Feature-Based Mosaic Construction
Proposal

Kristin M. Branson
Department of Computer Science and Engineering
University of California, San Diego
kbranson@cs.ucsd.edu

1 Introduction

In applications such as space-based remote imaging and surveillance, data is collected in the form of multiple small images that each contain a piece of a desired scene. For example, consider film taken by a satellite orbiting a planet. Each frame will show a small piece of the planet surface, but what is useful is an image of a large fraction of the surface. The goal of mosaicing is to stitch together a single image from these smaller, overlapping image frames.

Each 2-D point in each image is the projection of a 3-D point in the world. A mosaic of all the images will contain the projection of every 3-D point viewable in any image in a common coordinate system. For any pair of images, if the camera’s intrinsic parameters and motion between images is known, then image mosaicing is trivial. The intrinsic parameters define the 3 x 3 viewing matrix $V$, which is the transformation from the 3-D location of a point relative to the position of the camera to the 2-D location of that point in an image. The inverted viewing matrix, $V^{-1}$, can therefore be used to transform the 2-D image location of a point in the first image, $x^1$, to the 3-D world location of that point relative to the camera’s current position, $X^1$. $X^1 = V^{-1}x^1$. In this equation, the world coordinates are inhomogeneous, three-element coordinates and the image coordinates are homogeneous, three-element coordinates. The motion of the camera between images defines the transformation from the world coordinate system relative to the camera’s first position to the world coordinate system relative to the camera’s second position. The projection of point $X^1$ in the world coordinate system relative to the camera’s second position is $X^2 = RX^1 + t$, where $R$ is the camera’s rotation and $t$ is the camera’s translation. Because the same camera takes all the images, the viewing matrix also define the transformation from $X^2$, the 3-D location of a point relative to the second position of the camera, to $x^2$, the 2-D location of that point in the second image. Thus, the projection of each point in the first image to the coordinate system of the second image is

$$x^2 = V(RV^{-1}x^1 + t).$$

(1)

In general, the camera’s intrinsic parameters and motion are both unknown, thus the transformation from one image’s coordinate system to another image’s coordinate system is unknown. It is impossible to estimate $V$, $R$, and $t$ without ambiguity, given only the images. It is extremely difficult to estimate these parameters, even up to an ambiguity, because $V$ is common to all pairs of images. Thus the estimation cannot be broken up into parts, and must be performed over all image pairs at once.
However, if we assume that the camera does not translate between images, the problem is greatly simplified. Consider the previous equations when the translation vector \( \mathbf{t} = 0 \). Equation 1 reduces to

\[
\mathbf{x}^2 = \mathbf{VRV}^{-1}\mathbf{x}^1 = \mathbf{Hx}^1.
\]

Thus, there is a 3 x 3 matrix \( \mathbf{H} \) that projects points in the first image into the coordinate system of the second image (Szeliski, 1994). Any 3 x 3 matrix that projects coordinates from one image to another is a planar homography. More intuitively, if the camera does not translate, then there is no motion parallax. Parallax is the displacement of points in one image that are coincident in the second. It is observable when you switch from just your left eye to just your right eye and is the key to depth perception. Without parallax, no effects of points in a scene being at different depths are observable. Thus it can be assumed that all points in the scene lie on a plane. Because the transformation between one projection of a plane and a second projection of the same plane is a homography, the transformation between one frame and any other frame is a homography.

Assuming that all the images are related to each other by homographies, the first task is to estimate these homographies given only the images. Approaches to homography estimation all estimate point correspondences between images and the image transformations between one image and the other images. When the image transformations are known, all images can be transformed to be in the same frame, and fused together to create a single panoramic image.

In this project, I applied the feature-based described in (Torr and Zisserman, 1999) approach to homography estimation. The feature-based methods, in contrast with direct methods, only calculate point correspondences in areas where it is likely that an accurate correspondence can be found. In the next section, I explain the approach I used for homography estimation. In Section 3, I discuss methods for blending the projected images together. In Section 4, I present my results. In section 5, I discuss my conclusions and future work.

