Examination #1

CS 766: Computer Vision

October 20, 2005

Last (Family) Name: _____**SOLUTION**_____

First Name: _____

Problem	**Score**	**Out of**
1	_____	16
2	_____	20
3	_____	18
4	_____	10
5	_____	16
Total	_____	80

1. [16] **Image Projection**

(a) [4] Given the 3 · 4 camera matrix

$$P = \begin{bmatrix} 5 & -14 & 2 & 17 \\ -10 & -5 & -10 & 50 \\ 10 & 2 & -11 & 19 \end{bmatrix}$$

and a 3D point in homogeneous coordinates $\mathbf{X} = [0\ 2\ 2\ 1]^T$

(i) What are the Cartesian coordinates of the point **X** in 3D?

(0/1, 2/1, 2/1) = (0, 2, 2)

(ii) What are the Cartesian image coordinates, (u, v), of the projection of **X**?

su = -7, sv = 20, and s = 1, so (u, v) = (-7/1, 20/1) = (-7, 20)

(b) [11] An ideal pinhole camera has focal length 5mm. Each pixel is 0.02 mm · 0.02 mm and the image principal point is at pixel (500, 500). Pixel coordinates start at (0, 0) in the upper-left corner of the image.

(i) [6] What is the 3 · 3 camera calibration matrix, **K**, for this camera configuration?

$$K = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5\ \frac{1}{0.02} & 0 & 500 \\ 0 & 5\ \frac{1}{0.02} & 500 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 250 & 0 & 500 \\ 0 & 250 & 500 \\ 0 & 0 & 1 \end{bmatrix}$$

(ii) [2] Assuming the world coordinate frame is aligned with the camera coordinate frame (i.e., their origins are the same and their axes are aligned), and the origins are at the camera's pinhole, what is the 3 x 4 matrix that represents the extrinsic, rigid-body transformation between the camera coordinate system and the world coordinate system?

$$P_r = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(iii)[4] Combining your results from (a) and (b), compute the projection of scene point (100, 150, 800) into image coordinates.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = KP_r \begin{bmatrix} 100 \\ 150 \\ 800 \\ 1 \end{bmatrix} = \begin{bmatrix} 250 & 0 & 500 & 0 \\ 0 & 250 & 500 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 100 \\ 150 \\ 800 \\ 1 \end{bmatrix} = \begin{bmatrix} 425000 \\ 437500 \\ 800 \end{bmatrix}$$

so, the scene point projects to pixel (425000/800, 437500/800) = (531, 547)

2. [20] **Camera Calibration**

A camera is rigidly mounted so that it views a planar table top. A projector is also rigidly mounted above the table and projects a narrow beam of light onto the table, which is visible as a point in the image of the table top. The height of the table top is precisely controllable but otherwise the positions of the camera, projector, and table are unknown. For each of the following table top heights, the point of light on the table is detected at the following image pixel coordinates:

Table height	Image coordinates of beam of light
50 mm	(100, 250)
100 mm	(140, 340)

(a) [4] Using a *projective camera model* specialized for this particular scenario, write a general formula that describes the relationship between world coordinates (x), specifying the height of the table top, and image coordinates (u, v), specifying the pixel coordinates where the point of light is detected. Give your answer using homogeneous coordinates and a projection matrix containing variables.

The beam of light is a 1D line in the world, which projects to a line of points on the table top, which in turn projects to a line in the images. Therefore this configuration corresponds to a 1D to 1D projective transformation of the form

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ p_{31} & p_{32} \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

(b) [2] For the *first* table top position given above and using your answer in (a), write out the explicit equations that are generated by this one observation.

$$100s = 50p_{11} + p_{12} \quad \text{or} \quad 100 = \frac{50p_{11} + p_{12}}{50p_{31} + p_{32}}$$

$$250s = 50p_{21} + p_{22} \qquad 250 = \frac{50p_{21} + p_{22}}{50p_{31} + p_{32}}$$

$$s = 50p_{31} + p_{32}$$

(c) [1] How many degrees of freedom does this transformation have?

