EECS 16A Designing Information Devices and Systems I
Fall 2018 Homework 2

This homework is due September 7, 2018, at 23:59.
Self-grades are due September 11, 2018, at 23:59.

Submission Format
Your homework submission should consist of **two** files.

- `hw2.pdf`: A single PDF file that contains all of your answers (any handwritten answers should be scanned) as well as your IPython notebook saved as a PDF.

  If you do not attach a PDF "printout" of your IPython notebook, you will not receive credit for problems that involve coding. Make sure that your results and your plots are visible. Assign the IPython printout to the correct problem(s) on Gradescope.

- `hw2.ipynb`: A single IPython notebook with all of your code in it.

Submit each file to its respective assignment on Gradescope.

1. Lab Grades and Discussion Attendance

Lab grades and discussion attendance will be posted on Gradescope. These will be updated periodically; check the Piazza post for the most recent updates.

For lab grades, please submit a file named `sid.txt` to the "Lab Grades" assignment. This text file should contain only your SID number and nothing else (e.g. no "" marks or extra whitespace). You can find a template in the iPython folder.

For discussion grades, submit the same file `sid.txt` to "Discussion Attendance."

If you use a strange text editor, the autograder will give a strange error message complaining about the unicode format. If this happens, please use a different editor and reupload.

Describe how to find your lab and discussion grades.

2. (PRACTICE) Finding Charges from Potential Measurements

We have three point charges Q_1, Q_2, and Q_3 whose positions are known, and we want to determine their charges. In order to do that, we take three electric potential measurements U_1, U_2 and U_3 at three different locations. You do not need to understand electric potential to do this problem. The locations of the charges and potentials are shown in Figure 1.

For the purpose of this problem, the following equation is true:

$$U = k\frac{Q}{r}$$

at a point r meters away (for some fixed physical constant k; this problem does not require the numerical value of k).

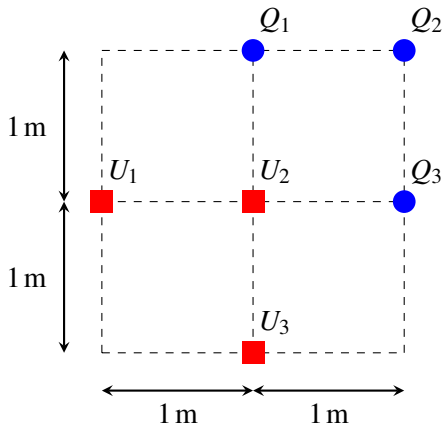


Figure 1: Locations of the charges and potentials.

Furthermore, the potential contributions from different point charges add up linearly. For example, in the setup of Figure 1, the potential measured at point U_2 is

$$U_2 = k\frac{Q_1}{1} + k\frac{Q_2}{\sqrt{2}} + k\frac{Q_3}{1}.$$

Given that the actual potential measurements in the setup of Figure 1 are

$$U_1 = k\frac{4+3\sqrt{5}+\sqrt{10}}{2\sqrt{5}},$$
$$U_2 = k\frac{2+4\sqrt{2}}{\sqrt{2}},$$
$$U_3 = k\frac{4+\sqrt{5}+3\sqrt{10}}{2\sqrt{5}},$$

write the system of linear equations relating the potentials to charges. Solve the system of linear equations to find the charges Q_1, Q_2, Q_3. You may use your IPython notebook to solve the system.

IPython hint: For constants a_i, b_i, c_i, y_i, you can solve the system of linear equations

$$a_1x_1 + a_2x_2 + a_3x_3 = y_1$$
$$b_1x_1 + b_2x_2 + b_3x_3 = y_2$$
$$c_1x_1 + c_2x_2 + c_3x_3 = y_3$$

in IPython with the following code:

```
import numpy as np
a = np.array([
    [a1, a2, a3],
    [b1, b2, b3],
    [c1, c2, c3]
])
b = np.array([y1, y2, y3])
x = np.linalg.solve(a, b)
print(x)
```

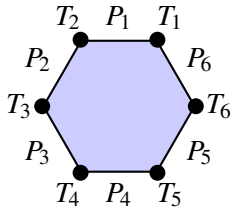
The square root of a number a can be written as `np.sqrt(a)` in IPython.

