Laser Curve Tracing for Robotic Arms

Timothy K. Findling, Marius C. Silaghi

Florida Institute of Technology
Computer Science
150 W University Blvd
Melbourne, FL 32901

tfindling2014@my.fit.edu, msilaghi@fit.edu

ABSTRACT

Applications such like soldering require robotic arms to follow the soldering line as present on a surface. We assume here that the surface to be soldered is remote from the robotic arm, that the soldering is performed via a laser beam, and that the line to be followed can be an irregular curve that may self-intersect (e.g. a crack in the material).

We describe research conducted using a robotic arm pointing a laser for tracing a remote line on a smooth surface. The trace is converted to a one pixel width skeleton line generated from images using a hit-and miss algorithm. The robotic arm guides the laser dot along a series of target positions based on a set of processed line segments. A camera is used to validate and correct the movement of the robotic laser arm by measuring position accuracy. Considered methods are compared based on various proposed metrics quantifying the areas traversed redundantly and the areas missed.

Keywords
Robotic Arm, laser, camera, skeletonize, templates, line segments

1 INTRODUCTION

Automating a welding process and maintaining a good welding quality requires the alignment of the torch along a welding seam. A robotic arm that guides the welding torch must be able to accurately follow a welding seam and compensate for tolerances in the machinery and local distortions in materials.

We address the problem of automating the control of a robotic laser arm that is tasked with soldering a crack on a material posted at a certain distance from the arm. The robot is supposed to solder the crack using a laser beam directed at the crack. The laser has to be moved along the crack to produce a good welding. The robot detects the crack using a camera. It processes the image and directs its laser beam based on the visual parameters estimated using its camera and the estimations of the point where the beam intersects the surface.

Many complex challenges can occur in this setting, such as surface irregularities and heat based fumes that can blur vision. The surface may also suffer deformations due to heat. In this research we assume a simpler case where all these additional complications are already solved and we just have to ensure that the torch is correctly directed and following the crack. One has to minimize the number of defects consisting of the laser abandoning the line and welding already correct areas of the surface, or skipping some segments of the crack. A couple of algorithms are investigated and their efficiency is measured by comparing the areas welded without need, and the total length of skipped segments.

In the next section we describe the related work and background concerning robotic soldering and visual line processing and tracing algorithms. In Section 3 we describe the addressed problem formally. The techniques investigated in this research are introduced in Section 4. After describing data collection and precision experiments with a laser robotic arm, we conclude with an analysis of the obtained quality and potential future work.

2 BACKGROUND

Robotics can be classified into two categories, servo and non-servo robots. Servo robots operate in a closed loop

controlled environment and non-servo robots operate in an open loop controlled environment. Robots that operate in an open loop controlled environment have discrete check points and are rigid in their preprogrammed operations. These robots cannot adapt to changes in their environment. Robots that operate in a closed loop environment are much more flexible to changes in their environment and find their applications in computer numerical control (CNC) of milling machines, painting, assembling, bio-medical, remote controlled mobile, inspecting and welding [2, 1]. Industrial applications further expand into laser mapping, distance measuring and target tracking, as well as laser cutting.

A laser visual sensing system for welding with robotic arms was described in [3]. Difficulties in laser tracking welding seams arise from variations in the depth of the seam and deviations in the surface reflectivity. The study in [3] includes research of a Missing-Point algorithm that interpolates a path where target points are missing. Laser spot detection is further described in [4], and a comprehensive historical review of robotics applied to welding with vision based seam identification is provided in [5].

3 DETAIL PROBLEM DESCRIPTION

We address the problem of soldering a line on a remote surface using a laser beam. Algorithms are proposed and evaluated for achieving this task. This research evaluates a low end robotic laser arm's execution of algorithms for soldering cracks on a surface. In evaluation experiments, the laser dot traces benchmark cases comprised of various types and shapes of skeletonized lines. Images of hand drawn lines are captured to a repository, and they are further processed to produce these templates. Templates are comprised of skeleton lines, which are further divided into line segments. Every n-th pixel on a line segment is declared a target position. The robotic arm must traverse these target positions in sequence and meet their coordinate positions within an accuracy $\mathcal{D}$. Moving the laser dot to each target position in sequence, effectively reproduces the skeleton line.

