The design of a compact laser scanner and an integrated simulation environment for smart manufacturing

Haowen Shi

CMU-RI-TR-21-45
Aug, 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

The development in material science and robotics makes 3D printing large building structures possible and desirable thanks to its flexibility and efficiency. Automated 3D printing of building structures has many benefits ranging from reducing construction cost to helping make habitats for space explorations. However, most of the existing robotic 3D printing solutions require human supervision for monitoring and intervention. We believe this lack of autonomy is partly because of the lack of a suitable sensor that could provide the necessary feedback information, and the difficulty in developing control algorithms on physical hardware that is expensive and dangerous to operate.

In this work, we present a design framework for a miniaturized laser scanner that could be used for *in-situ* inspection in additive processes by providing submillimeter-accurate, real-time depth reconstruction of the printed material and imagery feedback. We also present a novel, all-in-one simulation environment for accelerating control software development based on the simulated sensory feedback. The proposed simulation environment is equipped with photorealistic rendering, sensor simulation, additive fluid material simulation, robot arm and robotics software integration.

We show the precision of our laser scanner by comparing the measured widths of some 3D printed gaps against the ground truth. To show the effectiveness of our robotic manufacturing simulation environment, we modeled a full robotic additive printing process with our laser scanner, an additive thermal plastic extrusion nozzle and a servoing robot arm. We then showed a simulated *in-situ* scan result using a simple sensor placement strategy that maximizes scan coverage given our sensing constraints.

Acknowledgments

I would like to thank my advisor Professor Howie Choset for his unwavering support and guidance throughout my undergraduate and graduate studies. I thank Lu Li for the countless inspiring conversations and all the valuable advice. Next I would like to thank my lab mates in the Biorobotics Lab, especially Daqian Cheng, Albert Xu, Eliana Cohen and everyone who I have had the privilege to work with. My work would not have been possible without you. I am grateful for the feedback and suggestions I got from my thesis committee: Professor Srinivasa Narasimhan and Sudharshan Suresh. Special thanks to Michelle Crivella who has been supporting us along the way, and Boeing for funding and collaborating with us on this project.

Last but not least, I want to thank my family, friends and CMU staff members who have helped me over the years. Your support and love have been critical to my development and progress.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

There has been growing interest in using Additive Manufacturing (AM) technology to produce large building structures because of its flexibility, sustainability and safety [5]. Large scale, robotic AM is being used for several experimental construction projects with human supervision, as people are aiming toward using AM for autonomously building habitats before crew arrive in space explorations [6, 7]. Fig. 1.1 shows two example robotic AM construction systems working to produce large building structures.

Despite the recent development in material science and robotics technology that

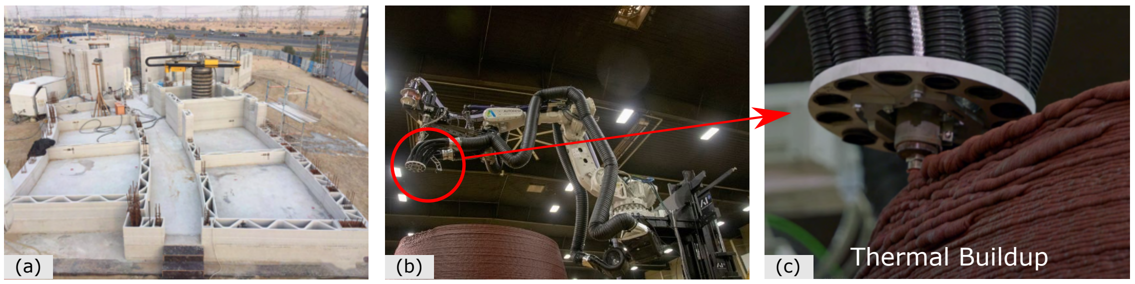


Figure 1.1: (a): an office building being additively manufactured in Dubai [1]. (b): AI Spacefactory MARSHA in the NASA 2019 3D-Printed Habitat Challenge [2]. (c): Print quality affected by thermal buildup.

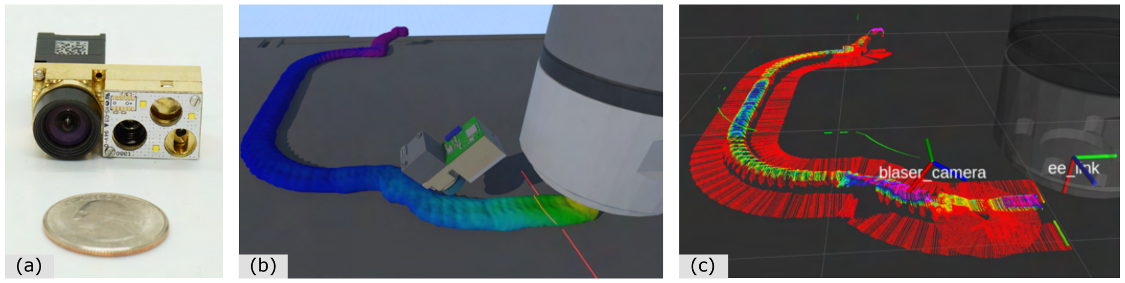


Figure 1.2: (a): Our proposed compact laser scanner. (b): Our all-in-one robotic AM simulation environment. (c): The in-situ 3D reconstruction of extruded material (in simulation).

make robotic AM more viable, most of the modern AM systems do not have the ability to make adjustments in case of a process error. This can be problematic especially for large scale AM projects such as building printing with a high do-over cost, if at all possible. Shown in Fig. 1.1 (c), during the NASA 2019 3D-Printed Habitat Challenge, team AI Spacefactory experienced material thermal buildup during the competition. Manual intervention was required by changing the printing trajectory speed for area with thermal irregularities. This shows even the state-of-the-art AM construction systems are not equipped with proper sensors and algorithms for closed-loop control. We believe this lack of smart adaptation is partly because of the lack of a suitable sensor that can provide the necessary feedback information, as well as the difficulty of developing and iterating control algorithms on large machines that are dangerous and costly to operate.

1.2 Approach and Contribution

In this work, we propose a novel, small form factor laser scanner (Fig. 1.2 (a)) that could be easily placed next to an additive material extrusion nozzle. The scanner provides *in-situ* 3D reconstruction as well as images of the printed material which can be utilized by closed-loop control algorithms. To make this dual data type output possible with a monocular setup, we propose the "alternating shutter" approach to produce two video streams with different exposure levels from the same camera.

To address the difficulty in developing control algorithms, we propose an all-in-one simulation environment (Fig. 1.2 (b)) that for the first time, to the best of our knowledge, provides a full stack modeling of the entire robotic AM process including material deposition and curing simulation, robot arm integration, and a photorealistic simulation for a laser based scanner providing the necessary feedback. We tested this simulation environment by creating a digital twin of a typical robotic AM setup, and used it to execute a real 3D printing G-Code trajectory.

Because of the sensing range and field of view (FOV) constraints of our compact scanner, a simple but effective sensor placement strategy called “virtual caster” is proposed to set the orientation of the sensor during printing to maximize scan coverage. Fig. 1.2 (c) shows an example scan coverage using this strategy.

Chapter 2

Sensor Design

In this chapter, we discuss the development of a miniature structured light sensor that can be mounted closely to an additive material extrusion nozzle for collecting *in-situ* feedback about the print quality. We will cover the sensor's theory of operation, different odometry sources for map stitching, calibration and hardware development.

2.1 Theory of Operation

2.1.1 Single-line Laser Scanner

Our laser scanner resolves the depth information using an active stereo setup. A laser stripe projector actively projects a pattern into the world which then gets observed by a single camera. Depth is then solved by triangulating the incident laser points on the object. Here, a laser line is our pattern of choice because it is easy to model and triangulate. Our scanner contains a dot laser emitter and a cylindrical lens. The cylindrical lens refracts the laser beam and projects a laser plane as shown in Fig. 2.1. The laser plane intersects with the object of interest and creates a laser profile which is then captured by the RGB camera to produce an incident pattern.

As shown in Fig. 2.1, the projected laser plane is modeled as $\Pi_l : \mathbf{n} \cdot \mathbf{X} + d = 0$ in the image frame, where $\boldsymbol{n}$ and d are plane parameters that needs to be calibrated. Details on how calibration is done is discussed in Section 2.3. Depth information of the incident world point $\mathbf{X}_i$ can be reconstructed by solving a ray-plane intersection problem. The ray is the line-of-sight (LOS) ray shooting from the origin of the camera frame C passing through the normalized image coordinate of incident laser point $\mathbf{X}_{ic}$. Given the laser plane Π_l we can solve for the ray-plane intersection point in 3D to triangulate the depth of the incident laser point.

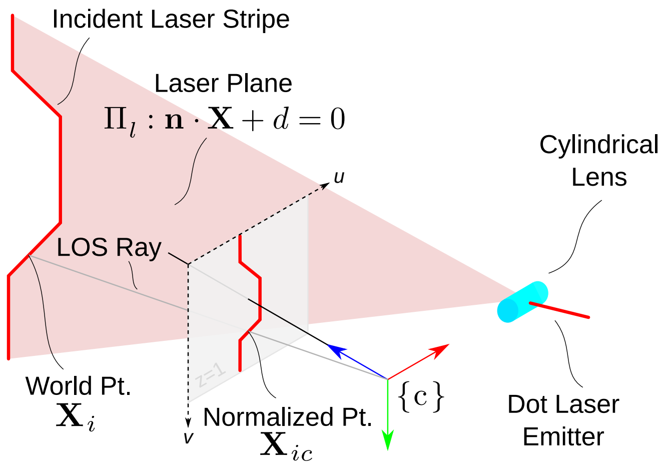


Figure 2.1: Theory of operation: Laser depth is triangulated by projecting a camera ray out from the camera origin and finding its intersection with the laser plane.

The triangulation process is as follows: from the laser plane calibration $\{\mathbf{n}, d\}$, we have the equation: $\Pi_l : \mathbf{n} \cdot \mathbf{X} + d = 0$. From the camera calibration we have the inverse projection function π_c^{-1}, which computes the normalized image coordinate $\mathbf{X}_{ic} = \pi_c^{-1}(\mathbf{x}_i)$ where $\mathbf{x}_i$ is the pixel coordinate. The world point $\mathbf{X}_i$'s 3D position is resolved using (2.2).

$$\mathbf{X}_i = \frac{-d}{\mathbf{n} \cdot \pi_c^{-1}(\mathbf{x}_i)} \mathbf{X}_{ic} \tag{2.1}$$

$$= \frac{-d}{\mathbf{n} \cdot \pi_c^{-1}(\mathbf{x}_i)} \pi_c^{-1}(\mathbf{x}_i) \tag{2.2}$$

This step is repeated for all incident laser pixel coordinates in the image. Details on how incident points are detected in the image is covered in Section 3.1.

2.1.2 Multi-line Laser Scanner

To increase the sensor measurement rate and improve scan coverage, we also created a multi-line laser scanning setup. We replace the cylindrical lens with a diffractive optical element (DOE) lens to produce multiple laser planes instead of just one.

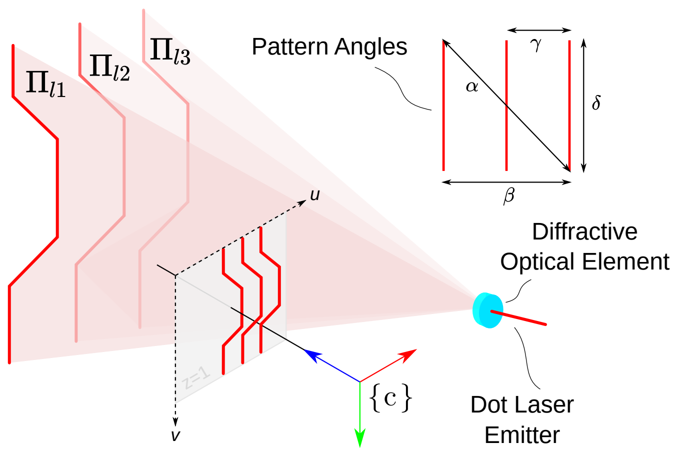


Figure 2.2: Theory of operation: The laser projector generates multiple laser planes Π_{l1}, Π_{l2}, Π_{l3}, etc.

