Recognition and Tracking of 3D Objects by 1D Search

Daniel F. DeMenthon and Larry S. Davis

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742

Abstract

We show that the bounded error recognition problem for images of *non-planar* 3D objects using point features can be decomposed into 1D search tasks, along lines joining the origin of the object coordinate system to the feature points chosen to model the object. Points are constructed along these lines at locations given by the coordinates of the detected image points; concurrent bracketing of these points by segment tree search along each of these lines provides maximal matchings between feature points and image points. Depth of search is limited by pixel resolution. This method is well adapted to the task of tracking objects in the presence of variable occlusions and clutter.

1 Introduction

The task considered here is model-based recognition and tracking using *non-planar* configurations of point features to describe 3D objects. We formulate the problem with the techniques introduced by Baird [1] and extended by Cass [4], Breuel [2, 3], and others [7, 8]. The approach consists of matching model features and image features to determine the pose of the object, while assuming that there are spurious or missing image features, and uncertainties in detecting these features. The image features are assumed to be located somewhere within bounded regions around their detected locations; the problem posed with this uncertainty model is sometimes called *bounded error* recognition problem. Authors have mostly applied this model to the matching between 2D images and to the recognition of flat objects. One of the major obstacles to the practical extension of past work to general non-planar objects has been that the search in the general case has to be performed in an 8-dimensional transformation space [4]. The proposed method reduces the search to a problem of *1D search by segment trees* along lines defined in the object model.

We introduce new equations for expressing the relationship between model points and image points in a perspective model of projection, generalizing the formulation introduced for iteratively computing object pose (the POSIT algorithm; see Appendix and [5]). These

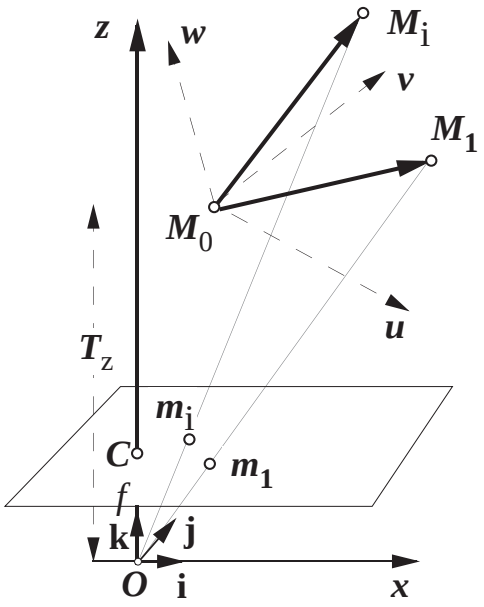

Figure 1: Perspective projections m_i for object points M_i

equations place the nonlinear terms of the transformation on the right hand side, in combination with the image coordinate terms. The advantage of this formulation is that when initial estimates are made for these nonlinear terms, *the uncertainty in these estimates can be modeled as additional image uncertainty.* We obtain linear constraints for two 4D vectors **I** and **J** proportional to the first and second rows of the homogeneous transformation matrix of the object. These linear constraints represent *slabs* of space which are perpendicular to *feature vectors* (joining the origin of the object coordinate system to the object feature points) at points depending on the image coordinates of these feature points. Regions of space where the largest numbers of slabs intersect correspond to maximal matchings between object points and image points. To find these regions we adapt the binary tree search advocated by Breuel [3] for this type of problem; however, with our formulation, the search can be decomposed into 1D search tasks by segment trees [9] along the feature vectors. Simultaneous search tasks are performed for regions containing **I** and **J**, and are pruned by mutual constraints resulting from the fact that **I** and **J** belong to slabs corresponding to the same image points. Other pruning criteria utilize the fact that the first three components of **I** and **J** define vectors which are perpendicular and equal in length. When an object is tracked, the nonlinear terms of the equations can be evaluated from the pose results obtained for the previous image frame, and the dimensions of the initial search space can be reduced because the position and extent of this search space can be deduced from predictive techniques and bounds on admissible motions. This method accommodates the disappearance of features due to self-occlusion during the object's motion.

2 Notation

In Fig. 1, we show the classic pinhole camera model, with its center of projection O, its image plane at distance f (the focal length) from O, its axes Ox and Oy pointing along the rows and columns of the camera sensor, and its third axis Oz pointing along the optical axis. The unit vectors for these three axes are called $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$. In this paper, the focal length and the intersection of the optical axis with the image plane (image center C) are assumed known.

An object with feature points M_0, M_1, ..., M_i, ..., M_n is located in the field of view of the camera. The object coordinate frame is $(M_0\mathbf{u}, M_0\mathbf{v}, M_0\mathbf{w})$. The coordinates (U_i, V_i, W_i) of the points M_i in this frame are known. The images of the points M_i are called m_i, and the image coordinates (x_i, y_i) of each m_i are known. In the recognition problem, one of the goals is to be able to say that m_i is indeed the image of M_i, which is not obvious since the pose of the object is also unknown. In other words, the correspondences between the image points and the object points have to be found.

The rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$ of the object in the camera coordinate system can be grouped into a single 4×4 transformation matrix which will be called the pose matrix $\mathbf{P}$ in what follows:

$$\mathbf{P} = \begin{bmatrix} i_u & i_v & i_w & T_x \\ j_u & j_v & j_w & T_y \\ k_u & k_v & k_w & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

To obtain the coordinates of an object point M_i in the camera coordinate system using this pose matrix $\mathbf{P}$ instead of the more traditional rotation matrix and translation vector, one simply multiplies $\mathbf{P}$ by the coordinates of point M_i or vector $\mathbf{M_0M}_i$ in the object coordinate system. This operation requires that point M_i or vector $\mathbf{M_0M}_i$ be given a fourth coordinate (a fourth dimension) equal to 1 (one). These four coordinates are said to be the homogeneous coordinates of the point or vector. In the following, vectors and points are four-dimensional (4D) entities in this homogeneous space, unless otherwise specified.

The first line of the matrix $\mathbf{P}$ is a *row vector* that we call $\mathbf{P}_1$. The other row vectors are called $\mathbf{P}_2$, $\mathbf{P}_3$ and $\mathbf{P}_4$. The coordinates $T_x, T_y, T_z, 1$, the fourth *column* of the matrix, are the coordinates of the translation vector $\mathbf{T}$ (in Fig. 1, the translation vector $\mathbf{T}$ is the vector $\mathbf{OM}_0$). In the first row vector, $\mathbf{P}_1$, the coordinates i_u, i_v, i_w are the coordinates of a 3D vector, $\mathbf{i}$, which is also the first row of the rotation matrix $\mathbf{R}$ of the transformation. Notice that vector $\mathbf{i}$ is also the unit vector for the x-axis of the camera coordinate system expressed in the object coordinate system $(M_0\mathbf{u}, M_0\mathbf{v}, M_0\mathbf{w})$. Similarly, in the second row vector, $\mathbf{P}_2$, the coordinates j_u, j_v, j_w are the coordinates of a 3D vector $\mathbf{j}$ which is the second row vector of the rotation matrix; the vector $\mathbf{j}$ is also the unit vector for the y-axis of the camera coordinate system, expressed in the object coordinate system $(M_0\mathbf{u}, M_0\mathbf{v}, M_0\mathbf{w})$. In the third row vector, $\mathbf{P}_3$, the coordinates k_u, k_v, k_w are the coordinates of a 3D vector $\mathbf{k}$ which is the cross product of the two vectors $\mathbf{i}$ and $\mathbf{j}$. In the following, we show how the vectors $\mathbf{I} = \frac{f}{T_z}\mathbf{P}_1, \mathbf{J} = \frac{f}{T_z}\mathbf{P}_2$ can be computed. Once these are obtained, the quantity T_z is found by

noticing that the first three coordinates of $\mathbf{I}$ and $\mathbf{J}$ define 3D vectors $\mathbf{R}_1$ and $\mathbf{R}_2$ with norms equal to f/T_z. The completion of the object pose matrix $\mathbf{P}$ is then straightforward (see step 3 in Appendix).

3 Fundamental Equations

The fundamental relations which relate the row vectors $\mathbf{P}_1$, $\mathbf{P}_2$ of the pose matrix, the coordinates of the object vectors $\mathbf{M}_0\mathbf{M}_i$ in the object coordinate system, and the coordinates x_i and y_i of the perspective images m_i of M_i are

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x_i',$$

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{J} = y_i', \tag{2}$$

with

$$\mathbf{I} = \frac{f}{T_z}\mathbf{P}_1, \quad \mathbf{J} = \frac{f}{T_z}\mathbf{P}_2, \tag{3}$$

$$x_i' = x_i(1+\varepsilon_i), \quad y_i' = y_i(1+\varepsilon_i), \tag{4}$$

and

$$\varepsilon_i = \mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_3/T_z - 1 \tag{5}$$

It is useful to introduce the unknown coordinates (X_i, Y_i, Z_i) of vector $\mathbf{M}_0\mathbf{M}_i$ in the *camera* coordinate system for the sole purpose of demonstrating that these equations are correct. We remember that the dot product $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_1$ is the operation performed when multiplying the first row of the transformation matrix $\mathbf{P}$ by the coordinates of an object point in the object frame of reference to obtain the x-coordinate X_i of M_i in the camera coordinate system. Thus $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_1 = X_i$. For the same reason, the dot product $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_3$ is equal to Z_i, thus $(1+\varepsilon_i) = Z_i/T_z$. Also, in perspective projection, the relation $x_i = fX_i/Z_i$ holds between image point coordinates and object point coordinates in the camera coordinate system. Using these expressions in the equations above leads to identities, which proves the validity of these equations.

Two problems must be addressed before applying these equations:

1. The terms ε_i are generally unknown. These terms depend on $\mathbf{P}_3$ (Eq. (5)), which can be computed only after $\mathbf{I}$ and $\mathbf{J}$ have been computed (Section 2).

2. These equations can be used only after the correspondences between image points and object points have been established. Only then can the correct values for the image coordinate x_i be written on the right hand side for a given vector $\mathbf{M}_0\mathbf{M}_i$ on the left hand side.