3. Figuring Out The Tips

A number of people gather around a round table for a dinner. Between every adjacent pair of people, there is a plate for tips. When everyone has finished eating, each person places half their tip in the plate to their left and half in the plate to their right. In the end, of the tips in each plate, some of it is contributed by the person to its right, and the rest is contributed by the person to its left. Suppose you can only see the plates of tips after everyone has left. Can you deduce everyone's individual tip amounts?

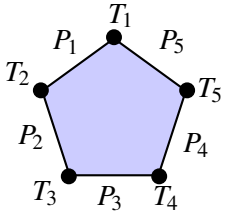
Note: For this question, if we assume that tips are positive, we need to introduce additional constraints enforcing that, and we wouldn't get a linear system of equations. Therefore, we are going to ignore this constraint and assume that negative tips are acceptable.

(a) Suppose 6 people sit around a table and there are 6 plates of tips at the end.



If we know the amounts in every plate of tips (P_1 to P_6), can we determine the individual tips of all 6 people (T_1 to T_6)? If yes, explain why. If not, give two different assignments of T_1 to T_6 that will result in the same P_1 to P_6.

(b) The same question as above, but what if we have 5 people sitting around a table?



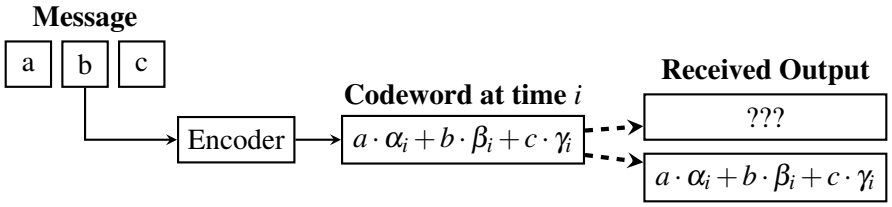
(c) If n is the total number of people sitting around a table, for which n can you figure out everyone's tip? You do not have to rigorously prove your answer.

4. Fountain Codes

Consider a sender, Alice, and a receiver, Bob. Alice wants to send a message to Bob, but the message is too big to send all at once. Instead, she breaks her message up into little chunks that she sends across a wireless channel one at a time (think radio transmitter to antenna). She knows some of the packets will be corrupted or erased along the way (someone might turn a microwave on...), so she needs a way to protect her message from errors. This way, even if Bob gets a message missing parts of words, he can still figure out what Alice is trying to say! One coding strategy is to use fountain codes. Fountain codes are a type of error-correcting codes based on principles of linear algebra. They were actually developed right here at Berkeley! The company that commercialized them, Digital Fountain, (started by a **Berkeley grad, Mike Luby**), was later acquired by Qualcomm. In this problem, we will explore some of the underlying principles that make fountain codes work in a very simplified setting.

In this problem, we concentrate on the case with transmission erasures, i.e. where a bad transmission causes some parts of the message to be erased. Let us say Alice wants to convey the set of her three favorite ideas covered in EE16A lecture each day to Bob. For this, three real numbers a, b, c can be used to represent three different ideas and convey the 3 element vector $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ (let us say there are an infinite number of ideas covered in EE16A).

At each time step, she can send one number, which we will call a "symbol" across, so one possible way for her to send the message is to first send a, then send b, and then send c. However, this method is particularly susceptible to losses. For instance, if the first symbol is lost, then Bob will receive $\begin{bmatrix} ? \\ b \\ c \end{bmatrix}$, and he will have no way of knowing what Alice's favorite idea is.



(a) The main idea in coding for erasures is to send redundant information, so that we can recover from losses. Thus, if we have three symbols of information, we might transmit six symbols for redundancy. One of the most naive codes is called the repetition code. Here, Alice would transmit $\begin{bmatrix} a \\ b \\ c \\ a \\ b \\ c \end{bmatrix}$. How many lost symbols can this transmission recover from? Are there specific patterns it cannot handle?

(b) A better strategy for transmission is to send linear combinations of symbols. Alice and Bob decide in advance on a collection of 3 element transmission row vectors $\vec{v}_i^T = \begin{bmatrix} \alpha_i & \beta_i & \gamma_i \end{bmatrix}$, $1 \leq i \leq 6$. (The ^{T} is the transpose operator, which converts row vectors to column vectors and vice-versa.) These transmission row vectors define the code: at time i, Alice transmits the scalar

$$k_i = \vec{v}_i^T \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \alpha_i a + \beta_i b + \gamma_i c.$$

Let $\vec{k}$ represent the vector Alice sends to Bob, where

$$\vec{k} = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \end{bmatrix}$$

.

