CSE252a – Computer Vision – Assignment 1
Instructor: Prof. David Kriegman.
Revision 1

Instructions:

- This assignment should be solved, *and written up* in groups of 2. If you can't find a partner, you can work alone. No groups of 3 are allowed.
- Attempt all questions
- Submit your assignment electronically by email to obeijbom@cs.ucsd.edu with the subject line *CSE252 Assignment 1.* The email should have two files attached.
  1. A pdf file with your writeup. This should have all code attached in the appendix. Name this file: CSE_252_hw1_writeup_lastname1_lastname2.pdf.
  2. A compressed archive with all your matlab code files. Name this file: CSE_252_hw1_code_lastname1_lastname2.zip.

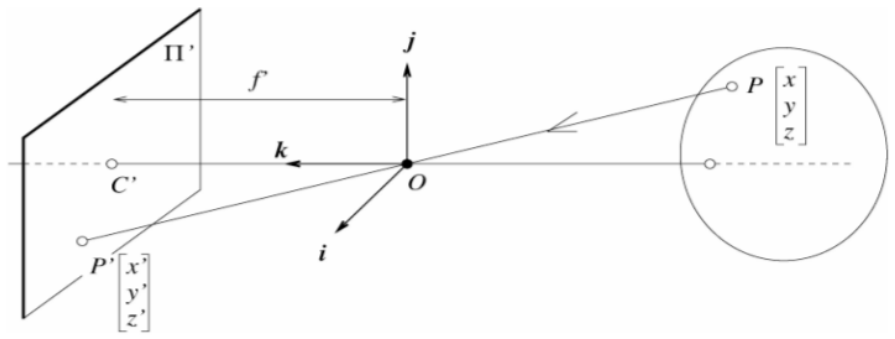
  The code is thus be attached *both* as text in the writeup appendix and as m-files in the compressed archive.
- Please make this a proper report, with methods, thoughts, comments and discussions on *every problem.* All code should be tucked away in an appendix.
- No physical hand-in for this assignment.
- You may do problems on pen an paper, just scan and include in the writeup pdf file.
- In general, MATLAB code does not have to be efficient. Focus on clarity, correctness and function here, and we can worry about speed in another course.

Perspective Projection [2 pts]

Consider a perspective projection where a point

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

is projected onto an image plane Π' represented by $k = f' > 0$ as shown in the following figure.



The first, second, and third coordinate axes are denoted by i, j, and k respectively. Consider the projection of a ray in the world coordinate system

$$Q = \begin{bmatrix} 2 \\ -3 \\ 0 \end{bmatrix} + t \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}$$

where $-\infty \leq t \leq -1$. Calculate the coordinates of the endpoints of the projection of the ray onto the image plane.

Thin Lens Equation [2 pts]

An illuminated arrow forms a real inverted image of itself at a distance w = 90cm, measured along the optic axis of a convex thin lens (see Figure 1). The image is just half the size of the object.

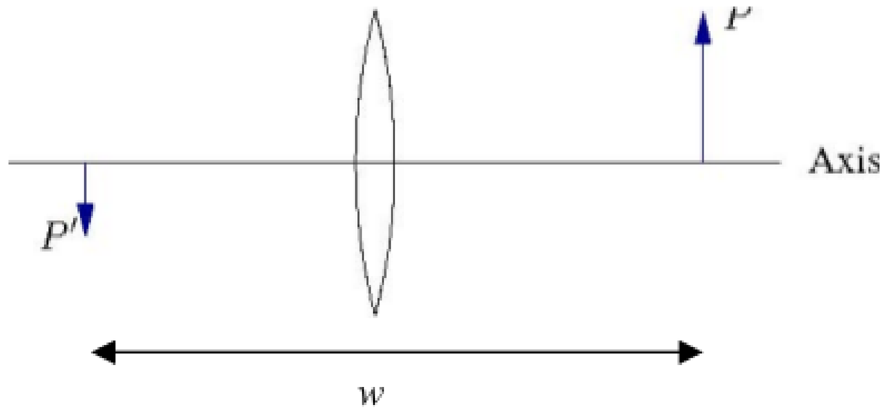


Figure 1: Problem 3 Setup

1. How far from the object must the lens be placed?
2. What is the focal length of the lens?
3. Suppose the arrow moves x cm to the right while the lens and image plane remain fixed. This will result in an out of focus image; what is the radius of the corresponding blur circle formed from the tip of the arrow on the image plane assuming the diameter of the lens is d?

Affine Projection [2pts]

Consider an affine camera and a line in 3D space. Consider three points (A, B, and C) on that line, and the image of those three points (a, b and c). Now consider the distance between a and b and the distance between a and c. Show that the ratio of the distance is independent of the direction of the line.

