

CSE152a – Intro to Computer Vision – Assignment 1

Instructor: Prof. David Kriegman.

Instructions:

- Attempt all questions
- Turn in a **hard copy in class**. This should include answers to the paper-and-pen questions and a report on the programming questions. For the last two problem, also include your MATLAB code in the hard copy that you turn in. This is *not* required for problems 1 and 2.
- In addition, **email a copy of all your code** with the subject *CSE152 Assignment 1* to oboejbom@cs.ucsd.edu. Attach code as a *zip* or *tar* file. Important: include the “CSE152” in the subject line so my email filter catches it.

1. Geometry [15 points]

Consider a line in the 2D plane, whose equation is given by $ax + by + c = 0$. This can equivalently be written as $l^T x = 0$, where $l = (a, b, c)^T$ and $x = (x, y, 1)^T$. Noticing that x is a homogeneous representation of $(x, y)^T$, we can view l as a homogeneous representation of the line $ax + by + c = 0$. We see that the line is also defined up to a scale since $(a, b, c)^T$ and $k * (a, b, c)^T$ with $k \neq 0$ represents the same line.

All points (x, y) that lie on the line $ax + by + c = 0$ satisfy the equation $l^T x = 0$. We note this down as a fact:

Fact 1: A point x in homogeneous coordinates lies on the homogeneous line l if and only if the dot products $x^T l = l^T x = 0$

- [2 points] Using Euclidean coordinates, what is the equation of the line passing through the points (1,2) and (3, 4).
- [4 points] Prove the following two statements that follow from Fact1:
 - The cross product between two points gives us the line connecting the two points
 - The cross product between two lines gives us their point of intersection
- [3 points] What is the line, in homogenous coordinates, connecting the points (1, 2) and (3,4).
- [10 points] When a rectangle is observed under pinhole perspective, the image will be arbitrary quadrilateral, and figure 1 shows a projected rectangle. Answer the following questions using homogeneous representations. Find the line equations (i.e. $ax + by + c = 0$) of the four edges. Calculate the Euclidean coordinates of vanishing point for the image of each pair of parallel edge (lines 2 & 4 and lines 1 &3).

2. Image formation [38 points]

Suppose we are viewing a square with vertices $P_1 = (-1, -0.5, 0)^T$, $P_2 = (1, -0.5, 0)^T$, $P_3 = (1, 0.5, 0)^T$, $P_4 = (-1, 0.5, 0)^T$ in world coordinates. Recall that the image formation can be divided into two matrices, P_i , the image formation (intrinsic) matrix, and P_e , the rotation / translation (extrinsic) matrix. P_i is 3×4 , and P_e is 4×4 . Once these are found, the image of a point, S , can be calculated as $P_i * P_e * S$. We will consider three different settings of focal length, viewing angles and camera positions below. For each of these, calculate

- the extrinsic transformation matrix,
- intrinsic camera matrix under the perspective camera assumption.

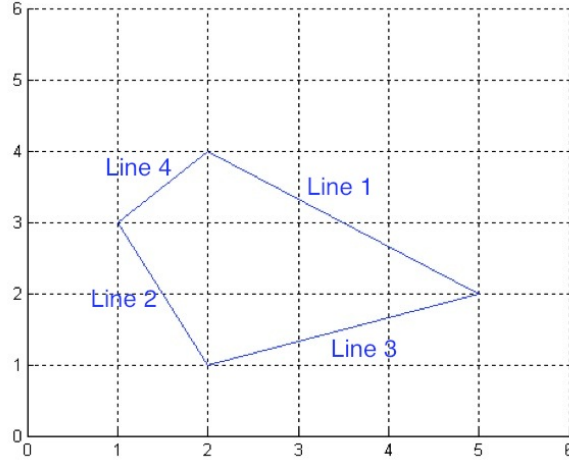


Figure 1: Illustration for problem 1

- (c) intrinsic camera matrix under the affine camera assumption. (Use the camera center (${}^A O$) as the center of the Taylor expansion).
- (d) Calculate the image of the four vertices and plot (see e.g. output in figure 2).

Each of these are worth 3 points. Note: we will not use a full intrinsic camera matrix (e.g. that maps centimeters to pixels, and defines the coordinates of the center of the image), but only parameterize this with f , the focal length. In other words: the only parameter in the intrinsic camera matrix under the perspective assumption is f , and the only ones under the affine assumption are: f, x_0, y_0, z_0 , where x_0, y_0, z_0 is the center of the Taylor series expansion.