Write $\vec{k}$ as a product of a matrix and a vector.

(c) What are the transmission row vectors $\vec{v}_i^T = \begin{bmatrix}\alpha_i & \beta_i & \gamma_i\end{bmatrix}$, $1 \leq i \leq 6$ that generate the repetition code strategy in part (a)?

(d) Suppose now they choose a collection of seven transmission row vectors:

$$\vec{v}_1^T = \begin{bmatrix}1 & 0 & 0\end{bmatrix}, \quad \vec{v}_2^T = \begin{bmatrix}0 & 1 & 0\end{bmatrix}, \quad \vec{v}_3^T = \begin{bmatrix}0 & 0 & 1\end{bmatrix}, \quad \vec{v}_4^T = \begin{bmatrix}1 & 1 & 0\end{bmatrix},$$
$$\vec{v}_5^T = \begin{bmatrix}1 & 0 & 1\end{bmatrix}, \quad \vec{v}_6^T = \begin{bmatrix}0 & 1 & 1\end{bmatrix}, \quad \vec{v}_7^T = \begin{bmatrix}1 & 1 & 1\end{bmatrix}$$

Again, at time i, Alice transmits the scalar $k_i = \vec{v}_i^T \begin{bmatrix}a\\b\\c\end{bmatrix}$.

Bob would like to find the transmitted message. Suppose, using this collection of transmission row vectors, Bob receives $\begin{bmatrix}7 & ? & ? & 3 & 4 & ? & ?\end{bmatrix}^T$. Can you express the problem as a system of linear equations using matrix/vector notation? What was the transmitted message?

(e) They continue to use the transmission row vectors from part (d). Under what conditions, (i.e. what patterns of losses) can Bob still recover the message?

(f) Fountain codes build on the principles explored in this problem. The basic idea used by these codes is that Alice keeps sending linear combinations of symbols until Bob has received enough to decode the message. So at time 1, Alice sends the linear combination using $\vec{v}_1$, at time 2 she sends the linear combination using $\vec{v}_2$ and so on. After each new linear combination is sent, Bob will send back an acknowledgement *if* he can decode her message (i.e. figure out the original $\begin{bmatrix}a\\b\\c\end{bmatrix}$ that she intended to communicate). So clearly, the minimum number of transmissions for Alice is 3. If Bob receives the first three linear combinations that are sent, Alice is done in three steps! But because of erasures, she might hear nothing (he hasn't decoded yet). Suppose Alice used

$$\vec{v}_1^T = \begin{bmatrix}1 & 0 & 0\end{bmatrix}, \quad \vec{v}_2^T = \begin{bmatrix}0 & 1 & 0\end{bmatrix}, \quad \vec{v}_3^T = \begin{bmatrix}0 & 0 & 1\end{bmatrix}$$

as her first three vectors, but she has still not received an acknowledgement from Bob. Should she choose new vectors according to the strategy in part (a) or the strategy in part (d)? Why?

5. Kinematic Model for a Simple Car Building a self-driving car first requires understanding the basic motions of a car. In this problem, we will explore how to model the motion of a car.

There are several models that we can use to model the motion of a car. Assume we use the following kinematic model, given in the following four equations and Figure 2.

$$x[k+1] = x[k] + v[k]\cos(\theta[k])\Delta t \tag{1}$$
$$y[k+1] = y[k] + v[k]\sin(\theta[k])\Delta t \tag{2}$$
$$\theta[k+1] = \theta[k] + \frac{v[k]}{L}\tan(\phi[k])\Delta t \tag{3}$$
$$v[k+1] = v[k] + a[k]\Delta t \tag{4}$$

where

- k, a nonnegative integer, indicates the time step at which we measure the variable (e.g. $v[k]$ is the speed at time step k and $v[k+1]$ is the speed at the following time step)

