Examination #1

CS 766: Computer Vision

October 21, 2004

Last (Family) Name: SOLUTIONS

First Name: __________

Problem	Score	Out of
1	____	15
2	____	15
3	____	20
4	____	10
5	____	15
6	____	10
Total	____	85

1. [15] **Camera Calibration**
The relationship between a 3D point at world coordinates (X,Y,Z) and its corresponding 2D pixel at image coordinates (u,v) can be defined as a projective transformation using a 3×4 camera projection matrix **P**.

(a) [3] Can the matrix **P** incorporate any lens distortions that might be in the camera? Briefly explain.

No because lens distortions are usually highly nonlinear, which cannot be represented in **P**.

(b) [4] Give two lists, one specifying the intrinsic camera parameters and the other giving the extrinsic camera parameters.

The five intrinsic parameters are the focal length, f, principal point coordinates, (u_0, v_0), and the pixel size scaling parameters, k_u, k_v.
The six extrinsic parameters are the three translational and three rotational parameters that define the rigid body motion between the world coordinate frame and the camera coordinate frame.

(c) [4] Show how **P** can be decomposed into a product of matrices that contain elements expressed in terms of the intrinsic and extrinsic camera parameters.

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{T}] = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

(d) [4] Give the main steps of an algorithm for computing the matrix **P** from a single image of a known 3D "calibration object."

Since **P** is a 3 x 4 matrix with 11 unknowns (the overall scale of **P** does not matter), observing a 3D scene containing at least 6 known points (in a non-degenerate configuration) is sufficient. Each point gives a pair of equations of the form

$$u = su/s = (p_{11}X + p_{12}Y + p_{13}Z + p_{14}) \,/\, (p_{31}X + p_{32}Y + p_{33}Z + p_{34})$$
$$v = sv/s = (p_{21}X + p_{22}Y + p_{23}Z + p_{24}) \,/\, (p_{31}X + p_{32}Y + p_{33}Z + p_{34})$$

Rewrite these 12 equations in the form **Ap** = 0, where **p** is a 12 x 1 vector of unknowns, and **A** is a 12 x 12 matrix of coefficients. Use least squares to solve for **p** by computing the eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$. Note: This is only an approximate solution and often this is then used as a starting point for a nonlinear optimization.

2. [15] **Planar Transformations**
The planar façade of a building is captured in an image taken by a camera. Assume this plane corresponds to the world coordinate frame's Z=0 plane, and scene point (X,Y) on the building projects to image pixel coordinates (u,v).

(a) [3] What is the *planar projective transformation* that describes the relationship between (X,Y) and (u,v)? Give your answer using homogeneous coordinates.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

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

8

(c) [1] How many point correspondences are required to determine this transformation?

4

(d) [2] Would having more correspondences than your answer to (c) be helpful in any way? If no, briefly explain why not. If yes, explain how they could be used.

```
Yes, if more points are available, the calibration accuracy can be
improved by using linear least squares.
```

(e) [2] Give one invariant of a planar projective transformation.

```
Invariants include concurrency, collinearity, order of contact,
tangent discontinuities and cusps, cross-ratio of 4 collinear
points, and measurements in a canonical frame.
```

(f) [2] Give one invariant of a planar affine transformation that is not an invariant for a planar projective transformation.

```
Invariants include parallelism, ratio of areas, ratio of lengths on
collinear or parallel lines (e.g., midpoints).
```

(g) [4] If the building has sets of lines on it running parallel to both the X and Y axes, how could we use the corresponding lines in the image to determine if the building plane is parallel to the image plane?

```
If both sets have their vanishing point at infinity then the
building is fronto-parallel.
```

3. [20] **Edge Detection**

(a) [5] Show how an approximation to the first derivative of an image can be obtained by convolving the image with the kernel [1 −1] where the image is defined as

[56 64 79 98 115 126 132 133]

Ignore computing a value for the first and last image pixels (in other words, your result will be 6 values). In addition to showing the result of the convolution, indicate where edges would be detected and why.

```
8  15  19  17  11  6  1
```

```
Intensity discontinuities occur at the maxima of the first
derivative, which this approximates.  Here the maximum of 19
corresponds to the position where an edge is detected,
corresponding to a position between the pixels with
intensities 79 and 98 (or associated with one of these two
pixels).
```

(b) [6] What property of the coefficients of a kernel ensures that an appropriate output is obtained for regions of constant intensity in an image when

(i) The kernel is approximating a first derivative.

