we determine the light location region by finding the lighting regions for the vertices of the target. The light location regions for three target surfaces are superimposed in Figure 8. The size of the bounding shapes follows the diffuse surface reflectance: the largest shape is the "r1" curve for one vertex of the light-gray surface; the medium-sized shape corresponds to all vertices of the medium-gray background; the smallest "r1" shape corresponds to the vertices of the dark-gray surface. The "r2" shapes for the three target surfaces are superimposed in the lower portion of the figure.

The region $R_{diffuse}$ is calculated by intersecting all the minimum-brightness regions and then subtracting the union of the maximum-brightness regions. For this example, it was not necessary to form the convex hull of the maximum-brightness regions (as previously described) because the light region shapes are much larger than the target surfaces. We calculate region $R_{specular}$ and subtract it from region $R_{diffuse}$, leaving the region of acceptable light positions, as shown in Figure 9.

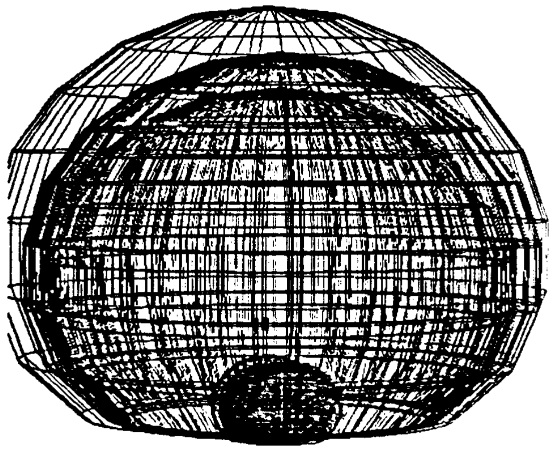


Figure 8: Superimposed Light-Location Regions for Three Target Surfaces: Light Gray (largest light-location region), Medium Gray (mid-size region), and Dark Gray (smallest region).

To test these results we positioned the camera as described above and placed the light source on the surface of maximum brightness. The image obtained at one such light position is shown in Figure 10. As expected, one vertex (the lower-right vertex of the light-gray surface in this image) appears to be saturated. As a further test we then calculated the distance such that the medium-gray vertex (adjacent to the light-gray one used above) would saturate and moved the light source there (see Figure 11). As expected, there is a loss of contrast along the edge between the surfaces.

3.4 Light-Source Placement for Edge Detection

The light-source placement techniques described above, while a necessary first step, do not provide all the functionality required for vision system design. One key aspect of this type of design is enabling the detection of particular object features, such as object edges. The next two subsections describe our model of image digitization and the relevant characteristics of edge detection. The last subsection describes how

