Problem 1 Structure from motion

Consider two cameras with projection matrices $P_1$ and $P_2$ looking at a planar surface. The images $x_1' = P_1 X'$ and $x_2' = P_2 X'$ of any point $X'$ on the plane are related by a homography

$$x_2' \propto H x_1'$$

where $H$ is a $3 \times 3$ invertible matrix.

Given the projections of four 3D points that belong to the plane, i.e. four pairs of matching 2D image points, we can recover the matrix $H$ (the algorithm was given in B14 and reviewed in the lectures). However, because all the 3D points belong to the same plane, which is a non-general configuration, we cannot recover the epipolar geometry.

In this question we want to determine how many additional 3D points (not on the plane) one must be observed such that, given their matching projections and the matrix $H$, the epipolar geometry can be recovered.

In order to do so, consider a 3D point $X$ not on the plane whose projections in the two views are $x_1 = P_1 X$ and $x_2 = P_2 X$, respectively. The parallax vector in image 2 is defined as the vector going from $x_2' = H x_1$ (the point transferred by the homography) to the actual projection $x_2$.

Show that the parallax vector lies along an epipolar line. Hence, argue that the location of the epipole can be determined using, in addition to $H$, as few as two correspondences generated from two 3D points $X$ not on the plane.

Problem 2 RANSAC for structure from motion

Using a simple point correspondence algorithm 200 point matches between two views have been established. It is estimated that up to 25% are outliers.

1. If the 8-point algorithm is used to compute the Fundamental Matrix, determine minimum number of trials that will be required in order to be 99% confident that the correct motion has been found.

2. Repeat the calculation for the 7-point algorithm and comment on the results.

Problem 3 Feature points

The SIFT (Scale Invariant Feature Transform) is a popular technique for detecting and representing features for structure-from-motion and object recognition tasks.
1. Explain how feature locations are determined in the SIFT operator.

2. Using MATLAB, or otherwise:
   (a) plot the graph of the 2nd derivative of a 1-D Gaussian
   (b) plot the graphs of the “difference of Gaussians” (in 1-D) for the case of the scales differing by factors of 1.2, 1.4, 1.6, 1.8 and 2.0, and suggest which gives the best approximation to the 2nd derivative.

3. The 2D equivalents of the plot above are rotationally symmetric. Hence infer to what type of image structure a Laplacian of Gaussians (or Difference of Gaussian approximation) will respond?

4. Explain how the SIFT operator achieves invariance to (i) illumination; (ii) scale; (iii) rotation.

**Problem 4 Instantaneous image motion**

1. A scene $\mathbf{X}$ has instantaneous translational velocity $\mathbf{V}$ and angular velocity $\Omega$ relative to a perspective camera with focal length $f$. Derive an expression for the projected image motion, assuming that the image plane lies in front of the optic centre.

2. Explain the origin of the depth-speed scaling ambiguity.

3. The scene now moves such that $\Omega = 0$. Explain how you could solve graphically for the focus of expansion using the position and motion of (at least) two image points.

4. The minimal problem involved intersecting two lines. State the condition under which the construction is (i) most ill-conditioned (ii) best conditioned.

5. Suppose now that the scene is also rotating, but the rotation is known. Show that the translation can be determined (up to the depth-speed scaling ambiguity) by measuring the motion field at 2 points.

**Problem 5 Template tracking**

The Lucas-Kanade tracker for the case of a pure image translation is derived from the sum of squared differences

$$\sum_{x,y} [I(t_x + \delta t_x + x, t_y + \delta t_y + y) - T(x, y)]^2$$

by taking a first-order Taylor expansion of the $I$ term about $I(t_x + x, t_y + y)$ and then minimising over the small displacement $[\delta t_x, \delta t_y]$.

