Needle Detection Algorithm

Sam Lovvoll, Prasanna Kumar Peram

Abstract

There is a need to identify and locate a curved needle in the context of a surgery robot. There is currently an algorithm to precisely identify the entire needle, but it needs a seed position to begin the search. The current algorithm being used to retrieve that seed position is inaccurate, hallucinating needles where there aren't, and ignoring existing needles. Our algorithm is to take an unprocessed image containing possibly several needles and return the coordinates in mm relative to the camera of the needles. First, we use some gaussian blur, Hough transform to detect the edges of the needle. Then, with those pixels isolated, we can perform a constricted Hough transform on those pixels to find the rotation of the needle. With the rotation values known, we can geometrically determine the relative position of the needle given the diameter of the needle (which is provided as an input parameter). Due to time constraints, we were unable to complete the algorithm, but were able to complete aspects of it, which will be discussed in the results section.

Introduction

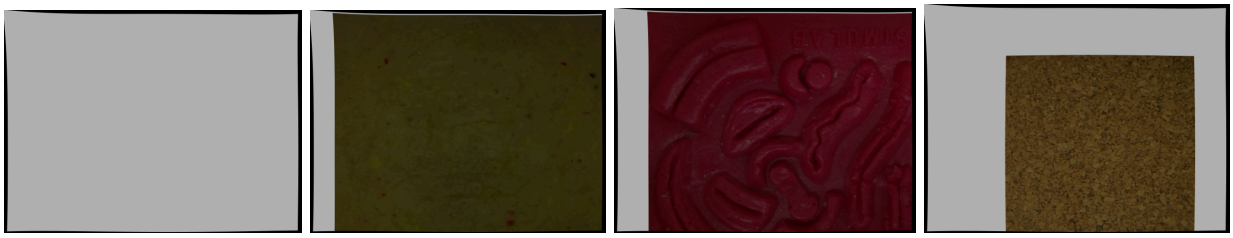
This algorithm will be used to identify curved needles for medical use with a DaVinci surgery robot. The algorithm should be able to run in real time (around 20 frames per second), and should have a low error rate. Due to the context in which it is being used, the algorithm should be able to work even with complex backgrounds. These backgrounds will have other curved objects, being internal organs in the application setting.

The needle is a semi-circle, with a pointed end and a flat end. The diameter is given as an input parameter to the algorithm in mm. In the application case, the needle has a metallic texture and is shiny, but some of the test images have a black needle (more on the test images later). There are two, stereo cameras being used to take pictures on the robot, and these cameras are 5mm apart. Since the FOV of the cameras are unknown, it is left as an input parameter for the algorithm.

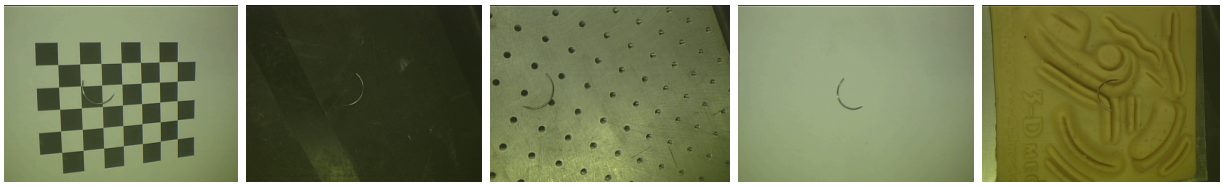
Although there is plenty of literature on ellipse detection, this problem is unique due to the fact that the needle is a partial ellipse, being half of a perfect circle. Due to this condition, a number of ambiguous cases can be disputed, and the search space for a matching function can be significantly reduced. This is filling a hole in current knowledge, optimizing the needle search for this specific case.

There are two batches of test images, simulated and real. The simulated test images are taken over set backgrounds and the needle is a 3D model of the expected shape, still with a pointy end and a flat end. The rendered needle is solid black, rather than metal. Additionally, the

rendered needle has a diameter of 13mm. Alongside the simulated images, x y and z coordinates are given, along with roll pitch and yaw for rotation. There are 100 pairs of simulated images, where each pair has a left and right side, for stereo vision. 75 of the images are duplicates of the first 25 with different simulated backgrounds. The 4 different simulated backgrounds are shown here:



The second set of images, the real ones, were taken with the camera that the application will be running on, the medical robot. The needle in the real set of images is silver, and also has a diameter of 13mm. There are 5 different image pairs for the real set of images, and no repeat backgrounds. These backgrounds are different from the backgrounds in the simulated images, though the 5th background is similar to the third background from the simulated images. The left camera for the 5 real images are as follows:



These images were provided to us by Thomas Shkurti.

