Computer Vision CS543/ECE549
Homework 4

Due Date: 26 April 2011

1 Assorted Short Answer

For the questions below, explain using an illustration and additional words, as needed.

1.1 Stitching (12%)

Suppose you have two pictures from a camera that has rotated but not translated:

a. Can you estimate depth from the two images? Why or why not? (3 pts)

b. Can you stitch the images into a larger image? If so, what artifacts might you see? (3 pts)

Suppose you have two pictures from a camera that has translated and rotated:

c. Can you estimate depth from the two images? Why or why not? (3 pts)

d. Can you stitch the images into a larger image? If so, what artifacts might you see? (3 pts)

1.2 Stereopsis (10%)

Suppose you have two images taken by a pair of cameras at different positions with known translation, and you want to find the point in the 2nd image (x') that corresponds to some point (x) in the first image. Why is it that you can search for the point x' along a single line, rather than searching everywhere in the image? Explain using both an illustration and words. (10 pts)

2 Tracking and Structure from Motion

For this problem, you will recover a 3D pointcloud from the image sequence `hotel.seq0.png` ... `hotel.seq50.png`. Since this is a multipart problem, we

have included precomputed intermediate results in the supplemental material in case you're unable to complete any portion.

To receive credit, you must submit your code for this problem in a .zip file. Please also include pseudocode in your report. Furthermore, do not use existing keypoint detectors, trackers, or structure from motion code, such as found in OpenCV.

2.1 Keypoint Selection (20%)

For the first frame, use the second moment matrix to locate strong corners to use as keypoints. These points will be tracked throughout the sequence.

You can either use the Harris criteria (1), *or* the Shi-Tomasi/Kanade-Tomasi criteria (2). Here M is the second moment matrix, λ_1, λ_2 are the eigenvalues of M, and τ is the threshold for selecting keypoints:

$$\det(M) - \alpha \cdot \text{trace}(M) \geq \tau \tag{1}$$

$$\min(\lambda_1, \lambda_2) \geq \tau \tag{2}$$

If using the Harris criteria, it is good to choose $\alpha \in [0.01, 0.06]$. Choose τ so that edges and noisy patches are ignored. Do local non-maxima suppression over a 5x5 window centered at each point. This should give several hundred good points to track.

Required output:

1. Display the first frame of the sequence overlaid with the detected keypoints. Ensure that they are clearly visible (try `plot(..., 'g.', 'linewidth',3)`).

Suggested Structure:

Write this as a function such as `[keyXs, keyYs] = getKeypoints(im, tau);` Be sure to smooth the gradients when constructing the second moment matrix.

Useful functions:

`imfilter.m`

References:

C. Harris and M. Stephens. *A Combined Corner and Edge Detector.* 1988

J. Shi and C. Tomasi. *Good Features to Track.* 1993

2.2 Tracking (30%)

Apply the Kanade-Lucas-Tomasi tracking procedure to track the keypoints found in part 2.1. For each keypoint k, compute the expected translation from $(x, y) \rightarrow (x', y')$:

$$I(x', y', t+1) = I(x, y, t) \tag{3}$$

This can be done by iteratively applying (6): Given the i^{th} estimate of (x'_i, y'_i), we want to update our estimate $(x'_{i+1}, y'_{i+1}) = (x'_i, y'_i) + (u, v)$. Here, W is a 15x15 pixel window surrounding the keypoint, I_x, I_y are the x, y gradients of image $I(x, y, t)$, computed at each element of W at time t. $I_t = I(x', y', t+1) - I(x, y, t)$ is the "temporal" gradient.

$$(x'_0, y'_0) = (x, y)$$
$$I_t = I(x'_i, y'_i, t+1) - I(x, y, t)$$

$$\begin{bmatrix} \sum_W I_x I_x & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_W I_x I_t \\ \sum_W I_y I_t \end{bmatrix} \tag{4}$$

$$(x'_{i+1}, y'_{i+1}) = (x'_i, y'_i) + (u, v)$$

This should be applied iteratively, that is, begin with $(x'_0, y'_0)^T = (x, y)^T$, which is needed to compute I_t. Use this I_t to estimate $(u, v)^T$, which can in turn be used to compute $(x'_1, y'_1) = (x'_0, y'_0) + (u, v)$, and so on. Note that $(x', y')^T$ (and $(x, y)^T$) need not be integral, so you will need to interpolate $I(x', y', t+1)$ ($I_x, I_y, ...$, etc.) at these non-integral values.