Starting with the first problem of dealing with unknown ε_i, notice that the coordinates $x_i' = x_i(1+\varepsilon_i)$ and $y_i' = y_i(1+\varepsilon_i)$ are the image coordinates of the object points M_i by a model of projection which is a scaled orthographic projection. Indeed $x_i = fX_i/Z_i$ can be written as $x_i = \frac{f}{1+\varepsilon_i}X_i/T_z$, and we obtain $x_i' = fX_i/T_z$. In other words, image points (x_i', y_i') are obtained by "flattening" the object by orthographic projection of its points onto the plane $z = T_z$ through M_0 before performing a perspective projection. To obtain estimates for $\mathbf{I}$ and $\mathbf{J}$, we first use x_i and y_i instead of x_i' and y_i' in Eqs. (2), thereby making errors $x_i\varepsilon_i$ and $y_i\varepsilon_i$ which are added to the estimates of the image errors. Once estimates for $\mathbf{I}$ and $\mathbf{J}$

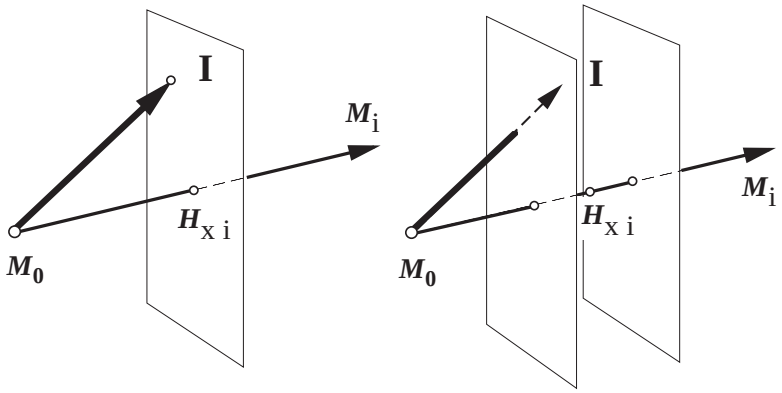

Figure 2: *Left:* In the absence of uncertainties, the head of vector $\mathbf{I}$ belongs to a plane orthogonal to the feature vector $\mathbf{M}_0\mathbf{M}_i$ at the *x-point* H_{xi} located at abscissa $x'_i/|\mathbf{M}_0\mathbf{M}_i|$. *Right:* Because of uncertainties in image detection and ε_i, the head of $\mathbf{I}$ lies in a 4D slab perpendicular to $\mathbf{M}_0\mathbf{M}_i$.

have been obtained, these estimates can be used to find more precise values of ε_i, which in turn lead to better estimates of $\mathbf{I}$ and $\mathbf{J}$.

Regarding problem (2), in some computer vision applications the correspondences can be obtained prior to the pose information. For example, in calibration applications, the feature points may be the centroids of marks that are easy to differentiate on the target calibration object. Then Eqs. (2) can be solved iteratively by first making rough estimates for ε_i (setting them to zero when no information is available), solving the linear systems for $\mathbf{I}$ and $\mathbf{J}$, finding better estimates for ε_i, and repeating the process. A least square object pose is generally found in a few iterations. An appendix summarizes the steps of this algorithm, which was introduced in a more geometrical form in [5]. The rest of this paper addresses the more difficult situation where the correspondences between image points and object points cannot be obtained prior to the pose information.

4 Geometric Constraints for the Solutions I and J

The following discussion shows that the solutions $\mathbf{I}$ and $\mathbf{J}$ are located within small polyhedral regions which can be identified with respect to the 4D homogeneous coordinate system of the object.

Equations such as $\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x'_i$ (Eq. (2)) can be viewed as geometric constraints on the vector $\mathbf{I}$ in space with respect to the feature vectors $\mathbf{M}_0\mathbf{M}_i$: If the foot of vector $\mathbf{I}$ coincides with M_0, the head of $\mathbf{I}$ must project on the feature line $\mathbf{M}_0\mathbf{M}_i$ onto a point H_{xi} with abscissa $x'_i/|\mathbf{M}_0\mathbf{M}_i|$. Equivalently, the head of $\mathbf{I}$ must belong to a plane perpendicular to $\mathbf{M}_0\mathbf{M}_i$ at H_{xi} (Fig. 2). In the following, the points H_{xi} are called *x–points*. They are points constructed on the feature lines $\mathbf{M}_0\mathbf{M}_i$ using the x-coordinates of the image points. Similarly, the points H_{yi} considered in constructing the vector $\mathbf{J}$ have abscissae $y'_i/|\mathbf{M}_0\mathbf{M}_i|$ and are called y–points.