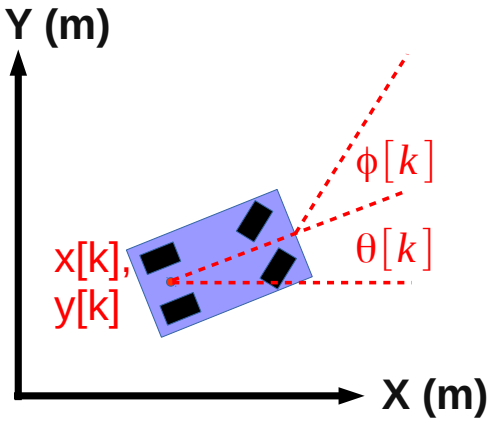


Figure 2: Vehicle Kinematic Model

- $x[k]$ and $y[k]$ denote the coordinates of the vehicle (meters)
- $\theta[k]$ denotes the heading of the vehicle, or the angle with respect to the x-axis (radians)
- $v[k]$ is the speed of the car (meters per second)
- $a[k]$ is the acceleration of the car (meters per second squared)
- $\phi[k]$ is the steering angle input we command (radians)
- Δt is a constant measuring the time difference (in seconds) between time steps $k+1$ and k
- L is a constant and is the length of the car (in meters)

For this problem, let L be 1.0 meter and Δt be 0.1 seconds.

The variables $x[k], y[k], \theta[k], v[k]$ describe the **state** of the car at time step k. The state captures all the information needed to fully determine the current position, speed, and heading of the car. The **inputs** at time step k are $a[k]$ and $\phi[k]$. These are provided by the driver. The current value of these inputs, along with the current state of the vehicle, will determine the state of the vehicle at the next time step.

We note that the problem is nonlinear, due to the sine, cosine and tangent functions, as well as terms including the product of states and inputs.

The purpose of this problem is to show that we can approximate a nonlinear model with a simple linear model and do reasonably well. This is why, despite many systems being nonlinear, linear algebra tools are widely used in practice.

For Parts (b) - (d), fill out the corresponding sections in prob2.ipynb.

(a) We assume that the car has a small heading ($\theta \approx 0$) and that the steering angle is also small ($\phi \approx 0$), where $\approx$ means "approximately equal to." In this case, we could use the following approximations:

$$\sin(\alpha) \approx 0,$$
$$\cos(\alpha) \approx 1,$$
$$\tan(\alpha) \approx 0.$$

where α is the small angle of interest. Here, we use a very simple approximation for small angles; in later classes, you may learn better approximations.

Draw, by hand, graphs of of $\sin(\alpha)$ and $\cos(\alpha)$, for α ranging from $-\pi$ to π. Using these graphs can you justify the approximation we are making for small values of α?

(b) Applying the approximation described in the previous part, write down a linear system that approximates the nonlinear vehicle model given above in Equations (1) to (4). In particular, find the 4 x 4 matrix **A** and 4 x 2 matrix **B** that satisfy the equation given below.

$$\begin{bmatrix} x[k+1] \\ y[k+1] \\ \theta[k+1] \\ v[k+1] \end{bmatrix} = \mathbf{A} \begin{bmatrix} x[k] \\ y[k] \\ \theta[k] \\ v[k] \end{bmatrix} + \mathbf{B} \begin{bmatrix} a[k] \\ \phi[k] \end{bmatrix}$$

(c) Suppose we drive the car so that the direction of travel is aligned with the x-axis, and we are driving nearly straight, i.e. the steering angle is $\phi[k] = 0.0001$ radians. (Driving exactly straight would have the steering angle $\phi[k] = 0$ radians.) The initial state and input are:

$$\begin{bmatrix} x[k] \\ y[k] \\ \theta[k] \\ v[k] \end{bmatrix} = \begin{bmatrix} 5.0 \\ 10.0 \\ 0.0 \\ 2.0 \end{bmatrix}$$

$$\begin{bmatrix} a[k] \\ \phi[k] \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.0001 \end{bmatrix}$$

You can use these values in the IPython notebook to compare how the nonlinear system evolves in comparison to the linear approximation that you made. The IPython notebook simulates the car for ten time steps. Are the trajectories similar or very different? Why?

(d) Now suppose we drive the vehicle from the same starting state, but we turn left instead of going straight, i.e. the steering angle is $\phi[k] = 0.5$ radians. The initial state and input are:

$$\begin{bmatrix} x[k] \\ y[k] \\ \theta[k] \\ v[k] \end{bmatrix} = \begin{bmatrix} 5.0 \\ 10.0 \\ 0.0 \\ 2.0 \end{bmatrix}$$

$$\begin{bmatrix} a[k] \\ \phi[k] \end{bmatrix} = \begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}$$

You can use these values in the IPython notebook to compare how the nonlinear system evolves in comparison to the linear approximation that you made. The IPython notebook simulates the car for ten time steps. Are the trajectories similar or very different? Why?