Some keypoints will move out of the image frame over the course of the sequence. Discard any track if the predicted translation falls outside the image frame.

Required Output:

1. For 20 random keypoints, draw the 2D path over the sequence of frames. That is, plot the progression of image coordinates for each of the 20 keypoints. Plot each of the paths on the same figure, overlaid on the first frame of the sequence.

2. On top of the first frame, plot the points which have moved out of frame at some point along the sequence.

Useful functions:

`interp2` - For computing I_x, I_y and $I(x', y', t+1)$ when x, y, u, v are not integers.

`meshgrid` - For computing the indices for `interp2`

Suggested Structure:

`[newXs newYs] = predictTranslationAll(startXs, startYs, im0, im1);` - Compute new X,Y locations for all starting locations. Precompute gradients Ix,Iy here, then compute translation for each keypoint independently:

`[newX newY] = predictTranslation(startX, startY, Ix, Iy, im0, im1);`
- For a single X,Y location, use the gradients `Ix, Iy`, and images `im0, im1` to compute the new location. Here it may be necessary to interpolate `Ix,Iy,im0,im1` if the corresponding locations are not integral.

References:

Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features.* 1992

2.3 Affine Structure from Motion (28%)

Now, use the discovered tracks found in part 2.2 as input for the affine structure from motion procedure described in *Shape and Motion from Image Streams under Orthography: a Factorization Method* 1992 by Tomasi and Kanade. See section 3.4 of this paper for an overview of the algorithm.

Note that there should be a sufficient number of tracks that persist throughout the sequence to perform the factorization on a dense matrix. There is no need to fill in missing data for this problem.

To eliminate the affine ambiguity (i.e. apply the metric constraints) by discovering QQ^T (eq. 16 of Tomasi and Kanade), let $L = QQ^T$ and solve for L using least squares. Note that $\hat{\mathbf{i}}_f, \hat{\mathbf{j}}_f$ refer to rows of $\hat{R}$. Finally, compute the Cholesky decomposition of $L = QQ^T$ to recover the appropriate Q.

Required Output:

1. Plot the predicted 3D locations of the tracked points for 3 different viewpoints. Choose the viewpoints so that the 3D structure is clearly visible.
2. Plot the predicted 3D path of the cameras. The camera position for each frame is given by the cross product $\hat{\mathbf{k}}_f = \hat{\mathbf{i}}_f \times \hat{\mathbf{j}}_f$. For consistent results, normalize all $\hat{\mathbf{k}}_f$ to be the same length (i.e. unit vectors). Give 3 plots, one for each dimension of $\hat{\mathbf{k}}_f$.

Useful functions:

`svd`, `chol`

`plot3` - for plotting 3D points

References:

Tomasi and Kanade. *Shape and Motion from Image Streams under Orthography: a Factorization Method.* 1992

2.4 Extra Problems

Any of the following can be done for extra credit. Points from up to two problems (up to 30 pts) will be awarded:

Shi-Tomasi Affine Verification (15 pts) Use the Shi-Tomasi approach to infer the affine transformation between each keypoint from the first frame and frames $F = [10, 20, 30, 40, 50]$. Plot the points whose appearance drifts too far over time. Also plot the unrectified and affine-rectified patches for several points for each frame in F.

Missing Track Completion (15 pts) Some keypoints will fall out of frame, or come into frame throughout the sequence. Use the matrix factorization to fill in the missing data and visualize the predicted positions of points that aren't visible in a particular frame.

Optical Flow (15 pts) Implement the optical flow approach from lecture to estimate the translation at every point. Use convolution to compute sums over W efficiently.

Coarse to Fine Tracking (15 pts) Implement the coarse-to-fine tracking procedure. To test the accuracy on large translations, tracks the keypoints by only using frames $F = [0, 10, 20, 30, 4050]$. Compare this to if you run original tracker over frames F.