A camera system provides position feedback to the controlling software by capturing the current position of the laser dot. The feedback is used to create new position commands which are issued to the robot in order to minimize tracking errors. As the robot moves the laser dot towards the target positions, pixel coordinates of the laser dots are recorded and superimposed onto a trajectory record.

The tracking error is calculated by overlaying the trajectory of the laser dots onto the template. The area of the surface between the two lines is computed. A greater area expressed in pixels indicates a larger tracking error. The percentage of pixels missed along a skeleton line is also provided.

Formally we define the problem as follows:

Definition 1 *Given a surface S, a band of maximum width d drawn on the surface, and a laser positioned in the arm of a robot located at point O, with dot of diameter D, the problem is to define a plan and a control scheme for the robot handling the laser such that the laser dot traverses the band with a minimum number of interruptions, such that the laser dot covers the whole band but covers a minimum area outside of the band.*

4 TECHNIQUES

The software developed for this research is comprised of three major components:

- the Arduino micro-controller embedded software (sketch),
- the image processing software, and
- the control software.

The sketch configures the Digital IO and the six Pulse Width Modulated (PWM) output signals for the servo motors. The sketch also enables communications via the USB port between the control software and the micro-controller. When the control software issues a position command, the micro-controller processes the commands and returns the current positions of the servo motors.

Skeleton-based Jumping The algorithm we report here for this problem is called Skeleton-based jumping. Figure 1 shows the top level diagram of the command processing. Namely, each image is loaded and a set of filters is applied on it to skeletonize the crack line that has to be soldered. The skeletonized line is then processed into a path with a start and an end position. Further, in a loop, a control algorithms focuses the laser dot within a given distance from various positions selected along the skeletonized line being followed.

Figure 3 shows in more detail the steps applying the filters to the images. It can be observed that line thinning is interleaved with smoothing. A tree is built with the obtained skeleton at the end of this processing. The root node of the tree includes the list of pixels from the starting point to the first intersection. Each subsequent

node contains the list of pixels from the parent node's intersection to the next sequential intersection or end point. A simplified example is shown in Figure 2.

4.1 Control Software

Algorithm 1 Control Software

Given Laser Position L and Target Position T
while Target Available **do**
  if $L.X < T.X$ and $|L.X - T.X| > \mathcal{D}$ **then**
    X Motor += $2us$
  else if $L.X > T.X$ and $|L.X - T.X| > \mathcal{D}$ **then**
    X Motor -= $2us$
  end if
  if $L.Y < T.Y$ and $|L.Y - T.Y| > \mathcal{D}$ **then**
    Z Motor += $2us$
  else if $L.Y > T.Y$ and $|L.Y - T.Y| > \mathcal{D}$ **then**
    Z Motor -= $2us$
  end if
  if $|L.X - T.X| <= \mathcal{D}$ and $|L.Y - T.Y| <= \mathcal{D}$ **then**
    Target Met
    Increment Target
  end if
end while

Any PID motion controller can be used to trace a recorded line. In our case we settled for the Algorithm 1 as an example of controller to calibrate our metric, but other controllers can be used in the future. The control software issues position commands to the micro-controller to traverse the line based on the predetermined target positions. The target positions are visited in the order as determined by the target list algorithms. Camera feedback and position information obtained from the micro-controller corrects the robotic arm to place the laser dot onto each target position within $\mathcal{D}$ pixel accuracy. On the tested robotic arm, the relative position commands may be as small as half a degree, incrementally steering the laser dot to its target.