2 Feature-Based Methods to Homography Estimation

Feature-based methods find correspondences and the corresponding homographies for select sets of points in each image where there is a chance of estimating the correct correspondence. As an illustration of points in an image that would not be useful in correspondence calculations, consider points where the gradient of the pixel intensities in all or one direction is small. If the gradient in all directions is small, then the point is inside a flat region, and there is no way of telling one point in this flat region from another point in this flat region. If the gradient in one direction is small, then the point is on a line, and there is no telling one point on this line from another point on this line. Thus, points where the gradient has these properties are not useful for correspondence calculation, so feature-based methods ignore them and concentrate on calculating correspondences between interest points where the gradients in both directions are large. In addition, this makes the search space smaller. Thus, the first step in feature-based methods is to calculate a set of interest points in each image. This is done using a corner detection algorithm, which finds interest points where the gradient is high in both directions.

The Harris corner detection algorithm finds the points for which the criterion \( \text{det}(\mathbf{M}) - k\text{tr}(\mathbf{M}) \) is large. \( k \) is a constant that is usually set to 0.04 and \( \mathbf{M} \) is a matrix of gradients of intensity,

\[
\mathbf{M} = \begin{pmatrix}
I_x^2 & I_xI_y \\
I_xI_y & I_y^2
\end{pmatrix},
\]

where \( I \) is the pixel intensity at every point, \( I_x \) is the gradient of the pixel intensity in the \( x \) direction, and \( I_y \) is the gradient in the \( y \) direction. For this criterion to be largest, the
Figure 1: Example interest points detected using the Harris corner detector. The standard deviation of the Gaussian used to smooth the image gradients is 4 and the threshold determining which points are corners and which are not is 10,000.

gradient in both directions must be large. The extension of Harris corners to color images is very similar; the pixel intensity gradients are replaced with the summed intensity gradients of each color channel:

$$M_{\text{color}} = \begin{pmatrix}
R_x^2 + G_x^2 + B_x^2 & R_xR_y + G_xG_y + B_xB_y & R_y^2 + G_y^2 + B_y^2 \\
R_xG_x + G_xG_y + B_xB_y & R_x^2 + G_x^2 + B_x^2 & R_y^2 + G_y^2 + B_y^2 \\
R_xB_x + G_xG_y + B_xB_y & R_x^2 + G_x^2 + B_x^2 & R_y^2 + G_y^2 + B_y^2
\end{pmatrix},$$

where \( R \) is the red pixel intensity, \( G \) is the green pixel intensity, and \( B \) is the blue pixel intensity. Figure 1 shows corners detected using the color Harris corner detection (Montserratos et al., 1998).

The goal of mosaicing is to project all the images into a common coordinate system, for example the coordinate system of one of the images. It is not desirable to calculate the homography between this one image and all other images, because the amount of movement between images in these cases may be large, and the images may not overlap much. Instead, the homographies between pairs of sequential images is calculated, as the motion between sequential frames will in general be small and there will be a lot of overlap. The homography between any two frames can be calculated by concatenating homographies, for example, \( H_{j,k} = H_{j,j+1} \ldots H_{k-1,k} \) for \( j < k \). Thus the homography between every image and the basis image can be calculated in this way.

The next steps in these algorithms are done for each sequential pair of images. Once a set of interest points in two sequential images has been calculated, correspondences between these points that fit a homography transformation need to be calculated. Because the homography determines the point correspondences, and vice-versa, the ideal algorithm would find both of these at the same time. However, there are only methods for finding the correspondences between interest points and finding the optimal homography from a set of approximate correspondences separately. After an initial estimate for the point correspondences are found, feature-based methods iteratively estimate the homography from the approximate point correspondences and the point correspondences from the approximate homography to converge on an optimal solution for both.