6. Show It

Let n be a positive integer. Let $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k\}$ be a set of k linearly dependent vectors in $\mathbb{R}^n$. Show that for any $n \times n$ matrix **A**, the set $\{\mathbf{A}\vec{v}_1, \mathbf{A}\vec{v}_2, \ldots, \mathbf{A}\vec{v}_k\}$ is a set of linearly dependent vectors. Make sure that you prove this rigorously for all possible matrices **A**.

7. Image Stitching

Often, when people take pictures of a large object, they are constrained by the field of vision of the camera. This means that they have two options to capture the entire object:

- Stand as far as away as they need to to include the entire object in the camera's field of view (clearly, we do not want to do this as it reduces the amount of detail in the image)
- (This is more exciting) Take several pictures of different parts of the object and stitch them together like a jigsaw puzzle.

We are going to explore the second option in this problem. Daniel, who is a professional photographer, wants to construct an image by using "image stitching". Unfortunately, Daniel took some of the pictures from different angles as well as from different positions and distances from the object. While processing these pictures, Daniel lost information about the positions and orientations from which the pictures were taken. Luckily, you and your friend Marcela, with your wealth of newly acquired knowledge about vectors and matrices, can help him!

You and Marcela are designing an iPhone app that stitches photographs together into one larger image. Marcela has already written an algorithm that finds common points in overlapping images. **It's your job** to figure out how to stitch the images together using Marcela's common points to reconstruct the larger image.

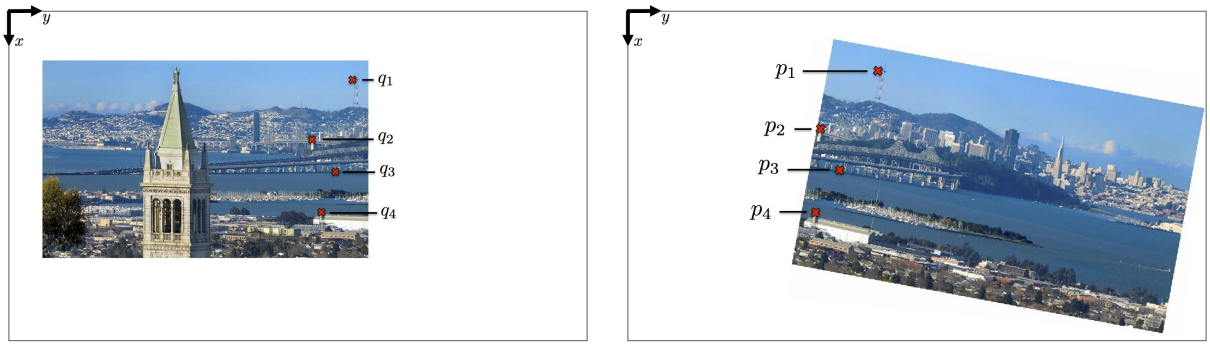


Figure 3: Two images to be stitched together with pairs of matching points labeled.

We will use vectors to represent the common points and these related by a linear transformation. Your idea is to find this linear transformation. For this you will use a single matrix, $\mathbf{R}$, and a vector, $\vec{T}$, that transforms every common point in one image to that same point in the other image. Once you find $\mathbf{R}$ and $\vec{T}$ you will be able to transform one image so that it lines up with the other image.

Suppose $\vec{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$ is a point in one image and $\vec{q} = \begin{bmatrix} q_x \\ q_y \end{bmatrix}$ is the corresponding point in the other image (i.e., they represent the same object in the scene). You write down the following relationship between $\vec{p}$ and $\vec{q}$.

$$\begin{bmatrix} q_x \\ q_y \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{R}_{xx} & \mathbf{R}_{xy} \\ \mathbf{R}_{yx} & \mathbf{R}_{yy} \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \underbrace{\begin{bmatrix} T_x \\ T_y \end{bmatrix}}_{\vec{T}} \tag{5}$$

This problem focuses on finding the unknowns (i.e. the components of $\mathbf{R}$ and $\vec{T}$), so that you will be able to stitch the image together.