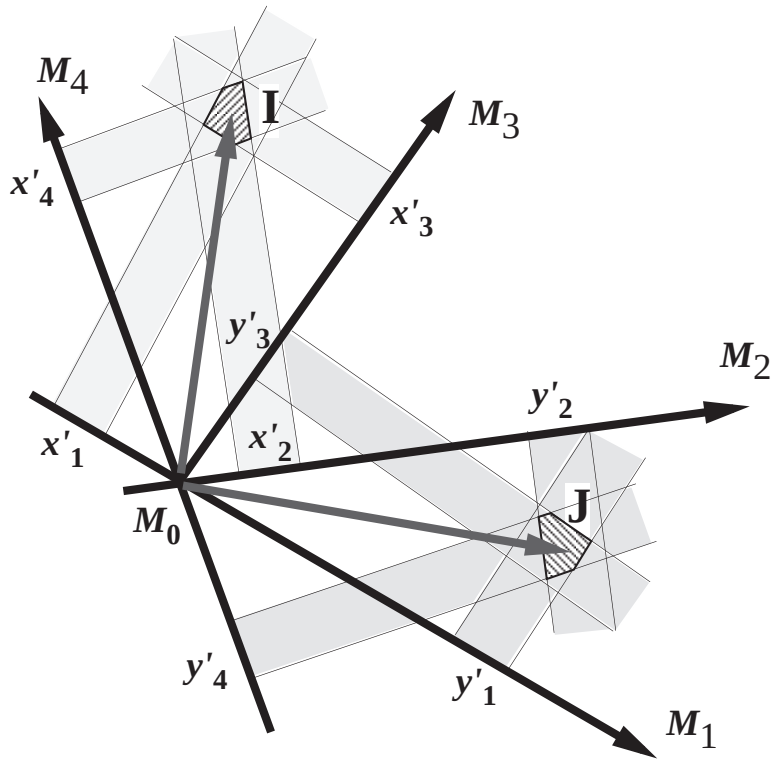

Figure 3: The head of **I** is found at the intersection of the slabs corresponding to the x-coordinates x'_i (corrected by $1+\varepsilon_i$) of the feature points M_i. The vector **J** is then found at the intersection of the slabs using the y-coordinates y'_i for the same correspondence. A further verification is obtained from the property that **I** and **J** (or 3D vectors from the first three coordinates of **I** and **J** in 4D) are perpendicular and have same lengths.

In most situations, the terms $x'_i = x_i(1+\varepsilon_i)$ are known only approximately. Bounds for the uncertainties in these terms can be computed by adding the image error e and the error e' made in estimating $x_i\varepsilon_i$. The ε_i terms are the projections of the vectors $\mathbf{M_0M}_i$ on the camera optical axis, divided by the distance T_z from the camera to the point M_0 along the camera optical axis (Eq. (5)). Therefore an upper bound for these terms is R/D, where R is the radius of a sphere centered at M_0 containing the object and D is a lower bound for the distance T_z. Clearly, estimating tight upper bounds for these errors is made easier if we have some idea of the range in which the object is expected to be found. When the object is being tracked and an approximate pose for the object has been found from a previous image, ε_i can be estimated from this previous pose, and the uncertainty interval can be reduced and centered around $x_i(1+\varepsilon_i)$.

Because of these uncertainties, all we can say is that the head of **I** projects onto the feature vector within the uncertainty interval around an x-point H_{xi}. Equivalently, the head of **I** must belong to a *slab* perpendicular to $\mathbf{M_0M}_i$ at x-point H_{xi} with a thickness defined by the uncertainty interval (Fig. 2).

For **I** to be solution of a system of n Eqs. (2) for $i = 1, 2, 3, ..., n$, the head of vector **I**

must belong to the slab S_{11} defined at x-point H_{x1} on the feature vector $\mathbf{M}_0\mathbf{M}_1$, the slab S_{22} defined at H_{x2} on $\mathbf{M}_0\mathbf{M}_2$, etc. Therefore $\mathbf{I}$ must belong to the intersection of these n slabs. A necessary condition for this to occur is that there exists a region Σ in space included in at least n slabs S_{ii} (Fig. 3). This region Σ is helpful for locating $\mathbf{I}$ only if it is a closed polyhedral region. In the 4D space of homogeneous coordinates, this occurs if the number of slabs is at least $n = 4$ and if the feature vectors $\mathbf{M}_0\mathbf{M}_i$ are not coplanar.

Similarly, $\mathbf{J}$ must belong to the intersection of n slabs T_{ii}. A necessary condition for this to occur is that there exists a region Θ in space included in at least n slabs T_{ii} (Fig. 3).

Furthermore, the n slabs S_{ii} including the region Σ and the n slabs T_{ii} including the region Θ must be defined by the same feature vectors $\mathbf{M}_0\mathbf{M}_i$ and the same image points $\mathbf{m}_i$. Therefore the n slabs T_{ii} at the y-points H_{yi} computed from the same points $\mathbf{m}_i$ which produced the slabs S_{ii} must intersect in a non-empty polyhedral region Θ.

As additional conditions, the solution vectors $\mathbf{I}$ and $\mathbf{J}$ are constrained in relative amplitude and orientation; the first three coordinates of $\mathbf{I}$ and $\mathbf{J}$ define two 3D vectors $\mathbf{R}_1$ and $\mathbf{R}_2$. These vectors are proportional to $\mathbf{i}$ and $\mathbf{j}$ respectively, with the same coefficient of proportionality f/T_z. Therefore $\mathbf{R}_1$ and $\mathbf{R}_2$ must be orthogonal and have equal lengths. Therefore a pair of regions Σ and Θ can contain the heads of vectors $\mathbf{I}$ and $\mathbf{J}$ *only* if (1) the range of 3D distances from M_0 to the points of Σ overlaps the range of distances from M_0 to the points of Θ, and (2) the extrema of the 3D dot products between pairs of vectors with heads in each region are of opposite signs.