In the multi-line triangulation mode, the sensor takes more depth measurements in the same image frame, so at the same camera frame rate the multi-line scanner effectively has higher measurement rate compared to a single-line one. However, this advantage does not come free. When we back project an incident laser pixel out to the 3D space as a LOS ray, the ray could intersect with multiple laser planes, creating more than one possible solutions for the depth estiamte, as shown in Fig. 2.3.

Having a stereo camera is helpful for addressing this depth ambiguity, as illustrated in [8]. However, because of the sensor size constraint, we are limited to one camera and hence we need some heuristic to distinguish which laser plane an incident laser pixel belongs to. [9] introduced a method to solve this ambiguity by classifying laser stripe segments into several types and grouping the segments into the most likely plane by making incremental connections from one segment to the next. In this work we implemented the method proposed in [9] and evaluated its effectiveness, see Section 2.1.3 for details.

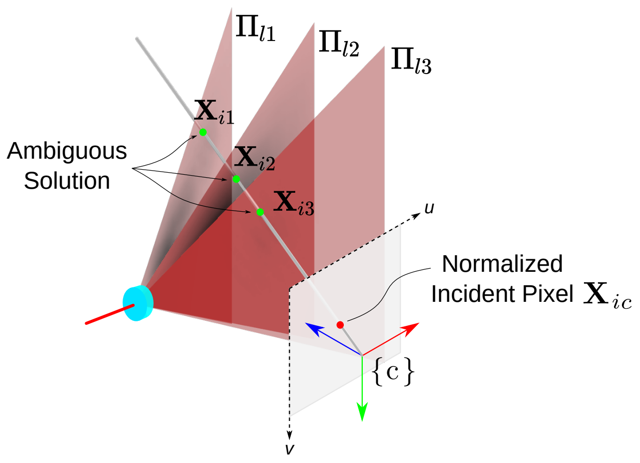


Figure 2.3: In the multi-line configuration, depth estimate can have multiple possible solutions for each incident laser pixel.

2.1.3 Multi-line Laser Plane Identification

After extracting all the laser pixel centers as described in Chapter 3, we need to associate them with the correct laser plane to obtain the their correct depth. For multi-line projector with N laser stripes, we make the following requirements and assumptions:

- The laser planes are wide enough to extend beyond the camera observable field of view.
- The sensor is placed next to the scanned surface such that all N laser stripes are observable within a single frame.

The first step in plane identification is to identify line segments. This can be easily done using the 8-way connected components algorithm [10]. Once we have a set of connected laser pixel segments, we can classify them into four types: left-edge-connected (LEC), light-edge-connected (REC), left-right-edge-connected (LREC) and floating (F). An example classification is shown in Fig. 2.4b. Edge connectivity is defined as having part of the segment within a few pixels away from the image edges that are perpendicular to the laser plane. A floating (F) segment means it is not connected to the “left” or the “right” edge, hence floating in the middle. The group of laser stripe segments can be classified using the following algorithm:

Once the segments have been classified into the aforementioned four types, we

Algorithm 1 Stripe segment type classification

```
procedure ClassifySegments(segmentGroups)
    for segment in segmentGroups do
        for each point in segment.points do
            if isPointLEC(point) then
                switch segment.type do
                    case F
                        segment.type = LEC
                        break
                    case REC
                        segment.type = REC
                        break
            end if
            if isPointREC(point) then
                switch segment.type do
                    case LEC
                        segment.type = LREC
                        break
                    case F
                        segment.type = REC
                        break
            end if
        end for
    end for
end procedure
```

can start grouping the laser segments that most likely belong to the same laser plane together. We can do this by first creating a list of segments that belong to the same plane for each LEC segment. For each of the lists, we check the remaining segments for suitable ones to move to the list based on a heuristic proposed in [9]. The high level idea is to find shortest connection vectors from the end of previous segment to the next segment among a candidate list. and stop when the last suitable segment in the laser plane group is of type REC, which indicates that we have found all the segments that most likely belong to one laser plane in the image.

When we are done grouping all stripe segments, under the assumption that all laser stripes are observed in a single frame, we should have N total laser stripe groups. We can then assign the laser plane ID incrementally from "left" to "right" to associate them with the correct laser plane calibration. For each of these segment groups, depth is obtained by solving the same ray-plane intersection as described in Section 2.1.1. Fig. 2.4 shows an example stripe classification and grouping result. To demonstrate the effectiveness of our implementation, we show a scan result with five laser planes in Fig. 2.5.

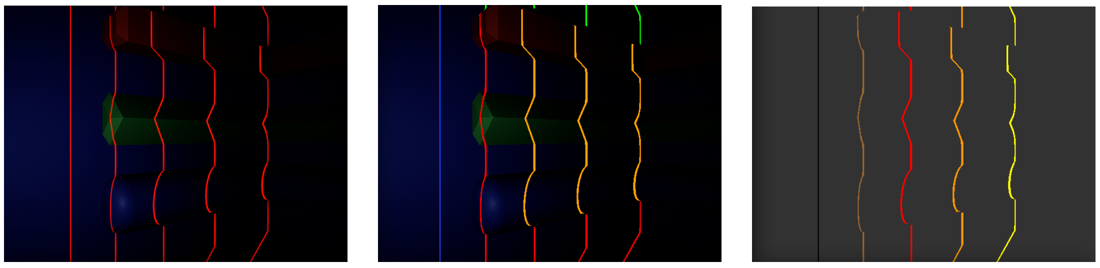

(a) Camera image (b) Stripe segments classification (c) Stripe segments grouping

Figure 2.4: Multi-line segment classification and grouping results. In the middle: blue is type LECREC, green is type LEC, orange is type F, and red is type REC. To the right: each color correspond to one laser plane group. An ID is assigned incrementally from left to right.

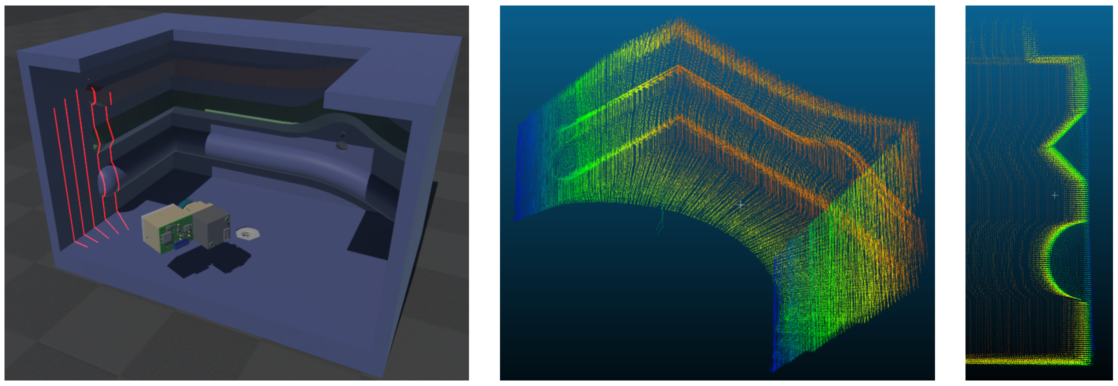

(a) Scan setup (b) Scan result perspective view (c) Side view

Figure 2.5: A scan result using our multi-line laser plane identification implementation.

2.1.4 Sensor Pose Estimation and Map Stitching

Our structured light laser scanner produces a local, sectional depth reconstruction of the scanned object at a given time. To stitch up a full 3D reconstruction of an object, we need the sensor's pose information at each time step and transform the local points to the world coordinate frame for point cloud accumulation. There are primarily three ways to provide the sensor's pose information during a scan:

- Robot arm kinematics: mount the sensor on a robot arm and use the arm kinematics to generate the scanning trajectory.
- Visual fiducial: put fiducials such as Vicon pearl markers on the sensor and use external cameras to provide accurate pose estimates.
- Visual odometry: use the onboard camera to estimate the sensor trajectory without external positioning infrastructure.

Robot Arm Kinematics

Modern industrial robot arms have high repeatibility and submillimeter precision [11], which are ideal for providing an accurate pose trajectory. Non-actuated measurement arms are widely used in commercial coordinate measurement machines (CMM) such as a FARO ScanArm (FARO Technologies, Lake Mary, Florida, USA). In the robotic manufacturing context, our laser scanner can be mounted next to the additive material extrusion nozzle. The pose of the scanner can be obtained by composing the robot arm end-effector link pose with the sensor's hand-eye extrinsics, as illustrated in Fig. 2.6. Details on how the hand-eye transform $\boldsymbol{X}$ is obtained is described in Section 2.3.3.

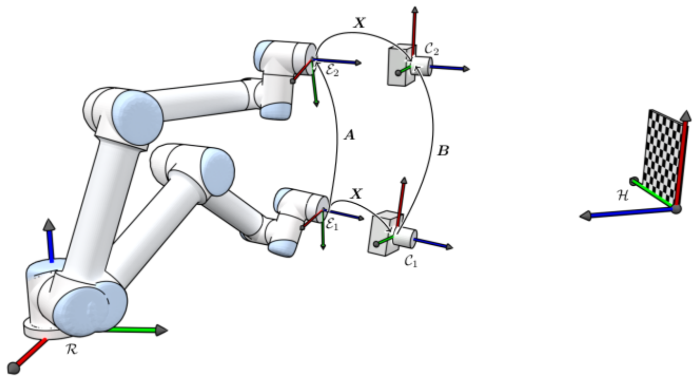


Figure 2.6: **X** is the hand-eye transform from the robot arm end-effector link to the camera on the laser scanner. Figure from [3].

Visual Fiducial

With a well calibrated high resolution camera, visual fiducials can also provide submillimeter accuracy pose estimation for the sensor [12]. There are two primary ways visual fiducials can be used:

1. Fiducial markers are placed in the environment. The sensor estimates its pose relative to the fixed markers by visually observing them, as described in works like [13, 14, 15].

2. Fiducial markers are rigidly attached to the sensor. External camera(s) observe the markers and estimate the location of the sensor. Examples of this include motion capture systems like Vicon Vantage (Vicon, Oxford, UK), and contact-free CMM's such as Absolute Scanner AS1 (Hexagon AB, Stockholm, Sweden), MetraSCAN 3D (Creaform, Levis, Canada), etc.

Visual Odometry

Visual Odometry (VO) [16, 17] provides camera based pose estimates which can be used for accumulating local scans. In VO, pose drift could cause ghosting in the reconstructed map. To improve the mapping consistency and quality, a global map optimization component can be added to VO, forming what is called visual SLAM (vSLAM) [18, 19, 20, 21]. Monocular VO or vSLAM suffers from scale ambiguity [22], so techniques like [23, 24, 25] are developed to estimate metric scale by taking advantage of Inertial Measurement Units (IMU). These are generally referred to as Visual-Inertial Odometry (VIO).

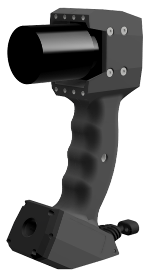
The advantage of using visual odometry as a source of localization is that the laser scanner can operate free of external infrastructure. This not only makes sensor deployment faster and more convenient, but also eliminates much of the workspace constraints associated with robot-arm based CMM's. When compared with the fiducial marker based approach, self-localizing sensors do not have to be in the line of sight of external positioning reference systems, making confined space inspection in tight corners possible with our laser scanner.

There are challenges involved with vSLAM in confined spaces because visual and geometric features can be sparse in many close-up situations. To address these problems, we developed Visual-Inertial-Laser Odometry (VILO) and VIL-SLAM [26] and achieved notable performance increase compared to the baseline VINS-Mono algorithm [27], producing sub-centimeter level pose estimation accuracy.