1. By instead considering

$$\sum_{x,y} [I(t_x + x, t_y + y) - T(x - \delta t_x, y - \delta t_y)]^2$$

and taking a first-order expansion about $T(x, y)$ derive an alternative form of the tracker whose update depends on the template’s spatial gradients.
2. Comment on the relative efficiency of the two algorithms

Problem 6 3D Model-based tracking

The “RaPiD” model-based tracker due to Harris (’90) assumes a small motion between frames \( \delta t \mathbf{v}, \delta t \mathbf{\Omega} \) such that
\[
\mathbf{X}' = \mathbf{T} + \delta t \mathbf{V} + \mathbf{P} + \delta t \mathbf{\Omega} \times \mathbf{P}
\]
For a calibrated camera \( \mathbf{x} = \mathbf{X}/Z \).

1. Hence show that to first order
\[
\mathbf{x}' = \begin{bmatrix} \mathbf{x}'/Z' \\ \mathbf{Y}'/Z' \end{bmatrix} \approx \begin{bmatrix} x + \delta t (V_x + \Omega_y P_z - \Omega_z P_y - x(V_z + \Omega_x P_y - \Omega_y P_x)) / (T_z + P_z) \\ y + \delta t (V_y + \Omega_z P_x - \Omega_x P_z - y(V_z + \Omega_x P_y - \Omega_y P_x)) / (T_z + P_z) \end{bmatrix}. \tag{3}
\]

2. Apart from the linearization above, explain why the small motion assumption is fundamental to the RaPiD tracker algorithm?

Problem 7 AdaBoost

1. In the AdaBoost algorithm (given in Figure 1) show that the weight \( \alpha_t \) of a weak classifier increases as its error \( \epsilon_t \) decreases.

2. For a set of training examples \( x_i, i = 1, 9 \), the ground truth classification is
\[
[1, 1, 1, 1, -1, -1, -1, -1, -1]
\]
and four weak classifiers \( h_j \) give the following results
\[
\begin{align*}
h_1(x_i) &= [1, 1, -1, 1, -1, -1, 1, 1] \\
h_2(x_i) &= [1, -1, 1, 1, -1, 1, -1, -1] \\
h_3(x_i) &= [1, 1, 1, -1, 1, -1, -1, -1] \\
h_4(x_i) &= [-1, 1, -1, 1, -1, 1, -1, -1]
\end{align*}
\]
Use the AdaBoost algorithm to show that a strong classifier, \( H(x) = \text{sign} \sum_{t=1}^{T} \alpha_t h_t(x) \), can be built by combining three of these weak classifiers, and gives zero error on the training data. (Write a MATLAB program for AdaBoost).
• Given example data \((x_1, y_1), \ldots, (x_n, y_n)\), where \(y_i = -1, 1\) for negative and positive examples respectively.

• Initialize weights \(\omega_{1,i} = \frac{1}{2m}, \frac{1}{2l}\) for \(y_i = -1, 1\) respectively, where \(m\) and \(l\) are the number of negatives and positives respectively.

• For \(t = 1, \ldots, T\)
  1. Normalize the weights,
     \[
     \omega_{t,i} \leftarrow \frac{\omega_{t,i}}{\sum_{j=1}^{n} \omega_{t,j}}
     \]
     so that \(\omega_{t,i}\) is a probability distribution.
  2. For each \(j\), train a weak classifier \(h_j\) with error evaluated with respect to \(\omega_{t,i}\),
     \[
     \epsilon_j = \sum_{i} \omega_{t,i}[h_j(x_i) \neq y_i]
     \]
  3. Choose the classifier, \(h_t\), with the lowest error \(\epsilon_t\).
  4. Set \(\alpha_t\) as
     \[
     \alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}
     \]
  5. Update the weights
     \[
     \omega_{t+1,i} = \omega_{t,i}e^{-\alpha_t y_i h_t(x_i)}
     \]

• The final strong classifier is
   \[
   H(x) = \text{sign} \sum_{t=1}^{T} \alpha_t h_t(x)
   \]

Figure 1: The AdaBoost algorithm.