5 Finding solution regions with unknown correspondences

In the problem addressed here, the correspondences between the N feature points and some of the n' detected image points are not known. Given an object point M_i, we do not know which image point among $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$, etc, is the image of M_i. Furthermore, some points M_i may not have images, and some image points may not correspond to any of the object points. However, we assume for the moment that the number n_0 of image points that match the object points is defined (finding this number is the objective of the next section).

The best we can then do is to consider, for the N feature points M_i, all the slabs defined from the n' detected image points. On each feature vector $\mathbf{M_0M}_i$ we can construct an x-point H_{xij} for each detected image point $\mathbf{m}_j$, and consider the corresponding slab. Slabs for a given feature vector $\mathbf{M_0M}_i$ are parallel. Slabs from two different feature vectors intersect each other (object feature points M_i can be chosen so that lines M_0M_i are well separated).

The proposed method finds small regions of space Σ and Θ including the heads of vectors $\mathbf{I}$ and $\mathbf{J}$. If there are indeed at least n_0 image points of the object feature points among the detected image points, and if the bounds for the image and ε_i uncertainties are correct, there exists a pair of regions (Σ, Θ) such that Σ is included in at least n_0 slabs, defined by x-points H_{xij} located on feature vectors $\mathbf{M_0M}_i$ and corresponding to image points $\mathbf{m}_j$, and such that Θ is also included in at least n_0 slabs, defined by y-points H_{yij} located on feature vectors $\mathbf{M_0M}_i$ and corresponding to the same image points $\mathbf{m}_j$.

The method is based on eliminating pairs of regions of space which do not satisfy the geometric constraints defined in the previous section, proceeding from coarse to fine regions by bisection of space. Considering one region Σ' and one region Θ', these *cannot* contain the heads of $\mathbf{I}$ and $\mathbf{J}$ if:

1. Σ' or Θ' are not intersected by n_0 slabs (then no point inside either region can be included in n_0 slabs).

2. Σ' and Θ' are not intersected by n_0 slabs constructed from the same image points.

3. The range of 3D distances from M_0 to the points of Σ' does not overlap the range of distances from M_0 to the points of Θ'. (Hence the two regions cannot respectively contain heads of $\mathbf{I}$ and $\mathbf{J}$ at equal distances from M_0).

4. The extrema of the 3D dot products between pairs of vectors with heads in each region are of the same sign. (Hence the two regions cannot respectively contain heads of perpendicular vectors $\mathbf{I}$ and $\mathbf{J}$).

There exists a pair of regions (Σ, Θ) which cannot be eliminated by the above criteria, and we can recognize and label in the image the n_0 points that contributed to these regions. We find these regions by recursive bisection of space to find region Σ, and simultaneous bisection of space to find region Θ. We simultaneously explore two binary trees by depth-first search, pairing branches from both trees and pruning paired branches excluded by the above criteria.

As a further verification of the matching of the n_0 points (also providing a more accurate pose matrix), we proceed as follows: The terms ε_i can be computed at this point from the pose matrix, using the expression given with Eq. (2). The terms $x'_i = x_i(1 + \varepsilon_i)$ and $y'_i = y_i(1 + \varepsilon_i)$ can then be computed, as well as reduced uncertainty intervals. These corrected coordinates and intervals define thinner slabs at slightly different locations which result in smaller regions Σ and Θ, less ambiguity between possible matchings, and a more accurate pose.

6 Finding the best region

The explanations so far have focused on finding regions Σ or Θ at the intersection of at least n_0 slabs. We would actually like to find the regions at the intersection of the highest number of slabs, because this provides the maximal number of matches between image points and object points. We start the search with $n_0 = n'$, the total number of image points detected. Generally, some image points do not have any matches, and the search quickly fails. We then decrement n_0 until a search succeeds.

7 Search for regions Σ and Θ

A binary tree search was advocated by Breuel for this type of problem [3]. To search for a single region, say Σ, the approach consists of starting with a large box which is guaranteed to contain all the regions of interest, and recursively dividing the box into two child boxes. At depth 1, the plane used for dividing the box is perpendicular to the x-axis of the 4D space, at depth 2 the plane is perpendicular to the y axis, at depth 3 to the z-axis, and at depth 4 to the k-axis. At depth 5 we are back to a division perpendicular to the x-axis, and so on. Eventually, we have divided the space into boxes so small that at least one of them is contained in the region Σ, at the intersection of n_0 slabs. The process is illustrated in 2D in Fig. 4.