Our approach had two distinct sections, where in the first part, pixels that likely contained a needle-like object were selected, and in the second part, those pixels were fit to the most likely needle orientation for positioning.

Once a selection of pixels that are deemed to be of a needle are segmented, some landmark features of the needle need to be selected to orient the rest of the algorithm. The pixel location of both endpoints of the needle are found by starting at a random pixel on the needle, then finding the pixel furthest away from that initial one, which is the first endpoint, then finding the pixel furthest from the first endpoint, which is the second endpoint. From those two, the midpoint of the needle can be found by averaging those two points.

Next, the image is rotated so that the endpoints of the needle line up vertically, this is the first degree of rotation, and eliminates that variable from the search. Then the pixel radius of the needle is found by finding the furthest pixel from the midpoint of the needle. Regardless of the orientation of the needle, there is always one point in the image where the needle touches the bounding circle of the needle. The bounding circle is a circle where, again, regardless of orientation, no part of the needle will go past. Once the pixel radius is calculated, the image is cropped so that the midpoint of the needle is in the center of the image and the image

dimensions are 2r x 2r (r being the pixel radius of the needle). This constrains the possible needle orientations to 2 dimensions, rotation around the vertical and rotation along the horizontal since all positional elements are eliminated by constraining the needle to be centered and consistent size.

Next, a Hough transform is performed on the pre-processed image. This Hough transform is applied over two variables, rotation around the vertical axis, and rotation around the horizontal axis. From now on, rotation around the vertical axis will be referred to as angle α, and rotation around the horizontal axis will be referred to as angle β. The range of α is 0 to π, and the range of β is -π to π.

From the Hough transform α and β, the geometric position of the needle can be directly calculated. First, using α and the known diameter of the needle, the length of a line that when projected into 2D space, is the same as the line between the two endpoints is calculated, but this line is actually perpendicular to the camera, and can be used to find the distance in mm from the needle to the camera. Using this distance and angular properties of the image, we can calculate the z y and x coordinates of the midpoint of the needle.

Methods

Pixels that likely contained a needle-like object were selected:-

1. First we will convert the RGB image to Grayscale image
2. Apply the gaussian blur:
    - Why is gaussian blur:-Gaussian blur is a commonly used image processing technique that is applied to smooth out an image and reduce noise.
    - We need to specify the kernel size for blurring. A larger kernel size results in stronger smoothing.
3. Edge Enhancement with Bilateral Filtering:
    - Bilateral filtering is an image processing technique used to smooth images while preserving edges and important details.
    - During bilateral filtering, each pixel in the image is processed by combining its neighbors' values using both the spatial and range Gaussian kernels.
    - Diameter of the pixel neighborhood used for filtering, Larger values consider more pixels in the neighborhood.
4. Detect Edges with Canny Edge detector:-
    - The primary purpose of the Canny edge detector is to identify edges in an image. Edges represent significant changes in intensity or color, such as object boundaries, contours, and textures.
5. Line Detection using Probabilistic Hough Transform:-
    - What is hough transform :- The Hough Transform is a technique used to detect shapes, such as lines, circles, and other geometrical patterns, in images. In the standard Hough Transform for line detection, each edge pixel in the image is

transformed into a parameter space representation (Hough space) that represents possible lines in the original image.

- What is Probabilistic hough transform:- it is a variation of the Hough Transform that focuses on detecting lines efficiently, especially in cases where the entire parameter space is not required for line detection.Unlike the standard Hough Transform, which considers all possible lines in the image, the Probabilistic Hough Transform samples a subset of edge points and performs line detection probabilistically based on these samples.
- While calculating Angle resolution in radians of the Hough accumulator, Distance resolution in pixels of the Hough accumulator,Minimum number of intersections in the Hough grid cell, Minimum line length in pixels to be considered a line, Maximum allowed gap between line segments to be considered a single line places a crucial role.

