CS 81 - Principles of Robot Design & Programming

Carter Kruse

Homework 6 - Coverage/Exploration, ML, CV

Purpose

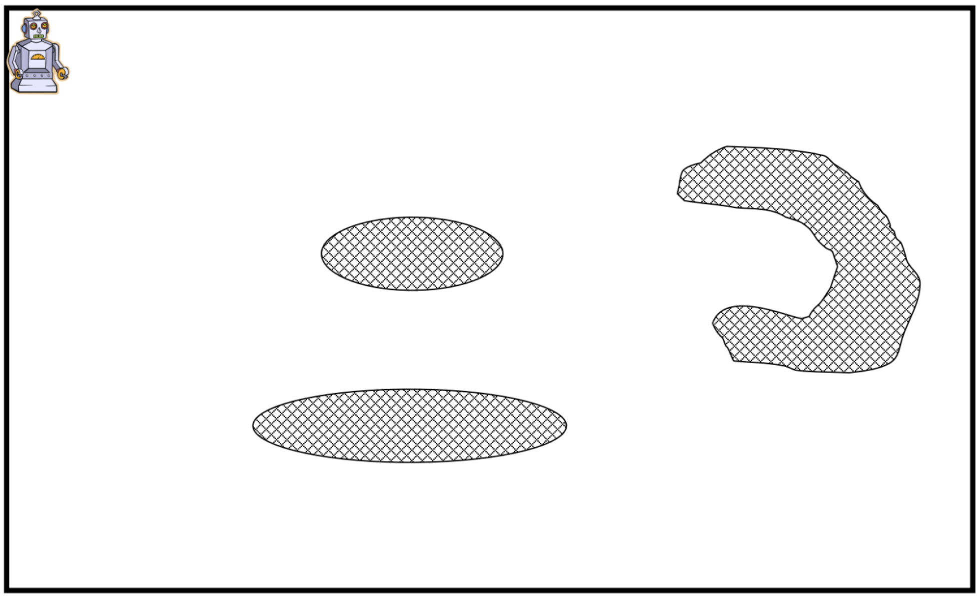
In this assignment, you will put in practice some of the coverage/exploration, machine learning, and computer vision concepts learned in class.

General Instructions

Answer the following questions, showing all the relevant drawings and intermediate math work by uploading a PDF file containing the work. You can include code written by yourself if used for doing some of the repeated calculations, but write down the formulas used on paper. Feel free to add comments in the text box.

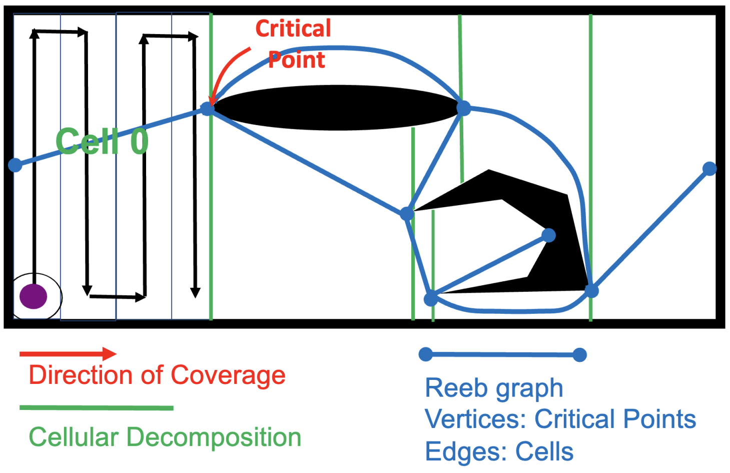
Question 1

A robot positioned in the top left corner needs to cover the following environment.



Draw the Reeb graph on that environment and mark a plausible optimal order of cell coverage. (Hint: Remember that certain edges may be doubled.)

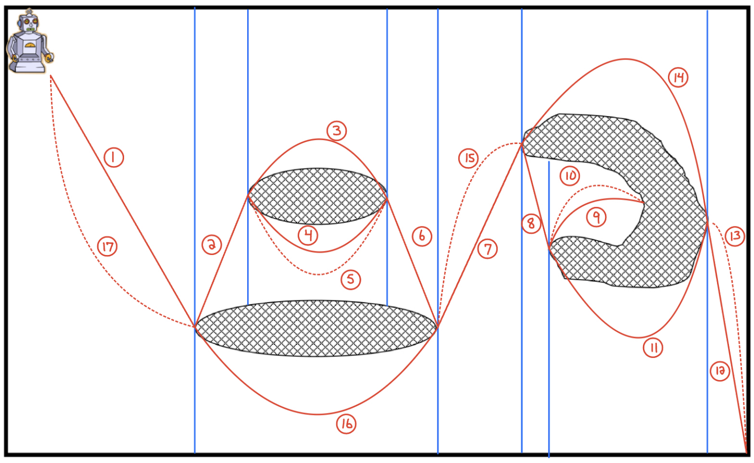
Using a single robot, we may approach the task as indicated:



This systematic approach allows us to consider critical points as decision points for the robot. The goal is to minimize the traveling distance of the robot. Thus, to find an order for traversing the Reeb graph such that the robot would not go through a cell more times than necessary, we consider the Chinese Postman Problem.