7.1 Simultaneously searching for two regions

We start with an initial box A_0 large enough to include the head of **I** and an initial box B_0 large enough to include the head of **J** (from Eqs. (2) and (3), one can show that the coordinates of these vectors can be expressed in pixels and are smaller than the largest image point coordinates). The box A_0 is divided into two boxes A_1 and A_2. The box B_0 is divided into two boxes B_1 and B_2. Then the pair of boxes A_1 and B_1 is considered, and the elimination criteria of the previous section are applied to the pair. If none of the elimination criteria applies, the two boxes are themselves divided, and the process is repeated recursively. If any elimination criterion applies, another pair at the same depth, say A_1 and B_2, is considered. If all four possible pairs are eliminated, we backtrack to a pair which has not be considered at the previous depth. If the previous depth is the root depth, the search has failed. The search succeeds when two boxes of small predefined dimension (a few pixels if the coordinates of **I** and **J** are expressed in pixels) survive the elimination criteria, and

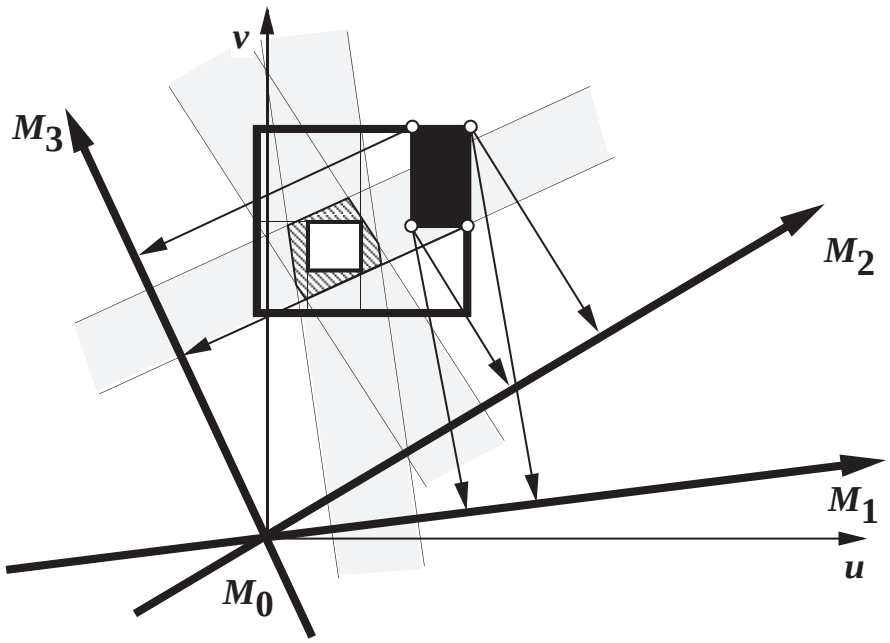

Figure 4: A search by bisection of space locates a box included in a region at the intersection of three slabs (white box). Boxes which are not *intersected* by three slabs are pruned (black box). Tests for intersections between boxes and slabs are performed using box projections on the feature vectors.

are included in n_0 corresponding slabs; this second test is added when the boxes become small enough to fit in the slabs; the elimination criteria test only for the necessary (but not sufficient) condition that a box be intersected by n_0 slabs.

7.2 Tests for intersection of a box with n slabs

If a box does not intersect n slabs, no subdivision of this box will be included in n slabs, and this branch of the tree can be pruned [3]. This is one of the elimination criteria defined above. The tests for inclusion and intersection are simpler here than in Breuel's formulation. A box intersects n slabs if, for each of n feature vectors $\mathbf{M}_0\mathbf{M}_i$, the 1D projection of the box on $\mathbf{M}_0\mathbf{M}_i$ intersects the uncertainty interval around an x-point H_{xij}.

Instead of checking for intersections of intervals, we augment the interval of the box projection on each side by the amplitude of the uncertainty interval, and we count the x-points contained in this interval. The uncertainty intervals and the lengths p_{di} of projection of a box at depth d on feature vectors $\mathbf{M}_0\mathbf{M}_i$ are precomputed (Fig. 5).

The count of intersections between boxes and slabs is incremented by 1 for each feature vector when we find at least one x-point inside the (augmented) projection interval of the box. Accordingly, we have to keep track of which x-points are contained in the augmented projection interval of the box. Having done the same task with the parent box, we know the list of x-points included in the parent interval. Each child interval shares with the parent interval one bound. The other bound is inside the parent interval and must be located with

respect to the x-points (Fig. 5). This is achieved by bisection over the parent list of x-points (the root list of x-points was sorted). The location of this bound provides the element count for the new list. The list of x-points for the left child has the same address than the parent list, whereas the list for the right child has an address offset by the index position of its left bound in the parent list.

7.3 Tests for inclusion of a box in a region

We verify that a box at depth d is included in n_0 slabs by verifying that for each of at least n_0 feature vectors $\mathbf{M}_0\mathbf{M}_i$, the 1D projection of this box on $\mathbf{M}_0\mathbf{M}_i$ is included in the uncertainty interval around an x-point H_{xij}.

The depth for which all the lengths p_{di} of projection of a box at depth d on feature vectors $\mathbf{M}_0\mathbf{M}_i$ are smaller than the lengths u_i of the uncertainty intervals is also precomputed, and we start checking for inclusion of the box projections in the uncertainty intervals only when the tree search has reached this depth. Instead of checking whether this projection is included in the uncertainty interval around a point H_{xij}, we check whether there is a point H_{xij} in the interval of length $(u_i - p_{di})$.

