Cornell University
Computer Science 664

Stereopsis and Motion: Extracting Shape from Images

1 Middle Level Vision

One of the goals of so-called 'middle level' visual processing is to extract three-dimensional geometric information from one or more images. Extracting three-dimensional geometry from images is often referred to as *shape-from-x*, because there are a number of different sources of information that can be used to recover the three-dimensional structure of a scene (or shape) from two-dimensional images. For instance, shading in an image reveals information about three-dimensional shapes (e.g., much of the way that the shape of a sphere in a photograph is perceived as being a solid rather than a disk is due to the uniform change in brightness away from the light source). Shape-from-shading is an active area of research, but we will not study it here because it requires a substantial amount more knowledge of surface reflectance properties than we will have time to cover (also its use in practice is quite restricted).

Another source of three-dimensional shape information is provided by the change in location of an object from one image to the next, in a set of two or more images. The two main techniques for extracting image shape from multiple images are *stereopsis* and *structure from motion*. These are the two approaches that we will be considering in some detail in this section of the course.

In the first approach, stereopsis, two images are taken of the same scene from slightly different viewpoints. Those objects in the scene that are far away from the two cameras will appear nearly identical in the two images, whereas those objects that are near the cameras will change significantly between the two images. The key idea underlying stereopsis (or stereo vision) is to make use of the *disparity*, or change in image location, of an object from one view to the next. The closer an object is to the camera, the larger the disparity will be. This allows us to reconstruct three-dimensional shape from disparity. Generally the depth (or distance) information recovered from stereo is quite noisy, and thus a surface interpolation process is often applied to the data (cf. [3]). Such interpolation methods have broad applicability beyond computer vision.

The second approach that we will consider is structure from motion. Here the idea is to take a sequence of images, and to use the motion of an object with respect to the camera to reconstruct its three-dimensional shape. There are many methods for solving this problem, but all of them involve first *tracking* the object (finding corresponding points in successive frames), and then applying some sort of technique to recover the three-dimensional positions of the points from their two-dimensional motions. The tracking problem is itself is quite difficult, particularly when it is necessary to identify corresponding points in successive frames. Many methods require the object to be stationary, and the camera to undergo a restricted type of motion. This helps with both the tracking problem, and the subsequent shape reconstruction.

Stereo vision is the more structured of the two problems, because we assume that the cameras and the objects are fixed. It is also generally assumed that something about the relation between the two camera frames is known. Thus we will first investigate the stereo vision problem, and then turn to the more general problems of tracking and recovering structure from motion.

2 Stereopsis

In the basic stereo vision paradigm, there are two cameras observing a static scene (i.e., where nothing is moving). The relative coordinate systems of the two cameras are known, or are constrained in some fashion. Various modifications include adding a third camera, and adding small motions of the cameras to help resolve possible ambiguities. In the basic two-camera case, the images are generally referred to as L and R (resulting from the left and right cameras respectively). The idea underlying stereopsis is to determine a correspondence (or matching) between each location p_l of L and some location p_r of R. In other words, to find the pairs of points p_l and p_r that result from the projection of the same point p into the two images. Note that a given point p need not have an image in both L and R — there may be some other point in the scene that hides p from view in the left or right image, causing there to be no correspondence.

The *disparity*, or difference in image location, of p_l and p_r then indicates the distance from the cameras to the point p in the world. If an object is infinitely far away, then its projection into the two camera planes will be at the same location, and the disparity will be zero. If an object is close to the cameras then the disparity will be large. In other words, disparity is inversely proportional to the distance between an object and the camera system. Stereopsis is a common technique for recovering shape both in artificial vision systems and in the human visual system. It is not indispensable, however, as a significant percentage of people have little or no stereo vision.

To make the discussion more precise, we will consider the geometry of the camera system. We will use a simple pinhole-camera model, where optical effects due to the lens are ignored completely. This model of the camera geometry is illustrated in Figure 1. A simple camera thus consists of a focal point (or center), o, through which all the rays of light pass, and an image plane I onto which these rays are projected. The *optical axis* of the camera is the line perpendicular to the image plane, I, and through the focal point, o. We will call the intersection of the optical axis with the image plane o'. The distance from o to o' is called the *focal length*, f, of the camera. If we place the origin of the world coordinate system at o, and the origin of the image plane at o', and assume that the optical axis points in the $\hat{z}$ direction, then the following two equations describe the projection of a point at location (x, y, z) in the world into location (x', y') in the image

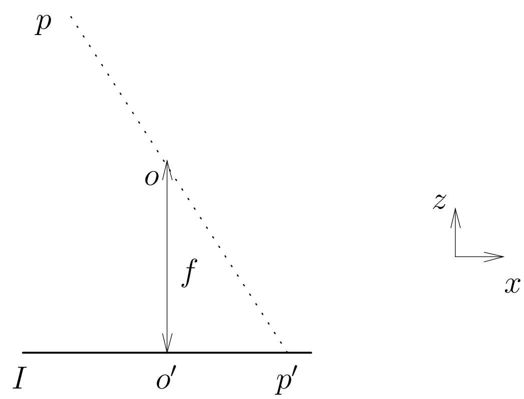


Figure 1: A pinhole camera model.

plane:

$$
\begin{aligned}
\frac{x'}{f} &= \frac{x}{z} \\
\frac{y'}{f} &= \frac{y}{z}.
\end{aligned}
$$

These equations are referred to as the *perspective equations.* The projection of the world onto a plane through a central point in this manner is referred to as perspective projection (or central projection). In general we will assume that the world origin is at o, the image origin is at o', and the optical axis is in the $\hat{z}$ direction, and use the above equations.

For stereo vision there are two cameras at some fixed relative position and orientation with respect to one another. First we will consider a simple stereo camera geometry in which the optical axes of the two cameras are parallel to one another, and are perpendicular to the *baseline* connecting the two camera centers (which are denoted by o_l and o_r). Moreover, we will assume that the focal length, f of the two cameras is the same. This situation is illustrated in Figure 2, where the length of the baseline (distance between the camera centers) is denoted by b. We will let the origin of the coordinate system for the left image plane, L, be the projection of its optic axis, o'_l (and similarly the origin of the right image plane, R, is at o'_r). We will let the origin of the world coordinate frame be along the baseline, at the point equidistant between the two camera centers (at distance $b/2$ from each o_l and o_r). Note that this camera geometry makes L and R the same plane, and with the same coordinate frames except for a translation of the origin in the x-direction.

Consider a point $p = (x, y, z)$ in the world which is imaged into L at location $p_l = (x'_l, y'_l)$ and into R at location $p_r = (x'_r, y'_r)$. By the perspective equations and the geometry

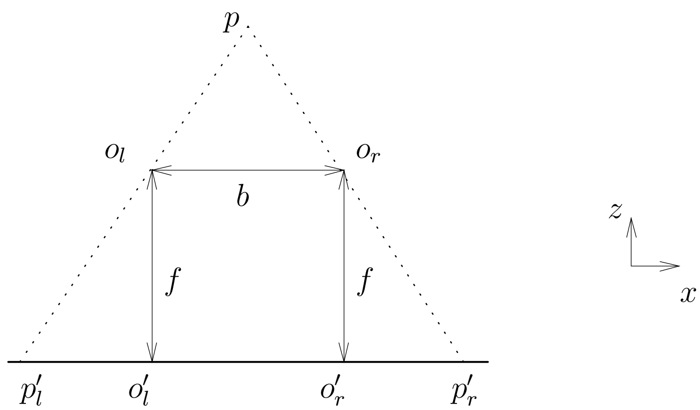


Figure 2: A simple stereo camera geometry.

of the two cameras,

$$\frac{x'_l}{f} = \frac{x + b/2}{z}$$
$$\frac{x'_r}{f} = \frac{x - b/2}{z}$$
$$\frac{y'_l}{f} = \frac{y'_r}{f} = \frac{y}{z}$$

where b is the baseline width and f is the focal length of both cameras. Thus in this simple camera geometry, only the x location of a projected point differs between the left and right images. The y location of a given point in space is the same for both images.

Recall that the disparity is defined as the distance between (x'_l, y'_l) and (x'_r, y'_r), which in this case is just the magnitude of the difference between the x coordinates, $x'_l - x'_r$. From the above equations, we see that

$$\frac{x'_l - x'_r}{f} = \frac{b}{z}.$$

Thus if b and f are known then the depth z of the point p can be computed from the disparity. Note that as z gets infinitely large the disparity goes to zero (as we noted in the introduction, things that are very far away have no disparity). If b and f are unknown then we can compute the *relative* depths of points, but not their absolute distance from the camera (because b and f although unknown are fixed, and thus there is simply a constant factor difference). In other words, for an uncalibrated camera system (unknown f and b) all that we can conclude is disparity is inversely proportional to depth.

The disparity is directly proportional to the focal length, f. Thus a larger focal length camera system will produce bigger disparities for the same distance, z. Disparity is also directly proportional to the baseline width, b. Note that if there is some fixed error in determining disparity, then increasing b and f will reduce the error in the depth computation (because increasing these quantities increases the amount of disparity for a fixed depth difference). The focal length, f is effectively limited for most cameras. The baseline is quite easy to increase, however this results in other problems. As b is increased, the two images become less and less similar to one another (in the worst case two finite size images contain nothing in common). Even when the images contain common subparts, the large disparities make it quite difficult to identify corresponding points p_l and p_r in the two images that both result from the same point p in the world (because the points may be very far apart). This is a *fundamental tradeoff* in stereo imaging systems: a wider baseline provides more accurate depth estimates for fixed errors in disparity, however it also makes the problem of determining a correspondence much more difficult.

A point $p = (x, y, z)$ in the world and the two camera centers define a plane called the *epipolar plane* (alternatively this plane is defined by p and its two images $p_l = (x'_l, y'_l)$ and $p_r = (x'_r, y'_r)$). In other words, for each point in space there is a corresponding epipolar plane defined by that point and the two camera centers (or the two images in the stereo camera system). A given epipolar plane intersects with the left camera plane, L, defining an epipolar line e_l. Analogously the intersection with R defines an epipolar line e_r. These two lines, one in each image, are referred to as a corresponding pair of epipolar lines. Note that in the simple camera model illustrated in Figure 2, the epipolar lines e_l and e_r are both parallel to x-axis, and have the same y-coordinate.