Image formation and rigid body transformations [10 points]

In this problem we will practice rigid body transformations and image formations through the projective and affine camera model. The goal will be to 'photograph' the following four points given by ${}^AP_1 = (-1, -0.5, 2)^T$, ${}^AP_2 = (1, -0.5, 2)^T$, ${}^AP_3 = (1, 0.5, 2)^T$, ${}^AP_4 = (-1, 0.5, 2)^T$ in world coordinates. To do this we will need two matrices. Recall, first, the following formula for rigid body transformation

$$ {}^BP = {}^B_AR\ {}^AP + {}^BO_A \tag{1} $$

where ${}^BP$ is the point coordinate in the target (B) coordinate system, ${}^AP$ is the point coordinates in the source (A) coordinate system, ${}^B_AR$ is the rotation matrix from A to B, and ${}^BO_A$ is the origin of coordinate system A expressed in the B coordinates. The rotation and translation can be combined into a single 4×4 *extrinsic parameter* matrix, Pe, so that ${}^BP = Pe * {}^AP$. Once transformed, the points can be photographed using the *intrinsic camera* matrix, Pi which is a 3×4 Once these are found, the image of a point, ${}^AP$, can be calculated as $Pi * Pe * {}^AP$. We will consider four 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.
- intrinsic camera matrix under the affine camera assumption. In particular, around what point do you do the taylor series expansion?
- Calculate the image of the four vertices and plot using the supplied plotsquare.m function (see e.g. output in figure 2).

1. [**No rigid body transformation**]. Focal length = 1. The optical axis of the camera is aligned with the z-axis.
2. [**Translation**] ${}^BO_A = (0, 0, 1)^T$. The optical axis of the camera is aligned with the z-axis.
3. [**Translation and rotation**]. Focal length = 1. ${}^B_AR$ encodes a 30 degrees around the z-axis and then 60 degrees around the y-axis. ${}^BO_A = (0, 0, 1)^T$.
4. [**Translation and rotation, long distance**]. Focal length = 5. ${}^B_AR$ encodes a 30 degrees around the z-axis and then 60 degrees around the y-axis. ${}^BO_A = (0, 0, 13)^T$.

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.

In your report, include a image like Figure 2. Note that the axis *are the same for each row*, to facilitate comparison between the two camera models. Also include

1. The actual points around which you did the taylor series expansion for the affine camera models.
2. How did you arrive at these points?
3. How do the projective and affine camera models differ? Why is this difference smaller for the last image compared to the second last?
4. I'm expecting you to implement this in MATLAB rather than by pen and paper.

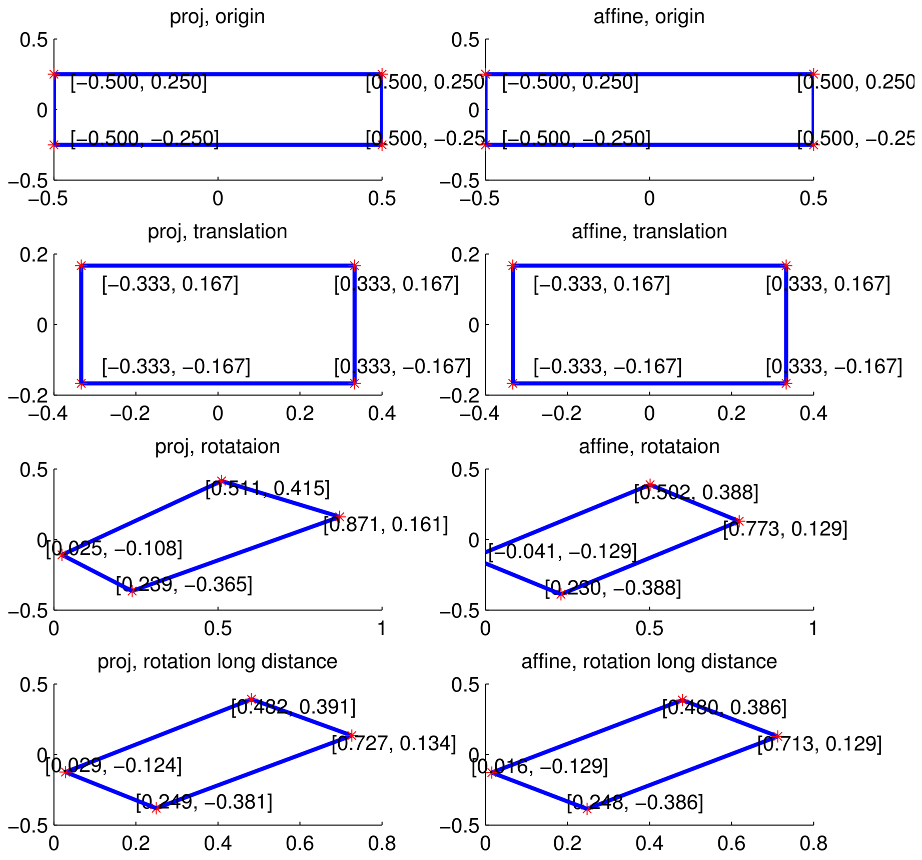


Figure 2: Example output for image formation problem. Note: the angles and offsets used to generate these plots are different from those in the problem statement, it's just to illustrate how to report your results.