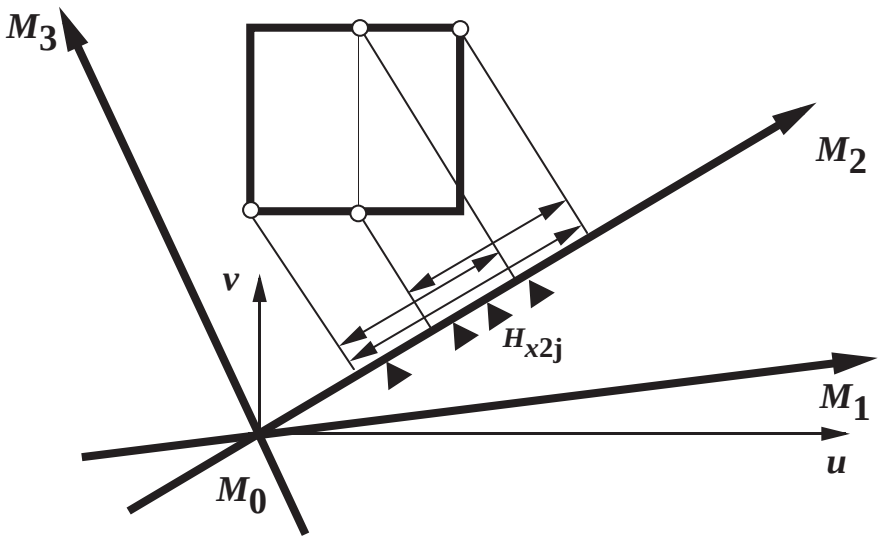

Figure 5: The projection segment of a child box shares one bound with the projection segment of the parent box. A binary search finds the position of the other bound of this segment in the sorted parent list of x-points. This position provides the element count and the address for the child list.

8 Tracking

In a tracking task, the object has been found in the previous image field, and its pose has been estimated after finding acceptable vectors **I** and **J** (Section 2). We propose to perform the tracking in the 4D space where the search for **I** and **J** takes place. First, the previous pose allows us to compute estimates for the terms ε_i, and to take $x_i' = x_i(1 + \varepsilon_i)$ and $y_i' = y_i(1 + \varepsilon_i)$ to define the positions of the x-points and y-points on the feature vectors. The error made by using ε_i from a previous frame contributes to the uncertainty intervals around these points and can be computed from upper bounds $d\theta$ and dT on possible rotation angle and translation increments between two frames. Second, the vector **I** transformed into $\mathbf{I}' = \mathbf{I} + d\mathbf{I}$ between two frames, and the search for the head of $\mathbf{I}'$ can be limited to a box centered around the head of **I** and of dimension larger than $|d\mathbf{I}|$, also depending on $d\theta$ and dT. Predictive techniques can be applied to predicting $\mathbf{I}'$ and the uncertainty on $\mathbf{I}'$ to further reduce the size of the initial search box for $\mathbf{I}'$, and similarly for $\mathbf{J}'$.

9 Summary

We have introduced new equations for expressing the relationship between model points and image points in a perspective model of projection, which place the nonlinear terms of the perspective transformation on the right hand side in combination with the image coordinate terms. The uncertainty on the estimates of these nonlinear terms can be modeled as additional image uncertainty. We obtain linear constraints on two 4D vectors **I** and **J** proportional to the first and second row of the homogeneous transformation matrix of the object. These linear constraints represent slabs of space which are perpendicular to feature vectors (joining the origin of the object coordinate system to the object feature points) at points depending on the image coordinates of these feature points. Regions of space where the largest numbers of slabs intersect locate the vectors **I** and **J** and correspond to maximal matchings between object points and image points. Simultaneous binary tree search tasks are performed for regions containing **I** and **J**, and are pruned by mutual constraints resulting from the fact that **I** and **J** belong to slabs corresponding to the same image points. Other pruning criteria utilize the fact that the first three components of **I** and **J** define vectors which are perpendicular and equal in length. Most of the search consists of 1D search by segment trees along the feature vectors.

Appendix: Iterative Pose Computation from Point Correspondences

Here we summarize a simple iterative algorithm for finding the pose of an object when a matching between object feature points and image points is known. It is an analytic formulation of the POSIT (Pose from Orthography and Scaling with Iterations) algorithm [5] in homogeneous form, which removes the need to locate the image of the origin M_0 of the object coordinate system. Note that this pose calculation is presented independently of the search method described above, which finds the matching and the pose by binary search of space when the matching is not known.