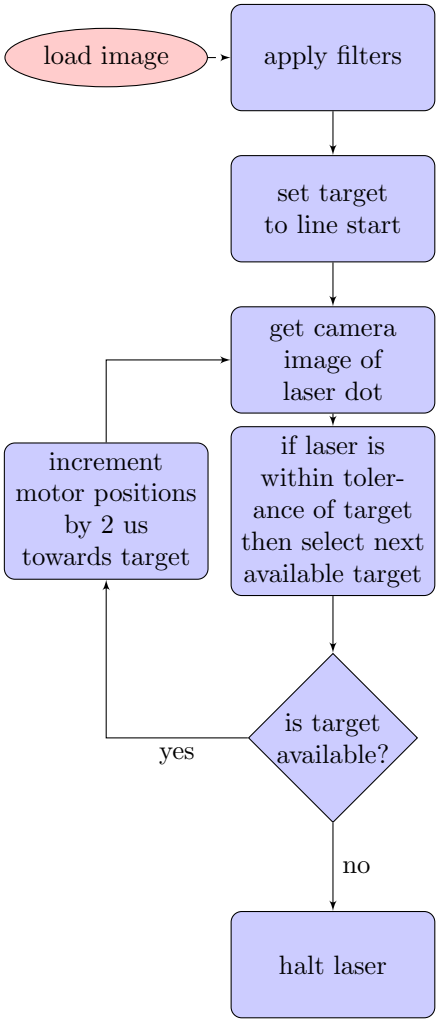

Figure 1: Control Software

4.2 Image Processing Software

An image is captured by the camera of a curved line forming a loop. The image is first converted into a B&W image. During this conversion image noise and variations in the background are removed. A hit-and-miss algorithm [6] is looped on the pixels of the line to minimize the line thickness. The algorithm uses several 3x3 transforms shown in Figure 4, which are applied to the B&W line to reduce the line to one pixel width while maintaining a continuity of the line. This process is completed

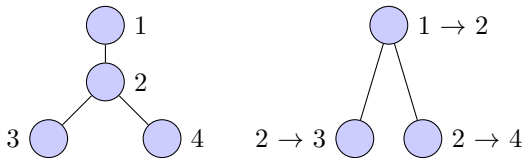

Figure 2: Example Line (Left) and Generated Tree (Right)

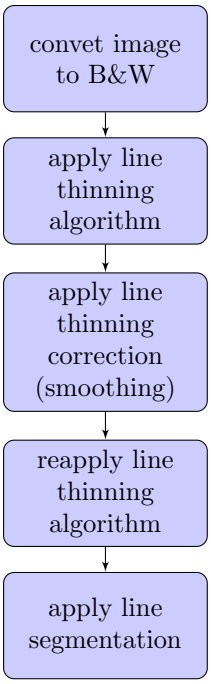

Figure 3: Filters

by looking for black pixels that match the operators and corresponding 90 degree variants of Figure 4. Each black pixel of the line is tested as the center point of the operator. The pixel is converted to white if a match is found. End points and intersections are located during the last iteration of the algorithm. An intersection is defined as a point with three or more neighbors.

A smoothing operator is applied to the skeleton line to reduce the number of neighboring intersections. This process further matches pixels to specific operator cases, where a pixel is either shifted or removed. The line thinning algorithm is called a second time after the smothering operator in order to relabel the intersections. Once a template is created, a list of line segments are determined.

Each line segment is further divided into target positions. For this research every n-th pixel is used as a target position, where n is a function of the distance between the laser and the remote surface. Every n-th pixel of the line is declared a target point that the laser dot must meet before moving on to the next target point. Feedback from the camera and the micro-controller ensures that the laser dot meets the target within $\mathcal{D}$ pixel accuracy.

Algorithm 2 Line Thinning (Hit-and-miss Transform)

Given structuring pairs B1, ..., B8 from Figure 3
while Image X not converged **do**
  $X \oplus B1 \oplus B2 \oplus ... \oplus B8$
end while
Mark pixels with 3 or more neighbors as intersections
Mark pixels with 1 neighbor as end points

	1	
0	1	1
0	0	

0	0	0
	1	
1	1	1

Figure 4: Structured pairs B_1 and B_2 including their 90°, 180°, and 270° rotations B_3-B_8

5 TARGET LIST ALGORITHMS

The following algorithms are used to develop the list of targets for the robot to follow when tracing the curve. The target list is developed after application of the filters and the curve is skeletonized.

5.1 Long Path Algorithm

Algorithm 3 Long Path

1: Set starting point as current pixel P
2: Start new line segment
3: **while** Unvisited neighboring pixels > 1 **do**
4: Add P to line segment
5: **if** P equals intersection **then**
6: Start New Line Segment
7: Count pixels to next intersection or end point in both available paths
8: Increment P in direction of larger count
9: **else**
10: Increment P to neighboring pixel.
11: **end if**
12: **end while**
13: Add end point to line segment

The long path algorithm builds a series of line segments for the robot to use as a target list. The algorithm is ideal for situations where the curve contains minimum intersections and where it is not desired to allow the laser to retrace a portion of the curve already visited. The algorithm begins in Line 1 and 2 by setting the starting point

of the line to P and creating a new line segment. A while loop is started in Line 3 where the loop continues while an unvisited neighboring pixel is available. The first part of the loop at Line 4 adds P to the newest created line segment. The P value is checked in Lines 5 – 7 to detect if P is an intersection. If P is an intersection then a new line segment is created. The intersection would signify that there are two available paths to select the next neighboring pixel from. The count of pixels in both paths are counted from the intersection to the next intersection or end point. The path with the largest count is selected to be taken and P is incremented in the direction of the larger count during Line 8. If P was not detected to be an intersection during Line 5 then P is incremented to the only unvisited neighboring pixel available. The loop will exit when no more neighboring pixels are available. At that time, P would be equal to the end point and would be added to the final line segment in Line 13.

