CS280 - Computer Vision - Spring 2018 - Assignment 1
Due: Sunday, Feb 4, 2018, 11:55pm

1 Perspective Projection

Humans can estimate the spatial layout of a scene from a single image. Can computers also do this? In computer vision, we have made some progress towards solving this problem (i.e. recovering depth/spatial layout from a single image; interested students may want to have a look at [1]). There are various methods through which spatial layout can be recovered from a single image. Some of these methods rely on the knowledge of perspective projections and vanishing points. Here, we will further our understanding of perspective projections.

1. Show that the vanishing points of lines on a plane lie on the vanishing line of the plane.
2. Show that, under typical conditions, the silhouette of a sphere of radius r with center $(X,0,\ Z)$ under planar perspective projection (XY is the image plane, and the center of projection is at the origin) is an ellipse of eccentricity $X/\sqrt{(X^2+Z^2-r^2)}$. Are there circumstances under which the projection could be a parabola or hyperbola? (Hint: See the definition of conic sections at: `http://en.wikipedia.org/wiki/Conic_section#Features`)
3. An observer of height h is standing on a ground plane looking straight ahead. We want to calculate the accuracy with which she will be able to estimate the depth Z of points on the ground plane, assuming that she can visually discriminate angles to within $1'$. Derive a formula relating depth error δZ to Z. For simplicity, just consider points straight ahead of the observer($x = 0$). Given a Z value (say 10 m), your formula should be able to predict the δZ.

[1] Criminisi, Antonio, Ian Reid, and Andrew Zisserman. "Single view metrology." *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on.* Vol. 1. IEEE, 1999.

2 Rotations

Rotations are routinely encountered in both Computer Vision and Graphics. One application where rotations turn out to be very handy is rendering images of a object from different viewpoints. Rodrigues' formula provides an elegant way to deal with rotations. `Wikipedia` can serve as a good reference.

1. Rodrigues' formula converts a rotation ϕ about an axis of rotation $\widehat{\mathbf{s}}$, a unit vector. Performing a cross product with vector $\widehat{\mathbf{s}}$ is equivalent to multiplying with a skew-symmetric matrix $\mathbf{S} \in \Re^{3x3}$. In other words, $\widehat{\mathbf{s}} \times \mathbf{b} = \mathbf{S}\mathbf{b}$ for any $\mathbf{b} \in \Re^3$. What is $\mathbf{S}$ in terms of $\widehat{\mathbf{s}}$?
2. Rodrigues showed that the matrix exponential of $\phi\mathbf{S}$ produces a rotation ϕ about an axis of rotation $\widehat{\mathbf{s}}$. In other words, $\mathbf{R} = \exp(\phi\mathbf{S})$. Use this to derive the Rodrigues' formula, stated below.
$$\mathbf{R} = \mathbf{I} + (\sin\phi)\mathbf{S} + (1-\cos\phi)\mathbf{S}^2$$
3. Write a `Matlab` or `Python` function for computing the matrix $\mathbf{R}$ corresponding to rotation ϕ about axis vector $\widehat{\mathbf{s}}$. Choose a random unit vector $\widehat{\mathbf{s}}$ and a random initial point with unit magnitude. Plot the axis of rotation, along with the point after rotations of $\phi \in \{0, \frac{\pi}{12}, \frac{\pi}{8}, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. Do this for a few pairs of axes and points and include this in your report.
4. Find the eigenvalues of $\mathbf{R}$ and their corresponding eigenvectors and verify analytically. Express eigenvectors in terms of unit vectors $\widehat{\mathbf{u}}, \widehat{\mathbf{v}}, \widehat{\mathbf{s}}$, where $\widehat{\mathbf{u}} \times \widehat{\mathbf{v}} = \widehat{\mathbf{s}}$.
5. Analytically verify the formula $\cos\phi = \frac{1}{2}(\mathrm{trace}(\mathbf{R}) - 1)$. Hint: Eigenvalues are your friend.
6. Write a function in `rot_to_ax_phi.m` or `rot_to_ax_phi.py` for computing the axis of rotation $\widehat{\mathbf{s}}$ and ϕ from matrix $\mathbf{R}$. Include the function in your report.