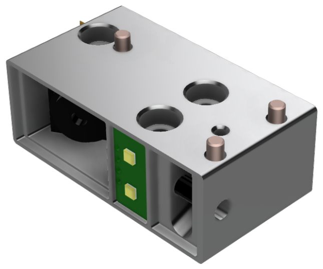
2.2 Hardware Development

Based on our active stereo laser scanner configuration, we iterated on a few hardware implementations as shown in Fig. 2.7. Fig. 2.7a and Fig. 2.7b are medium-sized hand-held scanner prototypes used for larger scale reconstructions. Fig. 2.7c and Fig. 2.7d are smaller sensors that could either be hand-held for close range scans in confined spaces or mounted on a robot arm for high precision *in-situ* scanning.

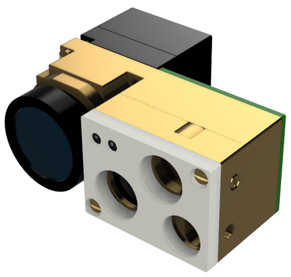
(a)



(b)



(c)



(d)

Figure 2.7: Hardware design iterations.

To make development more streamlined for different sensor configurations with different cameras and microcontroller units (MCU), we developed a modular software framework with hardware drivers as plugins that can be easily changed under different hardware configurations. Fig. 2.8 shows our sensor driver software framework. Some camera driver plugins we implemented include the following:

- A driver for generic USB Video Class (UVC) cameras.
- A Video4Linux2 (V4L2) driver for MT9M114 camera using RGB565 16-bit color format on a NXP i.MX RT1064 board.
- A driver for XIMEA cameras, specifically the MU181CR-ON model.
- A dummy driver for replaying recorded camera frames in ROS bags.

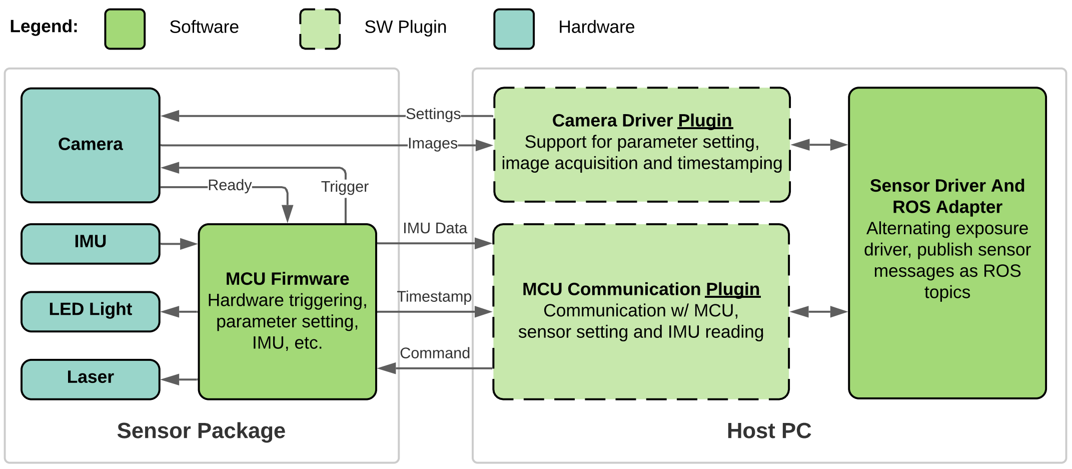

Legend:
Software
SW Plugin
Hardware
Camera
Settings
Images
Ready
Trigger
IMU
IMU Data
LED Light
Laser
MCU Firmware
Hardware triggering, parameter setting, IMU, etc.
Timestamp
Command
Camera Driver Plugin
Support for parameter setting, image acquisition and timestamping
MCU Communication Plugin
Communication w/ MCU, sensor setting and IMU reading
Sensor Driver And ROS Adapter
Alternating exposure driver, publish sensor messages as ROS topics
Sensor Package
Host PC

Figure 2.8: Our sensor driver framework has swappable driver plugins for different hardware configurations.

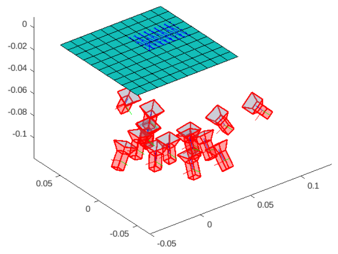
Because computer vision and vSLAM tasks are computationally expensive, it is unrealistic to run the them onboard in the sensor itself with limited space and computational power. Hence, a connection to a more powerful host computer is required to perform high level vision tasks.

The latest model of our laser scanner shown in Fig. 2.7d communicates with the host PC through two USB connections. The first one is a direct USB 3.0 connection for the high speed XIMEA camera and the second is a serial-over-USB connection for the onboard MCU. Our latest scanner is equipped with a powerful blue laser pattern projector, a low-cost BNO085 IMU and an LED flood illuminator array to light up dimly lit environments or increase camera frame brightness under low exposure times. The compact nature of this model allows us to mount it closely to an additive extrusion nozzle for inspection.

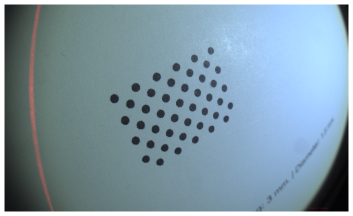
2.3 Calibration

2.3.1 Camera Parameters

There are two models we can choose from for the camera intrinsic parameter calibration: pin-hole and fisheye. The pin-hole model is suitable for cameras with a smaller field of view and less distortion. The fisheye model [28] is used for wide-angle lenses which can observe more visual features that help vSLAM to more accurately estimate its pose. We use the Matlab Computer Vision Toolbox to obtain the correct camera intrinsic and extrinsic parameters for each frame. Fig. 2.9a shows the calibrated extrinsics of an example calibration dataset. Fig. 2.9b and Fig. 2.9c show the camera images before and after undistortion.



(a) Calibrated camera extrinsics


(b) Raw image

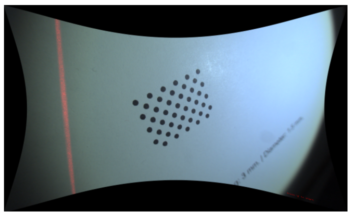

(c) Undistorted image

Figure 2.9: Camera parameter calibration and lens distortion correction.

2.3.2 Laser Plane Calibration

Once we have the intrinsic and extrinsic parameters of the camera, the next step is to calibrate the laser plane Π_l in the camera frame C. When the calibration dataset is collected, we make sure the laser stripe is also observable on the checkerboard plane. Because know the transform from C to the checkerboard frame b from the extrinsic parameters, for each laser pixel, we can generate a line-of-sight ray emanating from C, passing through the normalized laser pixel, then intersecting with Π_l to give us the laser incidence location in 3D. We sample these laser locations across all calibration image frames and perform RANSAC [29] plane fitting to find the laser plane Π_l, as shown in Fig. 2.10.

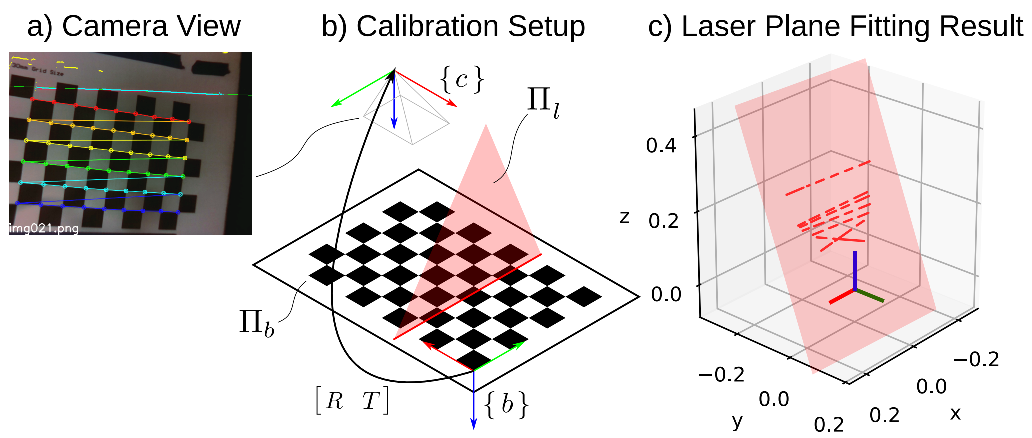


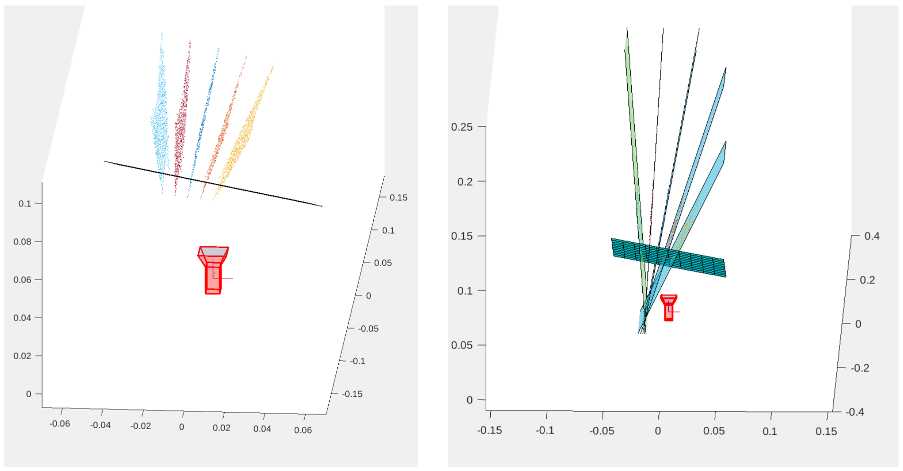
Figure 2.10: The laser plane Π_l can be found by fitting a plane over the sampled 3D laser incidence locations across calibration image frames.

The calibration for multiple laser planes in the multi-line setup is similar. With the correct laser stripe grouping, each plane can be calibrated separately just like the single-line case. Because during calibration the laser pattern is projected onto a flat plane, we can easily distinguish laser stripe segments and group them with an incrementing ID from "left" to "right", without needing to run the more involved plane identification routine described in Section 2.1.3. An example of multi-line calibration result is shown in Fig. 2.11.

2.3.3 Calibration of The Fixed Transform Between Camera Frame and Pose Tracking Frame

As described in Section 2.1.4, there are three main ways in which the pose of the scanner can be tracked. In each case, some kind of fixed transform needs to be calibrated between the frame of pose tracking and the camera frame. This is because the laser depth measurements are made in the camera frame C but the frame of pose tracking is not necessarily the same.

In the robot arm case, the pose of the end-effector is provided using forward kinematics. The end-effector frame can be related to the camera frame using a fixed transform $\mathbf{X}$. This transform is called the "hand-eye transform" and can either be directly calculated from the CAD design of the modeled system, or obtained using a



(a) Sampled laser points (b) Plane fitting result

Figure 2.11: An example calibration for multi-line laser configuration.

visual based hand-eye calibration routine. CAD based hand-eye transform calculation can be convenient but inaccurate due to manufacturing and installation error, so we chose to calibrate the hand-eye transform by framing the problem as a classic AX=XB problem [30] to optimize the transform for minimum reprojection error, and solve it using LM optimization, as formulated in [31].

In the visual fiducial case, particularly when the visual fiducial is placed on the sensor and observed by external cameras, we need to calibrate the fixed transform between the fiducial tracking frame and the camera frame. This is similar to the robot arm case and the fixed transform can be calibrated using the same method and optimization framework.

Finally in the visual odometry case, although the frame of pose tracking is just the camera frame, we need to calibrate the transform between the camera frame and the inertial measurement frame when an IMU is added to the system to help resolve the scale ambiguity of monocular VO. We use an visual-inertial calibration toolbox called "Kalibr" [32] to obtain this transform.