5.2 Tree Search Algorithm

Algorithm 4 DFS with Backtracking Algorithm

```
1:  if isNull(Node.Left) and isNull(Node.Right) then
2:    Add Node.lineSegment to target list
3:    while Node.Parent.Left.Last not equals Node.Last
      or isNull(Node.Parent) do
4:      Add Node.lineSegment in reverse to target list
5:      Node equals Node.Parent
6:      if Node.Parent.Left.Last not equals Node.Last
        then
7:        Add Node.lineSegment in reverse to target list
8:      end if
9:    end while
10:   if isNull(Node.Parent) or Node.Parent.Left.Last
      equals Node.Last then
11:     return;
12:   end if
13: end if
14: if not isNull(Node.Left) then
15:   Add Node.lineSegment to target list
16:   DFS(Node.Left)
17: end if
18: if not isNull(Node.Right) then
19:   Add Node.left.lineSegment to target list in reverse
20:   DFS(Node.Right)
21: end if
```

The tree search algorithm is ideal for situations where every portion of the line is desired to be traced regardless of the laser retracing a portion of the line already visited. The algorithm builds a tree of line segments and transverses the line segments based on a modified depth first search algorithm.

Line 1 of the algorithm checks if the current node does not contain a left and right branch. The pixels of the line segment are then added to the target list during Line 2. Lines 3 through 10 are implemented to reverse the trace when the trace has followed a right branch down to an end point.

Line 14 of the algorithm checks if the left branch of the current node exists. The pixels in the line segment are then added to the target list in Line 15. The tree search algorithm is then called recursively with the left branch of the current node in Line 16.

Line 18 of the algorithm checks if the right branch of the current node exists. The pixels in the line segment are then added in reverse to the target list in Line 19. This will reverse the trace when the trace follows a left branch down to an end point. The tree search algorithm is then called recursively with the right branch of the current node in Line 20.

5.3 Adhoc Neighbor Algorithm

Algorithm 5 Ad-Hoc Neighbor

```
Set starting point as current pixel P
while Unvisited neighboring pixels >= 1 or intersec-
tion Stack > 0 do
  if Unvisited neighboring pixel is available then
    if P is an intersection then
      Add P to the intersection stack
      Increment P towards pixel with the shortest
      path to next intersection or end point
    else
      Increment P to unvisited neighboring pixel.
    end if
  else if Neighboring pixel is on top of intersection
  stack then
    Increment P to intersection
    Remove P from stack and longer treat P as an
    intersection
  else
    Reverse P to previous visited pixel.
  end if
end while
```

The Adhoc Neighbor algorithm is similar to the tree algorithm. Every portion of the line is desired to be traced regardless of the laser retracing a portion of the line already visited. In this implementation the algorithm does

not precompute the target list but instead computes the list as the trace is occurring. The algorithm follows the trace of the line by incrementing the target to the next neighboring pixel. When an intersection is encountered the algorithm stores the location in a stack. The algorithm selects the first direction available at the intersection and continues forward until an end point is found. While the intersection array contains values, the trace will reverse towards the first value on the stack. Once the intersection is reached again, the value is removed from the stack. The trace will halt once no values are remaining on the stack.

6 EXPERIMENTS

For this research the robotic laser arm was positioned 32 inches away from a whiteboard as seen in Figure 5. The camera was mounted next to the robotic laser arm separately. A small repository of images was created. The images are comprised of a set of hand drawn lines which vary in complexity and size. Figure 6 displays the original line images used in the experiment. The robotic laser arm is directed by the control software to trace the skeleton line. For this research the accuracy of the trace was set to $\mathcal{D} = 0.4$ mm (which has to be calibrated based on the camera resolution, lenses, and distance to the traced line). Feedback from the camera is recorded to file and then the position points are superimposed onto the skeleton line.