The epipolar lines are important in stereo vision because if the corresponding pairs of epipolar lines in L and R are known, then this constrains the possible locations of corresponding pairs of points in the left and right images. If a point p_l lies on a given epipolar line e_l, then the corresponding point p_r must lie on the corresponding line e_r in the right image (if it occurs at all in the bounded image region of the plane R). For example, in the simple camera geometry $y'_l = y'_r$, and thus the corresponding epipolar lines of the left and right image are those lines with the same y coordinates.

The central computational problem in stereo vision is to determine for each point p_l in the left image, what matching point p_r in the right image corresponds to p_l (that is what pairs of points p_l and p_r are projections of the same point p). Thus knowing the corresponding pairs of epipolar lines in the two images constrains the search for corresponding pairs of points to just a line, rather than the entire image plane. In the case of the simple camera geometry a given point in the left image $p_l = (x'_l, y'_l)$ must have a match on the line $p_r = (x'_r, y'_l)$ (if there is any matching point at all in the right image).

In case of *general* camera geometry, the corresponding epipolar lines in R and L form pencils of lines through the images of the other camera centers (a pencil of lines in the plane is the set of all lines through some given point). That is, all the epipolar lines in L go through projection of right camera center into that image (the image of o_r in L), and analogously all the epipolar lines in R go through the projection of the right camera center into that image. In the case of the simple camera geometry, the right camera center o_r projects to infinity rather than into L (and analogously for o_l and R). Thus the point that all the epipolar lines go through is at infinity — in other words the epipolar lines are parallel. (Note that the image of the left (right) camera center need not actually be visible in the right (left) image, because the image is just a finite portion of R (L).)

The correspondence of epipolar lines in the left and right images must be discovered through some sort of calibration process that relates the coordinate systems of L and R. We will not discuss such camera calibration further, but only note that accurate calibration is a difficult and tedious process. We will generally use cameras that are setup in (approximately) the simple geometry, where corresponding epipolar lines are lines parallel to the x-axis with the same y coordinate.

2.1 Finding Matching Point Pairs

The geometry of stereo vision is the 'easy part', now we must consider the problem of how to identify pairs of image points p'_l and p'_r that correspond to projections of the same scene point p into the left and right images. In order to recover depth (or relative depth in the case where f and b are not known) we must identify such corresponding pairs of points. The first problem in identifying corresponding points is that a given point p'_l need not have a corresponding point p'_r. It may be that p was not imaged inside the bounded camera image, or it may be that p was occluded (hidden from view by some other points) when viewed from R (but not from L). There is nothing to be done in such cases; but we assume that most of the time p is imaged in both L and R.

The second problem in identifying corresponding pairs of points is the fact that there are inherent ambiguities in determining which points match in the left and right images. We can make various assumptions to resolve the ambiguities, such as preferring the match that results in the least change in depth (disparity). Consider a scene that contains two points, and the images of those points a'_l and b'_l in the left image and a'_r and b'_r in the right image. There are two possible physical interpretations of just these images of two points. The situation is illustrated in Figure 3. The two possible interpretations are

1. a_l and a_r are both images of B, and b_l and b_r are both images of A
2. a_l and b_r are both images of C, and b_l and a_r both images of D

None of the other pairings are possible, because they involve pairs of points that lie along the same line.

Ambiguities such as this also arise in human stereo vision. There is a well known optical illusion, called the 'double nail illusion', where two nails that actually are at locations C and D are seen as if they are at A and B. You can approximate this by holding your two index fingers in front of your eyes at about arms length and focusing on them. Don't move your eyes though, because eye movements allow disambiguation of the match (i.e., by moving your eyes slightly you can tell that they are actually behind each other, and not next to each other). You should then see your two fingers as if they are next to one another, even though they are behind one another (note: for some people this is not easy to do). Thus the human stereo vision system prefers the 'flatter' interpretation that the two points are at about the same depth (A and B) rather than the true situation (C and D).

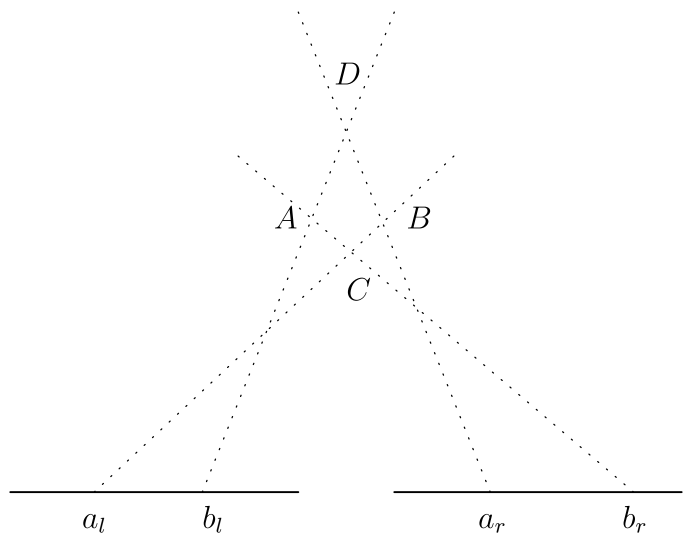


Figure 3: Inherent ambiguities in stereo correspondence.

In general, the problem of resolving such ambiguities involves making some assumptions that allow choosing one interpretation over another. The most common such assumption is that of 'continuity' of disparity (and depth values). That is, it is assumed that disparity (and depth) values vary slowly almost everywhere. Intuitively, this means preferring the flatter interpretation (the one with less change in disparity, and hence depth). For the example in Figure 3, this assumption would result in choosing the first rather than the second of the above two interpretations, because

$$| \, |a_l - a_r| - |b_l - b_r| \, | < | \, |a_l - b_r| - |a_r - b_l| \, |$$

In practice, most stereo matching methods use some sort of 'continuity' constraint in identifying pairs of points p_l and p_r. Thus we have limited the problem of determining corresponding pairs of points from that of comparing every pair of points in L and R to that of comparing every every pair along corresponding epipolar lines, and then to those pairs that preserve continuity of depth values. We will now turn to some simple algorithms for identifying correspondences between points along epipolar lines, and then consider some more involved methods.

For the simplified imaging geometry, the epipolar line pairs are corresponding lines parallel to the x-axis with the same y-coordinate. Thus in a digitized video image, each successive scan line of the left and right image forms a pair of corresponding epipolar lines. The stereo matching problem is therefore that of identifying which portions of each scan line of L correspond to which portions of the corresponding scan line of R. We assume that the disparities will vary slowly almost everywhere, so that in general if the point at x'_l is paired with that at $x'_r = x'_l + d$ (i.e., the disparity is d at x'_l) then $x'_l + 1$ will be paired with $x'_r + 1$. In other words, there will usually be a single a disparity, or shift, d for a set of neighboring pixels of a scan line of L.

One method of identifying the shift between pixels along corresponding epipolar lines is using correlation. Correlation is a standard technique for finding the best shift of two functions with respect to each other, by maximizing the product of the two functions. That is, given $a(x)$ and $b(x)$, the correlation is defined as,

$$a \star b = \int_{-\infty}^{\infty} a(\xi)b(\xi - x)d\xi \ .$$

Note the similarity of correlation to convolution. The value of x that maximizes this product is the best correlation of a with b (for example, the maximum value of $a \star a$ is at $x = 0$).

Another measure that is commonly used to find the best shift of two functions with respect to each other is to minimize some L_p norm of the difference between the functions,

$$\left[\int_{-\infty}^{\infty} |a(\xi) - b(\xi - x)|^p d\xi\right]^{1/p}$$

for integral values of $p \geq 1$. Note, however, that this is a nonlinear operation due to the absolute value and the power p. Thus correlation is used much more commonly.

In order to use correlation for stereo matching, we cannot simply take a and b to be the entire corresponding epipolar lines. There is no reason to believe that there is a single disparity, d, for all the pixels along a given scan line (a scan line may contain some pixels due to objects that are close, and some due to objects that are far). Therefore correlation is usually applied by selecting some fixed-size window (region) of the left scan line, and correlating that window with the right scan line to find the best position. Then the next window of the left scan line is taken, and so on. For example, if a window of ± 3 pixels is used, then each pixel at e_l defines 7 values that can be correlated with e_r to find the best shift, d. This then defines a value of d at each location of e_l.

There are a number of issues with the simple correlation-based stereo matching method just described,

1. *There need to be identifiable points in the two images.* For a uniform intensity surface all matches (all disparities) are equally good, so the correlation has very wide peak.

2. *The two images need to be the same intensity.* Overall differences in intensities in the left and right image cause there to be no good correlation, or even causes mismatches to be found.

3. *The window size must be appropriate.* If the window is too small then many false matches will be found (e.g., just one pixel matches nearly everywhere). If the window is too large then different disparity regions will be combined together, and there will be no single shift (disparity) that matches.

4. *There cannot be significant foreshortening in the images.* Foreshortening is caused by something that is changing depth quickly with respect to the camera, which causes the disparities to change quickly. Even worse is the situation where the two images have different amounts of foreshortening, which 'warps' e_l with respect to e_r.

The first of these issues is problematic for *all* stereo matching methods, although some are more sensitive than others. The second issue is mainly a problem for methods that rely on absolute intensity values in the two images (such as grey-level correlation). The main means of dealing with this problem is to use edges or other binary features instead. We will delay talking about edge-based stereo matching for a bit longer. The third and fourth issues are problems primarily for correlation based stereo matching, thus we now turn to techniques that do not suffer from these latter two problems.