Chapter 3

Laser Stripe Extraction

The first step of measuring depth using our scanner is to find the incident laser points. This is commonly referred to as "laser stripe extraction". Laser stripe extraction and center finding is a well studied problem, with works like [33, 34, 35] modeling and extracting laser stripe profiles in different ways.

The accuracy of our laser scanner is not only affected by the quality of sensor modeling and calibration, but also the accuracy of the laser stripe extraction algorithm. It is important that the laser stripe center is found accurately for improving the quality of the 3D reconstruction. Additionally, sub-pixel center finding algorithms can improve the resolution of the laser scanner and reduce the stair-stepping discrete sampling artifacts. This chapter contains a discussion and comparison some laser stripe center extraction methods.

3.1 Maximum Search

Given there is only one laser plane approximately perpendicular to the camera image rows, Algorithm 2 shows one naive solution to find laser center locations is to find the index of last peak pixel intensity.

The problem with maximal search is that it extracts laser stripe center at pixel level accuracy, which causes stairstepping effect in reconstructed surfaces due to the discrete nature of measurements.

Algorithm 2 Naive maximum search for laser center

```
procedure MAXIMUM SEARCH(image)
    maxLaserLocations ← []
    for r = 1 to image.rows do
        maxValue ← -1; maxIndex ← 0
        for c = 1 to image.cols do
            if image[r][c] $\geq$ maxValue then
                maxValue ← image[r][c]; maxIndex ← c
            end if
        end for
        maxLaserLocations.append((r, maxIndex))
    end for
    return maxLaserLocations
end procedure
```

3.1.1 Maximum Center of Mass Search

Maximum center of mass [36] is a popular way of extracting laser stripe center at subpixel level accuracy. The idea is to create a window around the peak intensity pixel and calculate the "center of mass" of that window where the mass is the pixel intensity. [36, 37] argue that center of mass based method performs better in many cases than other traditional subpixel center extraction methods including Gaussian approximation, Linear approximation, Blais and Rioux detector, and Parabolic estimator. Center-of-mass based center extraction can be easily implemented, as shown in Algorithm 3.

3.1.2 Multi Maximum Center of Mass Search

The Maximum center of mass search handles single-line laser well but does not work for the multi-line case because it only finds one peak. We solved this problem by doing multi-maximum center of mass search (Algorithm 4), which yields more than one detections per row as long as each peak has a width within a given range in the search direction.

Algorithm 3 Maximum Center of Mass For Subpixel Laser Center

```
1: procedure MAXIMUM CENTER OF MASS SEARCH(image)
2:     maxLaserLocations ← Maximum Search(image, halfWindowSize)
3:     subPixelMaxLaserLocations ← []
4:     for (r, c) in maxLaserLocations do
5:         if c < halfWindowSize or c > image.cols − halfWindowSize then
6:             continue
7:         end if
8:         trueCenter ← 0
9:         for cc = (c - halfWindowSize) to (c + halfWindowSize) do
10:            trueCenter += image[r][cc]
11:        end for
12:        trueCenter ← trueCenter / (halfWindowSize * 2)
13:        laserLocations.append((r, trueCenter))
14:    end for
15:    return subPixelMaxLaserLocations
16: end procedure
```

3.1.3 Steger's Method

Algorithm 4 finds center along a search direction based on assumption that the laser stripe transverse direction is parallel to the search direction. This is often not true because the stripe pattern orientation could be of any orientation depending on the shape of the scanned object. [38] identified this problem with Maximum center of mass search and introduced an improved method to search along the true stripe transverse direction by estimating the local stripe curvature. Another method that searches for center along true stripe transverse directions is the Steger's method for extracting curvilinear structures in images [39]. To understand it, we first look at the 1D case where we want to find the center of an 1D parabolic peak signal with peak height h and width $2w$:

$$f_p(x) = \begin{cases} h(1-(x/w)^2), & |x| \le w \\ 0, & |x| > w \end{cases} \tag{3.1}$$

A simple idea to solve for extrema in a function is to find locations where the first order derivative is zero. The first order derivative can be done by convolving the

Algorithm 4 Multi maximum search for laser centers

```
1: procedure Multi-Maximum Center of Mass Search(image, threshold, minWidth, maxWidth)
2:     weights ← []
3:     maxLaserLocations ← []
4:     for r = 1 to image.rows do
5:         inState ← OUT
6:         for c = 1 to image.cols do
7:             if image[r][c] ≥ threshold then
8:                 if inState == OUT then
9:                     inState ← IN
10:                    $i \leftarrow 0$
11:                end if
12:                weights.append(image[r][c])
13:                $i \leftarrow i + 1$
14:                if i > maxWidth then                         ▷ Peak is too wide
15:                    inState ← OUT
16:                end if
17:            else
18:                if inState == IN then
19:                    inState ← OUT
20:                    if i > minWidth then                     ▷ Peak detected
21:                        centerCOM ← Center of Mass(weights)
22:                        maxLaserLocations.append((r, c - i + centerCOM))
23:                    end if
24:                end if
25:            end if
26:        end for
27:    end for
28:    return maxLaserLocations
29: end procedure
```

signal with a first order derivative kernel. Additionally, for real world signals, we need to filter the signal with a Gaussian kernel first to eliminate the high frequency noise. Because the linearity of both kernels, we can compose the two filters into one Derivative of Gaussian (DoG) filter:

$$g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.2}$$

Now with the response $r_p(x, \sigma, w, h) = g_\sigma(x) * f_p(x)$, we can look for zero crossings and identify that as a stripe center if the second order derivative at that point is bigger than a threshold (indicating rate of change in intensity). A problem with this approach, however, is that in the discrete case, the zero crossings would be at discrete pixel level locations, producing a center estimate that is not subpixel level. To solve this problem, we can use Taylor expansion on the signal $f_p(x)$ at each pixel location n:

$$f_p(n, x) \approx r_p + r'_p x + \frac{1}{2} r''_p x^2 \tag{3.3}$$

Where r_p, r'_p and r''_p are filter responses to the DoG filter g_σ, the first derivative of the DoG filter g'_σ and the second derivative of the DoG filter g''_σ respectively. Given the expansion (3.3), we set its first derivative to zero:

$$f'_p(n, x) = r'_p + r''_p x = 0 \tag{3.4}$$

$$r''_p x = -r'_p \tag{3.5}$$

$$x = -\frac{r'_p}{r''_p} \tag{3.6}$$

If this estimated center x lies within the pixel n (i.e. $x \in \left[-\frac{1}{2}, \frac{1}{2}\right]$) and the second order derivative r''_p at this location is large enough, we consider the subpixel center of detected peak to be $n + x$.

Now we generalize to extracting laser stripe centers in a 2D image. [39] argues that 2D curves exhibit the same profile characteristic as the 1D case along the transverse direction of the curve. In this case, we need to estimate the local direction of the stripe for the same 1D analysis to be applied here. To estimate the local curve orientation at point (x, y), we can first calculate the Hessian matrix at that point:

$$H(x, y) = \begin{bmatrix} r_{xx}, r_{xy} \\ r_{xy}, r_{yy} \end{bmatrix} \tag{3.7}$$

where $r_{xx}, r_{xy}, r_{xy}, r_{yy}$ are partial derivatives around point (x, y), and find the orientation by taking the eigenvector corresponding to the maximum eigenvalue of the Hessian $H(x, y)$. The normalized eigenvector is denoted as (n_x, n_y), and with some geometry we can apply (3.6) and produce the subpixel center fit (p_x, p_y) along the laser stripe's transverse direction:

$$(p_x, p_y) = (tn_x, tn_y), \quad \text{where: } t = -\frac{r_x n_x + r_y n_y}{r_{xx} n_x^2 + 2r_{xy} n_x n_y + r_{yy} n_y^2} \tag{3.8}$$

3.2 Experimental Comparison

Fig. 3.1 shows an experiment comparing the laser stripe extraction results between Maximum search, Maximum center of mass and Steger's method. We first observe that subpixel center extraction methods (2b and 3b) effectively reduce the stairstepping effect in the pixel level extraction method (1b). We further observe that the laser stripes detected by Steger's method are smoother when compared with that produced by both the Maximum search and Maximum center of mass search. Because the partial derivatives of spatially close pixels are similar, the estimated direction of curves at these pixels change gradually. This causes the detected curves to be smooth and more continuous. One advantage of having smooth laser stripe detections is that the reconstructed 3D map will have have less discontinuity in curvature. However, the smoothing effect causes Steger's method to perform poorly at sharp edges. Shown in Fig. 3.1 (3a) is a clear example of the Steger's method over-smoothing a few sharp edges, producing inaccurate center estimates.

3.3 Realtime Implementation

Real-time 3D reconstruction is needed for in process closed loop control. Maximum search, maximum center of mass search and multi-maximum center of mass search are computationally cheap and *embarrassingly parallelizable* so they can easily run at high framerates. For computationally expensive stripe extraction algorithms such as the Steger's method, an efficient implementation is required to make real-time stripe extraction possible.

We first implemented the Steger's method in Matlab. Through time profiling We identified four main sub-procedures that take the most time:

- Pre-processing: color space conversion.
- Hessian: DoG filtering and Hessian computation.

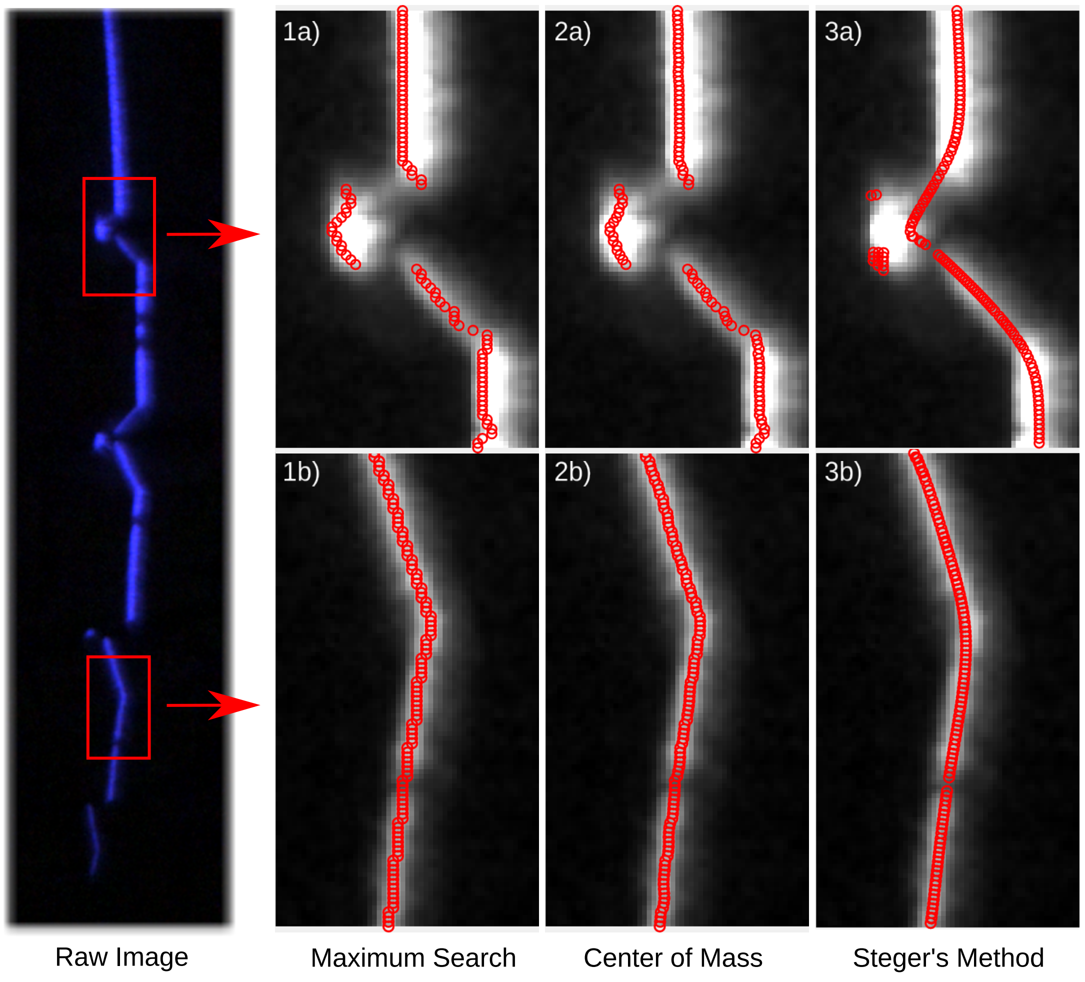


Figure 3.1: A comparison between laser stripe extraction results from Maximum search, Maximum center of mass and Steger's method.

