

Final Exam

Assigned: 11/21/02

Due: 12/05/02 at 2:30pm

Problem 1 Line Fitting through Segmentation (Matlab)

- a) Write a Matlab function to generate noisy line segment data with outliers, similar to that shown in Figure 15.9/15.10. You should presume Gaussian distributed noise (in y) and uniformly distributed outlier noise. Your function should have the following syntax:

```
function [X,Y] = create_line(a,b,x1,x2,var,perc_out,range_out)
```

The output parameters $[X, Y]$ are samples of the line $y = ax + b$ between $x=x_1$ and $x=x_2$. The input parameter `var` is the variance of the Gaussian distributed noise, `perc_out` is the percentage of outliers and `range_out` is the outlier range.

For each implementation (b), (c), (d) and (e), below,

- (i) Plot cases where your implementation succeeds;
- (ii) Plot cases where your implementation fails and discuss why it fails;
- (iii) Discuss the effect of each input parameter on the algorithm;
- (iv) Submit your files and print a copy of your code.

- b) Implement a simple least-squares fit of the line. Your function should have the following syntax:

```
function [a, b] = fitline(X, Y)
```

- c) Implement a Hough transform accumulator array, and a simple maximum peak finder. Test this for different granularities of parameter sampling on single-line datasets with varying amounts of normal and outlier noise. Your function should have the following syntax:

```
function [a,b,M] = hough(X, Y, r_min, r_max, r_step, theta_step)
```

where $[r_{\min}, r_{\max}]$ is the range of the line distance from the origin, `r_step` is the incremental step between radius samples and `theta_step` is the incremental step between angle samples. The mask vector `M` is set to 1 when the point is an inlier and 0 when an outlier.

- d) Implement a RANSAC approach (Algorithm 15.4) using the least squares distance metric described in section 15.2.1. Test it on single-line datasets with varying amounts of normal and outlier noise. Your function should have the following syntax:

```
function [a, b, M] = ransac(X, Y, n, max_it, thresh, min_inliers)
```

where `n` is the number of points sub-sampled to fit a line, `max_it` is the maximum number of iterations, `thresh` is the threshold used to identify inliers and `min_in` is the minimum number of inliers.

- e) Implement an EM approach to estimate a fixed number of noisy lines (Algorithm 16.4) and add an outlier model (section 16.4.2). The simplest way to do this is to consider a mixture model with the last mixture component is not a line, but a process that generates outlier points with uniform probability. Your function should have the following syntax:

```
function [a, b, W] = em(X, Y, perc_out, range_out, W_initial)
```

where `perc_out` models frequency with which outliers occur, `range_out` is the range of the uniform outlier process and the vector `w_initial` is the initial weight values. The output parameter `w` is the final weight vector. Don't forget to define a convergence criterion.

Test your implementation on single- and multiple-line datasets with varying amounts of normal and outlier noise.

- f) Outline a possible graph theoretic approach to segmenting this data using a normalized cut criterion.

Problem 2 Line Fitting through Filtering (Matlab)

In this problem, we view the line fitting task of Problem 1 as an online filtering problem. The x coordinate of each point on the line will be treated as the (discrete) time variable. We will then estimate the sequence of y coordinates based on noisy measurements generated as in Problem 1(a). For both of the implemented algorithms, present your results as in Problem 1, (i)-(iv).

- a) Implement the Kalman filter for tracking linear dynamic systems based on observations corrupted by Gaussian noise. Assume the line is generated by a constant velocity state space model. Your filtering function should have the following syntax:

```
function xHat = kalman(Y,D,SigmaState,M,SigmaObs,Sigma0)
```

Here, Y is a vector of scalar observations, $(D, \text{SigmaState}, M, \text{SigmaObs})$ are the standard state space model parameters, and Sigma0 is the covariance of the (zero mean) initial state. The output `xHat` should be a matrix where each column gives the mean estimate of the hidden state at a different point in time. Choose values for the system noise parameters that are appropriate to the line fitting problem. In particular, you should have a large initial state variance, but a fairly small process noise variance.

Test your Kalman filter implementation on measurements containing only the assumed Gaussian noise, and also on data sets with large numbers of outliers.

- b) Particle filters do not share the Kalman filter's restriction to Gaussian observations. Implement the standard particle filtering algorithm (*Forsyth & Ponce*, Extra Chapter 2, Alg. 2.5) for tracking a linear dynamical system based on a sequence of scalar, non-Gaussian observations. Your filtering function should have the following syntax:

```
function xHat = pfilter(y,D,SigmaState,M,SigmaObs,Sigma0,
                      N,perc_out,range_out)
```

The first six parameters are the same as for the Kalman filter, N is the number of particles, and $(\text{perc_out}, \text{range_out})$ are the parameters of the outlier process. Your particle filter implementation should resample at every iteration, and use the appropriately weighted average of particle locations as its best estimate of the line position.