2.1 Normalized Cross-Correlation

The initial correspondences between interest points are chosen based on cross-correlation. For any interest point in one image, interest points around that location in the second image
are searched for a correspondence. This search space could be the entire second image, if the amount of motion between images is unlimited, or a small section if the motion is known to be small. For every interest point in the first image, a small patch around that interest point is compared to a small patch around every interest point in the second image, and the correspondence with the highest correlation is kept, if it is above a threshold. This search is repeated for every point in the second image. If a point belongs to multiple correspondences, the best correspondence is chosen and the others are removed.

Cross correlation works especially well for image mosaicing because there is no observable parallax. Thus, the distance between two points in any frame is constant relative to the other between point distances in that image. Overlapping surfaces, for example a chair in front of a tree, will overlap the same amount in all images, thus corners detected at surface junctions will appear the same in all images and cross correlation will achieve excellent results. Examples of the correspondences chosen by cross-correlation are shown in Figure 2.

2.2 Robust Estimation through RANSAC

Once the initial correspondences between interest points in two sequential images are found, RANdom SAmple Consensus (RANSAC) is run to prune incorrect correspondences (outliers) using the knowledge that the images are related by a homography. As a homography has eight degrees of freedom (there are nine entries in the 3 x 3 homography matrix, and this matrix can only be determined up to a scale factor) and each point correspondence provides two constraints, four point correspondences determine a homography. So in general a larger set of correspondences will not be fit by a homography. Given an initial estimate of point correspondences, RANSAC finds the homography that is supported by the largest subset of these correspondences. A homography is supported by a correspondence if the distance between the homography model and the correspondence is below a threshold. This set of correspondences are the inliers, and are deemed correct correspondences by RANSAC. The rest of the correspondences are the outliers, and are deemed incorrect and thrown away. The homography found by RANSAC ignores outliers and is thus robust. What does the distance between a homography $H$ and a correspondence mean? We can think of any set of $n$ correspondences $\{x_i, x'_i\}$ as a $N = 4n$ length measurement vector $X$ in $\mathbb{R}^N$, where $X$ is made up of the inhomogeneous coordinates of the matched points in
both images, i.e., \( \mathbf{X} = (x_1, y_1, x_1', y_1', \ldots, x_n, y_n, x_n', y_n') \). Let \( S \) be the set of all points in \( \mathbb{R}^N \) (i.e., the set of all sets of correspondences) that satisfy the homography \( \mathbf{H} \). The distance between \( \mathbf{X} \) and \( \mathbf{H} \) is the perpendicular distance between \( S \) and \( \mathbf{X} \). That is, it is the minimum distance between \( \mathbf{X} \) and any \( \hat{\mathbf{X}} \) such that \( \hat{\mathbf{X}} \in S \). This is equivalent to the minimum cost of the reprojection cost function for correspondences \( \{x_i, x_i'\} \) and homography \( \mathbf{H} \) described later.

Calculating the reprojection error requires nonlinear optimization to estimate the correspondences that satisfy the homography that are closest to the correspondences. As this calculation must be made on every iteration, an approximation is made. The symmetric transfer error is a much simpler function that provides an adequate estimate. Thus the distance between a homography \( \mathbf{H} \) and a correspondence is

\[
d(\mathbf{H}, (x, x')) = d(\mathbf{Hx}, x')^2 + d(\mathbf{H}^{-1}x', x)^2.
\]