- Direction Estimate: computing eigenvalue and eigenvectors.
- Center Finding: sub-pixel offset calculation.

Among these four sub-procedures, the DoG filtering and construction of Hessian took the longest time, which is expected for high resolution images. Tab. 3.1 shows that the baseline Matlab implementation only ran at 2Hz@2688x1620 even with all vectorized operations. We then implemented a C++ version using the OpenCV's well optimized filtering methods, yielding a significant performance increase of 3.6x for the serial C++ implementation for high resolution at 2688x1620. Because Hessian calculation and stripe direction estimate take the most time, we attempted to further speed up these processes using threading. The Hessian computation is parallelizable by computing the four partial derivatives in 3.7 concurrently. The stripe direction estimate is sped up by breaking down the whole image into smaller blocks for parallel eigenvalue/vector computation. As shown in Tab. 3.1, the parallel C++ implementation achieved a 4.0x performance increase over Matlab, which is 11% better than the serial C++ implementation. This improvement from threading is less than expected on a 8-core machine because the Gaussian filtering process takes the most time and its implementation in OpenCV is already well optimized using the separable property. It is not easy to further speed up the Gaussian filtering sub-procedure[1].

Decreasing the image size would reduce total computation time, as shown in Tab. 3.3. However, running the camera at lower resolution or downsampling sacrifices the depth resolution of the scanner. One way to work around this is to apply a region of interest (ROI) to the camera image to reduce the laser stripe search space. This is possible because given the depth sensing range of the scanner, part of the image where the laser incidence would be too out of focus to be reliable can be skipped. Tab. 3.2 shows a reasonable real-time performance ($>$ 20Hz) of the Steger's method at a reduced search resolution using ROI without sacrificing the resolution of depth measurements.

3.4 Conclusion And Future Work

In this chapter, we compared several laser stripe center extraction methods including Maximum search, maximum center of mass and Steger's method. We observe that maximum center of mass provides the benefit of added sub-pixel resolution compared to maximum search and also reduces the stair-stepping effect due to discrete sampling. Steger's method provides the smoothest stripe detections but perform poorly on sharp edges. It is also computationally more expensive than both other methods. A hybrid approach using Steger's method for long, continuous laser stripes and maximum center of mass for sharp edges might combine the advantage of both methods. We believe there is room for further improving the accuracy and resolution of the scanner by developing better laser stripe center extraction methods.

[1]GPU accelerated implementation of Gaussian filtering is possibly faster but not tested here.

Resolution	@2688*1620		
Sub-procedure	**Matlab**	**C++ Serial**	**C++ Parallel**
Pre-processing (ms)	9.00	3.53	3.67
Hessian (ms)	204	109	108
Direction Est. (ms)	250	20.1	8.20
Center Finding (ms)	28.0	2.10	2.10
Total (ms)	491	134	122
FPS (Hz)	2.03	7.42	**8.19**

Table 3.1: Steger's method performances at high resolution

Resolution	@2688*480		
Sub-procedure	**Matlab**	**C++ Serial**	**C++ Parallel**
Pre-processing (ms)	2.00	0.227	0.283
Hessian (ms)	59.0	29.3	28.2
Direction Est. (ms)	60.0	12.0	6.50
Center Finding (ms)	7.00	2.09	2.11
Total (ms)	128	43.6	37.1
FPS (Hz)	7.81	22.9	**27.0**

Table 3.2: Steger's method performances at high resolution with ROI

Resolution	@640*480		
Sub-procedure	**Matlab**	**C++ Serial**	**C++ Parallel**
Pre-processing (ms)	1.00	0.0523	0.219
Hessian (ms)	13.0	2.42	4.86
Direction Est. (ms)	12.0	4.14	4.07
Center Finding (ms)	6.00	0.51	0.50
Total (ms)	32.0	7.12	9.65
FPS (Hz)	31.2	**140**	103

Table 3.3: Steger's method performances at low resolution

Chapter 4

Simulation

4.1 Motivation

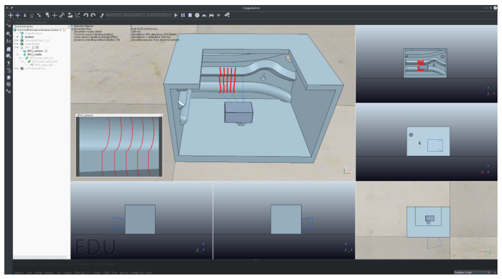
In robotic additive manufacturing scenarios, having a full simulation of the manufacturing process and sensor feedback in a mockup environment speeds up the development of control algorithms by enabling parallel development of software and hardware, as well as reducing the risks and cost associated with working on physical systems. In this work, we developed a photorealistic simulation environment with a digital twin of our proposed laser scanner, an integrated fluid solver for simulating additive material fluids, robot arm and robotics software (specifically ROS) integration.

4.2 Photorealistic Laser Scanner Simulation

Photorealism in the simulation of our sensor is desirable because the scanner is largely visual based. The projection and reflection of the laser pattern needs to be realistic for effectively developing and testing the vision pipeline such as the laser stripe center finding algorithm.

Most popular robotics simulation environments such as Gazebo [40] and V-REP (Now CoppeliaSim) [41] provide great support for robotics applications but lack photorealism. To create a photorealistic simulation for a laser scanner, we take advantage of the graphics capabilities of modern game engines. In particular, Unreal Engine 4 (UE4) [42] provides great graphics support including ray-tracing for real-time rendering of scenes with dynamic light sources.

Fig. 4.1a and Fig. 4.1b shows the V-REP and UE4 simulation scenes for our scanner respectively. From our experiment we found out the UE4 based simulation was superior than V-REP based version, in the metrics that we care about shown in Tab. 4.1.


(a) V-REP simulation

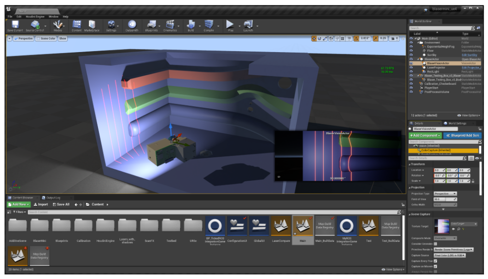

(b) UE4 photorealistic simulation

Figure 4.1: Two different simulation environments of our proposed laser scanner.

Comparison done on the same PC with a RTX 2070 graphics card		
Metric	**V-REP**	**UE4**
Framerate	Around 8 FPS[1]	Capped at 80 FPS
Rendering Laser Projector	CPU serial	GPU parallel
Laser Pattern Quality	Poor, discrete	Great, smooth
Dynamic Lighting	Yes	Yes
Physics Based Camera[2]	No	Yes
Ray-Tracing	Very slow [43]	Real time
ROS Interface	Yes, built-in	Yes, with plugin[3]

[1] Complexity is $O(n \times d)$ where n represents the size of projected pattern and d represents the sampling density.
[2] Meaning the virtual scene capture can be configured using real world parameters such as exposure, aperture, white balance, etc.
[3] UE4-ROS plugins include: `ROSIntegration` and `ROSIntegrationVision` [44]. Contributions back to these open source libraries were made to make this work.

Table 4.1: A comparison between V-REP and UE4 based laser scanner simulation

Fig. 4.2 demonstrates the advantage of having photorealism for the simulation of a vision-based laser scanner. The intensity profile of a UE4 simulated laser stripe in 3b) resembles that of a real laser stripe shown in 2b), with a Gaussian-like [35] profile in the transverse direction of the stripe pattern. On the other hand, the profile of a V-REP simulated laser stripe in 1b) does not follow such a distribution, plus having visible discontinuity artifacts on curves due to low sampling rate (higher sampling rate would cause unacceptable framerate).

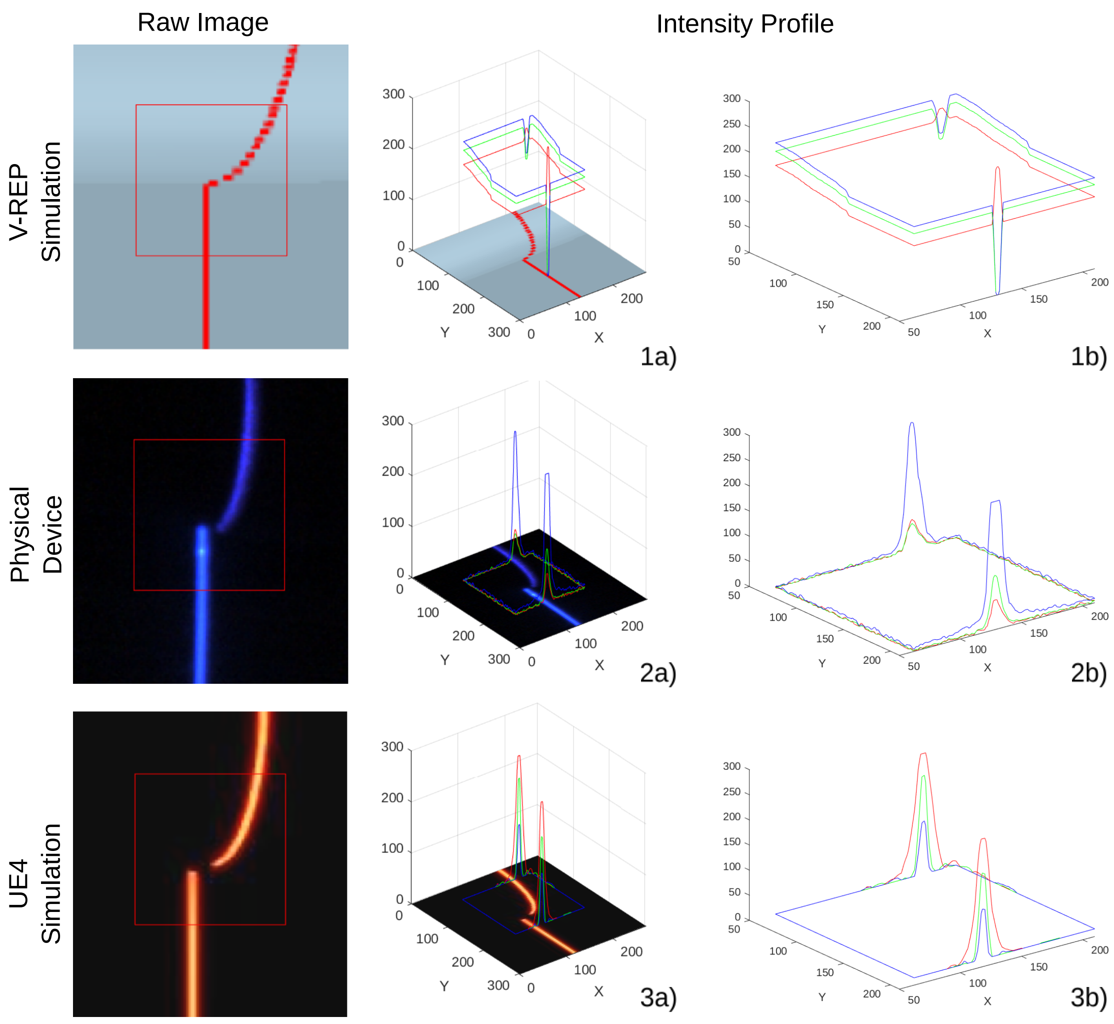


Figure 4.2: An RGB intensity comparison between simulated laser in V-REP, real world laser and simulated laser in UE4.