The Chinese Postman Problem (CPP), is to find a shortest closed path that visits every edge of a (connected) undirected graph. When the graph has an Eulerian circuit (a closed walk that covers every edge once), that circuit is an optimal solution.

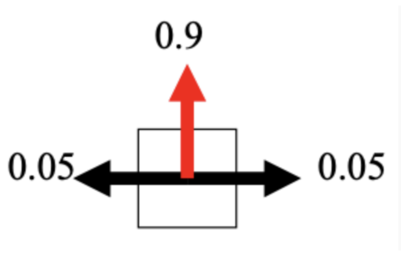
To accomplish the task, we use Boustrophedon Cellular Decomposition (BCD), followed by the Chinese Postman Problem, as shown.



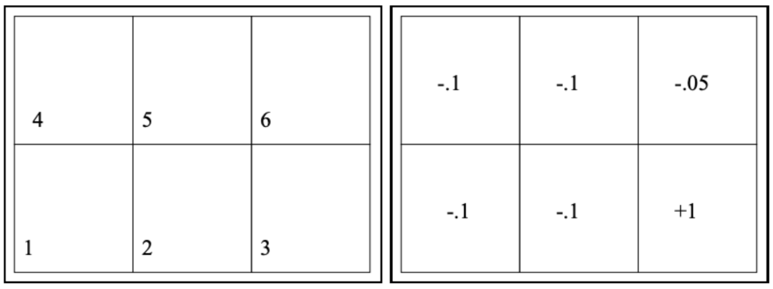
The solution of the CPP guarantees that no edge is doubled more than once. That means that some cells have to be traversed twice. Cells that have to be traversed/covered are divided in half. By dividing the cell diagonally, we control the beginning and end of the coverage.

Question 2

A robot is learning how to act in a 3×2 world. The robot can perform 5 possible actions: north, east, south, west and stay still. The success of the first four actions is with probability 0.9; there is a probability of 0.05 to go to a right angle of the desired direction. See the figure below.



The last action has a success probability of 1. Each state has a reward associated that is gained by the robot arriving to that state, as follows: $R(1:6) = [-0.1, -0.1, +1, -0.1, -0.1, -0.05]$. The only terminal state (a state where the robot "ended the mission", i.e., no action is available) is the third state. See the figure below for the world and the corresponding reward.



(a) States of 3x2 World (b) Rewards of 3x2 World

Answer the following, showing the math calculations. Write down the results of each of the three iterations of value iteration. The value vector is initialized with zeros for all the states, including the terminal state, as indicated in the value iteration algorithm. The discount factor is 0.9.

When an agent is within an environment and makes sequential decisions, at each time step t, the agent takes an action u_t and the world updates given such action, emitting an observation z_{t+1} and reward r_{t+1}. The agent receives this observation and reward, and updates its future actions accordingly.

The goal is to select actions to maximize the total future reward, with the consideration that actions may have long term consequences, and that the reward may be delayed. In other words, it may be better to sacrifice immediate reward to gain more.

Thus, we consider a similar example to the one presented in class, with a different payoff given the actions of the robot. That is, we have a maze, where the agent lives in a grid and walls block the agent's path. Further, there is "noisy" movement, as actions do not always go as planned, as specified. For the purpose of this activity, we consider that if there is a wall in the direction the agent would have been taken, the agent stays put.

Typically, the agent receives rewards at each time step, which includes a small "living" reward (which may be negative) and a big "goal" reward (which could be positive or negative). The objective is to maximize the sum of the rewards, in this stochastic grid model.

While it is reasonable to maximize the sum of the rewards, it is further reasonable to prefer rewards now rather than later, which leads to discounting (rewards in the future). This allows for planning in which we consider the possible future rewards, and was discussed in our class sessions.

Now, let us consider "optimal quantities" which determine the robot's actions. The value (utility) of a state s is given by $V^*(s)$ and equals the expected utility starting in s and acting optimally. The value (utility) of a q-state (s, a) is given by $Q^*(s, a)$ and equals the expected utility starting out having taken action a from state s and (thereafter) acting optimally).