RANSAC is an iterative algorithm. On each iteration, four point correspondences are randomly selected and the homography defined by these correspondences is computed. As there is a homography that fits these four correspondences, the solution is easily computed using a linear algorithm. This solution is found by noticing that the cross product \( x' \times Hx = \mathbf{mathbf{m}} \times \mathbf{mathbf{H}} \mathbf{x} = \mathbf{mathbf{m}} \mathbf{H} \mathbf{x} = \mathbf{mathbf{m}} \mathbf{H} \mathbf{x} \mathbf{h} = \mathbf{0} \) since \( \mathbf{c} \) is any scale factor. Thus from this point correspondence, a 3 x 3 matrix \( \mathbf{A}_i \) can be determined such that \( \mathbf{A}_i \mathbf{h} = \mathbf{0} \), \( \mathbf{A}_i \) depends only on the correspondence, and \( \mathbf{h} \) is the homography matrix strung into a vector. From the \( \mathbf{A}_i \), we can construct a 12 x 3 matrix \( \mathbf{A} \) s.t. \( \mathbf{Ah} = \mathbf{0} \) and \( \mathbf{A} \) constrains \( \mathbf{h} \) completely. As only two of the rows of each \( \mathbf{A}_i \) are linearly independent, the rank of \( \mathbf{A} \) is 8 (for nondegenerate cases). \( \mathbf{h} \) is thus in the null space of \( \mathbf{A} \), and is the eigenvector corresponding the 0 eigenvalue of \( \mathbf{A} \).

After the homography is calculated from the randomly selected point correspondences, the number of inliers for \( \mathbf{H} \) is calculated. That is, the distance between each point correspondence and \( \mathbf{H} \) is calculated, as described above. These steps are iterated until some termination condition is met. The randomly selected homography with the largest number of inliers is returned by RANSAC.

There are two parameters that must be set for the RANSAC algorithm, the distance threshold to separate inliers from outliers and the number of iterations to be run. The distance threshold can be set so that the probability that a point is an inlier is some value \( \alpha \) if it is assumed that the measurement error follows a Gaussian distribution with mean 0 and variance \( \sigma^2 \). As corners were detected with only pixel accuracy, I assumed a standard deviation of 0.5 pixels. The number of iterations required can be set so that with high probability \( p \) (e.g., \( p = .99 \)), at least one of the random samples of four correspondences does not contain outliers, given the fraction of points that are outliers, \( \varepsilon \). Since in general this fraction is not known, the number of samples can be determined adaptively by starting with a very high estimate of \( \varepsilon \) and then dropping \( \varepsilon \) as larger numbers of inliers are found (Hartley and Zisserman, 2000).

The inlying correspondences found by RANSAC are shown in Figure 3. The distance between the projection of the interest points in image 2 into the coordinate system of image 1 and the measured location of the corresponding interest points in image 2 is shown in Figure 4(a).

### 2.3 Reprojection Error for Over-Determined Systems

The homography returned by RANSAC may not be the best fit for the inlying correspondences. For more accurate results, the homography and exact point locations are reestimated using a more complex error function, the reprojection error:

\[
\sum_i d(x_i, \hat{x}_i)^2 + d(x_i', \hat{x}_i')^2,
\]
Figure 3: Example of inliers found at three iterations of RANSAC. The lines in each image connect corresponding pixels.

Figure 4: (a) The red x’s are the measured locations of the inliers found by RANSAC, and the yellow +’s are the projection of the corresponding points in the other image. Ideally, the distance between the points would be 0. (b) The red x’s are the measured locations of the inliers found by RANSAC, and the yellow x’s are the improved estimate of the projection of the corresponding points in the other image found by optimizing reprojection error.
subject to the constraint $\hat{x}_i = H\hat{x}_i$. The correspondences and homographies that minimize this cost function are the Maximum Likelihood Estimates, if the measurement error follows a Gaussian distribution with mean 0 and variance $\sigma^2$. Minimizing the reprojection error is difficult because it is also nonlinear and requires determining the values of $2n + 8$ parameters, where $n$ is the number of correspondences. This criterion is optimized using the Levenberg-Marquardt optimization algorithm, and requires a good initial guess of point locations and the homography.

The homography that minimizes the algebraic error is used to initialize the Levenberg-Marquardt optimization. This error criterion is $\|Ah\|$, where $A$ and $h$ are discussed in the previous subsection. This measure has the advantage that it is linear and thus $h$ is easily solved for (the eigenvector corresponding to the smallest eigenvalue of $A$). This measure is not used alone because it is not geometrically or statistically meaningful, and is sometimes counterintuitive (Hartley and Zisserman, 2000).