Fig. 4.3 shows the architecture of our UE4 based simulation of our laser scanner. The simulation is designed to seamlessly work with the high level robotics software stack, with the same ROS messages and services used across physical devices and simulated devices. Compared to the hardware architecture shown in Fig. 2.8, the physical camera is replaced with a perspective scene capture in UE4; the LED flood illuminator is replaced with a dynamic rectangle light source; and the laser projector is replaced with a GPU accelerated texture projection module [1] which could project an arbitrary 2D image pattern onto 3D objects just like projectors in the real world. We did not spend time modeling the IMU output because it is only needed for vSLAM, which is not the focus of our proposed simulation environment.

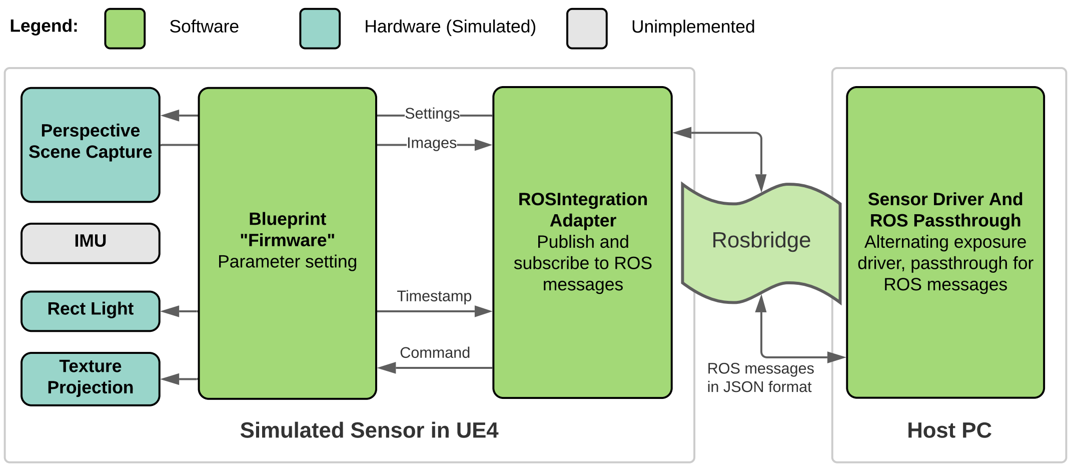

Legend:
Software
Hardware (Simulated)
Unimplemented
Perspective Scene Capture
IMU
Rect Light
Texture Projection
Blueprint "Firmware" Parameter setting
Settings
Images
Timestamp
Command
ROSIntegration Adapter Publish and subscribe to ROS messages
Rosbridge
ROS messages in JSON format
Sensor Driver And ROS Passthrough Alternating exposure driver, passthrough for ROS messages
Simulated Sensor in UE4
Host PC

Figure 4.3: The simulated sensor in Unreal Engine 4 (UE4) communicates with and responds to host PC via ROSIntegration data adapter and rosbridge for cross host and/or platform compatibility.

4.3 Robot Arm Simulation

The robot arm we use for development is the UR5e (Universal Robots A/S, Odense, Denmark). The physical arm is controlled using a Linux machine running UR PolyScope software. This software can be run in a detached simulation mode without the physical hardware. This conveniently allows us to use the same commands to servo a virtual robot arm, as illustrated in Fig. 4.4. With the ROS driver [45] for UR

[1]Inspired by: https://github.com/ChristopherRemde/Projection_shadow

Arms, the robot arm joint states were published as a ROS topic. We then built a virtual UR5e robot arm in UE4 using the CAD models and implemented scripts to make the virtual robot arm mirror the published joint states of the simulated arm in PolyScope, as shown in Fig. 4.5.

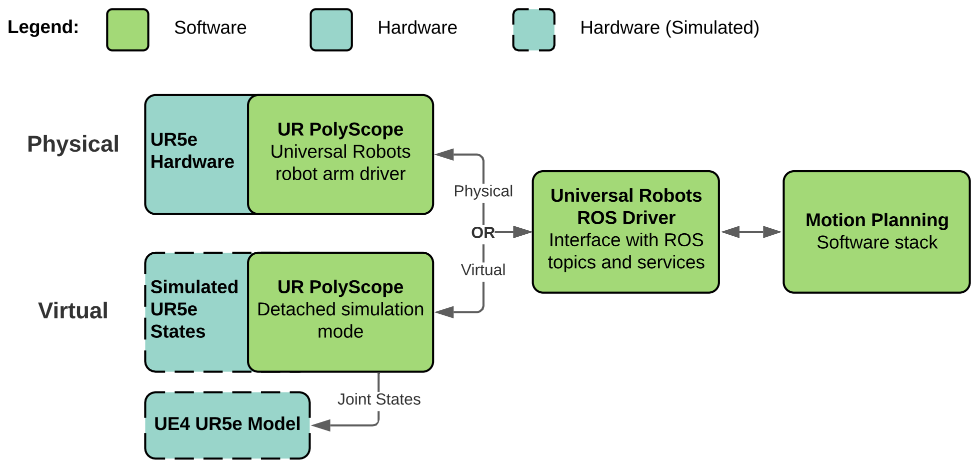
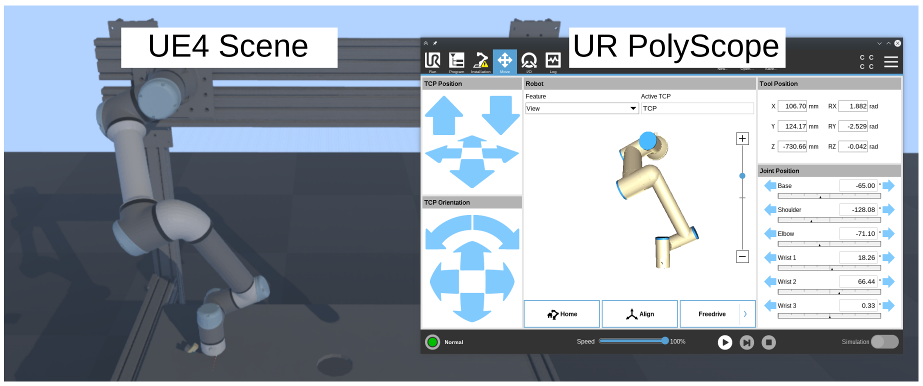

Legend:
Software
Hardware
Hardware (Simulated)
Physical
UR5e Hardware
UR PolyScope Universal Robots robot arm driver
Physical
OR
Virtual
Universal Robots ROS Driver Interface with ROS topics and services
Motion Planning Software stack
Virtual
Simulated UR5e States
UR PolyScope Detached simulation mode
Joint States
UE4 UR5e Model

Figure 4.4: Switching between physical and virtual robot arm is seamless with the same robot arm command interface through PolyScope.



Feature: View
Active TCP: TCP

Tool Position

X	106.70	mm	RX	1.882	rad
Y	124.17	mm	RY	-2.529	rad
Z	-730.66	mm	RZ	-0.042	rad

Joint Position

Base	-65.00 °
Shoulder	-128.08 °
Elbow	-71.10 °
Wrist 1	18.26 °
Wrist 2	66.44 °
Wrist 3	0.33 °

Figure 4.5: The visualized robot arm in Unreal Engine 4 (UE4) mirrors the simulated robot arm states in UR PolyScope.

4.4 Fluid Simulation for Additive Manufacturing

To simulate the material extrusion and curing process, we first model the material as a fluid with changing viscosity with respect to temperature. Then, we use a fluid solver to simulate the extrusion, thermal cooling and curing of the additive material. The fluid solver we chose to work with is the FLIP fluid solver plugin for Houdini (SideFX, Toronto, Canada). Houdini is a software used for generating digital assets including many physics based simulations for creating convincing visual effects and is heavily used in the digital content creation industry. Because of this, Houdini implements a plugin in UE4 for "baking" digital assets. In our case, the digital asset being baked and imported into our UE4 additive simulation scene from Houdini is the simulated material. However, the asset baking and importing plugin provided by Houdini only supports updating assets during editing, not running. Assistance provided by Albert Xu is greatly appreciated, for modifying the plugin to make run-time asset updates possible. To make the simulated material extrusion and curing realistic, we set the material temperature-viscosity curve based on an experimentally derived table for a particular PLA material in [46].

Fig. 4.6 shows the software architecture of our additive material extrusion simulation in Houdini and UE4. The green blocks represent software that we wrote / modified; the gray blocks are third party software that we took advantage of. To print a 3D object using a robot arm, we first use Ultimaker Cura slicer to slice it into G-Code that regular 3D printers understand. We then break down the G-Code trajectory into smaller pieces and convert them into waypoints. Next, we send the waypoints to the MoveIt planning and execution library for executing the 3D printing motion on the robot arm. The motion of the robot arm is captured by the UE4 scene and triggers fluid simulation updates which advances the simulation time of the fluid, requests the updated fluid mesh from Houdini, and then updates the visualization. Because the fluid simulation cannot happen at real-time, the next waypoint in the printing trajectory is only executed after the previous fluid simulation update finishes.

Fig. 4.7 shows two simulated prints in our environment. The rainbow color mapping of the printed material correspond to the temperature of the material. The material is initially hot and fluid (red/orange), then cures as it cools down over time (turns blue). Thermal transfer between layers is possible so that heat clusters causing

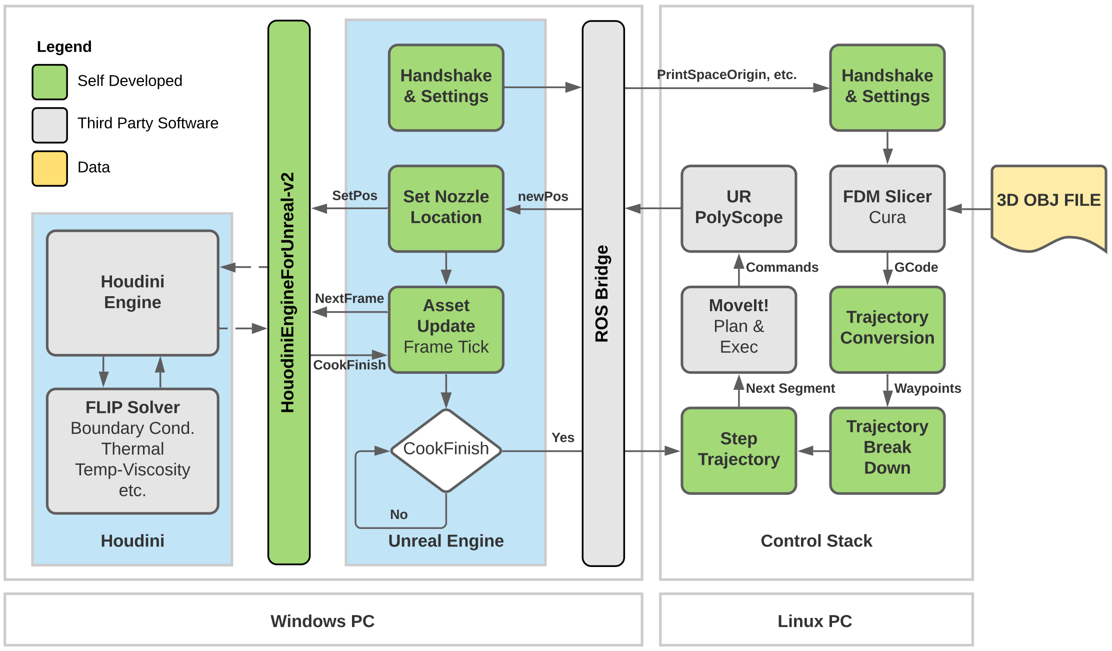
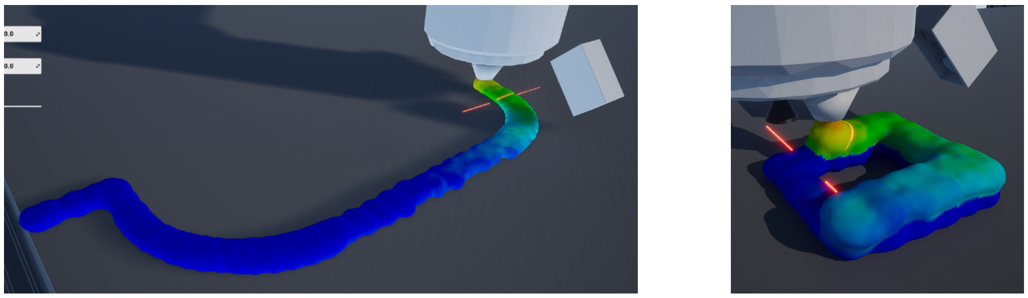


Figure 4.6: Architecture diagram and data flow of the fluid simulation in Houdini, visualization of the fluid in UE4 and the robot printing trajectory control software.

defects in the printed product is modeled. Our laser scanner is placed to the side of the FDM nozzle for in-situ inspection by examining the 3D profile of newly extruded material. An example simulated robotic additive printing scenario as well as the *in-situ* inspection result is demonstrated in Chapter 6, Section 6.2.