The fundamental operation is to compute (the expected maximum) value of a given state, which is the expected utility under optimal action, and an average sum of the discounted rewards. To this end, the following expressions are helpful.

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^*(s')\right]$$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^*(s')\right]$$

The expression T represents the transition, R represents the reward, and V represents the value.

Using this information, we can perform "value iteration" as indicated by the statement of this exercise. We repeat until convergence, or a terminating condition. Ideally, we will converge to unique optimal values, which allows us to determine the appropriate policy.

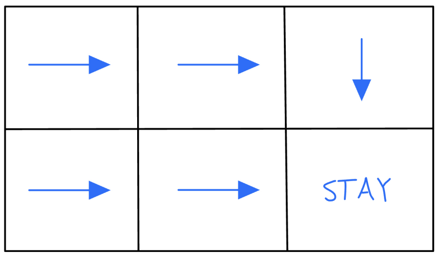
Rather than perform the calculations by hand, an Excel sheet was used to compute the results. Please view the formulas/results as appropriate here.

Write down the best policy after these 3 iterations.

In a deterministic, single-agent search problem, we aim for an optimal plan, or sequence of actions from start to a goal. That is, we aim for a policy π that gives an action for each state. An optimal policy is one that maximizes the expected utility if followed.

In other words, a policy maps the state of the robot to an action, according to the preferences of the robot based on the rewards.

The spreadsheet displays the optimal policy for the agent, given the constraints (set-up) of this particular scenario. This was determined by considering the optimal reward for a given state-action pair after three value iterations. The results are further displayed here.



Question 3

Kira Yamato wants to use reinforcement learning to train her robotic pet "Birdy". She chooses 0.2 for her learning rate and 0.4 for her discount factor. She uses an extremely simple model for her robot that has two states named a and b and three actions named $x, y,$ and z. The Q-table after a few rounds of training is shown.

State	Action	Q
a	x	90
a	y	100
a	z	80
b	x	7
b	y	8
b	z	10

Starting from this state, Kira's robot is in state b and executes action z. As a result of this action, the robot moves to state a and receives a reward of 10.

Write down the computations that Kira's robot would need to perform to update its Q-table, and fill in the new Q-table that results from this update.

Reinforcement learning incorporates training experience (online interactions with the environment), tasks (to collect as much reward as possible), and performance (indicating the amount of rewards). Essentially, it is a way for an agent to learn to make good sequences of decisions.

The agent learns through trial-and-error interactions, with the goal of maximizing the amount of reward received from the environment. This involves computing a value function $Q(z_t, u_t)$ mapping state-action pairs into expected future payoffs.

In the reinforcement learning model, a Markov decision process is used, which is similar to planning formulation, with states X, actions U, transitions (probability distribution) $P(x_{k+1}|x_k, u_k)$ and rewards $P(r_{k+1}|x_k, u_k)$. That is, a forward kinematics model is used, and the current state depends solely on the previous.

Reinforcement learning typically considers an *infinite horizon* instead of a specific goal region, transitions are probabilistic because they are *unpredictable*, and rewards are the duals of cost.

When considering an optimal policy, the policy is chosen in such a way that the robot's discounted future reward is maximized.

$$R = \sum_{i=k}^{\infty} \gamma^{k-i} r_i$$

where $\gamma \in (0, 1)$ is the discount factor that weighs immediate rewards compared to future rewards. Further information regarding reinforcement learning is provided on the slides from class.

As highlighted previously, in reinforcement learning, we still assume a Markov decision process (MDP), with a set of states $s \in S$, a set of actions (per state) A, a model $T(s, a, s')$ and a reward function $R(s, a, s')$. The twist is that we do not know T or R, and thus, do not know which states are good or what the actions do. As such, the aim is to actually try the actions and states to learn, in a process called "online learning".

The Q-learning algorithm learns Q values as the robot selects new actions and observes the world. The Q table is initialized, and an action u_t is executed. The new state s_{t+1} is observed and the reward r_{t+1} is received. Following this, the Q table is updated, according to the following.

$$Q(s_t, u_t) \leftarrow (1 - \alpha) Q(s_t, u_t) + \alpha \left(r_{t+1} + \gamma \left[\max_{u \in U} Q(s_{t+1}, u) \right] \right)$$

The learning rate parameter α controls how rapidly Q is changed.