For Example

1. Simulated images with clear background

Original image line detection

2. Unfortunately by this method we are not able to detect the lines with 5 real images.

The input for the coordinate detection part of the process should be a binary image, where the pixels that contain a needle are 1, and any background pixels are 0. The resolution can be anything, and the width of the needle does not have to be any specific thickness. The following

are the steps I used to get the coordinates from the camera position to the needle position. All of the following was done in matlab.

First step in the process is to locate the pixel locations of the endpoints of the needle. As mentioned in the introduction, first take a random pixel on the needle, then find the furthest pixel from that within the needle. This is done through geodesic dilation, where, starting from the initial pixel as the seed in the accumulator, the accumulator is dilated with a circular structuring element with a pixel radius of 5. Then the dilated accumulator is combined with the actual image with logical and, only keeping pixels where both are 1. This process continues until the accumulator and original image are the same. At this point, we get the last iteration of the process, where all pixels still left as 1 in the original image and 0 in the accumulator are considered potential candidates for the furthest pixel, so one is chosen at random. This process is done again, this time with the first endpoint as the seed, returning the second endpoint. With these, the pixel location of the midpoint of the needle can be found by averaging the two midpoint locations.

Next, using the atan2 function, I find the angle from one endpoint to the other and I rotate the image by the negative of that angle using imrotate to align the endpoints. Next, I need to find the radius of the needle section. This is done in two parts, the first being finding the furthest pixel from the midpoint. Since the midpoint is not guaranteed to be on one of the needle pixels, the process with geodesic dilation does not work for this, so instead I did a single pass of all the pixels in the needle, returning the one with the highest distance from the midpoint. Then, I apply a thinning algorithm to the needle, continuously applying morphological opening with a circular structuring element with progressively larger radii until the needle begins to disappear (using a threshold of >10% of the needle disappearing), then erodes the needle with a structuring element one smaller than the one that caused >10% of the needle to disappear when closing is applied. Additionally, this radius is returned with the thinning function, and the outer radius is converted to a “true” radius by subtracting the thinning radius from the outer radius. Finally, the image is cropped to have the midpoint be in the center of the image and the image to span 1 true radius in every cardinal direction, leading to an image size of 2r x 2r.

From this point, the Hough transform algorithm can be applied. Although this process is not one invented by me, the algorithm has no known libraries I could use to apply it specifically for searching for a rotated semi-circle, so I coded my own implementation. The algorithm goes over every 1 pixel in the image, and marks down in the accumulator every possible set of arguments that could lead to a needle containing this pixel. For a generic ellipse search problem, a Hough transform would have 6 degrees of freedom, 3 dimensions of size, and 3 degrees of rotation. But after constraining the needle to be in the center of the screen and cropped so that the pixels fit within a bounding circle, that reduces it to 3 degrees of rotation, and by manually rotating the image to line up the endpoints, it reduces it down to 2 degrees. Any pixel location belonging to a needle under these constraints can be determined with a 3 variable function, two of which being the rotation values α and β, and a third being t. While the needle is completely defined by α and β, a specific point on the needle needs a third value to determine where on the needle it is. The formula to determine the α and β for any given t is as follows:

Equation 1:

$$\alpha = cos^{-1}(\frac{x}{cos(t)})$$

Equation 2:

$$\beta = 2tan^{-1}(\frac{sin(\alpha)cos(t) - \sqrt{sin^2(\alpha)cos^2(t) + sin^2(t) - y^2}}{sin(t) + y})$$

Where x and y are the x and y positions of the pixel, and are constrained to have a range/domain of (-1, 1] by linear mapping of the coordinate values. t is determined by simply iterating through values from π/2 to 3π/2 (this range creates a semi-circle rather than a full ellipse, as these equations are applicable to finding any ellipse defined similarly to our needle but being a full circle rather than a half circle). My implementation of this does 300 evenly spaced iterations of t. This is to ensure a solid line is drawn, as certain parts of the function have a very high derivative. Additionally, rather than adding 1 to the accumulator for each iteration of t, I set the accumulator to 1. At the end of iterating t, the accumulator temporary accumulator layer is added to the full accumulator of all pixels. This is to prevent the uneven derivative of the function from skewing the results of the Hough transform. Once all the pixels have been iterated over, the highest values of α and β are returned as the most likely fits.