(a) To understand how the matrix $\mathbf{R}$ and vector $\vec{T}$ transform a vector, $\vec{v}_0$, consider this similar equation,

$$\vec{v}_2 = \begin{bmatrix} 2 & 2 \\ -2 & 2 \end{bmatrix} \vec{v}_0 + \vec{v}_1. \tag{6}$$

Using $\vec{v}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\vec{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, what is $\vec{v}_2$? On a single plot, draw the vectors $\vec{v}_0, \vec{v}_1, \vec{v}_2$ in two dimensions. Describe how $\vec{v}_2$ is transformed from $\vec{v}_0$ (e.g. rotated, scaled, shifted).

(b) Multiply Equation (5) out into two scalar linear equations. What are the known values and what are the unknowns in each equation? How many unknowns are there? How many independent equations do you need to solve for all the unknowns? How many pairs of common points $\vec{p}$ and $\vec{q}$ will you need in order to write down a system of equations that you can use to solve for the unknowns?

(c) What is the vector of unknown values? Write out a system of linear equations that you can use to solve for the unknown values (you should use multiple pairs of points $\vec{p}$'s and $\vec{q}$'s to have enough equations). Transform these linear equations to in matrix equation, so that we could solve for the vector of unknown values.

(d) In the IPython notebook `prob2.ipynb`, you will have a chance to test out your solution. Plug in the values that you are given for p_x, p_y, q_x, and q_y for each pair of points into your system of equations to solve for the matrix, $\mathbf{R}$, and vector, $\vec{T}$. The notebook will solve the system of equations, apply your transformation to the second image, and show you if your stitching algorithm works. You are not responsible for understanding the image stitching code or Marcela's algorithm.

(e) We will now explore when this algorithm fails. Marcela's algorithm gives us a new set of corresponding points between the images (Figure 4), but the new points are collinear (i.e. we can draw a straight line between them in the image). Marcela suspects that you will run into an issue when solving for your unknown vector. In the IPython notebook `prob2.ipynb`, try to plug in the new corresponding points and solve for new unknown values. What happens?

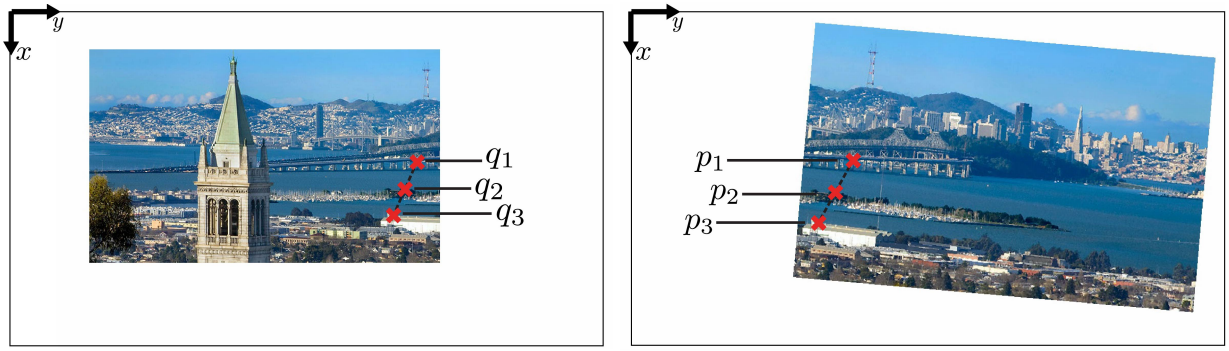


Figure 4: Two images to be stitched together with collinear pairs of matching points labeled.

(f) Now more generally, show that if $\vec{p}_1, \vec{p}_2, \vec{p}_3$ are *collinear*, the system of equations you created in part (c) will have an infinite number of solutions.

Fact: $\vec{p}_1, \vec{p}_2, \vec{p}_3$ are collinear *if and only if* $(\vec{p}_2 - \vec{p}_1) = k(\vec{p}_3 - \vec{p}_1)$ for some $k \in \mathbb{R}$.

8. Homework Process and Study Group

Who else did you work with on this homework? List names and student ID's. (In case of homework party, you can also just describe the group.) How did you work on this homework?