2.2 Marr-Poggio Method

The Marr-Poggio algorithm [6] is a local iterative method for computing disparities from *binary* image pairs, by matching pixels of the left and right images. The method is based on three constraints on stereo matching (recall that the matching of points in one frame to the other is not unique, thus we need some sort of constraints to find the 'best' match). These constraints, or criteria for a good match, are

1. 'continuity' - the disparity of matches varies slowly almost everywhere
2. compatibility - only 'same type' elements match (e.g., in binary images black pixels match with black pixels and white pixels with white pixels)
3. uniqueness - there is almost always a single match for each element
4. epipolar lines - matches must occur along corresponding epipolar lines of L and R

The Marr-Poggio method was developed in the context of Julesz's random dot stereograms — two images of black/white dots that produce the illusion of surface depth. Such a stereo pair is illustrated in Figure 4. The image in this case is a raised square in the center (as indicated by the corresponding disparities also shown). The key underlying assumption of the Marr-Poggio method is that the data are dense — that there are many and frequent changes from white pixels to black pixels in the image. In other words, the method assumes that there are many pairs of pixels on which to compute disparity. It then determines the subset of these possible pixel pairs for which the disparities best meet the above four criteria. Thus the best images for this type of stereo matching (and in fact for most stereo matchers) are highly textured; for example natural scenes, or manmade objects that have been spattered with paint.

In the simple imaging geometry, matching elements in the left and right images lie on corresponding horizontal lines e_l and e_r. Thus for a binary image, the possible matching elements are any pair of black pixels on e_l and e_r, or any pair of white pixels on e_l and e_r.

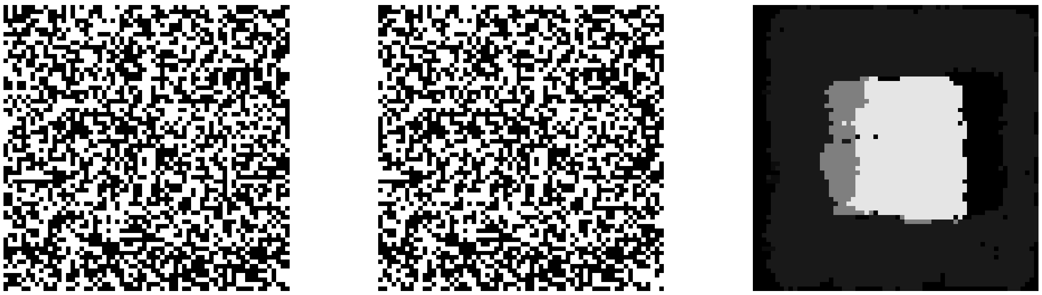


Figure 4: A random dot stereogram and the corresponding disparities computed using the Marr-Poggio method (larger disparity values are displayed as lighter grey levels).

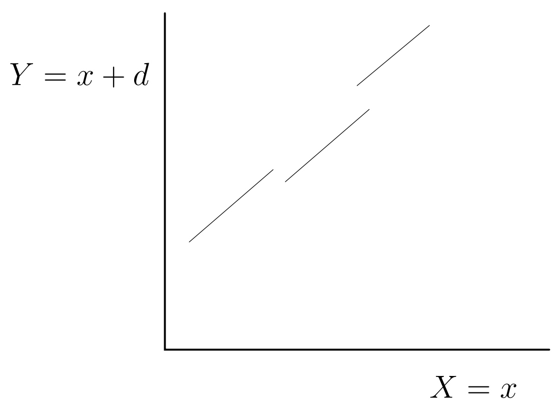


Figure 5: A simple disparity space for corresponding epipolar lines.

We will denote a matching pair of points as $l_y(x)$ and $r_y(x + d)$. Where the y subscript indicates which epipolar lines the two points are on, the left point is at location x on that line, and the right one is at location $x + d$ (the disparity is d). If we form the two dimensional space of $l_y(x) \times r_y(x)$ then we obtain a binary valued 'disparity space', where each point (X, Y) in the space has value 1 when $l_y(X) = r_y(Y)$ and value 0 otherwise. This space is illustrated in Figure 5, where the line segments indicate matching pairs of points in the left and right images (places where (X, Y) has value 1). Note that the disparity is encoded implicitly in this space. For instance, the disparity at $l_y(X)$ is $d = Y - X$.

The 'continuity' constraint means that the disparity lines should be nearly diagonal (slope 1) almost everywhere, as illustrated in Figure 5. A slope 1 line in disparity space indicates constant disparity. The uniqueness constraint means that over some disparity range, $\pm r$, there should only be a single match for a given pixel. In other words, given that $l_y(x_0)$ is paired with $r_y(x_0 + d)$, $l_y(x_0)$ should not match any $r_y(x_0 + d + i)$ and $r_y(x_0 + d)$ should not match any $l_y(x_0 + i)$, $-r \leq i \leq r$.

For intensity images, equality of pixels in the left and right image is not a good way of finding possible matching pairs of pixels, because the image intensities may be

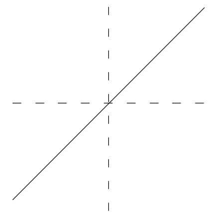


Figure 6: The pattern of excitatory and inhibitory connections for the simple disparity space.

substantially different for the two images. The two options are to produce a binary image from the intensity image (e.g., by edge detection, or the sign bit of $\nabla^2 I$) or to match pairs of points that are approximately equal in the two images (e.g., within some range $|l_y(X) - r_y(Y)| < \varepsilon$).

The 'continuity' and uniqueness constraints can be combined into a local pattern of connections in the disparity space, with excitatory connections on the slope 1 diagonal, and negative connections on the x and y axes, as illustrated in Figure 6. This pattern can be used to iteratively update matches in the disparity space. For example, initially each point of disparity space (X, Y) has value 1 whenever $l_y(X) = r_y(Y)$ and value 0 otherwise. Then the local neighbors are used to change the value of (X, Y) based on the values of its six neighbors using some combination of positive weights on the diagonal, and negative weights on the axes. This is the basic idea of the Marr-Poggio algorithm, although the exact details are a bit more involved.

Unlike the above description, the Marr-Poggio technique does not operate just on single pairs of epipolar lines. Even though matching pairs of points must lie along corresponding epipolar lines in the two images, information is lost by just looking at single pairs of epipolar lines because neighboring lines will generally be highly similar to one another. Thus we want to consider the disparity space for neighboring pairs of corresponding epipolar lines at the same time. This yields a three-dimensional disparity space, which can be thought of as a 'stack' of the two-dimensional spaces that result from just pairing $l_y(x)$ with $r_y(x)$. This space is a bit harder to visualize, however, because the three dimensions are x, y and d, where x and y are positions in the *left* image, and d is the disparity from the left image to the right image. Note that this differs from the two-dimensional space above, where X and Y were the x-coordinate in the left image and the x coordinate in the right image, respectively.

In this three-dimensional disparity space, the constant disparities form a plane parallel to the x-y-plane, rather than a line of slope one as before. A local neighborhood of radius r in this plane is the excitatory set of neighbors of a given point. The inhibitory neighborhood is any conflicting matches along the same epipolar line. In the three-dimensional space this neighborhood is a bit more complicated, and will be described in more detail below. Basically the inhibitory neighborhood is again the set of local conflicting matches (as represented in the simpler case above by dotted lines).

We will denote the three-dimensional disparity space by $D(x, y, d)$ which is a binary-valued space. We will denote points of the left image by $l(x, y)$, and points of the right image by $r(x, y)$, and disparity by d. The corresponding epipolar lines in the left and right images have equal y-coordinates. For a given pair of images, the disparity space is initialized to contain all possible matches of points along corresponding epipolar lines. That is

$$D_0(x, y, d) = \begin{cases} 1 & \text{if } l(x, y) = r(x + d, y) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

So, for example, given two identical images $l(x, y) = r(x, y)$ the initial disparity space would contain all ones in the $d = 0$ plane (i.e., $D(x, y, 0) = 1$). In general, however, for two identical images there will also be other values for which $D(x, y, d)$ is nonzero. In other words, there will be multiple possible disparities for which a given point of the left image matches some point on the corresponding epipolar line of the right image (i.e., other values of d where $l(x, y) = r(x + d, y)$).

In practice, the range of d is limited to some interval $d_{min} \leq d \leq d_{max}$. This limits the range of disparities that can be recovered by the stereo algorithm, but also limits the number of such 'false' matches that occur. Rather than pairing each point of e_l with all possible matching points of e_r, each point is only paired with those in the range $[d_{min}, d_{max}]$.

The basic idea of the algorithm is to iteratively update the disparity space, given the initial possible matches computed as just described. These iterative updates 'enforce' the criteria of uniqueness and 'continuity' described above. After sufficiently many iterations, the space should converge to a good match, according to the criteria. The uniqueness constraint says that if a given location in $D(x, y, d) = 1$ then certain neighbors should be zero. Those neighbors are the set $O(x, y, d)$, the inhibitory neighborhood. The 'continuity' constraints says that if a given location $D(x, y, d) = 1$ then certain neighbors should be one. Those neighbors are $S(x, y, d)$, the excitatory neighborhood.

The iterative update of the disparity space is thus

$$D_{t+1}(x, y, d) = \sigma\left(\sum_{x', y', d' \in S(x,y,d)} D_t(x', y', d') - \varepsilon \sum_{x', y', d' \in O(x,y,d)} D_t(x', y', d') + D_0(x, y, d)\right) \tag{2}$$

where D_0 is initial state described above, ε is the inhibition constant (which is generally 1 or 2), and $\sigma(z)$ is a threshold function,

$$\sigma(z) = \begin{cases} 1 & \text{if } z > \sigma \\ 0 & \text{otherwise} \end{cases}$$

where σ is generally abpit 3. This iterative updating can be implemented by a network of interconnected simple processors in a three-dimensional grid, or by updating arrays that store $D_t(x, y, d)$ and $D_{t+1}(x, y, d)$ on a serial machine.

Now we turn to the definitions of the excitatory and inhibitory neighborhoods $S(x, y, d)$ and $O(x, y, d)$ about a point (x, y, d) in the disparity space. The excitatory neighborhood is simply a 'circular' region about $D(x, y, d)$ with constant d; a circle in the plane $D(x, y, d)$. The radius of this circle is generally about 2 pixels (with an ε of 1 or 2 and a σ of 3).