`The kernel values sum to 0.`

(ii) The kernel is approximating a second derivative.

The `kernel values sum to 0.`

(iii) The kernel is approximating a Gaussian.

`The kernel values sum to 1 (after normalization).`

(c) [4] Describe a major advantage or disadvantage of using an isotropic operator instead of a non-isotropic operator with respect to the following issues:

(i) Computational efficiency.

```
Isotropic operators such as the Laplacian of a Gaussian are
generally more computationally efficient than non-isotropic
operators because they can be implemented using a single convolution
(or multiplication in the frequency domain) followed by zero-
crossing detection.  Directional operators such as the Canny
operator require a search for a local maxima.
```

(ii) Noise in the image.

```
Directional operators are generally more robust to noise by
smoothing pixels which are on only one side of an edge.  Also, an
operator which uses higher order derivatives is more sensitive to
noise.
```

(d) [5] What is the purpose of (i) non-maxima suppression and of (ii) hysteresis that are done in the Canny edge detector?

```
(i) Non-maxima suppression thins responses so that only a
single pixel in the gradient direction will be detected, thus
produces a one-pixel wide sequence of edge points.
(ii)  Hysteresis thresholding simultaneously attempts to
eliminate weak edge points that are caused by noise while
filling in gaps between strong edge points where only weak
edge response is detected.
```

4. [10] **Corner Detection**

(a) [2] When would detecting corners be more appropriate than detecting edges as an initial step in an application using computer vision?

Detecting corners would be more appropriate when only a sparse set of points are needed, especially facilitating the detection of corresponding points in multiple images. This is used, for example, in stereopsis and motion tracking.

(b) [4] The Harris corner detection algorithm computes a 2×2 matrix at each pixel based on the first derivatives at that point and then computes the two eigenvalues of the matrix, λ_1 and λ_2, where $\lambda_1 \le \lambda_2$. How can these two values be used to label each pixel as either a locally smooth region (S), an edge point (E), or a corner point (C)? Give your answer by specifying "Label pixel S if …", "Label pixel E if …" and "Label pixel C if …"

The Harris corner detector first computes $R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$ and then
labels a pixel S if $|R| \approx 0$ (or, alternatively, $\lambda_1 \approx \lambda_2 \approx 0$);
labels a pixel E if $R < T_1 < 0$ (or, alternatively, $\lambda_1 \approx 0$ (corresponding to the direction of the edge) and λ_2 is large (corresponding to the normal direction at the edge)); or
labels a pixel C if $R > T_2 > 0$ (or, alternatively, λ_1 and λ_1 are both large).

(c) [4] Given the two eigenvalues specified in (b), explain in English the rationale and difference(s) between detecting corner points using the criterion $\lambda_1 > T_1$ (which is used in the Tomasi and Kanade algorithm) versus the criterion $\lambda_1\lambda_2 > T_2$ (which is used in the Harris algorithm), where T_1 and T_2 are appropriate thresholds.

Since $\lambda_1 \le \lambda_2$, λ_2 corresponds to the direction of maximum change in intensity and λ_1 corresponds to the direction of minimum change in intensity. Therefore, the criterion $\lambda_1 > T_1$ is used to test if there is more than one direction with a strong edge present. The criterion $\lambda_1\lambda_2 > T_2$ could hold if $\lambda_2 >> \lambda_1$ and yet λ_1 is relatively small compared to λ_2. The criterion $\lambda_1 > T_1$ ensures that both eigenvalues are large as defined by T_1. Ideally, we'd like to ensure that both eigenvalues are large and they are of similar size.