The range of position commands that can be issued to the Arduino micro-controller is between 600 – 2400 usec, which translates into 0.1 degree movement per 1 usec.

Although position commands can be giving as small as 0.1 degrees or in 1 usec increments, the servo motors cannot easily respond to such a small command. The servo motors must overcome friction and resistance of movement by the wires in order to move. In this experiment the position commands are given in 2 $u$sec increments.

7 ANALYSIS

The following section is a summary of the results of the laser trace. Data was collected after each movement of the robotic arm. The data was analyzed by comparing the error between the laser trace and the skeletonized line. Additionally the percent of the line traced is also presented.

7.1 Error between Trace and Line

The results of the three algorithms are shown in Table 1 The error is calculated by measuring the area in pixels be-

Figure 5: Experimental Setup

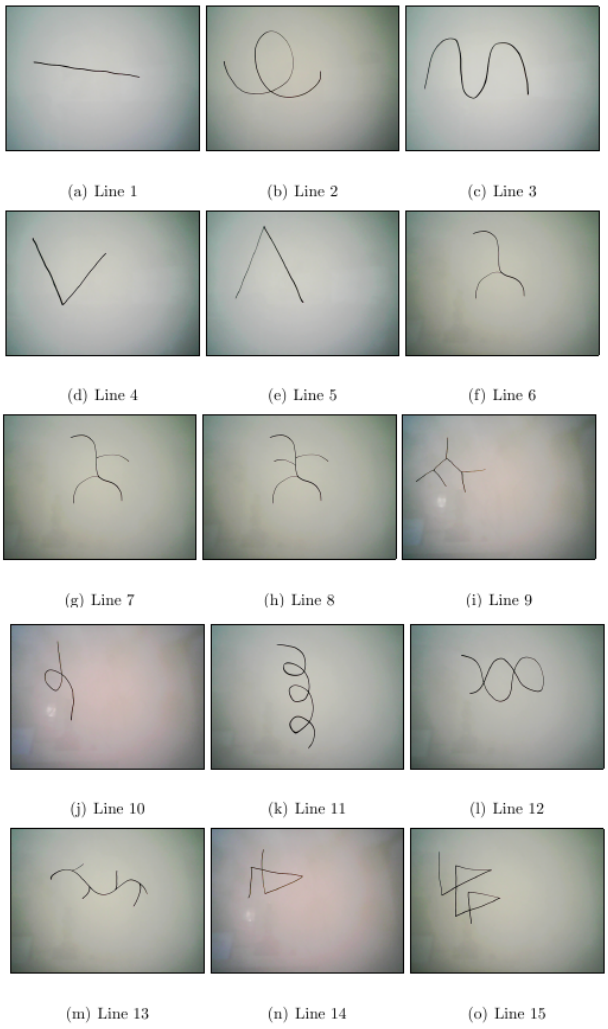

Figure 6: Original Lines

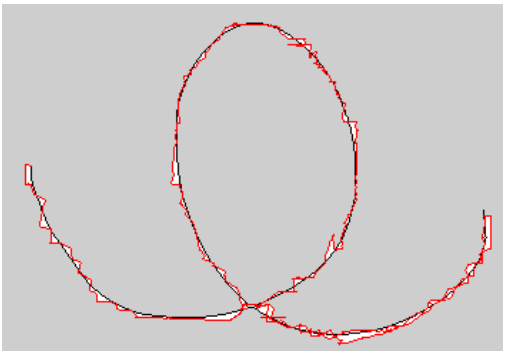

Figure 7: Laser trace of Line

Table 1: Error Between Lines - Results

Line	Pixels in Line	Longest Path	Tree Search	Ad Hoc
1	353	324	619	691
2	844	2124	2104	3141
3	807	803	1843	2634
4	394	592	1299	1643
5	491	533	1036	1487
6	427	330	1692	1405
7	533	364	2192	1809
8	601	313	2601	1973
9	419	72	1674	1674
10	439	661	959	959
11	815	1683	3134	2864
12	789	1913	2570	3178
13	548	1094	1695	2325
14	532	1138	1143	1800
15	926	801	3107	3048

tween the laser trace and the skeletonized line. A portion of the line may have not been traced due to the selected algorithm. The error in these cases are only calculated on the traced portion. A portion of the line may require the laser to retrace the section. The data points during the line retrace are included in the error calculations.