With this information, we may solve the activity. The robot is in state $s_t = b$ and executes action $u_t = z$. The reward from such action is $r_{t+1} = 10$, and the new state is $s_{t+1} = a$. The learning rate is $\alpha = 0.2$ and the discount factor is $\gamma = 0.4$. Thus, we have the following.

$$Q(b, z) \leftarrow (0.8) Q(b, z) + (0.2) \left(10 + (0.4) \left[\max_{u \in U} Q(a, u) \right] \right)$$

According to the table, this is as follows.

$$Q(b, z) \leftarrow (0.8)(10) + (0.2)(10 + (0.4)(100))$$

Thus, after performing the calculations, we may update the table to reflect the appropriate value.

$$Q(b, z) \leftarrow 18$$

Now that the computations that Kira's robot would need to perform to update it's Q-table, given the state, action, and result, we may fill in the new Q-table that results from this update.

State	Action	Q
a	x	90
a	y	100
a	z	80
b	x	7
b	y	8
b	z	18

Question 4

Assume that you have a GoPro camera which specifications are as follows.

Sensor Size: 1/2.5-Inch (5.75 mm $\times$ 4.28 mm)

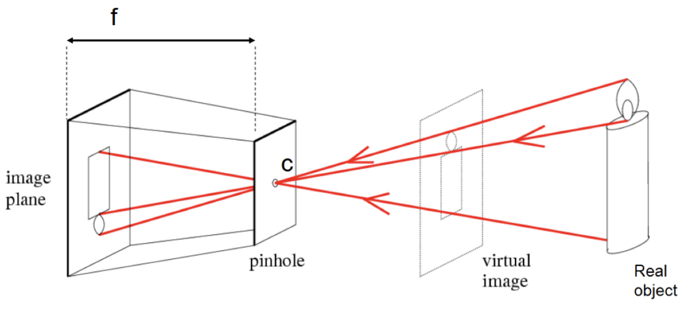
Focal Length: 5 mm

Resolution: 2592 (Cols) $\times$ 1944 (Rows) Pixels

The principal point is exactly centered at $(x, y) = (0, 0)$ on the image plane.

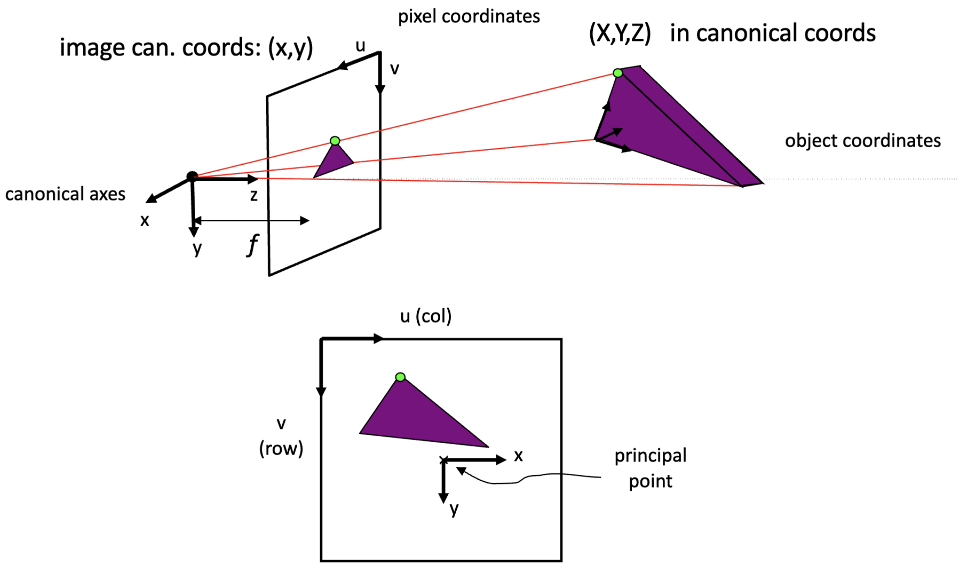
Showing the derivation (mathematically), write the projection matrix (in pixels).

In computer vision, the sensor model used plays a significant role on the way data is interpreted. In this situation, we consider a pinhole camera model, which takes a point from the 3D world an impresses it onto a 2D image that is the result. This is shown as follows.



f = Focal length
c = Optical center of the camera

In this way, the pinhole camera model describes the relationship between the coordinates of 3D points of objects in the world and its projection onto the image plane of an ideal pinhole camera.



At this point, we may consider similar triangles to determine the following: $x = \frac{fX}{Z}, y = \frac{fY}{Z}$

To determine the pinhole camera (projection) matrix, we use what is given on the in-class slides, with the corresponding pixel conversion.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} s_x f & 0 & 0 & 0 \\ 0 & s_y f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

The scaling factor is typically in pixel / mm. This is recovered by considering the image width x and height y divided by the sensor width and height (respectively) in mm.