5 because the solution is only defined up to a scale factor in the projection matrix.

(d) [1] How many table top positions and associated images are required to solve for all of the unknown parameters in the projective camera model?

Each table height yields two equations, so three positions are sufficient.

(e) [3] Once the camera is calibrated, given a new unknown height of the table and an associated image, can the height of the table be uniquely solved for? If so, give the equation(s) that is/are used. If not, describe briefly why not.

Given u, x can be uniquely determined by $\begin{bmatrix} su \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{31} & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$ so $x = \dfrac{p_{12} - u}{p_{31}u - p_{11}}$

Note that we could also have used just the v coordinate, though in practice it is a good idea to use *both* coordinates together and solve for a least squares solution so as to minimize measurement errors.

(f) [3] If in each image we only measured the u pixel coordinate of the point of light, could the camera still be calibrated? If so, how many table top positions are required? If not, describe briefly why not.

This problem involves a line-to-line transformation, where points on the line of light project to a line of image points where the beam intersects the table top. Hence if we know the distance d measured along the image line from an origin, then this can be represented using a single image parameter, d, as follows:

$$\begin{bmatrix} sd \\ s \end{bmatrix} = \begin{bmatrix} p & q \\ r & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

But, given only the image u coordinate, we cannot compute d. So, unless the line in the image is parallel to the u axis, the camera cannot be calibrated given only a set of (u, x) pairs.

(g) [3] If instead of assuming a projective camera for (a), we instead assume an *affine camera model* for this problem, write a general formula that describes the relationship between (x) and (u, v).

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

(h) [3] When would an affine camera model be appropriate instead of using a projective camera model? Give one advantage and one disadvantage of using an affine camera instead of a projective camera for this problem.

An affine camera model is appropriate when the scene has little depth variation relative to the viewing distance. A major advantage is computational because an affine camera is linear. A major disadvantage is that it is a weaker model of a true camera and can only be used in the situations described above where, for example, parallel lines do not converge in the image.

3. [18] **Edge Detection**
For each of the following properties of edge detectors, indicate whether the **Canny** edge detector or the **Marr-Hildreth** (also known as $\nabla^2 G$ or Laplacian-of-Gaussian) detector is better with respect to this property, and explain briefly why.

(a) [2] Fewer number of parameters that must be set.

Marr-Hildreth because it only requires 1 parameter.

(b) [2] Closed chains of edges detected.

Marr-Hildreth because it uses zero-crossings.

(c) [2] Computes the edge orientation.

Canny because it uses a directional derivative.

(d) [2] Better localization of the true edge position.

Canny because of non-isotropic smoothing and non-maxima suppression.

(e) [2] Can be implemented more efficiently (i.e., faster execution time).

Marr-Hildreth because it is an isotropic operator that can be implemented using two 1D convolutions.

(f) [2] Requires non-maximum suppression to thin edges.

Canny.

(g) [2] Is an isotropic operator.

Marr-Hildreth.

(h) [2] Less likely to round corners where the boundary curvature is high.

Canny because it does non-istropic smoothing.

(i) [2] Less sensitive to noise.

Canny because it smooths non-isotropically based on the edge normal direction.

4. [10] **Distance Transform**

Consider an 8 · 8 image with two points at pixel coordinates p = (3, 3) and q = (7, 5), specifying the row and column coordinates, respectively. We want to compute the set of points that are *equidistant* from p and q. If we use the Euclidean (L_2) metric, the set of points is a line defining the perpendicular bisector between p and q. For each of the following two alternative metrics, compute the **distance transform** for the image and circle those pixels that are equidistant from p and q as defined by that metric. Each square should show one number, which is the distance to the closest of the two given points.

(a) City-block (L_1) metric.

4	3	2	3	4	5	6	7
3	2	1	2	3	4	5	6
2	1	*p*	1	2	3	4	5
3	2	1	2	**3**	**4**	**5**	**6**
4	3	2	**3**	2	3	4	5
5	**4**	**3**	2	1	2	3	4
4	3	2	1	*q*	1	2	3
5	4	3	2	1	2	3	4

(b) Chessboard (L_∞) metric.