A sample image of the laser trace in Figure 7 show the traced line in red. The skeletonized line is show in black. The area outside the two lines is shown in gray and the area between the lines are shown in white. The area between the lines is calculated by counting the number of white pixels in the images.

7.1.1 Long Path Algorithm

The longest path algorithm selects the longest path available when detecting an intersection. The algorithm is the only method tested that does not have any backup capabilities. These properties allow the algorithm to complete its trace quicker than the other methods. With the laser in motion a smaller amount of time, the trace produces a smaller amount of errors.

7.1.2 Tree Search Algorithm

The tree search algorithm makes an attempt to trace the full line. The 0.5 mm diameter measurement is slightly larger than the tolerance allowed for the laser trace. It can be seen that no line contained a 100% trace under the 0.5 mm diameter laser. This could be potentially due to the laser tracing the line at the maximum tolerance away from the target position. The percentage increases to nearly 100% in all lines when the laser diameter is increased to 1.0 mm.

The tree search algorithm increases the number of retraces proportionally to the number of intersections. It could be seen in traces with a high number of intersections, such as line 15 (9 intersections), that the laser has a high trace percentage for the 0.5 mm laser.

7.1.3 Adhoc Neighbor Algorithm

The Adhoc Neighbor algorithm traces the line without predetermining the target list order. The trace is guided with only the knowledge of the length of the line segments. The trace always selects the shortest path available when at an intersection. The purpose of selecting the shortest path is to attempt to first cover line segments that form due to imperfections in the skeletonization process. This method significantly reduces retracing when the imperfection appears closer to the start of the line. The trace should not require retracing significant portions of the line to return to these imperfections.

7.2 Percentage of Line Traced

The laser in this experiment was allowed a tolerance of 4 mm from the target location. The target location is set on the edge of the original line due to the way the line thinning algorithm skeletonized the line. As a result the laser trace has not covered the entire width of the original line. The target list also did not add every pixel as a target. The target list has missed sections of the line with sudden changes in slope. Varying diameters of laser size were analyzed on each laser trace. For these experiments a laser diameter of 0.5 mm and 1.0 mm

Table 2: Long Path - Percent of Line Traced

Line	Pixels in Line	Untraced Pixels .5 mm	% Traced .5 mm	Untraced Pixels 1.0 mm	% Traced 1.0 mm
1	353	65	81.6 %	0	100.0 %
2	844	186	78.0 %	4	99.5 %
3	807	141	82.5 %	6	99.3 %
4	394	109	72.3 %	13	96.7 %
5	491	103	79.0 %	5	99.0 %
6	427	143	66.5 %	108	74.7 %
7	533	359	32.7 %	294	44.8 %
8	601	448	25.5 %	373	37.9 %
9	419	316	24.6 %	304	27.5 %
10	439	212	51.7 %	149	66.1 %
11	815	274	66.4 %	155	81.0 %
12	789	160	79.7 %	5	99.4 %
13	548	260	52.6 %	153	72.1 %
14	532	195	63.4 %	66	87.6 %
15	926	621	32.9 %	545	41.1 %

Table 3: Tree Search - Percent of Line Traced

Line	Pixels in Line	Untraced Pixels .5 mm	Percent Traced .5 mm	Untraced Pixels 1.0 mm	Percent Traced 1.0 mm
1	353	63	82.2 %	4	98.9 %
2	844	185	78.1 %	1	99.9 %
3	807	148	81.7 %	1	99.9 %
4	394	40	89.9 %	1	99.8 %
5	491	63	87.2 %	1	99.8 %
6	427	72	83.1 %	1	99.8 %
7	533	101	81.1 %	3	99.4 %
8	601	52	91.4 %	0	100.0 %
9	419	50	88.1 %	5	98.8 %
10	439	55	87.5 %	1	99.8 %
11	815	70	91.4 %	0	100.0 %
12	789	126	84.0 %	0	100.0 %
13	548	54	90.2 %	3	99.5 %
14	532	146	72.6 %	6	98.9 %
15	926	77	91.7 %	0	100.0 %

were compared. Data collected during backtracking was counted when calculating the percent of the line covered.

7.2.1 Long Path Algorithm

The longest path algorithm allows the trace to neglect portions of the line. The lines that contain side branches with insignificant length are shown to have the highest percentages. These traces can be seen in Lines 1 - 5. In other cases where the line contained side branches with significant length are shown to have the lowest percentages. The algorithm does not allow for backtracking and therefore the percentages do not benefit from the possibility of the laser getting a second chance to trace the line.