The inhibitory neighborhood is any conflicting match in the disparity range d_{min} to d_{max}. Recall that $D(x, y, d)$ means that point $l(x, y)$ in the left image matches $r(x+d, y)$ in the right image. Therefore,

1. $l(x, y)$ should not match any other $r(x+d_1, y)$, for $d_{min} \leq d_1 \leq d_{max}$. In other words, ideally $D(x, y, d_1) = 0$ except for $d_1 = d$. Thus the method should penalize $D(x, y, d)$ when these other locations are 1 (they are part of the inhibitory neighborhood).

2. $l(x + d_1, y)$ should not match $r(x + d, y)$ for $d_{min} \leq d_1 \leq d_{max}$. In other words, ideally $D(x + d_1, y, d - d_1) = 0$ except for $d_1 = 0$. Thus these locations are part of the inhibitory neighborhood.

In summary, the Marr-Poggio stereo matching algorithm consists of the following steps:

1. Create the initial state using equation (1).

2. Iterate using the update rule in equation (2). (Reasonable values for the parameters are a radius of 2 pixels for the excitatory region, $\varepsilon = 2.0$ and $\sigma = 3.0$.)

3. Display the results at each iteration using a grey level 'disparity image', where for x, y location in the disparity image the grey level is proportional to the disparity at that x, y location of the left image. Initially there are several different disparities for each location, use the mean, maximum, minimum or median.

4. Stop iterating when the number of locations in disparity space that change state from t to $t + 1$ is 'small'.

Figure 4 illustrates the output of the Marr-Poggio method for the random dot stereogram also shown in the Figure. There are a number of issues with the Marr-Poggio stereo algorithm. First of all, it can be quite sensitive to the settings of the parameters, ε, σ, and the radius of the excitatory region. Secondly, the method actually prefers *constant* disparity, rather than slowly changing disparity, which can be problematic. Thirdly, the method works best when limited to a relatively small range of possible disparities, d_{min} to d_{max} (8 or so pixels for the range). Finally, the method may not converge to any stable solution, although for 'reasonable' settings of the parameters the number of changes from one iteration to the next does generally get small quite quickly (5-10 iterations).

2.3 Area Based Methods

Another class of stereo matching methods is based on aggregating information about disparity estimates across local regions of the image. These methods are often referred to as 'area based' because they smooth or aggregate disparity values over some local area. In contrast with an iterative approach such as Marr-Poggio, these methods require only a single pass to smooth or aggregate the disparities. The averaging together of neighboring values in order to obtain estimates of disparity reflects the implicit assumption that disparity 'changes smoothly' (varies slowly), as was also assumed by the Marr-Poggio method. Unlike the Marr-Poggio method, however, area based methods generally do not exploit the uniqueness constraints that a given pixel in the left image should only be paired with one pixel in the right image and vice versa.

Most area based stereo methods consist of two steps: (i) determining where each pixel in the left image might have 'moved to' in the right image, and (ii) aggregating this information over some local area. The first step of this process is very similar to the initial matching of pixels in the left and right images by the Marr-Poggio method. Those pixels that are of the 'same type' and are within some maximum disparity range are possible pairs in the left and right image (e.g, those points along the proper epipolar line in the right image that are within distance $[d_{min}, d_{max}]$ and have similar intensity values to the point in the left image). This 'moved to' map is generally not a very good estimate of disparity, because the local pairing process is quite error prone. Thus the measures are aggregated over some region in order to obtain more accurate estimates.

One stereo method of this type [9] operates as follows. We assume that the simple camera geometry holds, so that corresponding epipolar lines e_l and e_r have the same y-coordinate. Denote the intensity value at a given location x_l on e_l by $i_l(x)$ (and analogously on e_r by $i_r(x)$). For each location x_l, a range of intensity values are used to find possible matching locations x_r. This range of intensities is determined from the minimum and maximum intensity values in a small neighborhood, $i_l(x-\delta), \ldots, i_l(x+\delta)$. Generally δ is just one pixel. Any pixel of e_r in the disparity range, $x_r = [x_l - d_{min}, x_l + d_{max}]$ for which $i_r(x_r)$ is in the intensity range is then a possible match. In other words, the possible matching pixels must lie within the displacement range specified by d_{min} and d_{max}, and also be within the intensity range determined by x_l and its neighbors.

In general each pixel p_l on e_l will have several possible matching pixels along e_r (resulting in several different disparities). The next step is to determine an aggregate disparity for each pixel of the left image. This is done by finding the modal disparity value for a local neighborhood around p_l. The neighborhood is generally a circle of some radius r (or a square of size $\pm r$ as this is easier to compute). Note that the modal value is the 'most popular' or most frequently occurring value in the region (in the case of a tie, one is chosen arbitrarily). This modal value is then chosen as the disparity at p_l. This results in a single disparity value for each pixel, which can be displayed as a grey level image by mapping each disparity value to a grey value. An image pair and the resulting disparity map for this method are shown in Figure 7.

2.4 Edge Matching Methods

The key problem underlying stereopsis is determining the correspondence between points in the left and right images. The basic methodology is to form likely pairs of points,

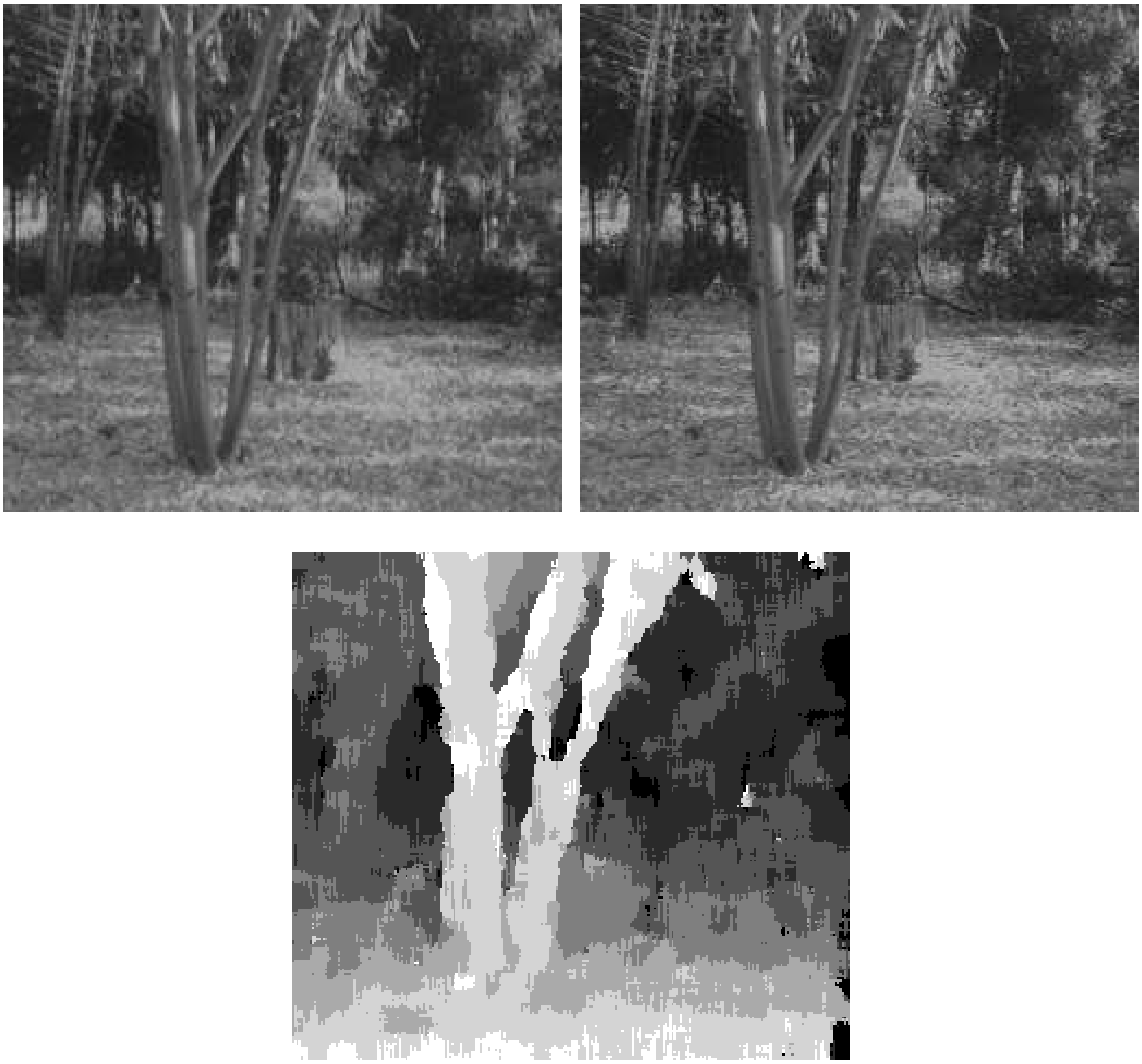


Figure 7: An image from a stereo pair and the output of an area based stereo matcher.

and then apply some sort of constraints (as in Marr-Poggio) or smoothing (as in area based methods) to generate a disparity map. In some ways, the problem of identifying corresponding points in the two images is worse for intensity images than it is for binary images such as edges. Part of the problem is that intensities may change from the left to the right image, making intensity matching difficult. The other part of the problem is that intensity information is dense (there are intensities at every point) making for more ambiguity in matching than when the points are sparser. In an attempt to address these problems, another class of stereo methods take explicit advantage of the fact that they operate on edge detector outputs.

Edge matching stereo methods produce sparse disparity maps, because disparities are only computed at edge locations, not at every pixel as was the case with the Marr-Poggio and area-based methods. Thus it is necessary to interpolate between these sparse values in order to obtain a dense disparity map. In practice, this is not really an issue, because the dense disparity maps produced by other methods are sufficiently noisy that they also need to undergo some sort of surface interpolation or approximation process in order to obtain a good estimate of three-dimensional shape. See [3] for a discussion of such surface interpolation methods.

Grimson's [2] edge based stereo method (which we will refer to as MPG, because it was developed initially with Marr and Poggio) estimates disparity at edge pixels along corresponding epipolar lines. The basic idea is to make use of information about the smoothing parameter, σ, of the edge detection process in order to aid in finding matching edge locations in the two images. Grimson uses a $(\nabla^2 G_\sigma) \otimes I$ (Laplacian of Gaussian) edge detector, which also divides the edges into two classes: positive (+) and negative (-) depending on the direction of the sign change at the zero crossing. That is, going left to right along an epipolar line, if an edge pixel corresponds to going from a negative sign region of $(\nabla^2 G_\sigma) \otimes I$ to a positive sign region, then this is a positive sign change (and the opposite is negative).