(a) Single layer path (b) Multi-layer stacking

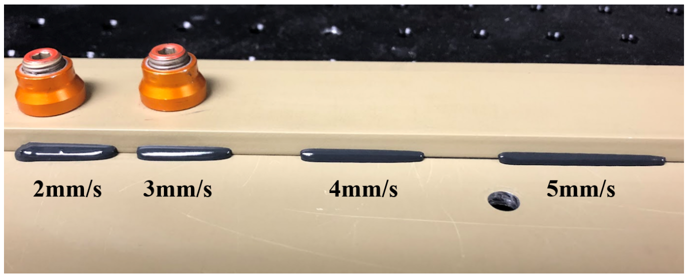
Figure 4.7: FDM additive process simulation using FLIP fluid solver with realistic temperature-viscosity curve.

Chapter 5

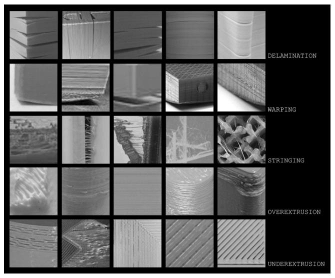
Alternating Shutter Approach

5.1 Motivation and Challenges

There are two types of information that really helps informing the control algorithm about the current state of the printed material. The first type is the depth profile of the extruded material. The second type is visual imagery. Depth profile can be used to actively control the nozzle tip velocity because there is a direct correlation between the nozzle tip velocity and the thickness of extruded material (given the same material flow rate and other parameters), as shown in Fig. 5.1a. Meanwhile, visual imagery can be used to detect manufacturing defects using machine learning based image classifications [47, 48, 49] such as the ones shown in Fig. 5.1b.



(a) Depth feedback can be used for servoing nozzle velocity



(b) Defect detection and classification

Figure 5.1: Example applications using depth profile and visual imagery feedback in additive manufacturing. Figure (b) from [4].

The challenge with providing both depth information and visual feedback using only one camera in our system is the conflicting requirement on camera exposure level. For depth measurements, it is ideal to have low exposure images so that only the bright laser incidence is visible in the camera images, which makes laser stripe extraction as described in Section 3.1 easier. On the other hand, for visual imagery, high exposure is desirable because a brighter image contains more visual textures that could be analyzed.

To solve this problem, we proposed an "alternating shutter" approach [50] to generate two video streams with bright and dim frames "simultaneously" from just one camera. We also created a tight, hardware synchronized implementation of this approach to make sure two video streams are perfectly separated.

5.1.1 Other Benefits

Aside from enabling two models of sensory feedback in the additive manufacturing inspection scenario, the alternating shutter approach also enables the following applications:

Point Cloud Colorization

We can use the adjacent bright image frames with texture information about the scanned object to colorize the reconstructed 3D point cloud. This is done by projecting the reconstructed sectional 3D point cloud from each dim frame to its adjacent bright image frame and colorizing the points using the color information on the projected pixel coordinates (assuming the sensor pose for both image frames are known).

vSLAM with our monocular laser scanner in confined space

Feature based vSLAM works best when there are an abundant amount of visual features that can be reliably tracked, so bright image frames are best for vSLAM, similar to the additive defect detection scenario. The alternating shutter approach enables this type of odometry source (VO or vSLAM) by providing bright image frames for feature tracking. Our work [26] outlines this novel setup for making confined space vSLAM possible using our laser scanner.

5.2 Alternating Shutter Approach

5.2.1 The Approach

The high level idea of "alternating shutter" is to toggle the camera exposure as well as the laser every other image frame. When acquired at a high framerate, the two image streams are taken in close spatial vicinity and look as if one camera is producing two distinct image streams.

In the bright frame, we illuminate the environment using the flood illuminator onboard and/or turn up the camera exposure time with the laser turned off. Laser is turned off to prevent incident patterns from interfering with image texture or feature extraction and analysis.

In the dim frame, we turn off the flood illuminator, turn on the laser and turn down the camera exposure time. This gives us images with only laser incidence visible with pitch-black background.

Fig. 5.2 shows some example scanning sequences where the images on the top row are bright frames for visual analysis and the bottom row are the adjacent dim laser frames taken in the near time and spatial vicinity.

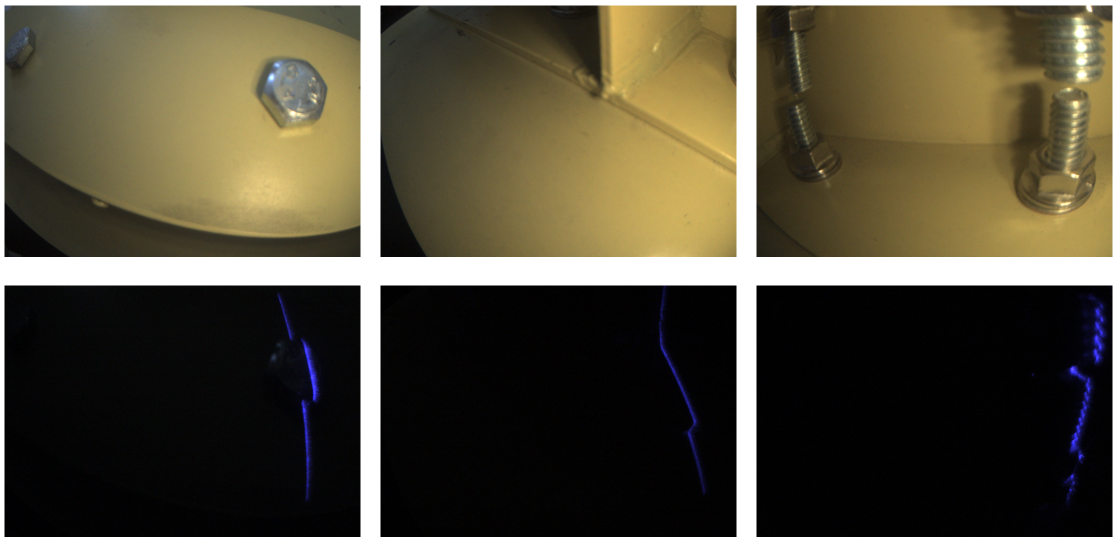


Figure 5.2: Alternating shutter method produces two distinct image streams using one single camera.

5.2.2 The Implementation

In the alternating shutter approach, it is necessary to toggle the laser and LED precisely before each frame exposure starts to prevent the laser stripes from being visible in the incorrect frame. If the synchronization is not precise, the laser stripe will "bleed" into the final reconstructed colorized point cloud, as shown in Fig. 5.3. To implement a precise hardware based synchronization, we choose the XIMEA MU181CR-ON camera which supports hardware signal triggering and outputs a signal indicating if it's ready to acquire the next frame.

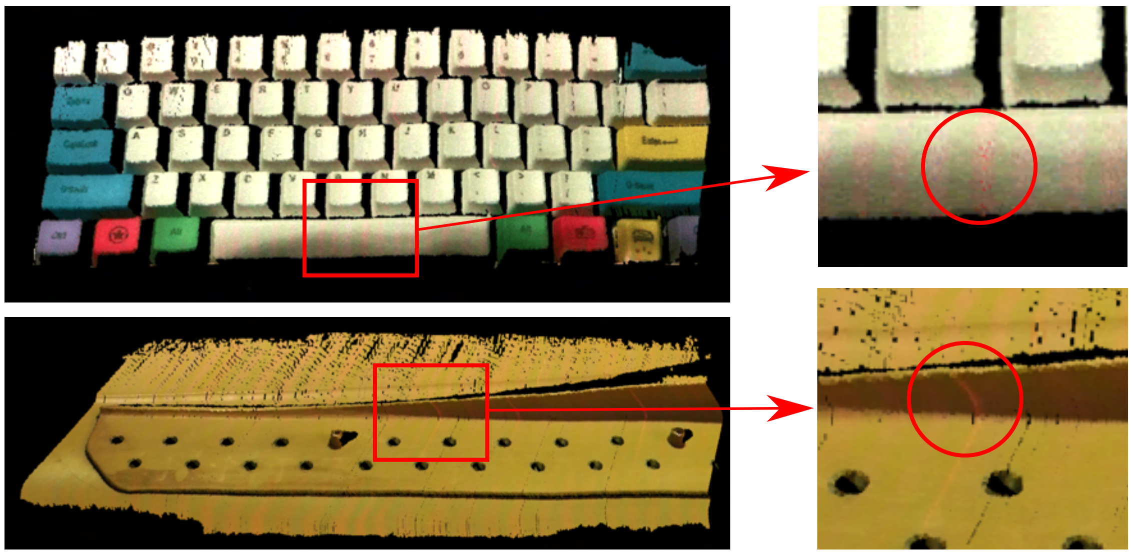


Figure 5.3: Without proper synchronization, the laser stripe may 'bleed' into the bright frames where it is not supposed to be in. This causes the colorized point cloud to show banding patterns with the color of the laser.

To correctly synchronize laser/LED with camera trigger signal while maximizing framerate, we developed a sensor firmware and driver suite to produce tight triggering right after the camera is able to take the next shot. Fig. 5.4 is a sequence diagram detailing the interaction and timing between hardware, firmware and software driver components. The laser and LED are toggled right before the microcontroller unit (MCU) triggers the camera exposure to prevent laser stripe from bleeding into bright frames used for colorization. In addition, the MCU clock is synchronized with the host computer's clock, providing synchronized timestamps for both the incoming

Resolution: 1000x700, statistics over window of 350 frames		
Metric	**Variable Exposure**	**Constant Exposure**
Average Framerate (FPS)[1]	29.738[2]	51.739
Average Frame Time (s)	0.0336	0.0193
Max Frame Time (s)	0.0430	0.0201
Min Frame Time (s)	0.0331	0.0182
Std Dev (s)	0.00021	0.00031

[1] The average framerate of the bright frames used for vSLAM. This is half of the total framerate where the other half frames are dim and used for laser stripe detection.
[2] To fairly measure the exposure setting overhead, "Variable Exposure" is toggling camera exposure time between 100us and 100us, the same with Constant Exposure.

Table 5.1: A comparison of bright images' framerate between variable camera exposure time vs constant exposure time.

images and IMU data.

