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 line 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.

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 points 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 in technical details. 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 [6].

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.*

A time constraint may also have to be addressed later, where it could be requested that the laser dot abides a minimal amount of time over each portion 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 first algorithm we report here for this problem is called Skeleton-based jumping, as additional algorithms are being currently investigated. 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 2 shows in more detail the steps applying the filters to the images. It can be observed that line thinning

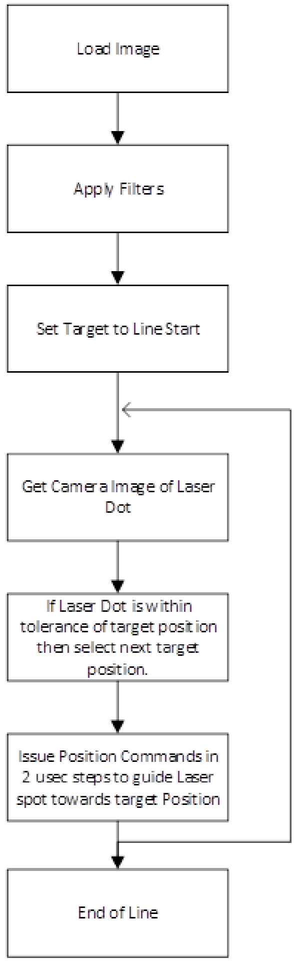


Figure 1: Control Software

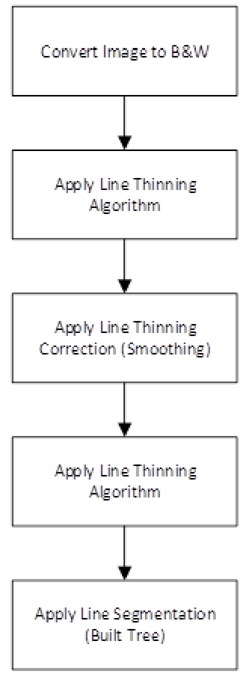


Figure 2: Filters

is interleaved with smoothing. A tree/graph is built with the obtained skeleton at the end of this processing.

4.1 Control Software

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 a tree/graph traversal algorithm. 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.

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

	1	
0	1	1
0	0	

0	0	0
	1	
1	1	1

Figure 3: Structured pairs B_1 and B_2 including their 90°, 180°, and 270° rotations B_3-B_8

4.2 Image Processing Software

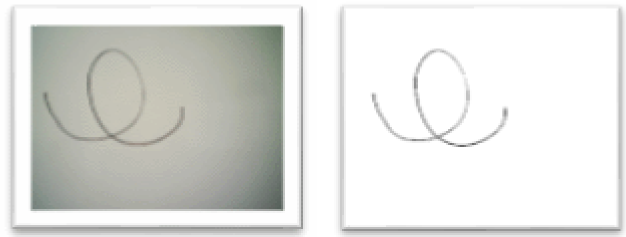
Figure 4 displays an image, captured by camera, of a curved line forming a loop. The image is first converted into a B&W image. During this conversion image noise and variations in the background are removed. A hit-and-miss algorithm [5] is looped on the pixels of the line to minimize the line thickness. The algorithm uses several 3x3 transforms shown in Figure 3, 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 by looking for black pixels that match the operators and corresponding 90 degree variants of Figure 3. 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.

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

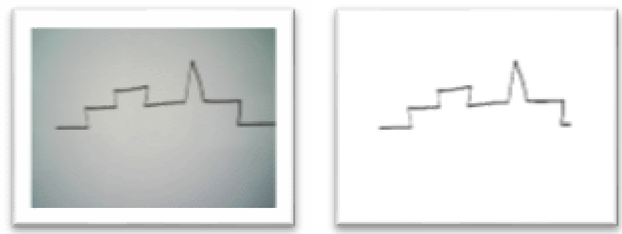
Given structuring pairs B1, ..., B8 from Figure 3
while Image X not converged **do**
$X \oplus B1 \oplus B2 \oplus ... \oplus B8$
end while
Mark pixels with 3 or more neighbors as intersections
Mark pixels with 1 neighbor as end points

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.

