Exercise 5: Draw Prepared Images

Plot sample image

Several sample images have already been acquired and processed for you to draw. Load one of them in MATLAB to see what the original image looks like. Start out with a drawing of the MathWorks logo.

```
imshow('MathWorksLogo.jpg')
axis on
title('Original Image')
```

Load pathways for sample image

In addition to the original image, a MAT-file is provided for you that contains the coordinates of pixels from this image that describe the line traces in the image. Load this data into the workspace.

```
load MathWorksLogo.mat segmentsPix xLimPix yLimPix
nSegments = length(segmentsPix);
```

Plot pixels chosen from sample image

Loop through the cell array `segmentsPix`. In each cell, extract the data, which is a two-column list of pixel coordinates. Plot each coordinate path. Call `hold on` after each `plot` so that the following `plot` command will add to the current plot instead of replacing it.

```
for ii = 1:nSegments
    coords = segmentsPix{ii};
    plot(coords(:,2),coords(:,1),'-*')
    hold on
end
hold off
axis equal ij
title('Paths to draw (pixels)')
```

Convert pixel coordinates to physical distances

In order to draw the image on the whiteboard, you'll have to convert from pixels to distance units. In the previous exercise, you identified the limits of the whiteboard in meters. Load these values from the MAT-file `WhiteboardLimits.mat` that you created in the previous exercise.

```
load WhiteboardLimits.mat xLim yLim
```

For each point in each line trace, convert the coordinates from pixels to meters using the `transformPixelsToMeters` function. Also, choose what fraction of the total available drawing space you want to use.

```
fraction = 0.7;
segmentsMeters = transformPixelsToMeters(segmentsPix,xLim,yLim,xLimPix,yLimPix,fraction);
```

Plot paths in meters

Just as you plotted the line traces when they were defined in pixel coordinates, now plot the line traces as they are defined in meters. The plot should look the same as before. However, now the units on the x and y axes should be different, and they should be bounded within the limits of the drawable area as defined by `xLim` and `yLim`.

```
for ii = 1:nSegments
    xy = segmentsMeters{ii};
    plot(xy(:,1),xy(:,2),'-*')
    hold on
end
hold off
axis equal ij
title('Paths to draw (meters)')
```

Reduce size of each segment

At this point, each segment may be composed of a large number of points that are very close together. It would be slow to try to draw all of the points we have defined. Instead, you can define the segment with a smaller number of points that still define roughly the same path. Specify a minimum distance between consecutive points and then remove points from each segment such that all points are at least that far apart. Here, we're using 2 mm (0.002 m) as the distance, but you can adjust this depending on how detailed your drawing is.

```
radius = 0.002; %Max distance between points to draw (meters)
for ii = 1:nSegments
    segmentsMeters{ii} = reduceSegment(segmentsMeters{ii},radius);
end
```

Plot reduced paths in meters

Plot the line traces one more time to observe the reduction of data and verify that the path still looks like the original drawing.

```
for ii = 1:nSegments
    xy = segmentsMeters{ii};
    plot(xy(:,1),xy(:,2),'-*')
    hold on
end
hold off
axis equal ij
title('Paths to draw (meters)')
```

Note: If the shape looks drastically different in the above plot, decreasing the radius value and repeating the last two sections can help retain more features from the original drawing.

Identify initial position in Z

Hang the robot on the whiteboard. Measure the initial string lengths *L_i*. Run this section of code, and enter these lengths in the dialog box.

```
Z_i = initialPosition();
```

Convert distances to encoder counts

Use the `xyToCounts` function created in a previous exercise to convert each coordinate trace from units of meters to encoder counts. Store this new set of line traces in a new variable `segmentsCounts`.

```
load RobotGeometry.mat Base
segmentsCounts = cell(size(segmentsMeters));
for ii = 1:nSegments
    segmentsCounts{ii} = xyToCounts(segmentsMeters{ii},Z_i,Base);
end
```

Connect to hardware

Connect to the Arduino and peripherals as you have done in previous exercises.

```
a = arduino;
carrier = addon(a,'Arduino/MKRMotorCarrier');
s = servo(carrier,3);
mL = dcmotor(carrier,2);
mR = dcmotor(carrier,1);
eL = rotaryEncoder(carrier,2);
eR = rotaryEncoder(carrier,1);
resetCount(eL)
resetCount(eR)
```

Load the servo position values. Raise both markers.

```
load ServoPositions.mat LeftMarker RightMarker NoMarker
writePosition(s,NoMarker)
```

Draw image on whiteboard

In a loop, draw each of the line traces stored in the `segmentsCounts` variable. Each cell in the variable corresponds to a line trace. For each trace, extract the counts data. Then move the robot to the first position, lower the marker, and draw all points in that trace. Finally, raise the marker before proceeding to draw the next segment.

```
for ii = 1:nSegments
    % Get counts for current segment
    countList = segmentsCounts{ii};
    % Move to first position and lower marker
    moveToCounts(countList(1,:),mL,mR,eL,eR)
    writePosition(s,LeftMarker)
    % Move to all positions of current segment
    moveToCounts(countList,mL,mR,eL,eR)
    % Raise marker
    writePosition(s,NoMarker)
end
```

Clear hardware variables when you're done using them.

```
clear a carrier s mL mR eL eR
```