Thus, given the relevant information, the projection matrix is written as follows.

$$\begin{pmatrix} s_x f & 0 & 0 & 0 \\ 0 & s_y f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \left(\frac{2592px}{5.75mm}\right)(5mm) & 0 & 0 & 0 \\ 0 & \left(\frac{1944px}{4.28mm}\right)(5mm) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2253.913px & 0 & 0 & 0 \\ 0 & 2271.028px & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Write in pixel coordinates the four 3D points that are at $(12, 11, 5)$, $(8, 11, 5)$, $(8, 9, 5)$, $(12, 9, 5)$ with the camera at $(10, 10, 0)$ with roll (x), pitch (y) , yaw (z), of $(0, 0, 45)$ degrees, all in the world reference frame. (Please remember that the coordinates are given in the world reference frame and the result is in homogeneous coordinates and the result should be adjusted by the scale.)

As the coordinates are given in the world reference frame, a transformation matrix is required to convert them to the "camera" reference frame. This transformation matrix is as follows.

$$_{camera}T^{world} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & x \\ \sin(\theta) & \cos(\theta) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The appropriate values are given as follows $\theta = -\frac{\pi}{4}$, $x = -10\sqrt{2}$, $y = 0$, as the transformation is from the world reference frame to the camera reference frame, not the other way around. Thus, the transformation (rotation and translation) is given in terms of the camera.

When we apply this transformation matrix to the 3D points in the global reference frame before applying the projection matrix, the results are as follows.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} s_x f & 0 & 0 & 0 \\ 0 & s_y f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} {}_{camera}T^{world} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} s_x f & 0 & 0 & 0 \\ 0 & s_y f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & x \\ \sin(\theta) & \cos(\theta) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Now, we may input the values given.

$$\begin{pmatrix} 4781.272 \\ -1605.859 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 2253.913px & 0 & 0 & 0 \\ 0 & 2271.028px & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\left(-\frac{\pi}{4}\right) & -\sin\left(-\frac{\pi}{4}\right) & 0 & -10\sqrt{2} \\ \sin\left(-\frac{\pi}{4}\right) & \cos\left(-\frac{\pi}{4}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 12 \\ 11 \\ 5 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} -1593.757 \\ 4817.578 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 2253.913px & 0 & 0 & 0 \\ 0 & 2271.028px & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\left(-\frac{\pi}{4}\right) & -\sin\left(-\frac{\pi}{4}\right) & 0 & -10\sqrt{2} \\ \sin\left(-\frac{\pi}{4}\right) & \cos\left(-\frac{\pi}{4}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 8 \\ 11 \\ 5 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} -4781.272 \\ 1605.859 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 2253.913px & 0 & 0 & 0 \\ 0 & 2271.028px & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\left(-\frac{\pi}{4}\right) & -\sin\left(-\frac{\pi}{4}\right) & 0 & -10\sqrt{2} \\ \sin\left(-\frac{\pi}{4}\right) & \cos\left(-\frac{\pi}{4}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 8 \\ 9 \\ 5 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1593.757 \\ -4817.578 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 2253.913px & 0 & 0 & 0 \\ 0 & 2271.028px & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\left(-\frac{\pi}{4}\right) & -\sin\left(-\frac{\pi}{4}\right) & 0 & -10\sqrt{2} \\ \sin\left(-\frac{\pi}{4}\right) & \cos\left(-\frac{\pi}{4}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 12 \\ 9 \\ 5 \\ 1 \end{pmatrix}$$

Given that the homogeneous coordinates should be adjusted by scale (specifically 5, which is representative of the Z coordinate in each case), let consider the following points. We apply the scale (by dividing by 5).

$$(4781.272, -1605.859) \to (956.254, -321.172)$$
$$(-1593.757, 4817.578) \to (-318.751, 963.516)$$
$$(-4781.272, 1605.859) \to (-956.254, 321.172)$$
$$(1593.757, -4817.578) \to (318.751, -963.516)$$

Now, we must recall that these points are given in the x, y coordinate system. To convert to the u, v coordinate system (which corresponds to the columns and rows of the image sensor, in pixels), we must apply a shift factor, as follows.

$$x_{shift} = \frac{2592}{2} = 1296$$
$$y_{shift} = \frac{1944}{2} = 972$$

This produces the following u, v pixel locations (rounded to the nearest integer).

$$(2252, 651), (977, 1936), (340, 1293), (1615, 8)$$