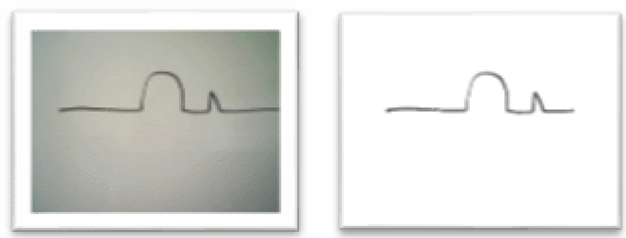
Once a template is created, a list of line segments are determined. These line segments are derived from the endpoints and existing intersections. The first line segment is measured from the starting point to the first intersection by following the path of the pixels. Line segments between two intersections are determined by calculating the length of the two possible paths. The majority of intersections formed are the result of very short branches



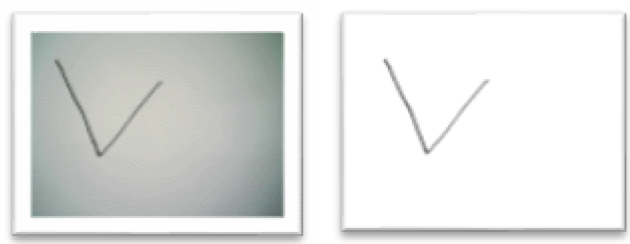
(a) Line 1



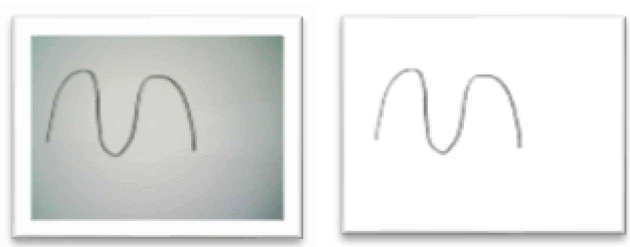
(b) Line 2



(c) Line 3



(d) Line 4



(e) Line 5

Figure 4: Original and B&W Image

Algorithm 3 Line Segmentation

Set starting point as current pixel P
Start new line segment
while Unvisited neighboring pixels > 1 **do**
  Add P to line segment
  if P equals intersection **then**
    Start New Line Segment
    Count pixels to next intersection or end point in both available paths
    Increment P in direction of larger count
  else
    Increment P to neighboring pixel.
  end if
end while
Add end point to line segment

splitting off from the main line. The longer path is selected to be kept. No pixel is allowed to be revisited once added to a line segment.

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. Figure 6 shows the movements of the robotic arm tracing the skeleton template generated from the lines in Figure 4. 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.

5 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 4 displays a few examples of these images. 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 results are displayed in Figure 6.

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.

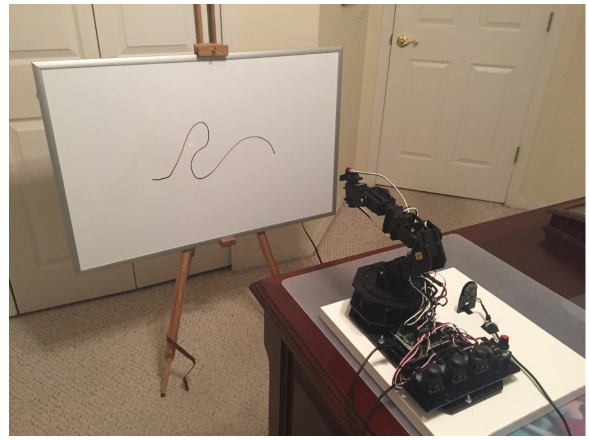


Figure 5: Experimental Setup

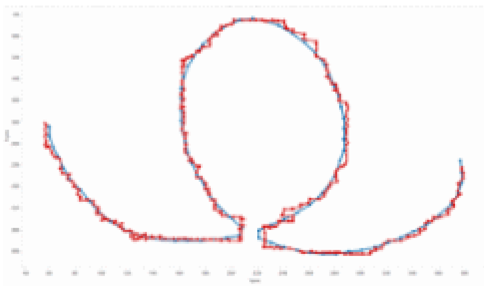
Although position commands can be giving as small as 0.1 degrees or in 1 usec increments, the servo motors cannot easily respond to such a small command. The servo motors must overcome friction and resistance of movement by the wires in order to move. Positioning within 0.1 degrees accuracy is possible as long as the position command itself is larger. In this experiment the position commands are given in 2 $u$sec increments. Figure 6 shows the differences between the skeleton lines and the trace of the laser dot.

The line traces show that every tenth pixel is a target position and therefore the laser dot follows the curves in small straight line segments. The disparity between the two curves can be expressed as an error by counting the pixels in the areas between the two curves. An area of 0 would imply a perfect trace. The results are documented in Table 1. The laser tracing the line may not be the same width of the original line. Table 1 also documents the percentage of pixels in the B&W image not covered by the laser during the trace.