Given that we have smoothed the image with a Gaussian of size σ, if there is an edge at some location x, of some type (say +), then it is very low probability that there will be another edge of the same type within some range $x \pm r$, for some distance r. In other words, for a given σ, we can determine the probability of seeing an edge of same type as a function of the distance r from the edge location, x. Grimson has shown that if $r = \sqrt{2}\sigma/2$ then the probability of finding another edge of the same type within the range $\pm r$ is less than 5%. In other words, with probability 95%, this is the only edge of this type within a range of $\pm\sqrt{2}\sigma/2$.

This fact can be used to constrain the problem of finding possible matching edges in the following fashion. If the true disparity d is no larger than $r = \sqrt{2}\sigma/2$, then 95% of the time a given edge at location x_l on the left epipolar line e_l should have exactly one corresponding edge (of the same type) in the range $[x_l - r, x_l + r]$ on the right epipolar line e_r. If, on the other hand, the true disparity d is not in range ($d > r$), then there is a much lower probability that an edge of the correct sign will be found in the range (Grimson has shown that this probability is about 40%). This yields the following means of finding corresponding points in the left and right images. For each edge location in the left image, search the region $[x_l - r, x_l + r]$ of the corresponding right epipolar line for edges of the same type. Count the number of times that exactly one match is found. If exactly one match is found closer to 95% of the time than to 40%, then the σ is large enough to allow for the true disparity. In other words, the known value of σ for the edge detector is used to solve the matching problem based on probabilities of matching edges within a range that depends on σ.

Note that one serious problem with this method as described so far is that a given σ not only rules out finding large disparities (which we can detect by looking at the fraction of matches that are found), but also prevents accurately measuring small disparities. Recall from our study of edge detectors, that larger values of σ produce more and more uncertainty in the locations of edges. Thus, while the MPG method will detect matches at large values of σ, the resulting disparity values will not be very accurate. Thus the full method operates at multiple scales, starting at the coarsest value of σ, and then reducing the size of σ (generally by one half on each iteration) in order to produce more accurate disparity estimates. This results in the following overall structure to the method.

1. Compute the zero crossing of $(\nabla^2 G_\sigma) \otimes I$ for the current scale, σ. Label each zero crossing as + or -.
2. For each zero crossing in L, find the matches (if any) on the corresponding epipolar line of R that are of the same type (+/-) in the range $x_l \pm r$, where $r = \sqrt{2}\sigma/2$.
3. Any region of the image with substantially fewer than 95% of the edges matched indicates that the disparity is not in the the range $\pm r$.
4. Use the disparities in those regions where the disparity is in the proper range as estimates of the true disparity.
5. Repeat with the next smaller value of σ, if any.

The result is a disparity estimate for each location in the left image that contained an edge. These disparities were computed from several different scales of smoothing, with the disparity at each point resulting from the smallest value of σ for which the disparity was in the range $\pm r$.

There are two refinements to this basic MPG edge matching stereo method that make it work better in practice. The first is to us the orientation of the edges to filter possible matches of pixels in the left and right image. That is, in addition to using the +/- type to limit pairings, the orientation of two matching edges must be approximately the same in the two images. The second refinement is to expand the disparity range r to be twice the size ($r = \sqrt{2}\sigma$). This results in a lower threshold for determining whether the disparity is too large for the given σ, but lowers by a factor of two the size σ that is needed for a given disparity range.

3 Motion and Optical Flow

In the case of stereo matching, we were able to make use of the structure of the two-camera system to define corresponding epipolar lines on the two image planes. In the case of a general motion of some object with respect to a camera, corresponding points from one frame to the next can move in any direction. If the motion of the object with respect to the camera is of some restricted form and is known, then a type epipolar constraint can generally be applied. In this section, however, we will consider the general problem of unconstrained and unknown motion of some object with respect to the camera (note that this can involve motion of the object, the camera, or both).

The quantity that we are interested in recovering is the actual motion of some object in three-space, which we call the *motion field*. All that we can observe, however, is changing intensities in the image. This results in several problems. First, the motion of a given point from one image to the next is only the projection of motion in the three-dimensional world. Second, and more importantly, intensity changes do not necessarily allow us to recover corresponding points from one image frame to the next. Thus all that we can observe is the change in intensities, or *optical flow*, and this may be substantially different than the motion field. We saw a manifestation of the fact that it is difficult to recover point correspondences in the case of stereo, but there we were able to apply the epipolar line and continuity constraints to help find a 'best' correspondence. In the case of general motion, this is not a viable approach.

The relation between the image motion field and the object motion field is relatively straightforward, and is governed by the projection equations discussed above. A given object point (x, y, z) projects to a point $x'/f = x/z$ and $y'/f = y/z$. As the object point moves with respect to the camera, the motion in space $v = (dx/dt, dy/dt, dz/dt)$ results in a corresponding motion $v' = (dx'/dt, dy'/dt)$. Given several corresponding points, and several frames of observation, if the velocities of the points in space are constrained in some fashion (for instance they correspond to rigid motion of an object in space), then the projection equations can be inverted. In other words, the three-dimensional object motion field and the three-dimensional shape of the object can be reconstructed from the two-dimensional image motion field, given certain assumptions about the motion. This is known as structure from motion, and will be considered below. Here we address the problem which precedes computing structure from motion: determining the image motion field.

The only measurements that we can make in the image are brightness changes, there is no way to tell what the corresponding points actually are between two images. For example, consider a rotating uniform sphere, in this case there is no optical flow (no change in brightness patterns in the image) but there is motion of the object. On the other hand, consider a shadow moving across a scene. In this case there is optical flow, but there is no motion of the object. Optical flow is all that is accessible, and to the extent that it does correspond to the true motions of image points, it allows us to determine image motion and thus eventually also scene motion.

3.1 Local Differential Methods

Most optical flow methods are based on measuring local intensity changes in the image with respect to time (cf. [5]). If we have an image $I(x, y, t)$ then what does the optical flow mean? It is the vector field that tells us where each point in the the image at time t 'moved to' at time $t + \delta t$. As long as there are no overall changes in brightness of the image, then we expect that

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t),$$

where $u(x, y)$ and $v(x, y)$ are the x and y components of the optical flow vector at (x, y). The Taylor series expansion of this is,

$$I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} + e = I(x, y, t),$$

where e is the second and higher order terms. Cancelling, dividing through by δt, and taking the limit as $\delta t \to 0$ we get

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dx} + \frac{\partial I}{\partial t} = 0$$

(which is actually just the expansion of $\frac{dI}{dt} = 0$, in other words the total derivative of I with respect to time is zero — there are no overall changes in brightness of the image).

Rewriting this equation using $u = \frac{dv}{dt}$, $v = \frac{dy}{dt}$, and $I_x = \partial I/\partial x$, etc., yields what is known as the optical flow constraint equation,

$$I_x u + I_y v + I_t = 0.$$

This equation gives the (linear) relation between the components of the optical flow, u and v, and the image measurements of the derivatives I_x, I_y and I_t. The optical flow equation by itself is not sufficient to determine the optical flow from the derivatives of the image with respect to time. The equation can be re-written in vector form as

$$(I_x, I_y) \cdot (u, v) = -I_t.$$

In other words, the component of the optical flow in the direction of the brightness gradient (I_x, I_y) is

$$\frac{I_t}{\sqrt{I_x^2 + I_y^2}}.$$

We cannot, however, determine the component of the optical flow along the isobrightness contour (normal to the intensity gradient).

This problem, of being able to measure the optical flow only in the direction of the intensity gradient, is known as the 'aperture problem'. The issue is that measuring the flow locally is like looking at the image of an edge through a small aperture window. The only component of the edge motion that can be determined is that in the direction normal to the edge (in the gradient direction). We can't tell how much the edge is moving in the direction along the edge (see Figure 8a). If, however, there is some identifiable point in the image, rather than just an edge, the complete motion of this point can be determined (see Figure 8b).

There are a number of ways to provide additional constraint so that the flow problem can be solved:

1. Assume that there is a single body moving rigidly; this is powerful but generally overly restrictive.

2. Assume that the motion field varies smoothly in most parts of the image (allows for nonrigid deformations). This is the method of choice when staying within the local differential framework.

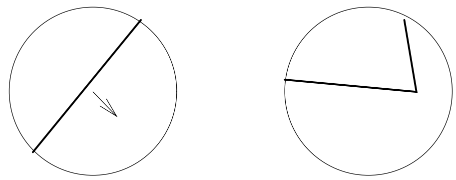


Figure 8: The aperture problem: a) only the motion normal to the edge can be determined, b) and edge with two distinct normals allows the full (planar) motion to be determined.

3. Track the motion of edge contours, rather than using local intensity differences with respect to time. As long as the contour is not a line segment, it is possible to extract the complete 2D motion (because there are at least two distinct normals to the edge, which thus span the plane).

We will not consider the first of these options any further. The second method requires minimizing some measure of the departure from smoothness of the motion field. One such measure is given by,

$$e_s = \int\int (u_x^2 + u_y^2 + v_x^2 + v_y^2) dx dy$$

the sum of the squared magnitude of the velocity field gradient. A slowly changing velocity field will have low gradient magnitude, and a rapidly changing one will have have gradient magnitude.

Moreover, from above we know that the error in optical flow constraint equation should be small (ideally zero). Thus we also want to minimize the sum of squares of the optical constraint equation,

$$e_c = \int\int (I_x u + I_y v + It)^2 dx dy \ .$$

Combining these two terms, yields the problem of minimizing

$$E(u, v) = e_s + \lambda e_c$$

where λ determines the relative importance of the smoothness criterion and the optical flow criterion. If $\lambda = 0$, then a vector field with zero gradient magnitude is produced, but it has nothing to do with the optical flow. As $\lambda \to \infty$ the flow field becomes less and less smooth, and more and more like the measured data. This weighting term, λ, is generally referred to as a *regularization* parameter. λ should be chosen proportional to the signal to noise ration of the data (i.e., λ should be small if the signal noisy, and large if it is not). $E(u, v)$ is a quadratic in the unknowns u and v, and thus the minimization can be solved using standard techniques (see, for example, the Numerical Recipes book).