The equations to be solved are Eqs. (2). The steps of the iterative pose algorithm can be summarized as follows:

1. ε_i= best guess, or $\varepsilon_i = 0$ if no pose information is available

2. Start of loop: Solve for $\mathbf{I}$ and $\mathbf{J}$ in the following systems (see next paragraph)

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x_i', \mathbf{M}_0\mathbf{M}_i \cdot \mathbf{J} = y_i'$$

   with

$$x_i' = x_i(1+\varepsilon_i), \quad y_i' = y_i(1+\varepsilon_i)$$

3. From $\mathbf{I}$, get

$$\mathbf{R}_1 = (I_1, I_2, I_3),$$
$$f/T_z = |\mathbf{R}_1|,$$
$$\mathbf{i} = (T_z/f)\mathbf{R}_1,$$
$$\mathbf{P}_1 = (T_z/f)\mathbf{I}$$

   Similar operations yield $\mathbf{j}$ and $\mathbf{P}_2$ from $\mathbf{J}$.

4. $\mathbf{k} = \mathbf{i} \times \mathbf{j}$, $\mathbf{P}_3 = (k_u, k_v, k_w, T_z)$, $\varepsilon_i = \mathbf{M}_0\mathbf{M}_i \cdot \mathbf{P}_3/T_z - 1$

5. If all ε_i are close enough to the ε_i from the previous loop, EXIT, else go to step 2.

6. $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, along with $\mathbf{P}_4 = (0,0,0,1)$, are the four rows of the pose matrix.

We now provide details on finding $\mathbf{I}$ and $\mathbf{J}$ for step 2 of the iterative algorithm. For example, the equations for $\mathbf{I}$ are:

$$\mathbf{M}_0\mathbf{M}_i \cdot \mathbf{I} = x_i'$$

The unknowns are the four coordinates (I_1, I_2, I_3, I_4), of $\mathbf{I}$, and we can write one equation with each of the object points M_i for which we know the position m_i of the image and its image coordinate x_i. One such equation has the form $U_iI_1 + V_iI_2 + W_iI_3 + I_4 = x_i'$, where $(U_i, V_i, W_i, 1)$ are the four coordinates of M_i. If we write equations for several object points M_i, we obtain a linear system of equations which can be written in matrix form as $\mathbf{AI} = \mathbf{V}_x$, where $\mathbf{A}$ is a matrix with i-th row vector $\mathbf{A}_i = (U_i, V_i, W_i, 1)$, and $\mathbf{V}_x$ is a column vector with i-th coordinate equal to x_i'.

Similarly the vector $\mathbf{J}$ can be found by solving the linear system $\mathbf{AJ} = \mathbf{V}_y$, where A is the same matrix, and $\mathbf{V}_y$ is a column vector with i-th coordinate equal to y_i'.

Since there are four unknown coordinates in vectors $\mathbf{I}$ and $\mathbf{J}$, the matrix $\mathbf{A}$ must have at least rank 4 for the systems to provide solutions. This requirement is satisfied if the matrix has at least four rows and the object points are noncoplanar; therefore at least four noncoplanar object points and their corresponding image points are required. The pseudo-inversion operation is applied to matrix $\mathbf{A}$. The pseudo-inverse of matrix $\mathbf{A}$ is called the object matrix $\mathbf{B}$. Since matrix $\mathbf{A}$ is defined in terms of the known coordinates of the object

points in the object coordinate system, the object matrix **B** depends only on the relative geometry of the object points and can be precomputed.

Experiments [5] show that this iterative approach generally provides an accurate pose of the object in a few iteration steps, as long as the points M_i are contained within a camera field of view of less than 90 degrees.

References

[1] Baird, H.S., 1985, "Model-Based Image Matching Using Location", MIT Press, Cambridge, MA.

[2] Breuel, T.M., 1991, "Model Based Recognition using Pruned Correspondence Search", Proc. IEEE Conf. Computer Vision and Pattern Recognition, Maui, HI, pp. 257–262.

[3] Breuel, T.M., 1992, "Fast Recognition using Adaptive Subdivisions of Transformation Space", Proc. IEEE Conf. Computer Vision and Pattern Recognition, Champagn, IL, pp. 445–451.

[4] Cass, T.A, 1992, "Polynomial-Time Object Recognition in the Presence of Clutter, Occlusion, and Uncertainty", Computer Vision–ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp.834–842, Springer-Verlag.

[5] DeMenthon, D.F., 1992, "Model-Based Object Pose in 25 Lines of Code", Proc. DARPA Image Understanding Workshop, San Diego, CA, pp. 753–761; also, Computer Vision–ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp.335–343, Springer-Verlag.

[6] Grimson, W.E.L and D.P. Huttenlocher, 1988, "On the Sensitivity of the Hough Transform for Object Recognition", Proc. International Conf. Computer Vision, Tarpon Springs, FL.

[7] Grimson, W.E.L, D.P. Huttenlocher, and D.W. Jacobs, 1991, "Affine Matching with Bounded Sensor Error: A Study of Geometric Hashing and Alignement", MIT AI Memo 1250.

[8] Grimson, W.E.L, D.P. Huttenlocher, and D.W. Jacobs, 1992, "Affine Matching with Bounded Sensor Error", Computer Vision–ECCV 92, Lecture Notes in Computer Science 588, G. Sandini (Ed.), pp.291–306, Springer-Verlag.

[9] Preparata, F.P., and M.I. Shamos, 1985, "Computational Geometry: An Introduction", Springer-Verlag.