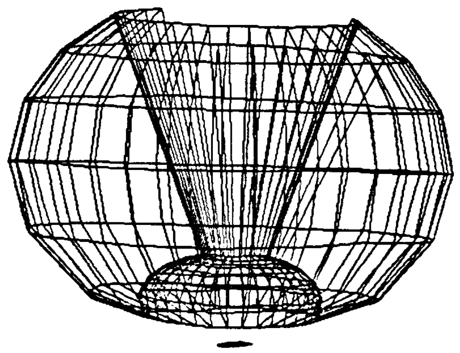


Figure 9: Region of Light Locations

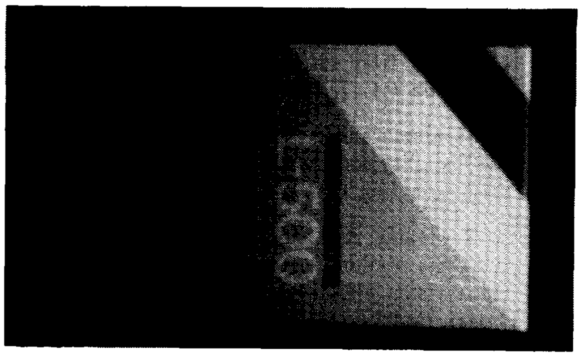


Figure 10: Image with Light on Surface of Maximum Brightness

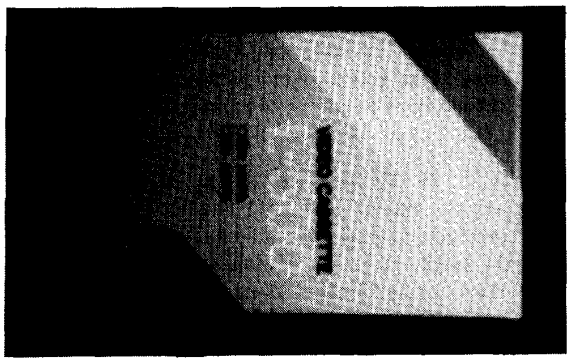


Figure 11: Image with Light on Surface of Maximum Brightness for Medium Gray Surface

lighting placement takes these into account to guarantee that specified object edges will be detected.

Image Digitization

An image is formed by light that passes through the camera lens, irradiating the pixels in the camera image plane. The light falling on the pixels generates an electric charge, and the camera periodically transfers the accumulated charge at each pixel to generate an analog signal. This signal is then sampled by the digitizer to produce a "gray level" for each image pixel. Such gray levels are typically encoded in 8 bits, resulting in a range of gray level values from 0 through 255. The transformation between image irradiance and digitized pixel values for our system was determined experimentally, producing the data shown in Figure 12. Based on these data, we use a linear model to relate image irradiance (I) to pixel value (P):

$$P = (20.5)I + 15 \tag{6}$$

Sobel Edge Operator

One of the most common and simplest methods to detect edges in an image is to calculate the gradient at each pixel and then threshold its magnitude. We are currently using a version of the gradient known as a Sobel operator. The relevant characteristic of such an operator is that, given an ideal step-edge of height ΔP (a pixel-to-pixel difference ΔP at the edge, with constant gray level on each side of the edge), the gradient magnitude is approximately proportional to ΔP. For the Sobel operator the proportional constant is 4.0 if the edge is aligned with the image rows and columns; the constant is 4.2 if the edge is diagonal to the image. Thus, the threshold on gradient magnitude for edge detection can be converted into a required difference in pixel values across an edge.

Pixel Contrast and Light Locations

Consider a straight, dihedral edge, having included angle ψ, and two small surface patches, A and B, on either side of the edge, as shown in Figure 13. Further assume that the reflectance properties of both patches are identical. If the light source is placed along the average surface normal of A and B, then the incident angles for the patches will be equal and they will appear equally bright, preventing detection of the edge between them. The contrast across the edge increases as the light source is moved farther from the average normal, until a limiting case is reached, where the light source becomes coplanar with one of the surfaces.

Figure 14 shows the two infinitesimal surface patches from Figure 13 with surface normals labeled N_A and N_B, respectively. Consider a point source located at an arbitrary location, L, forming incident angles θ_A and θ_B. Let R be the distance from L to the infinitesimal patches, ℓ be the projection of L onto the plane formed by N_A and N_B, α be the angle of ℓ measured from N_B, and β be the angle of L out of the plane of N_A and N_B.

Since we need only consider the diffuse component of reflection (we are avoiding locations causing specular reflection into the camera), the pixel gray levels of the two patches are proportional to the cosine of the incident angles, and inversely proportional to the square of the point-source distance. Therefore, the pixel difference requirement is

$$\Delta P = \frac{K}{R^2} |\cos\theta_A - \cos\theta_B| \tag{7}$$

Substituting and solving for the point-source distance as a function of angle,

$$R^2 = \frac{K}{\Delta P} |\cos\beta| \, |\cos(\psi - \alpha) - \cos\alpha| \tag{8}$$