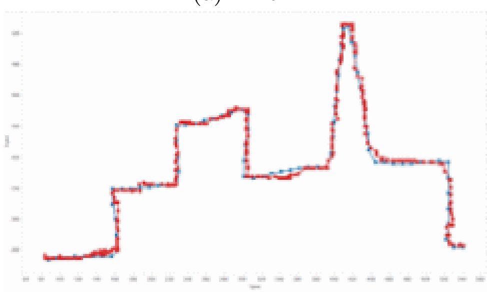
6 CONCLUSION

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.

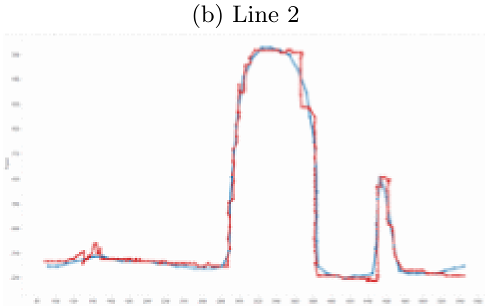
The operational speed of the robotic arm is limited as it has to process commands serially. The steps include issuing position commands, obtaining position data from the micro-controller, and obtaining and processing images



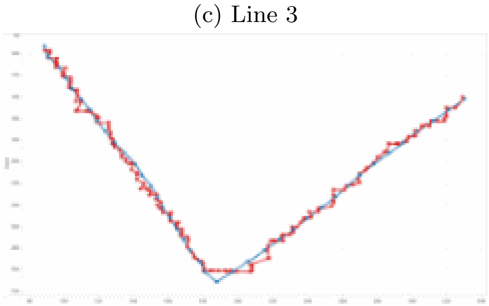
(a) Line 1



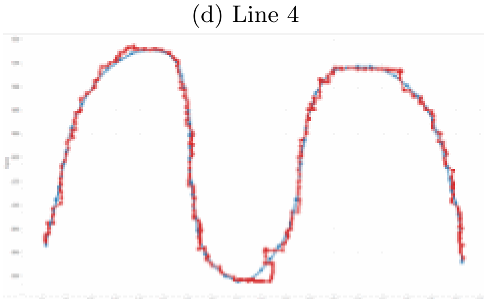
(b) Line 2



(c) Line 3



(d) Line 4



(e) Line 5

Figure 6: Tracing Robot Movement

Table 1: Results

Line	Pixels in B&W Line	Error (Area)	0.5 mm Wide Laser (% Error)	1.0 mm Wide Laser (% Error)	1.5 mm Wide Laser (% Error)
1	2190	927	26.67	2.15	0.68
2	2968	572	34.54	5.86	0.81
3	2539	358	26.82	2.72	0.39
4	2801	662	19.67	1.64	0.18
5	1613	499	27.71	6.51	2.11
Average	2422.2	603.6	27.08	3.78	0.83

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 and is ideal for many manufacturing environments.

Our research shows that the robotic arm successfully traced the benchmark lines with limited error. Comparison with additional algorithms is being currently investigated. 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. 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.

The effect of choosing the longest path between intersections to be traced maintains the original shape of the majority of the lines. The exception case involves lines that contain a loop. This scenario is reflected in Figure 6.a. When the longest path was selected between intersection points, a gap was formed at the bottom of the loop. The algorithm could be adapted to remove the gap in future work. However this would require allowing the laser to trace over pixels already traced when tracing through the intersection of the loop.

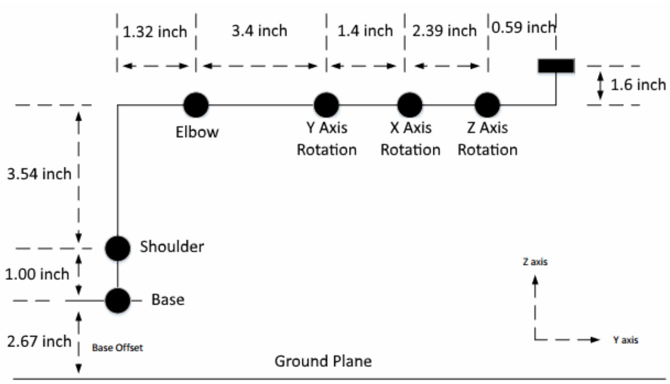


Figure 7: Block Diagram

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 7 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.

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

[5] Roger Boyle Milan Sonka, Vaclav HIavac. *Image Processing, Analysis, and Machine Vision.* Thomson Corp, 2007.

[6] 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.