Calculating the x, y and z coordinates is quite simple given the rotation value α. First, we need to use the known distance between the endpoints, the diameter, and project that to a line perpendicular to the screen, this distance is ds. This can be done by taking the sin of α and multiplying it by the input diameter of the needle, since α is the angle that the needle is rotated towards or away from the camera. See figure 1. Then we need to again project this line so it's perpendicular to the center of the screen, we do this with Equation 3. The distance of this line is dp, as seen in figure 2. From there, we can use Equation 4 to calculate the distance to the midpoint of the needle.

Equation 3: dp = cos(a1+a2-pi/2)

Equation 4: dist = dp * tan(a1)

Figure 1

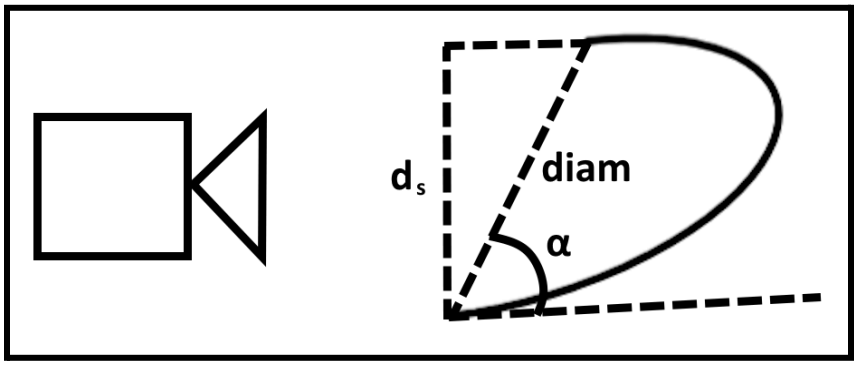
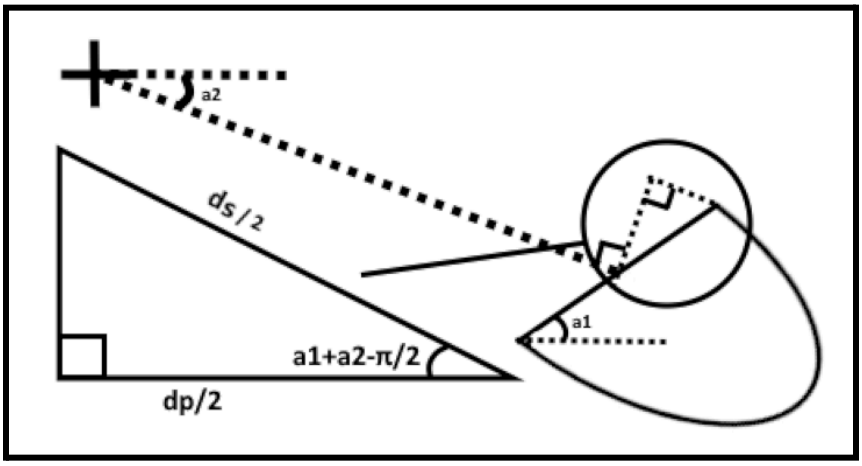


Figure 2



With the distance from the camera to the needle midpoint (dist) known, we can now start calculating the coordinates. As seen in the figure below, z=cos(ω)*dist. ω is calculated by using the known FOV of the camera to calculate a rads per pixel variable. This, like the name implies, is the radians per pixel, so ω = rads per pixel * the pixel distance between the center of the screen and the midpoint of the needle. (for the figure, assume the z axis stretches directly outwards from the camera, and the center of the screen is the point next to the camera). Likewise, X = sin(ω)*dist*cos(λ) and Y = sin(ω)*dist*sin(λ).

Results and Discussion

The needle edges detection methods which we tried are only working for clean background simulated images and are not working for noise background simulated images and real images also.

Due to time constraints, we were not able to calibrate the coordinate detection part of the project. In order to calibrate the results, we would need to know the FOV of the camera being used to take the pictures. This was left as a variable in the code to set, but without the FOV or adequate time, we were not able to confirm that the coordinates resulting from the code are accurate.

Conclusion

Conclusion

Given more time, the algorithm could be changed to detect multiple needles that are overlapping, as is, since the way that we detect needles is first a general runthrough to find needle-like pixels and then process each group of detected pixels, if two needles overlap, they will be considered part of the same group. Additionally, it would be useful to return a needle-ness confidence estimate with each detected needle.

Roles

Prasanna Kumar - To detect the needle like objects and endpoints of the needle
Sam Lovvoll - To process the detected needles and return the coordinate of the midpoint