2	2	2	2	2	3	4	5
2	1	1	1	2	3	4	**5**
2	1	*p*	1	2	3	**4**	4
2	1	1	1	2	**3**	3	3
2	2	**2**	**2**	**2**	2	2	3
3	**3**	2	1	1	1	2	3
4	3	2	1	*q*	1	2	3
4	3	2	1	1	1	2	3

5. [16] **Image Segmentation**

(a) [2] The Normalized-Cut segmentation algorithm computes the eigenvalues and eigenvectors of the normalized affinity matrix, $D^{-1/2}AD^{-1/2}$. Describe the meaning of the two eigenvectors corresponding to the two smallest eigenvalues.

The smallest eigenvalue is always 0 and its corresponding eigenvector is all 0s, corresponding to all pixels in the image. The second smallest eigenvalue has an eigenvector that divides the pixels into two sets, which we'll use to bi-partition the image into two regions.

(b) [2] Define an affinity measure, $A(i,j)$, which measures the similarity of pixels i and j and depends on the similarity of their intensities, $f(i)$ and $f(j)$, and inversely on their distance apart, $d(i,j)$. Briefly explain the rationale for your definition.

$$A(i,j) = e^{-\frac{\|f(i)-f(j)\|}{s_1^2}} \cdot \begin{cases} e^{-\frac{d(i,j)}{s_2^2}}, d(i,j) < \mathrm{T} \\ 0, otherwise \end{cases}$$

This definition combines the two measures, scaling each by their standard deviation in order to make them comparable. Also, only pixels that are within distance T of each other are combined so that the affinity between distant pixels in the image is always 0.

(c) [4] To minimize the computational burden of NCut, which requires computing a very large affinity matrix, suppose we first divide an input image into small sub-images, say each of size $50 \cdot 50$. Using these sub-images, describe the main steps of a procedure to use NCut in a two-stage fashion (first with the sub-images, and second with the results of the first stage) to segment an image.

Step 1: For each 50 x 50 sub-image, use the affinity function defined in (b) and apply NCut to compute its bi-partition.

Step 2: Compute the connected components of the binary image constructed by merging the results from Step 1.

Step 3: Define a new graph in which there is a node for each connected component from Step 2, and an arc for each pair of adjacent connected components. Define a new affinity function between pairs of regions based on their size, length of common boundary, average brightness, etc. If two regions are not adjacent, make their affinity 0. Otherwise, make it depend on the similarity of the mean brightness values for the two regions, and their sizes, for example. Apply NCut to this graph, which will compute a bi-partition of the original image. This two-stage approach will have benefits of both saving space and also reducing the tendency of NCut to over-segment images.

(d) [8] Consider an image that is empty except for two sets of points. One is a set of points distributed roughly uniformly on a circle of radius *r* centered at point C1, which is near the center of the image. The other set of points is distributed on a circle of radius 2*r*, which is centered at point C2, which is located inside the other circle. Assume the points in each set are distributed densely enough so that the distances between points on the same circle are smaller than the distances between points on different circles.

(i) Describe what segmentation the **k-means clustering** algorithm would produce for this example, and briefly explain why.

Assuming *k*=2 is given and the initial cluster centers are C1 and C2, the k-means algorithm will *not* find the correct two circular clusters. This is because it attempts to reassign points based on their distance to the class centers. The class separation boundary in this example will be the perpendicular bisector between the estimated class centers, which is initially C1 and C2. Hence all points on one side of this line will be assigned to C1 and all points on the other side of this line will be assigned to C2. At the following iterations the updated class centers will move farther and farther away from the center of the image, resulting in a final partitioning that corresponds to half-circles, i.e., half the points from each of the two circles will be grouped together.

(ii) Describe what segmentation the **normalized-cut** algorithm would produce for this example, and briefly explain why.

Using an affinity measure that is based on the distance between pairs of points, and because the distance between several points within a given circle is less than the distance between points in different circles, the normalized-cut algorithm will compute a bi-partition that *correctly* segments this image into the two sets of circular points.