Image warping and merging [10 pts]

This is a programming assignment, which should be done in Matlab as many of the necessary numerical routines are readily available (e.g., eig or svd for the computation of the eigenvalues and eigenvectors of a matrix). All data necessary for this assignment is available on the course web page.

Introduction

In this assignment, we consider a vision application in which components of the scene are replaced by components from another image scene. Consider for a moment that you are watching a sporting event on television whose audience, which has a broad, possibly multinational audience. During these events, it is advantageous and profitable to use different advertisements in different markets to target more directly viewers in those regions. In this assignment you will implement a simple version of this algorithm using multiple advertisements on a single scene. Given two scenes, the natural thing to do would be to compute how your object of interest is observed in the current scene and warp it to match the destination scene. This would allow you to paste together the two images where your new advertisement overlaps the particular billboard in the new scene even though these two images were obtained in different locations. This digital replacement is accomplished by a set of points for each advertisement in both the target (scene) and advertisement images. The task then consists of mapping the points from the advertisement to their respective points in the target image. In the most general case, there would be no constraints on the scene geometry, making the problem quite hard to solve. If, however, the scene can be approximated by a plane in 3D, a solution can be formulated much more easily even without the knowledge of camera calibration parameters. To solve this section of the homework, you will begin by deriving the transformation that maps one image onto another in the planar scene case. Then you will write a program that implements this transformation and uses it to warp some UCSD logos into an art gallery.

To begin, we consider the projection of planes in images. Imagine two cameras C_1 and C_2 looking at a plane π in the world. Consider a point P on the plane π and its projections $p = (u1, v1, 1)^\top$ in image 1 and $q = (u2, v2, 1)^\top$ in image 2.

Fact 1 *There exists a unique (up to scale) 3x3 matrix H such that, for any point P:*

$$q \equiv Hp$$

(Here $\equiv$ denotes equality in homogeneous coordinates, meaning that the left and right hand side are proportional.) Note that H only depends on the plane and the projection matrices of the two cameras.

The interesting thing about this result is that by using H we can compute the image of P that would be seen in camera C2 from the image of the point in camera C1 without knowing its three-dimensional location. Such an H is a projective transformation of the plane, also referred to as a homography.

Problem definition

Write files computeH.m and warp.m that can be used in the following skeleton code. warp takes as inputs the target (gallery) image, the image with the replacement graphics, the gallerypoints defining the area in the gallery image that should be replaced, and finally, H, the homography. Note that the homography should map points from the gallery image to the sign image, that way you will avoid problems with aliasing and sub-sampling effects.

The input and target output image are given in Figure 3. You may find the following MATLAB files useful: kron, svn, meshgrid, inpolygon, fix, interp2.

```
I1 = imread('joan_clancy_gallery.jpg');
subplot(1,2,1);
imshow(I1);
load points.mat

H = computeH(gallerypoints1, medlogopoints);
I1 = warp(I1, imread('ucsd_medcenter.jpg'), gallerypoints1, H);

H = computeH(gallerypoints2, logopoints);
I1 = warp(I1, imread('ucsd_logo.png'), gallerypoints2, H);
subplot(1,2,2);
imshow(I1);
```

Report

Run the skeleton code and include the output image in your report. Attach source code for computeH.m and warp.m in the appendix.

For fun (no credit or hand-in)

Instead of inserting a static photograph into the gallery scene, you can insert a movie. To do this, simply break the video up into frames, apply the code that you wrote here to each frame to create

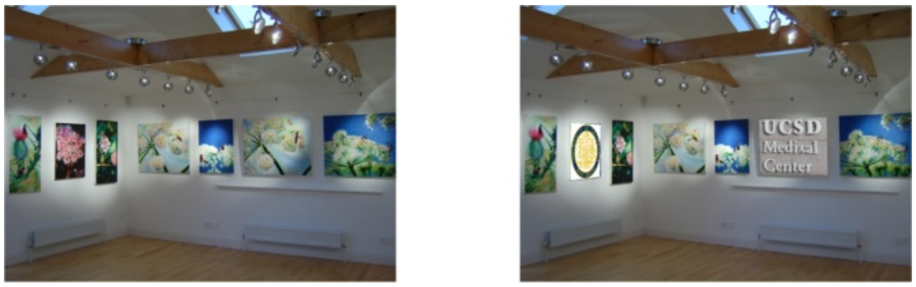


Figure 3: Input image (left) and target (right) for image mapping problem

new frames, and then pack the new frames together to create a new video. It's a bit harder to insert a still image into a video or one video into another video. Why?

Good luck!