One source of problems with such a method is that the smoothing process interpolates across motion boundaries. For example, given an image of a sphere rotating in front of some background, the sharp boundaries of the motion field at the edge of the sphere

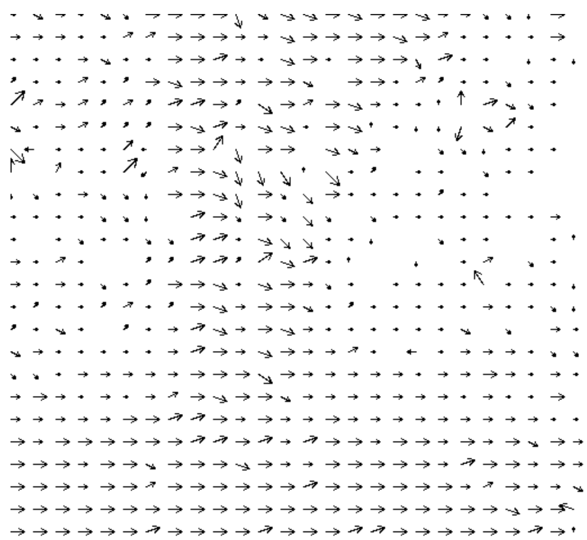


Figure 9: The optical flow for the two tree images above.

will tend to get smoothed out by minimization. One approach to this is to attempt to incorporate segmentation of motion boundaries into the iterative minimization process. In practice this is difficult to do.

A different approach to computing the optical flow is to identify 'popular' displacement vectors in a local region. This is the natural extension of the area based matching methods for stereo vision considered above. Recall that for stereo this computation consisted of two steps: (i) finding possible matches in the right image for each pixel of the left image, and (ii) determining the disparity at each point based on the modal value of the possible matches over some local neighborhood. This produced a 'moves-to' map, specifying where each pixel in one image moved to in the other image.

In the case of general motion, rather than stereo, the main difference is that the possible motion of a pixel from one frame to the next are not constrained to lie along an epipolar line. Thus, for each pixel (x, y) in the image, I_t, the neighboring pixels of I_t are used to determine a range of possible intensity values $[i_{min}, i_{max}]$. Then for each pixel of I_{t+1} that is within some distance d_{max} of (x, y), if that pixel is also in the intensity range it is taken as a possible match for (x, y) in I_t. This results in zero or more possible matching pixels for each (x, y) location of I_t (and for each matching pixel a vector specifying where (x, y) moved to).

The consensus step then picks the modal displacement vector over some region of radius r about (x, y). That is, each point (x', y') within distance r of (x, y) has zero or more matching points computed as above. Each such matching point (x_m, y_m) defines the displacement vector $(x' - x_m, y' - y_m)$. All of these vectors are histogrammed, and the most commonly occurring one (the modal value) is taken as the displacement vector for location (x, y). In practice, this type of mode filtering generally results in a relatively good motion field. The main problem, as with the smoothing method above, is that the motion boundaries become inaccurate because different motions are aggregated together.

The flow field for the two tree images from Figure 7 is shown in Figure 9. The flow field is drawn as a 'needle diagram' with the needle length proportional to magnitude, and the orientation pointing in the direction of the flow.

3.2 Edge Based Methods

Hildreth [4] notes that image motion is best measured at sharp intensity changes, such as intensity edges, as this is where the motion can be computed reliably. In areas of slow intensity change there is a high degree of ambiguity in matching pixels from one frame to the next, and thus she argues that it is better not to compute the change there at all. Her approach is to compute flow along connected contours of zero crossings in $(\nabla^2 G) \otimes I$.

Consider a curve c parameterized by arclength, s. Let $v(s)$ measure the velocity of c at a given location s along the curve. For example, a line segment rotating about its center will have zero velocity at $s = l/2$ where l is the length of the segment, and will have equal but opposite velocities at $s = 0$ and $s = l$. We can only measure the component of $v(s)$ that is normal to the curve at $C(s)$, as we saw above with the aperture effect. The goal is then to compute the 'true' velocities along the contour $v(s)$ given the measured velocities $\hat{v}(s)$ which reflect only the component of $v(s)$ normal to $c(s)$.

This is an underconstrained problem, for which various minimization techniques can be used to provide a unique solution. The particular measure of of variation that Hildreth minimizes is

$$\Theta = \int \left| \frac{\partial \hat{v}}{\partial s} \right|^2 ds$$

the squared magnitude of the change in velocity along the curve. This minimization guarantees a unique solution as long as the measured velocity is known in at least two distinct directions. (i.e., the motion of a straight line cannot be determined). This minimization problem can be solved using standard numerical methods, once the observed velocities have been determined (which can be done by simple local differences in the image). Note that it is assumed that the curve $c(s)$ does not deform over time, it just moves in the image plane.

The motion field computed by doing this minimization is not always the same as the true motion of the curve. In the case that the curve simply translates, the computed motion is identical to the real motion. If the curve is a polygon that translates and rotates, then the computed and real motion are also the same. However in the case of a rotating ellipse, the computed velocity field is quite different from the true motion. There is a reduced rotational component and an added radial component to the computed motion (as compared with the real motion). This has the interesting property that it predicts to some degree the human perception of such curves rotating. Those of you who still have an old-fashioned turntable at home can try putting a cutout ellipse on the turntable and rotating it. The ellipse will not not appear rigid, rather it seems to deform continuously (an added radial component to the motion). A similar effect happens for a rotating spiral. Thus there is evidence that Hildreth's method of minimizing the variation in the observed motion along the curve is similar to the type of processing done in human perception of some kinds of motion.

4 Tracking

The motion field computed from the optical flow gives a local 'moves-to' map, that specifies for each point of image I_t a corresponding point image I_{t+1}. This map is not necessarily a bijection — a given point of I_{t+1} may have many points of I_t that map to it, or no points that map to it. For many problems, we would like to *track* each point across a sequence of frames, $I_1, \ldots, I_k$. For example, in order to reconstruct the three-dimensional shape of an object from a sequence of frames, we need to know for each point p in the world, what are its images in each I_t. This involves finding a bijection from each frame I_t to I_{t+1}. (Note: In some situations, such as points that really move to the same location, there will not be a bijection, but we would like to recover such a mapping when possible).

One way to compute a bijection is to find the minimum weight total matching in a bipartite graph, where possible matching points in the two frames are connected by edges of the graph. Assume that points $p_1, \ldots, p_n$ are in I_t, and $q_1, \ldots, q_n$ are in I_{t+1}, and that each p_i, q_j are connected by an edge when p_i could have moved to q_j. These edges could be simply the optical flow vectors (which specify a q_j for each p_i) in which case each p_i will have just one edge incident on it. The edges could instead be all possible consistent pairs of points in the two frames (i.e., what was used as input to the optical flow computation). The edges form a bipartite graph, because there is no edge connecting any two points p_i and p_k, or q_j and q_l.

A *matching* in a bipartite graph is a subset of the edges such that each vertex has exactly one edge incident on it (i.e., it defines for each p_i a unique q_j and vice versa, which is a bijection). If the edges of the graph have weights associated with them then a *best* matching is a matching that minimizes the weights on the edges. In the case of the tracking problem, the weights could be something such as the distance from p_i to p_j (i.e., points that move less are better candidates for matches). The weighted bipartite matching problem is also called the *assignment problem* and can be solved in time $O(n^3)$, where n is the number of nodes in the bipartite graph.

Of course in practice, it may be that some points of I_t have no counterpart in I_{t+1} (or vice versa). Thus, rather than finding a matching, which pairs each point of one set with exactly one point of the other, we can find the minimum weight *cover* of a bipartite graph. A cover is a subset of the edges such that each vertex has at least one edge incident on it. Minimizing the total weight prefers subsets that contain fewer edges, so we get 'as close to a matching as possible'. This is the way in which Ullman [8] formulated the problem of tracking feature points from one frame to the next: finding the minimum weighted cover of a bipartite graph for each successive frame, where the weights are some function of the measured points in the successive frames (e.g., distance between points).

In more detail, Ullman sets up a linear programming problem corresponding to the minimum weighted cover. Let

$$x_{ij} = \begin{cases} 1 & \text{if } p_i \text{ paired with } q_j \\ 0 & \text{otherwise} \end{cases}$$

where $p_1, \ldots, p_n$ are the elements of frame t and $q_1, \ldots, q_m$ are the elements of frame $t+1$. By definition, a cover has at least one edge incident on each node. Thus the problem can be cast as a minimization, where the quantity

$$\sum_{i,j} x_{ij} \rho_{ij}$$

is minimized with respect to a set of constraints. The function ρ_{ij} is some measure of the 'cost' of pairing p_i with q_j (e.g., using the distance from p_i to q_j). This minimization is subject to a set of constraints that force the answer to be a covering (each p_i paired with at least one q_j and vice versa), which corresponds to,

$$\begin{aligned} \sum_j x_{ij} \geq 1 & \qquad 1 \leq i \leq n \\ \sum_i x_{ij} \geq 1 & \qquad 1 \leq j \leq m \\ x_{ij} \geq 0 & \qquad 1 \leq i \leq n, 1 \leq j \leq m \ . \end{aligned}$$

This is an integer programming problem in mn variables. That is, it is a restricted form of linear programming problem that must have integral solutions (where all x_{ij} are integers). There are various ways to solve such a linear programming problem, including simple parallel networks, and gradient descent minimization techniques (a discussion of which is beyond the scope of this course).

Using this formulation of the tracking problem, Ullman chose to minimize the distance between corresponding feature points (i.e., chose $\rho_{ij} = |p_i - q_j|$). This also has the effect of preferring paths that do not cross (due to the triangle inequality). Other cost functions that have been used include ones that constrain the change in the velocity vector for each point in successive pairs of frames. This amounts to the assumption that a point does not change its direction/magnitude of motion suddenly. These functions can be used to set up similar minimization problems. The major limitation with such approaches in general is that they are quite computationally expensive, particularly when the number of points in the two frames is moderately large.