Test the particle filter on the same measurement sequences used for the Kalman filter. Perform five independent runs of the particle filter for each of three different particle set sizes $N = 20, 100, 500$, and plot the estimated line positions.

Problem 3 Steerable Filters

Steerable pyramids received their name because the oriented filters at any one spatial scale form a "steerable basis set". They are all identical copies of the same filter, rotated to different orientations. One can synthesize that filter rotated to any orientation by taking an appropriate linear combination of the basis filters. This lets one analyze oriented filter response not just at a discrete set of orientations, but over the continuous range of possible orientations.

Let $F^\theta(x, y)$ be a filter kernel, $F(x, y)$, rotated through an angle, θ . For now, we ignore sampling issues and work in a continuous (x, y) space. $F(x, y)$ is a steerable filter requiring M basis filters if we can write

$$F^\theta(x, y) = \sum_{i=1}^M k_i(\theta) F^{\theta_i}(x, y)$$

Here, $F^{\theta_i}(x, y)$ are the basis filters and $k_i(\theta)$ are called the interpolation functions.

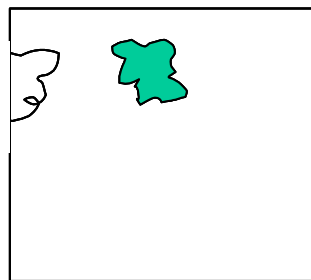
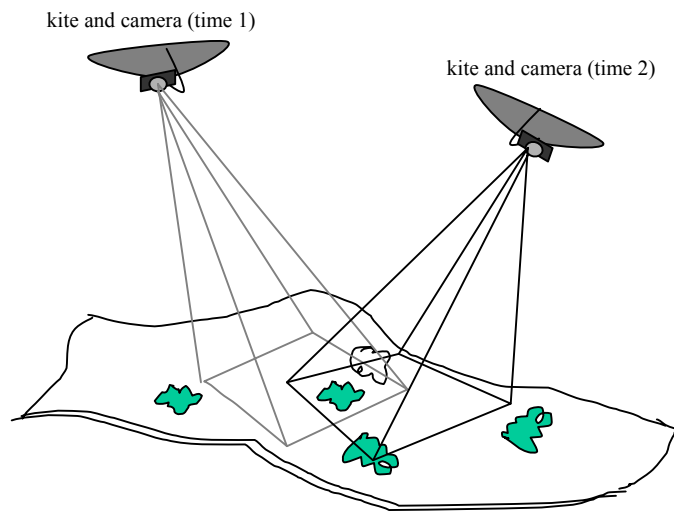
- The simplest possible steerable filter is the directional derivative of a radially symmetric function $G(r)$, where $r = \sqrt{x^2 + y^2}$. Let $G(r) = e^{-(x^2+y^2)}$ be a Gaussian function. Show that $\partial G / \partial x$ is a steerable filter. Give a set of basis filters and interpolation functions using the smallest possible number of terms M .
- Show that the second derivative of a Gaussian $\partial^2 G / \partial x^2$ is also a steerable filter. Give a set of basis filters and interpolation functions using the smallest possible number of terms M .
- Describe how sampling of these continuous functions at discrete locations on the pixel lattice affects steerability.

Note: In the general case, the basis filters don't need to be identical copies of the filter to be synthesized, nor do the steered filters need to be derivatives of radially symmetric functions.

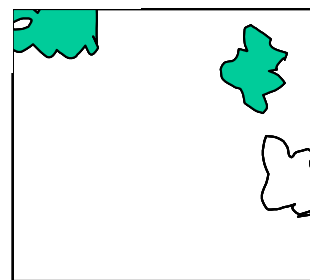
Problem 4 Mosaics of kite aerial photographs

You attach a camera to a kite, send it up, and take many pictures of the ground, each from a different position, orientation, and elevation of the camera. Then, you want to make a composite image, overlaying the photographs into a single image. (See <http://www.ai.mit.edu/people/wtf/kite.html>; for example <http://www.ai.mit.edu/people/wtf/kitePics/NVkiteCamFWSSSQuarterSize.jpg> or <http://www.ai.mit.edu/people/wtf/mvWeb/mvKite/quarter.jpg>.)