3 Make yourself famous!

Find a real image of your choosing (e.g. a portrait of yourself) as your source image. Pick another image to map onto (e.g. your hometown). Pick at least 5 planar surfaces of interest (e.g. building façades, posters, billboards, ground) in `/images/times_square.jpg` and a few planar surfaces for your target image. Mark the corner coordinates for each. They should be distributed across the image and have varying normal directions in 3D. You will be projecting your source image onto Times Square and your target image!
Given points $\{\mathbf{u_i}\}, \{\mathbf{v_i}\}$, where $\mathbf{u_i}$ is a corner in your chosen image, and point $\mathbf{v_i}$ is the corresponding corner in the Times Square image, both in homogenous coordinates, we would like to find the transformation $\mathbf{v_i} = T(\mathbf{u_i})$. Function $T : \Re^3 \rightarrow \Re^3$ is described as follows:

$$\mathbf{v_i} = \begin{bmatrix} v_{ix} \\ v_{iy} \\ 1 \end{bmatrix} = \begin{bmatrix} V_{ix}/V_{iz} \\ V_{iy}/V_{iz} \\ 1 \end{bmatrix}, \text{ where } \begin{bmatrix} V_{ix} \\ V_{iy} \\ V_{iz} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{ix} \\ u_{iy} \\ 1 \end{bmatrix} = \mathbf{H}\mathbf{u}_i$$

Include the following in your report:

1. Derive the least squares solution for $\mathbf{H}^* = \arg\min_{\mathbf{H}} \sum_{i=1}^{4} ||T(\mathbf{u}_i) - \mathbf{v}_i||^2$ when using
   (a) 2D affine transform (translation + rotation), and
   (b) homography.
2. What constraints are placed on $\mathbf{H}$ for the affine transform? How about for the homography?
3. Is the affine transform able to exactly transform the points from one image to the other? Why or why not? How about for the homography?
4. Implement both affine and homography transforms in `Matlab` or `Python` to map your chosen image onto the planar surfaces. In the report, you should include the following functions as well as the resulting images (saved as `times_square_affine[homography].jpg` and `myimage_affine[homography].jpg`) and any observations.
   (a) `H = affine_solve(u,v)`, `H = homography_solve(u,v)`, where `u,v` are 2xN matrices, representing N corresponding points. Functions should return `H`, a 3x3 matrix.
   (b) `v = homography_transform(u,H)`, where u is a 2xN matrix and `H` is a 3x3 matrix. Function should return `v`, a 2xN matrix.

   Matrices in `Python` should be represented as 2-D `numpy` arrays. Only use basic matrix operations to write these functions (e.g. multiplication, addition, concatenation, dot products, reshaping, etc.). Do not use a pre-written psuedo-inverse function. For reference, example results are shown in `/images/times_square_affine_efros.jpg` and `/images/times_square_homography_efros.jpg`.
5. Similar to 3.4, implement both affine and homography transforms in `Matlab` or `Python` to rectify the computer screen in `/images/computer_screen.jpg` and the black/whitefloor shown in `/images/the_flagellation.jpg`. For the computer screen, mark four points on the four corner of the screen and the floor mark as many points you deem appropriate for computing the transformations.

4 Instructions

1. This assignment is to be done individually.
2. Please submit the assignment using bCourses. Upload the following files:
   (a) `A PDF file`. The top of the first page should contain your name, student ID, and date of submission. The file should contain answers to all questions and all supporting images. Questions should be answered in order. Each problem should be on a new page.
   (b) `A tar/zip file`, containing any code you wrote for the assignment.
3. The HW is due on: Sunday, Feb 4, 2018, 11:55pm.