5. [15] **Active Contours**

The energy functional that is used with active contours (snakes) usually contains the three terms: $E_{continuity}$, $E_{smoothness}$, and E_{image} (the first two terms are often combined to define a term called E_{int}).

(a) [9] For each of these three terms explain briefly what it measures, how it is defined, and what happens if the term is omitted.

$E_{continuity} = ||dv/ds||^2$ measures the degree of rigidity or elasticity of the contour. If it is omitted, there can be discontinuities along the contour. Removing it can also cause the snaxels to "bunch up."

$E_{smoothness} = ||d^2v/ds^2||^2$ measures the degree of bending or stiffness of the contour. Omitting it allows tangent discontinuities along the contour. This may cause the snake to become quite jagged and may overfit noisy data.

E_{image} measures features in the image that act as either attraction or repulsion forces on the contour. For example, using edge magnitude as a term makes the contour attracted to edges in the image. Omitting this term means that the evolution of the snake will not be influenced by the image data at all; with usual definitions of the total energy functional this will mean that the contour will shrink to a point or a line.

(b) [3] What is the effect of giving a negative weight to $E_{continuity}$?

This will cause snaxels to bunch up and tend to cause the contour to expand.

(c) [3] How could the energy functional definition be specified so as to cause the contour to expand?

One approach would be to modify E_{image} so that it also includes a "balloon" force in the normal direction of the contour at each snaxel point.

A second approach, given in the online reading, is to run the snake algorithm, then segment the resulting contour to eliminate high energy segments. Grow each end of each segment in the direction of its tangents; then run the snake algorithm on each segment. Finally, merge all of the final segments.

6. [10] **Segmentation using Normalized Graph Cut**

(a) [3] Define cut(A,B) and explain intuitively what it measures.

$cut(A,B) = \sum_{i \in A, j \in B} aff(i,j)$ measures the similarity between two groups of pixels defined by the disjoint sets *A* and *B*.

(b) [5] Say we want to find "regions" corresponding to object boundaries by grouping connected chains of "strong" edge points that are adjacent and their gradients imply a smooth contour. Define an affinity measure for this purpose that uses the magnitude, mag($\nabla I(p)$), and direction, direc($\nabla I(p)$), of the gradient, ∇I, at a pixel *p* in image *I* to compute $\text{aff}_{\text{edge}}(x,y)$ for a pair of adjacent pixels *x* and *y*. Include any additional computational steps before computing the affinity that make sense for obtaining good results. Briefly explain your definition.

Intuitively, if there is an edge between pixels *x* and *y*, then *x* and *y* are on opposite sides of the edge and the dissimilarity should be high, meaning the affinity should be low. On the other hand, if the edge directions at both *x* and *y* are nearly the same, then *x* and *y* are likely on the same contour and the affinity should be high. Thus affinity can be defined by measuring the angle between the gradient directions at the two pixels, weighted by the gradient magnitude. If the angle is near 180 degrees the affinity is low and if it is near 0 degrees, then the affinity is high. One way to do this is to compute the dot product of the two gradient vectors, $\nabla I(x) \cdot \nabla I(y)$, which computes the cosine of the angle between these vectors. Non-maxima suppression should be done first so that "thick" contours are not created.

(c) [2] Say we want to segment images into regions of uniform texture using a measure of the texture at each pixel by computing some function over a $(2n+1) \times (2n+1)$ neighborhood centered on the pixel. Briefly explain why using this function to define the texture affinity between two pixels, $\text{aff}_{\text{texture}}(x,y)$, will be biased depending on how close the pixels are to a region boundary.

Using a fixed window to compute the texture at a pixel will be "polluted" when the pixel is near a texture boundary because some of the pixels in the window will be in one region and other pixels will be in a different region, each with their own texture characteristics.