Another approach to tracking is to use the constraint that nearby points should in general have similar motions. This is true when a given object is imaged as many points, but is not true when there are many moving objects in the image each of which has very few corresponding points (e.g., an 'asteroids' video game). This approach of using the 'smoothness' of the motion field is taken by [9], who define an object at time t to be a subset of the image pixels at that time. Thus, $O_t \subseteq I_t$ and $O_{t+1} \subseteq I_{t+1}$. The motion field, $M_t : I + t \rightarrow I_{t+1}$, is used to construct O_{t+1} from O_t, but since it is not a bijection each point of I_{t+1} may have several (or no) points mapped to it. Thus in constructing O_{t+1}

from O_t some additional processing needs to be done other than simply using M_t to map each point of O_t to a new point.

If we again denote the points of I_t by p_i and the points of I_{t+1} by q_j, then the set of points that are mapped to q_j by M_t is $M^{-1}(q_j) = \{p_i | M_t(p_i) = q_j\}$. If M_t were a bijection, this would always be a singleton set, but since it is not it may be the null set (no point is mapped to q_j) or may have several elements (all of which are mapped to q_j). In order to construct O_{t+1}, [9] use the following local aggregation rule: $q_j \in O_{t+1}$ when the the majority of the points of $M^{-1}(q_j) \in O_t$, moreover it is *undefined* whether $q_j \in O_{t+1}$ if there is a tie (no majority), or if $M^{-1}(q_j) = \emptyset$. This handles the case of multiple points mapping to q_j, by saying that $q_j \in O_{t+1}$ exactly when the majority of the points that map to it are in O_t.

Those points of I_{t+1} where O_{t+1} is undefined can then be filled in by using the neighboring points of I_{t+1} at which O_{t+1} is in fact defined. This is done in a 'second pass', after the above rule is applied. This second rule says that for each q_j where O_{t+1} is undefined, use the value of the majority of its neighbors in I_{t+1} for which O_{t+1} is defined. This has the effect of filling in 'holes' caused when no point of I_t is mapped to a given q_j (or by ties). The resulting object, O_{t+1} still has the drawback that uncertainty in M_t at motion boundaries tends to make the boundary of O_{t+1} very noisy.

In order to adjust the boundaries of O_{t+1}, [9] apply the following rule at motion boundaries. They define an adjusted object O'_{t+1}, where $q_j \in O'_{t+1}$ when

a) there is no motion boundary in the neighborhood around q_j, and $q_j \in O_{t+1}$, or

b) there is a motion boundary in the neighborhood around q_j, and the majority of the neighbors of q_j that have the same motion as q_j are also in O_{t+1}.

A motion boundary is identified where there M_{t+1} takes on substantially different values in some local neighborhood. Note that in areas where the magnitude of M_{t+1} is very small, the orientations may change quite a lot. Thus for, example, one reasonable definition of presence of a motion boundary is areas where the magnitude of the difference between the vectors is large. This adjustment of the model to produce O'_{t+1} helps ensure that the boundaries of the object are consistent with the boundaries of the motion field. This tracking method has been implemented on a Connection Machine, and used to track objects in real time. The major limitation of the method, is that it cannot track objects for which the motions or the 2D shape change are large from one frame to the next. This is an issue for all local differential tracking methods.

5 Structure from Motion

Determining the three-dimensional structure, or shape, of a point set from a sequence of two-dimensional views is a problem that has been studied extensively (starting with Ullman's 1979 book [8]). The input to the structure from motion process is generally a set of P feature points, where the two-dimensional image coordinates of each point are measured in each of F frames. In other words, it is assumed that the feature points have been *tracked* through the successive frames, so that the correspondence between points in each frame is known. Precisely what is to be recovered from these $2FP$ observed values depends on the exact phrasing of the structure from motion problem. We will consider the problem where the scene is assumed to be static, and the camera is moving. The task is thus to recover the three-dimensional structure of the scene points and the trajectory of the camera. Note that we are really concerned with the relative configuration of the scene with respect to the camera, and thus if the scene is moving (rigidly) this can be seen as associating the world coordinate system with some chosen points of the scene.

This formulation of the problem makes it easy to see what we are trying to recover, namely the $3P$ values that specify the locations of the P points in space and the kF values that specify the camera position at each frame as a function of some world coordinate system (where k depends on the particular camera model). Much of the early work on structure from motion was concerned with identifying the minimum number of points and frames such that it is possible to determine the $3P + kF$ values needed to recover the positions of the points and locations of the camera. We will not go into the derivation of these results here, because in practice it is necessary to have many more than the theoretical minimum number of observations in order to obtain a *reliable* solution. Ullman showed that under orthographic projection 3 views of 4 points are necessary and sufficient to recover three-dimensional shape. In other words, no fewer than three views, nor fewer than four corresponding points in each view are enough, and more than that lead to an over-constrained problem. Under perspective projection Longuet-Higgins showed that 2 views of 8 points are sufficient.

We will focus on the case of orthographic projection (all the rays are perpendicular to the image plane) and applications in which there are substantially more than the minimum number of observations required to obtain a solution. The key issue is to allow for measurement noise by using the redundancy of the observations to cancel out the (unbiased) noise in the locations of the image points and thereby recover the true structure and motion. There are number of possible techniques for doing this, we consider the work by Tomasi [7] which has been particularly effective.

As we noted at the beginning of this section, structure from motion methods assume that each of the P points in the scene is tracked through a sequence of F frames. We will make the further assumption that all P points appear in each of the F frames, in order to simplify the presentation. Small numbers of missing and mis-tracked points can be handled through suitable modifications of the basic method. Denote by (u'_{fp}, v'_{fp}) the image coordinates of the p-th point in the f-th frame, $1 \leq p \leq P$, $1 \leq f \leq F$. We first build a measurement matrix that contains these $2FP$ observed values. This measurement matrix uses normalized coordinates of the points, rather than the actual observed values (u'_{fp}, v'_{fp}). The normalization re-expresses each point in terms of the centroid of the points visible in that frame. In effect this places the origin of each image frame at the centroid of the points in that frame. Given that all of the points are visible in each frame, this is a valid way of "aligning the origins" of each image.

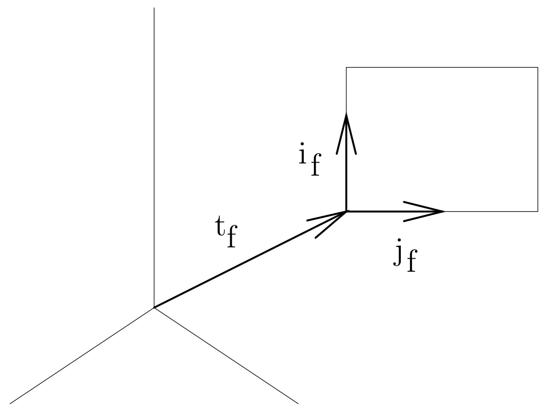


Figure 10: The location of the image plane for frame f in the world coordinate system.

More specifically, the measurement matrix is defined as

$$W = \begin{bmatrix} U \\ V \end{bmatrix}$$

where U and V are the $f \times p$ matrices $U = [u_{fp}]$ and $V = [v_{fp}]$ and where (u_{fp}, v_{fp}) are the normalized coordinates of each observed point (u'_{fp}, v'_{fp}). That is, $u_{fp} = u'_{fp} - \bar{u}_f$ and $v_{fp} = v'_{fp} - \bar{v}_f$ where $\bar{u}_f$ and $\bar{v}_f$ are the centroids of the u and v coordinates in frame f,

$$\bar{u}_f = \frac{1}{P} \sum_{p=1}^{P} u'_{fp}$$

and

$$\bar{v}_f = \frac{1}{P} \sum_{p=1}^{P} v'_{fp} \ .$$

In the case that there is no error in the observed data, the measurement matrix W is highly rank deficient. When there is error, if the magnitude of the errors is relatively small then the approximate rank of the matrix, as measured using the singular value decomposition (SVD) will still be small. Tomasi's structure from motion method uses this observation to decompose W into the $3P$ point coordinates and the kF camera frame coordinate frame values, by using the SVD to factor the matrix W into those components.

First we note that a point $s'_p = (x'_p, y'_p, z'_p)$ in the world projects to the image points

$$\begin{aligned} u'_{fp} &= i_f \cdot (s'_p - t_f) \\ v'_{fp} &= j_f \cdot (s'_p - t_f) \end{aligned}$$

where i_f, j_f are the unit vectors defining the orientation of the image plane in the world coordinate system, and t_f is the vector from the origin of the world coordinate system to the origin of the image plane (see Figure 10).

Now since centroids are preserved under projection (i.e., the centroid of the projection of a point set is the projection of the centroid of that set) we can use the centroid normalized coordinates

$$\begin{aligned} u_{fp} &= i_f \cdot s_p \\ v_{fp} &= j_f \cdot s_p \end{aligned}$$

where $s_p = s'_p - \bar{s}$ and $\bar{s}$ is the centroid of the points

$$\bar{s} = \frac{1}{P} \sum_{p=1}^{P} s'_p \, .$$

Given this, if there is no error in the observed image data, then the measurement matrix W can be factored into two parts $W = MS$, where

$$M = \begin{bmatrix} i_1^T \\ \vdots \\ i_F^T \\ j_1^T \\ \vdots \\ j_F^T \end{bmatrix}$$

represents the camera motion (the orientation of the image plane for each of the F frames) and

$$S = [s_1 \ldots s_P]$$

is the shape matrix (the locations of the points in space, with their centroid as the origin). In other words the observed image data, as represented by W, is the product of two parts, the camera motion and the three-dimensional positions of the points. Now the task is to recover M and S given W (and despite the fact that there is some error in the observed data).

Since M is a $2F \times 3$ matrix and S is $3 \times P$ matrix, when the measurements are exact $W = MS$ must be of rank at most 3. Of course in the case of even small errors in the locations of the image points, the rank of W will not be 3. However, we can use the singular value decomposition to determine the "approximate rank" of W, which we expect to be 3.