Figure 15 shows a perspective plot of distance (R) versus angle (α, β) for Equation 7, when the dihedral angle (ψ) is 90 degrees and ΔP is 30 gray levels. Vertical lines denote the limiting angles, $\alpha = 0$ and $\alpha = \psi$, at $\beta = 0$. An orthographic projection of these data appears in Figure 16. These data form two families of curves; each family forms a boundary between the regions of acceptable and unacceptable light locations. The unacceptable light locations, where insufficient contrast would result, lie between the two families of curves (see Figure 16). When $\alpha = \psi/2$, no value of R provides sufficient contrast (however, due to quantization error in the sampling of α angles for generating the display, the minimum data values in these figures are not zero).

In the future we plan to use the relationship in Equation 8 to remove the unacceptable light locations for each target edge by sweeping this shape along the edge and removing the locations that lie between the families of curves. We will then verify these techniques by detecting edges on a variety of objects and extend this analysis to other types of edge detectors.

Acknowledgement

The authors gratefully acknowledge the support provided by National Science Foundation Grant DMC-8714492.

References

[1] S. Sakane, M. Ishii, and M. Kakikura. Hand-eye simulator: A basic tool for off-line programming of visual sensors. In *Proc. 1985 Int. Conf. on Advanced Robotics*, 1985.

[2] S. Sakane, M. Ishii, and M. Kakikura. Occlusion avoidance of visual sensors based on a hand-eye action simulator system. *International Journal of the Robotics Society of Japan*, 2(2), 1987.

[3] C.K. Cowan, J.L. DeCurtins, P.D. Kovesi, P.G. Mulgaonkar, and D. Nitzan. Automated sensor placement. In *Proceedings of the 13th NSF Manufacturing Systems Research Conference.* Society of Manufacturing Engineers, Dearborn, Michigan, 1986.

[4] C. K. Cowan. Model-based synthesis of sensor location. In *Proc. IEEE Conference on Robotics and Automation*, April 1988.

[5] C.K. Cowan and P.D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Trans. on PAMI*, May 1988.

[6] K. Tarabanis and R. Y. Tsai. Viewpoint planning: the visibility constraint. In *Proc. DARPA Image Understanding Workshop*, 1989.

[7] K. Tarabanis, R. Y. Tsai, and P. K. Allen. Satisfying the resolution constraint in the "mvp" machine vision planning system. In *Proc. DARPA Image Understanding Workshop*, pages 850–860, 1990.

[8] C. K. Cowan and A. Bergman. Determining the camera and light-source location for a visual task. In *Proc. IEEE Conference on Robotics and Automation*, May 1989.

[9] S. Sakane, T. Sato, and M. Kakikura. Development of an active vision planning engine towards autonomous hand-eye coordination. In *Proc. 20th ISIR*, Tokyo, October 1989.

[10] S. Yi, R. M. Haralick, and L. G. Shapiro. Automatic sensor and light source positioning for machine vision. Technical Report EE-ISL-89-04, Dept. of Electrical Engineering, University of Washington, Seattle, Washington 98195, September 1989.

[11] Nicodemus, F.E., et al. Geometrical considerations and nomenclature for reflectance. National Bureau of Standards, Monograph 160, 1977.

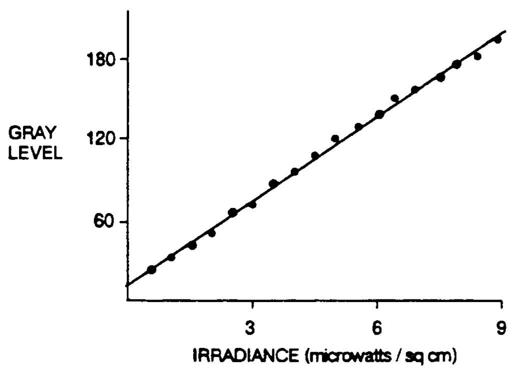


Figure 12: Experimental Data Relating Image Irradiance and Digitized Pixel Values