- (a) **[Pure translation, 12 points]**. Focal length = 1. The center of the camera is placed such that the origin is at ${}^A O = (0, 0, 2)^T$. The optical axis of the camera is aligned with the z-axis.
 - (b) **[Translation and rotation, 12 points]**. Focal length = 1. The center of the camera is placed such that the origin is at ${}^A O = (0, 0, 2)^T$. Now, it is also rotated 30 degrees around the z-axis and then 45 degrees around the y-axis.
 - (c) **[Translation and rotation, long distance, 12 points]**. Focal length = 5. The center of the camera is placed such that the origin is at ${}^A O = (0, 0, 10)^T$. Now, it is also rotated 30 degrees around the z-axis and then 45 degrees around the y-axis.
 - (d) **[Discussion, 2 points]**. Discuss your results. How does the projective and affine camera models differ?
3. **[Colors 15 points]** Write a matlab function `imgOut = TRANSFORMHSV(imgIn,  $\theta, s, v$ )` to manipulate the color of an image in HSV color space. The function should do the following:
- Convert the image from RGB to HSV color space
 - Apply a clockwise rotation of the hue channel by θ degrees
 - Multiply the saturation channel by s
 - Multiply the value channel by v
 - Convert the transformed HSV image back into RGB color space

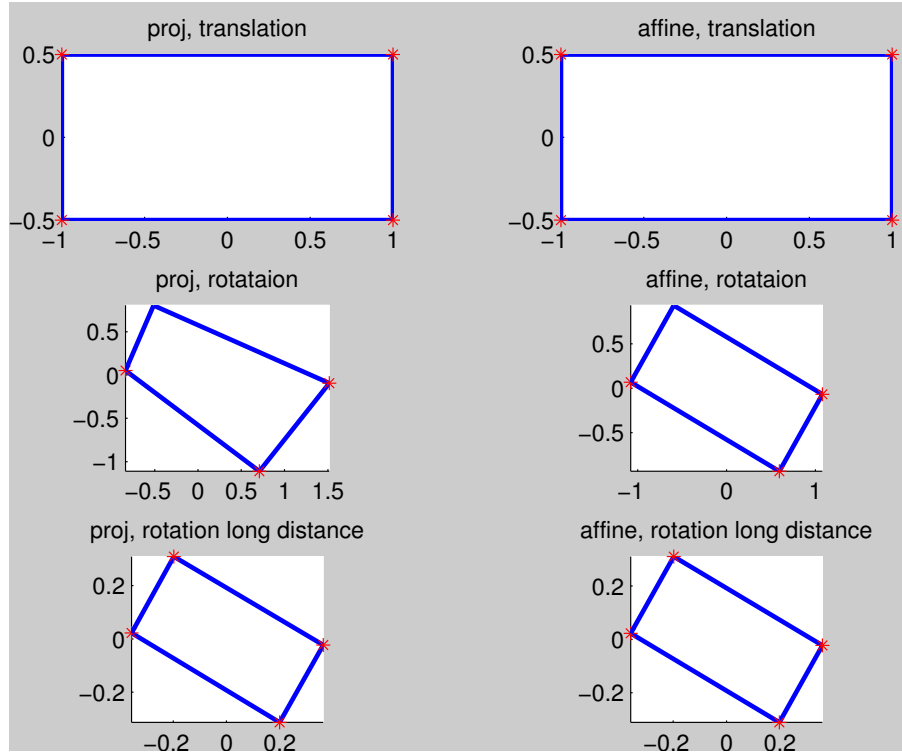


Figure 2: Example output for problem 2. Note: the angles and offsets used to generate these plots are different from those in the problem statement, it's just to illustrate how to report your results.

DO NOT use the built-in `rgb2hsv` or `hsv2rgb` functions in Matlab, except to compare your results. Test your function on the image `color.bmp`, and generate results for `TRANSFORMHSV(img,90,1,1)`, `TRANSFORMHSV(img,0,3,1)`, `TRANSFORMHSV(img,0,1,3)`. Include the generated images in your writeup.

4. **[Rendering 20 points]** In this exercise, we will render the image of a cone-like surface illuminated by two different point light sources using a Lambertian reflectance model. The surface is defined by the equation $z = -\sqrt{x^2 + 4y^2}$ for all $x^2 + 4y^2 \leq 1$. The first point light source is located at $(-1, .5, 0)^T$, and the second light source is located at $(1, -.5, 1)^T$. Both light sources have intensity .5. The ellipsoid surface is Lambertian, and has an albedo of 1 throughout the entire surface.

Write a Matlab function that renders a 401×201 image of the surface with pixels corresponding to the locations $x = -1 : 0.005 : 1$ and $y = -.5 : 0.005 : .5$. Set all pixels outside the circle $x^2 + 4y^2 > 1$ equal to 0. Generate an image of the surface that is illuminated by just the first light source, a second image that is illuminated by just the second light source, and a 3rd image that is illuminated by both light sources. Include all 3 output images in your writeup.

To facilitate this we have provided two matlab files. The first, `renderConeDriver.m`, is a wrapper to the second, `renderConeTemplate.m` that does the actual computations. Your task is to fill in the missing code in `renderCone.m`. Note: if you prefer to write everything from scratch, that is perfectly fine also.