Recall that if A is an $m \times n$ matrix then the singular value decomposition (SVD) of A is

$$A = U \Sigma V^T$$

where

$$U = [u_1, \ldots, u_m]$$

is an $m \times n$ orthogonal matrix,

$$V = [v_1, \ldots, v_n]$$

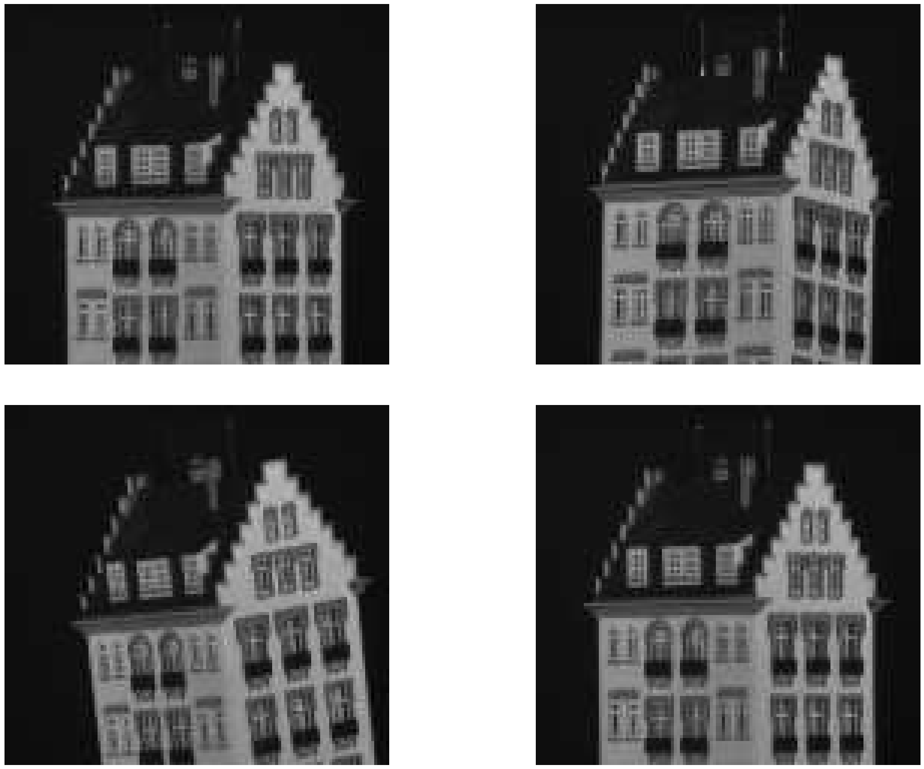


Figure 11: Four frames from a 150 frame sequence of camera moving with respect to a model house.

is an $n \times n$ orthogonal matrix, and Σ is a diagonal matrix with the values $\sigma_1, \ldots, \sigma_p$ along the diagonal, where each $\sigma_i \geq \sigma_{i+1}$ and $p = \min(m, n)$.

The number of nonzero singular values, σ_i, corresponds to the rank of the matrix A. That is, if $\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_p = 0$ then the rank of A is r. Moreover, the vectors $v_{r+1}, \ldots, v_n$ span the nullspace of A and the vectors $u_1, \ldots, u_r$ span the range of A. Most importantly, this decomposition provides a natural way of measuring the approximate rank of a matrix. In effect those singular values that are nearly zero can be taken to be zero — that is the approximate rank of a matrix is the number of singular values that are not nearly zero. The exact reasons for why this a reasonable thing to do are beyond the scope of this course (cf. [1]).

The SVD of the measurement matrix W is thus $W = L\Sigma R$ where L is $2F \times P$, Σ is a $P \times P$ diagonal matrix of singular values and R is $P \times P$. In principle the first three singular values (i.e., the upper-leftmost three diagonal elements of Σ) should be nonzero and the remaining ones zero, because we expect the matrix to be of rank 3. In practice measurement error will cause the remaining singular values to be nonzero, but they should still be small. As noted above, the information of interest corresponds to the three greatest singular values, so the best approximation to the "ideal" measurement matrix is

$$\hat{W} = L'\Sigma' R'$$

where L' is the $2F \times 3$ matrix corresponding to the first three columns of L, Σ' is the 3×3 diagonal matrix corresponding to the upper left part of Σ, and R' is $3 \times P$ matrix corresponding to the first three rows of R.

If we define $\hat{M} = L'(\Sigma')^{1/2}$ and $\hat{S} = (\Sigma')^{1/2}R'$, then $\hat{W} = \hat{M}\hat{S}$. The question is now whether $\hat{M}$ and $\hat{S}$ are good estimates of the ideal underlying motion M and shape S that we want to factor the measurements W into. In other words, we initially noted that W can be factored into the product MS of the camera motion and the three-dimensional shape of the point set. The question is whether this factorization of $\hat{W}$ into $\hat{M}$ and $\hat{S}$ is such that $\hat{M}$ is a good estimate of M and $\hat{S}$ is a good estimate of S.

One problem is that this is not a unique factorization of W, any linear transformation A of $\hat{M}$ and $\hat{S}$ yields a valid result, because $(\hat{M}A)(A^{-1}\hat{S}) = \hat{M}(AA^{-1})\hat{S} = \hat{M}\hat{S} = \hat{W}$. Thus we have only recovered the motion and shape up to an arbitrary linear transformation. We can solve for the "correct" motion and shape transformations by noting that the true motion matrix M is composed of unit vectors, and the first F rows are orthogonal to the remaining rows (because each row i and $i+F$ correspond to the two orthogonal unit vectors defining the image plane at frame i). This allows one to solve for A up to a rotational ambiguity. This remaining ambiguity corresponds to the initial position of the camera with respect to the world. In other words, the overall orientation of the points is only known relative to the initial orientation of the camera.

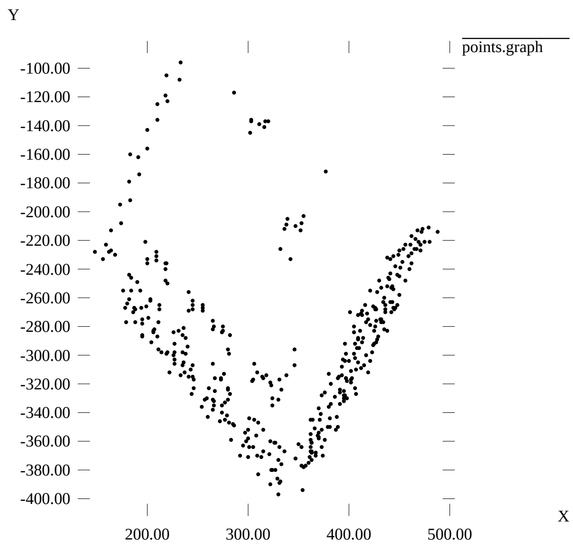
Thus, in summary the method is a follows,

1. Form the measurement matrix W from the centroid-normalized observed values.

2. Compute the SVD of W and keep just the part corresponding to the three largest singular values, $\hat{W} = L'\Sigma'R'$.

3. Solve for the A such that $M = L'(\Sigma')^{1/2}A$ is composed of unit vectors and the first F rows are orthogonal to the remaining rows. Let $S = A^{-1}(\Sigma')^{1/2}R'$. S is the shape and M is the motion.

Figure 11 shows four frames from a motion sequence in which a camera moved with respect to a model house. Feature points were extracted from each frame and their motion was tracked across successive frames (e.g., using a tracking method such as the ones discussed in the previous section). The two-dimensional locations of the tracked feature points were then processed according to the above method to recover their three-dimensional positions. Figure 12 shows a top view of the reconstructed three-dimensional positions of the features, next to a view of the house from this orientation. (Note that the house was never seen from a top view in the motion sequence.)



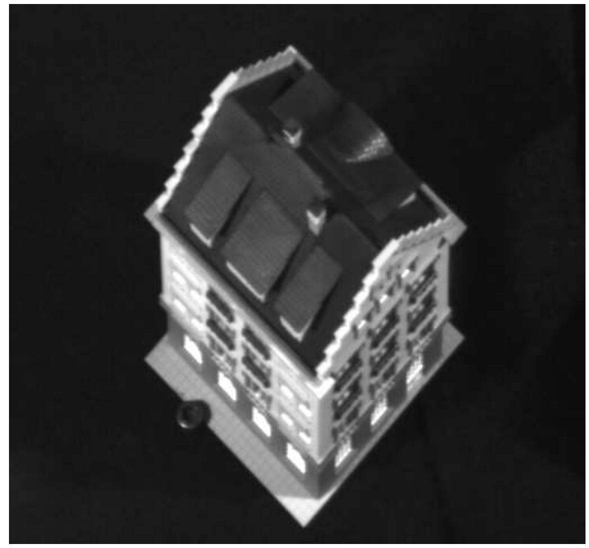


Figure 12: A top view of the reconstructed shape of selected feature points on the house, and an image of the house from that view.

References

[1] Golub, G.H. and VanLoan, C.F. *Matrix Computations*, Johns Hopkins University Press, 2nd ed., 1989.

[2] Grimson, W.E.L. "Computational Experiments with a Feature Based Stereo Algorithm", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 7, pp. 17-34, 1985.

[3] Grimson, W.E.L. *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambrdige Mass., 1981.

[4] Hildreth, E. *The Measurement of Visual Motion*, MIT Press, Cambridge Mass., 1983.

[5] Horn, B.K.P. *Robot Vision*, MIT Press, Cambridge Mass., 1986.

[6] Marr, D. and Poggio T. "Cooperative Computation of Stereo Disparity", *Science*, Vol. 194, No. 4262, pp. 283-287, October 1976.

[7] Tomasi, C. and Kanade, T. "Shape and Motion from Image Streams Under Orthography: A Factorization Method", *Intl. J. of Computer Vision*, vol. 9, no. 2, pp. 137–154.

[8] Ullman, S. *The Interpretation of Visual Motion*, MIT Press, Cambridge Mass., 1979.

[9] Woodfill, J. and Zabih R.D. "An Algorithm for Real-Time Tracking of Non-Rigid Objects", Proceedings of the American Association for Aritificial Intelligence, 1991.