7.2.2 Tree Search Algorithm

The tree search algorithm makes an attempt to trace the full line. The 0.5 mm diameter measurement is slightly larger than the tolerance allowed for the laser trace. It can be seen that no line contained a 100% trace under the 0.5 mm diameter laser. This could be potentially due to the laser tracing the line at the maximum tolerance away from the target position. The percentage increases to nearly 100% in all lines when the laser diameter is increased to 1.0 mm.

The tree search algorithm increases the number of retraces proportionally to the number of intersections. It could be seen in traces with a high number of intersections, such as line 15 (9 intersections), that the laser has a high trace percentage for the 0.5 mm laser.

7.2.3 Adhoc Neighbor Algorithm

The adhoc neighbor algorithm guides the laser trace through all pixels in the line. The algorithm shows similar percentages to the tree search algorithm. The error in the trace is most likely also attributable to the time the laser is tracing at the maximum tolerance allowed. The percent similarly increases to 100% when the laser diameter is increased to 1.0 mm.

Each retraced pixel gives the laser another opportunity to cover a portion of the missed line in the first trace pass. The adhoc neighbor algorithm selects the smaller of two line segments when encountering an intersection. This reduces the amount of pixels retraced in some lines in comparison to the tree search method. This has the potential to cause the percent traced to drop in comparison to the tree search algorithm.

8 CONCLUSION

For this research we used a low cost robotic laser arm comprised of six servo motors which are controlled by an Arduino micro-controller. The micro-controller converts the position commands into pulse width modulated signals which provide for 0.1 degrees or 1 microsecond position commands.

Table 4: Adhoc Search - Percent of Line Traced

Line	Pixels in Line	Untraced Pixels .5 mm	% Traced .5 mm	Untraced Pixels 1.0 mm	% Traced 1.0 mm
1	353	60	83.0 %	0	100.0 %
2	844	107	87.3 %	0	100.0 %
3	807	144	82.2 %	4	99.5 %
4	394	100	74.6 %	1	99.8 %
5	491	97	80.2 %	0	100.0 %
6	427	100	76.6 %	1	99.8 %
7	533	74	86.1 %	0	100.0 %
8	601	69	88.5 %	3	99.5 %
9	419	52	87.6 %	1	99.8 %
10	439	94	78.6 %	5	98.9 %
11	815	105	87.1 %	5	99.4 %
12	789	157	80.1 %	3	99.6 %
13	548	96	82.5 %	2	99.6 %
14	532	103	80.6 %	4	99.3 %
15	926	163	82.4 %	4	99.6 %

The operational speed of the robotic arm is limited as it has to process commands serially. The steps include issuing position commands, obtaining position data from the micro-controller, and obtaining and processing images from the camera.

A repository of images was created to test the procedure. The skeletonizing algorithm followed by the line segmentation algorithm successfully provided for position commands. These benchmarks can be executed at any time producing repeatable results, and will be made publicly available. The system provides for a high degree of flexibility.

Our research shows that the robotic arm successfully traced the benchmark lines with limited error. Some of the errors were caused by "stickiness" of the low-end physical system, but may be reduced on higher-end arms. The motor would randomly fail to move the appropriate distance when a position command is issued. Subsequent position commands caught the motor back up and caused the laser to slip from the target positions. The average change in angle between the laser trace and the original trace remained similar between the different algorithms. This implies the different algorithms did not impact the change in angles but were dependent on the accuracy of the motors. This error could be significantly reduced by moving the laser and camera closer to the surface, or introducing extra-delays for feedback and correction. This would improve the accuracy of the servo motor position commands.

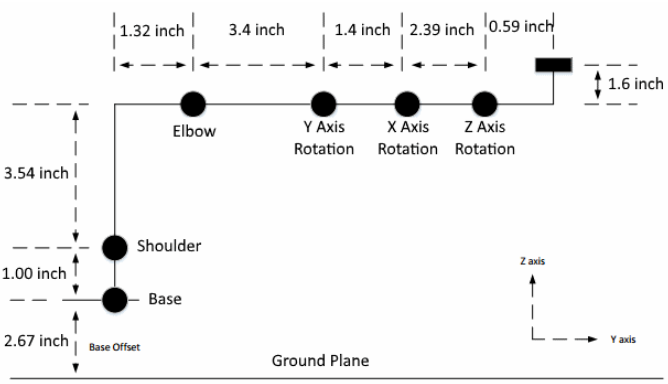

Figure 8: Block Diagram

The effect of choosing the longest path between intersections to be traced maintains the original shape of some of the lines. The trace loses accuracy when the line contains self intersections or significant branching. However, the algorithm prevents the laser from retracing portions of the line already visited.