This problem examines under what conditions the overlaid images can be made to fit together exactly. (See the figures for a visual description of the problem.)



photograph from camera at time 1



photograph from camera at time 2

We have two images taken of the same piece of land and we want to know under what combinations of

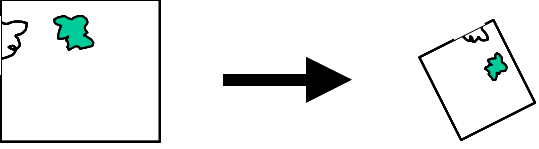
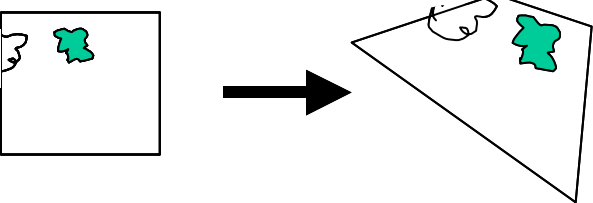
- (i) assumed camera model,
- (ii) assumed terrain model, and
- (iii) method of compositing the pictures

can the two images be made to look identical in their region of overlap.

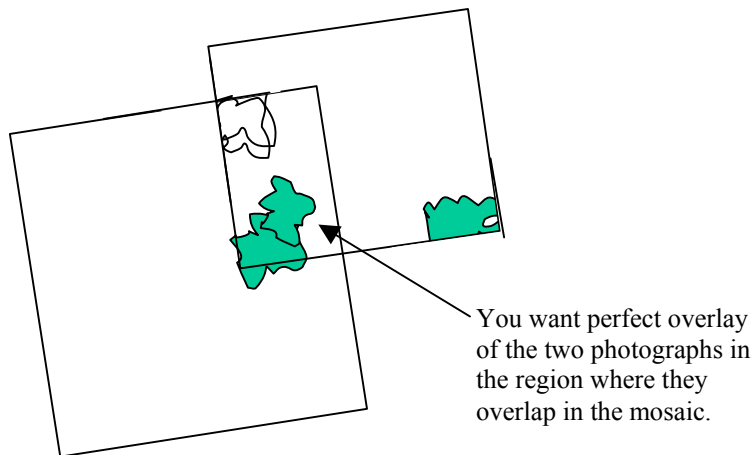
After you take the digital photographs, you form a mosaic using some image editing program. Assume the objects on the ground are stationary, and we only consider geometric effects, not photometric effects. (In other words, the objects in the photographs must line up in the mosaic, but their intensities may differ in the different images.)

- (i) **camera models:** we can assume either perspective projection, or weak perspective projection.
- (ii) **terrain models:** we can assume that the land being photographed lies in a plane, or has arbitrary 3-d shape.
- (iii) **method of compositing the pictures:** say we paste the photographs into the composite bitmap, then have a choice of two different methods of

adjusting the pasted photographs, corresponding to a choice of two different image manipulation programs. Using program A, we can translate, rotate, and uniformly scale each pasted image as we put it into the composite. Using program B, we can make all those changes and more: we can independently specify the location of each of the 4 corners of the pasted image (i.e., we can map the image into an arbitrarily shaped quadrilateral area, linearly interpolating the pixel positions inside the pasted image).

2 different methods of composing the pictures	
Composition method 1 allows scaling and rotation of the photographs:	
Composition method 2 allows arbitrary scaling, rotation, and linear warping of the photographs to fill an arbitrary quadrilateral:	

With two choices each for the camera, terrain, and image composition models, we have 8 possible systems to consider. Under which of those 8 systems can the overlapped pictures always be made to agree exactly within their regions of overlap? Explain your reasoning and justify your answers.



Problem 5 Projective Geometry

Exercise 13.6 from *Computer Vision: A Modern Approach* (Forsyth and Ponce)

Problem 6 State Space Observability

Consider the following linear state space model:

$$x_{t+1} = D_t x_t \qquad y_t = M_t^T x_t$$

Here, the measurement y_t is a one-dimensional scalar for each t , and x_t is a k -dimensional vector. We say that this model is *observable* if the state can be reconstructed from any sequence of k measurements.

- a) Show that the model is observable if and only if the following matrix has full rank:

$$\begin{bmatrix} M_t^T \\ M_{t+1}^T D_t \\ M_{t+2}^T D_{t+1} D_t \\ \vdots \\ M_{t+k-1}^T D_{t+k-2} \cdots D_t \end{bmatrix}$$

- b) Consider a point drifting in 3D ($D_t = I_3$), with a periodically varying observation matrix: $M_{3k} = [0 \ 0 \ 1]^T$, $M_{3k+1} = [0 \ 1 \ 0]^T$, $M_{3k+2} = [1 \ 0 \ 0]^T$. Show that the model is observable.
- c) Show that a point moving with constant velocity in an arbitrary number of dimensions, with the observation matrix reporting position only, is observable.
- d) Show that a point moving with constant acceleration in an arbitrary number of dimensions, with the observation matrix reporting position only, is observable.