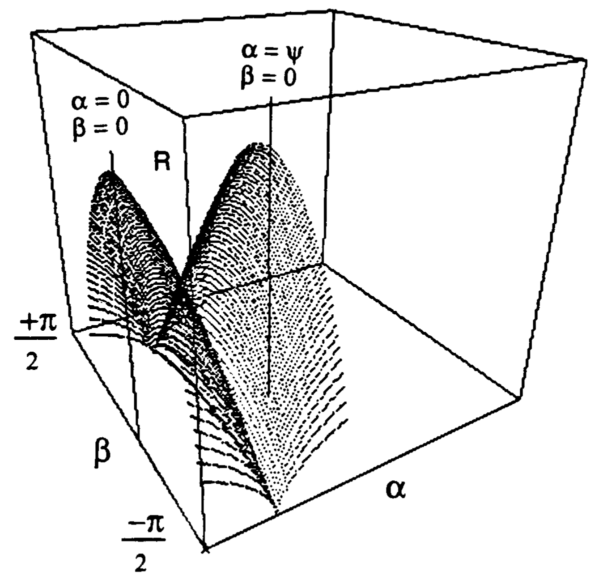


Figure 15: Point-Source Distance versus Angle

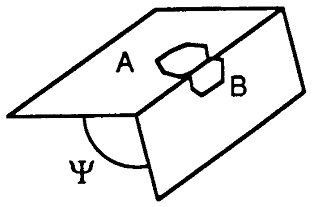


Figure 13: Infinitesimal Patches along a Dihedral Edge

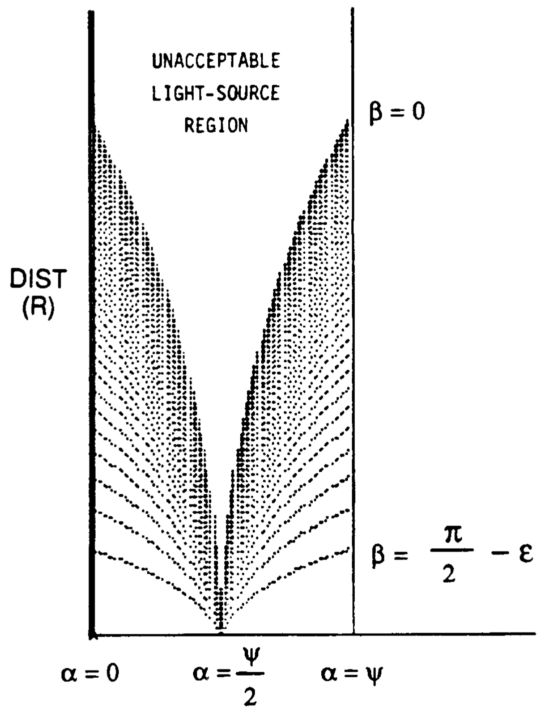


Figure 16: Projection of Distance versus Angle

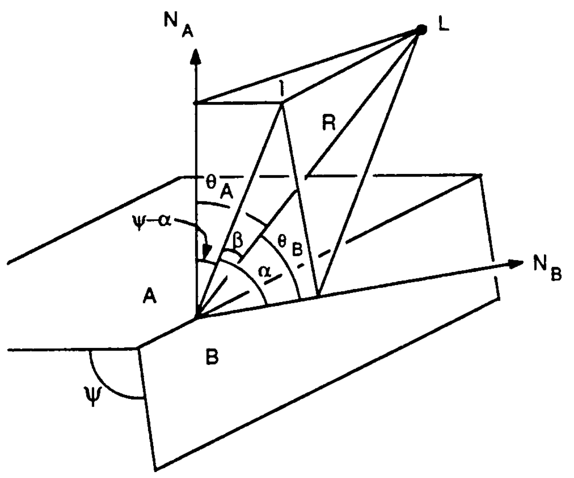


Figure 14: Light Placement for Dihedral Edge