Figure 4(b) shows the reduced estimate error found by minimizing reprojection error.

2.4 Bundles Adjustment

The previous discussion describes how to find the homography relating each pair of images in the sequence. The final mosaic will contain all the points in all the images. However, the homography relating the last image and the first image computed sequentially as the product of all the homographies directly computed is most likely incorrect. This is particularly important if the sequence wraps around on itself. Global realignment must be performed to resolve this issue. This is done through an extension of the reprojection error to all the image pairs,

$$d(H_1, \ldots, H_m, (\hat{x}_0, \hat{x}_1), \ldots, (\hat{x}_0, \hat{x}_m)) = \sum_{i=j}^m d(\hat{x}_0, H_j \hat{x}_j)^2.$$

In this equation, frame 0 is the coordinate system of the mosaic and $H_j$ is the homography that projects image $j$ into frame 0. This algorithm must estimate $2n + 9m$ parameters, where $n$ is the total number of interest points detected in all images. This criterion is optimized using the Levenberg-Marquardt algorithm. The initial estimates of the point locations in frames 1 through $m$ and the homographies are those optimized by the reprojection error earlier. The corresponding locations of these points in the base frame are estimated from these values (Hartley and Zisserman, 2000).

3 Mosaic Construction

3.1 Image Projection

Once the homographies relating the images have been calculated, the images must be projected into a common coordinate system. The simplest approach is to project all the images into the frame of one of the images, say project all the images into the coordinate system of the first frame. This technique works well if the rotation around any axis is small enough, significantly less than 180 degrees. The larger the rotation from the base frame, the larger and more skewed the projected image will be.

If most of the rotation is around one axis, this can be resolved by projecting the images onto a cylinder. For example, suppose that the x location of the images varies a lot but the y location varies little. The x coordinates of every image are converted to polar coordinates where the distance to the scene (the radius of the cylinder) is constant. For each pair of images, the second image is projected into the Cartesian coordinate system of the first image. Next, the x coordinates of both images are projected into polar coordinates in which the radius is a constant $z$, the distance from the camera to the scene, $x_{cyl} = k \tan(x_{cart}/z)$. $k$ is a
constant to normalize the width of the scene to the height of the scene. This is repeated for each pair of images, translating the coordinates of each pair. Because there are no artifacts of depth, \( z \) can be assumed to be constant for all images. \( z \) is the focal length of the camera. As I did not know this, I estimated it from the approximate viewing angle. If \( m \) images are estimated to translate a total of 360 degrees in the x direction and the overlap between each image is \( p \), then the average viewing angle of each image is \( \theta = \frac{360}{1 - p} \). The number of pixels in each image is known, thus an estimate for \( z \) is \( \frac{n_{\text{cols}}}{2\tan(\theta/2)} \). If this approximation must be refined, this can be done by seeing how much this estimate over/undershoots 360 degrees.

3.2 Blending

Once the images are projected into a common coordinate system, the pixel intensities must be blended together into one image. A weighted average of pixel intensities is used to create the final mosaic. To reduce effects of image edges, the pixels closest to the center of the image are weighted most heavily. I used a cubic weight of distance to the closest edge.

4 Results

75 images encircling the Price Center at UCSD were taken using a digital camera. Because of memory constraints, only 25 images could be projected into the frame of the first image taken. The results are shown in Figure 5. The images were also projected into a cylindrical coordinate system. The actual image projection is time consuming (it uses Matlab’s griddata to project the image into cylindrical coordinates), thus only preliminary results are shown in Figure 6.

References


Figure 5: Projection of 25 and 19 images of the Price Center into the Cartesian coordinate system of one of the images. The lower image shows the individual image boundaries in red.
Figure 6: Projection of 5 images of the Price Center onto a cylinder. A mosaic of more images is being created.