The tree search and adhoc neighbor search both showed similar results when tracing the lines. The tree search precomputed the target list. This allows the operator to estimate the percent of the line remaining to be traced while the laser is in motion. In contrast, the adhoc neighbor search does not contain knowledge of the remaining amount of untraced pixels.

The width of the laser played a role in the percent of the lines traced. The 0.5 mm wide laser did not succeed in tracing 100% of any of the lines. The error could be reduced for the 0.5 mm laser by moving the robotic arm closer to the line or by improving the accuracy of the servo motor positions. This would allow the tolerance between the laser and target to be reduced. The 1.0 mm wide laser was able to trace 100% or near 100% of the line in the majority of the lines. The exception cases would be where the longest path algorithm ignored a portion of the line. The few pixels missed in the traces with near 100% during the 1.0 mm wide laser traces could be improved upon by increasing the number of target points. This would allow the laser to better react to changes of slope in the lines.

DESCRIPTION OF THE ROBOTIC ARM

The robotic laser arm is controlled by positioning software running on a PC using visual feedback provided by

a single camera. The robotic arm is comprised of six servo motors controlling position and orientation of the endpoint, where a 5 mW laser (650 nm) is mounted. An Arduino micro-controller generates six pulse-width modulated (PWM) signals to position the servo motors. The servo motors can rotate 0 to 180 degrees, which corresponds to 600 - 2400 usec, respectively. The servo motors are physical centered at 90 degrees (1500 usec) at power of the robot.

The Arduino micro-controller receives position commands from the PC, which are converted to PWM signals to position the servo motors. Communication between the PC and the micro-controller is established using a universal serial bus (USB). Whenever a command is send from the PC to the micro-controller, the micro-controller executes the command, and returns the current servo motor positions. The laser state can be set to either on or off; and can also be set to blink at some periodic rate.

The analog joysticks are used to manually position the robotic arm. While manually positioning the robotic arm, the USB port to the Arduino micro-controller must be disconnected since PC positioning commands override analog commands. This feature is highly instrumental when manually bore sighting the system. The robotic arm should be adjusted such that traveling along the X-axis and the Z-axis does not generate significant cross-talk. The Y-axis is aligned orthogonal (depth) to the target range and the servo motors should remain centered or 90 degrees +/- a small offset.

Figure 8 shows the physical dimensions of the robotic arm. Each position and orientation servo motor is set to 90 degrees (1500 usec). While maintaining orthogonality to the range, the robotic arm can travel a maximum distance of 43.254 cm along the x-axis and 23.127 cm above its horizontal plane (z-axis). Moving below the horizontal plane is limited by the distance to the ground plane which is approximately 15 cm.

REFERENCES

[1] Yinshui He, Yanling Xu, Yuxi Chen, Huabin Chen, and Shanben Chen. Weld seam profile detection and feature point extraction for multi-pass route planning based on visual attention model. *Robotics and Computer-Integrated Manufacturing*, 37:251–261, 2016.

[2] M.S Hussin, Daut Firdaus, and Jufriadi A. Shahril. Robotics application in arc welding a review on current progress. *Int. J. of Mechanical Computational and Manufacturing Research*, 2(1):1–5, 2013.

[3] Hong Luo and Xiaoqi Chen. Laser visual sensing for seam tracking in robotic arc welding of titanium alloys. *The International Journal of Advanced Manufacturing Technology*, 26(9-10):1012–1017, 2005.

[4] Štefan TOTH Matej MEŠKO. Laser spot detection. *Journal of Information, Control and Management Systems*, 11(1), 2013.

[5] Hairol Shah, Marizan Sulaiman, Ahmad Shukor, Muhammed Jamaluddin, and Mohd Rashid. A review paper on vision based identification, detection and tracking of weld seams path in welding robot environment. *Modern Applied Science*, 10(2):1913–1952, 2016.

[6] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007.