Through experiments we discovered that setting the exposure of the camera through the XIMEA API is quite time consuming and can lower the overall framerate dramatically. With a bright LED flood illuminator to light up the scene, we can cut down on this time cost by keeping camera exposure time constant at a low level. Tab. 5.1 shows the overhead of setting the XIMEA camera exposure time is around 0.0143 second each time, which caused an overall 22 FPS (43%) drop in performance.

5.3 Experiment Results

We show the effectiveness of our hardware synchronized alternating shutter approach by scanning an industrial part (Fig. 6.4) and showing no sign of laser stripe bleeding in the colorized point cloud. The experiment is detailed in Section 6.3.

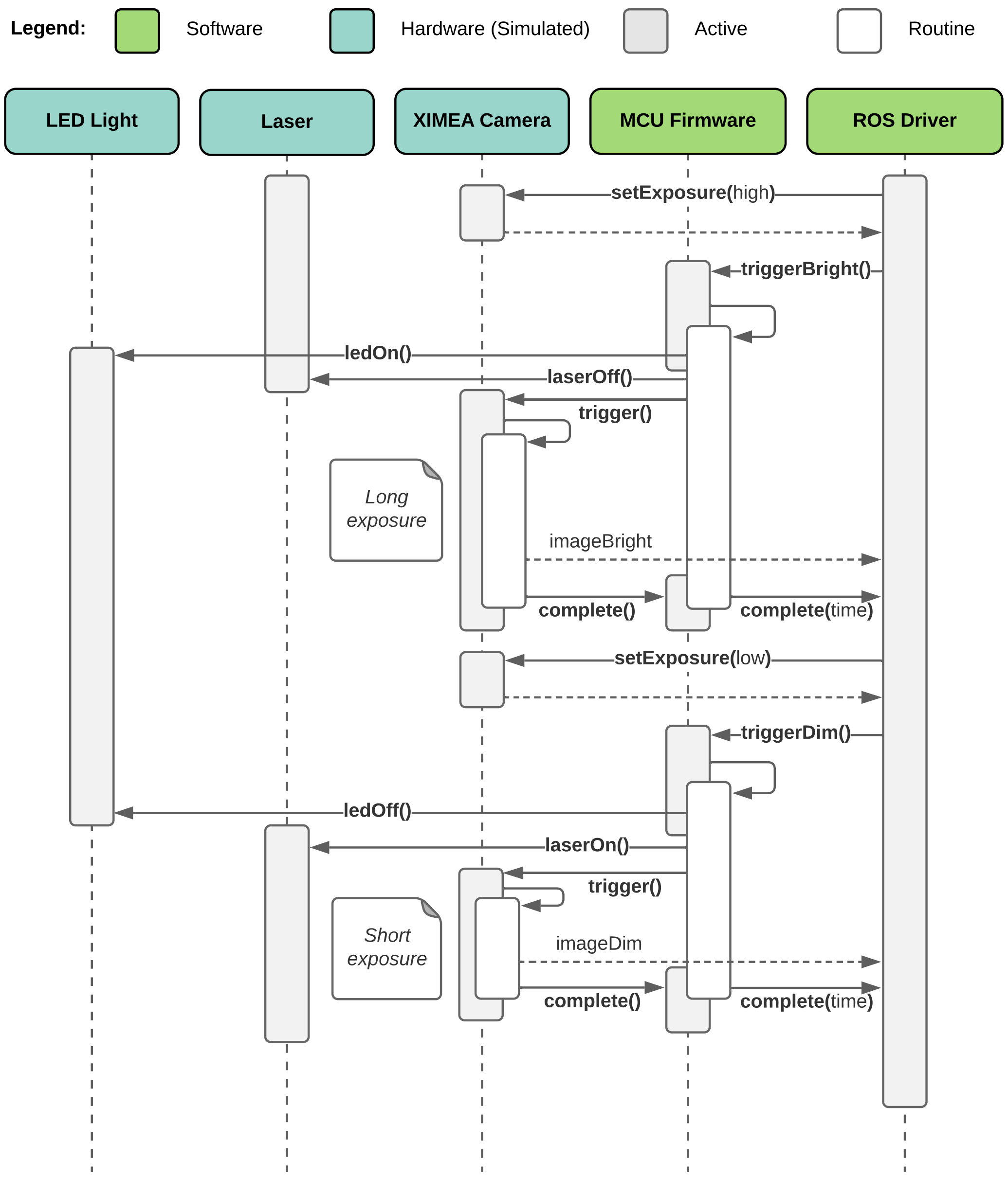


Figure 5.4: Sequence diagram of the synchronized alternating shutter method.

Chapter 6

Experiments

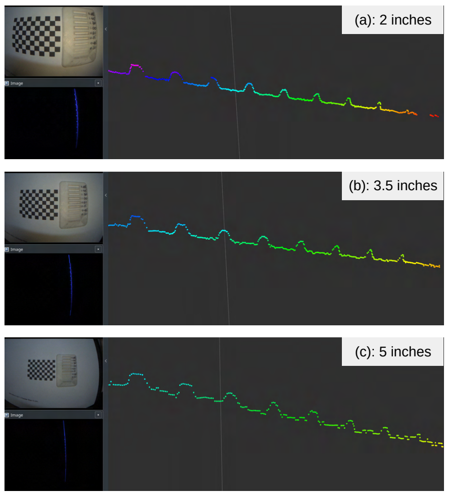
6.1 Scanner Depth Resolution Evaluation

One important evaluation metric for a laser scanner is its depth resolution. We tested the resolution of our scanner by measuring gaps of various widths. We 3D printed a test part with incrementing gaps from 0.2mm to 1.7mm, and measured the ground-truth gap widths using a vernier caliper and a set of steel feeler gauge. We compared the gap widths measured at three different heights from the surface against the ground-truth to understand the scanner's resolution performance at different sensing distances.

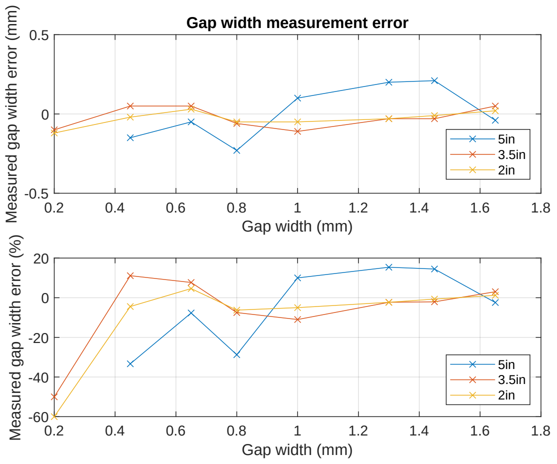
Fig. 6.1 shows the sensor resolution evaluation with gap widths measured at 2 inches, 3.5 inches and 5 inches away from the gaps. The laser depth profile shown in Fig. 6.1a does not have sharp edges because of the subsurface scattering of 3D printed plastic. Because of this, we get the gap width by measuring the middle part of the gap geometry. Fig. 6.1b shows the trend of the measurement error over the gap size, for the three different scanning distances respectively.

From the experiment we can make the following remarks about our scanner:

- The minimum detectable gap width is around 0.20mm when the sensor is within 3.5 inches away from the scanned object.
- The measurement resolution improves with closer scanning distances (no less than the minimum of 1 inch).



(a) Gap measurements made at different distances



(b) Measurement error trend over gap size

Figure 6.1: Two different simulation environments of our proposed laser scanner.

- There is no guarantee of less than 10% measurement error for small gaps less than 0.4mm.

6.2 Additive Simulation Evaluation

In this experiment, we test the effectiveness of our robotic additive manufacturing simulation environment by running a print trajectory and obtaining the simulated sensor feedback. Our proposed sensor has limited field of view and sensing range, allowing it to sample a sectional profile at a time. To increase the efficiency and quality of inspection, a strategic trajectory can be chosen such that coverage is optimized given a limited length or some other constraints. In this chapter, we propose a simple yet effective sensor placement strategy that maximizes scan coverage for our scanner when mounted on a robot arm next to the additive extrusion nozzle. We evaluate the effectiveness of this strategy by testing it in the AM simulation environment developed in Chapter 4.

Caster wheels are widely used in mechanical designs for passive and compliant support. One characteristic of a caster wheel is that its orientation does not change instantly when the driving force changes direction. The friction on the wheels slowly

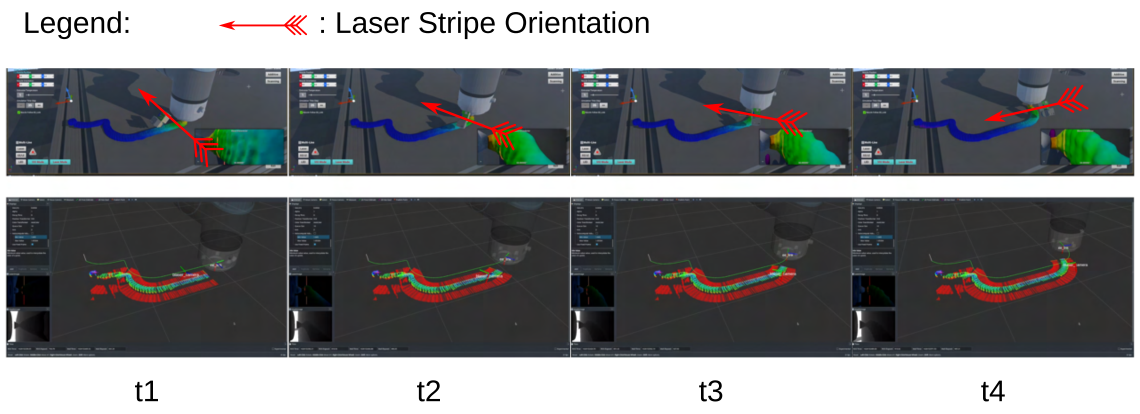
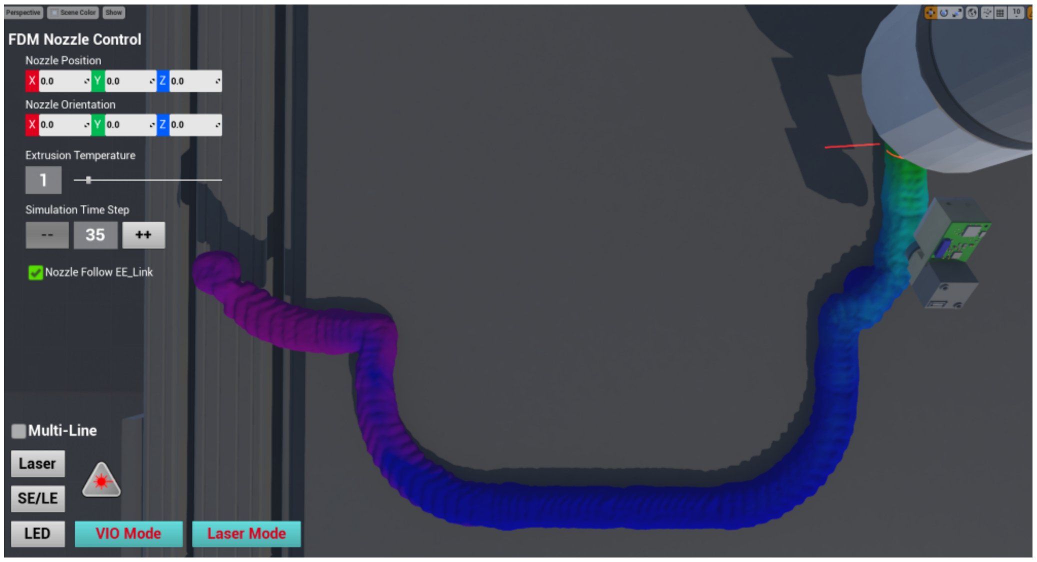


Figure 6.2: A scan demonstration using virtual caster wheel orientation planning strategy. Top row: simulation environment. Bottom row: reconstructed result using our scanner. Note the change in end-effector orientation is gradual but mostly perpendicular to the printing trajectory, giving us an efficient coverage of the printed material.

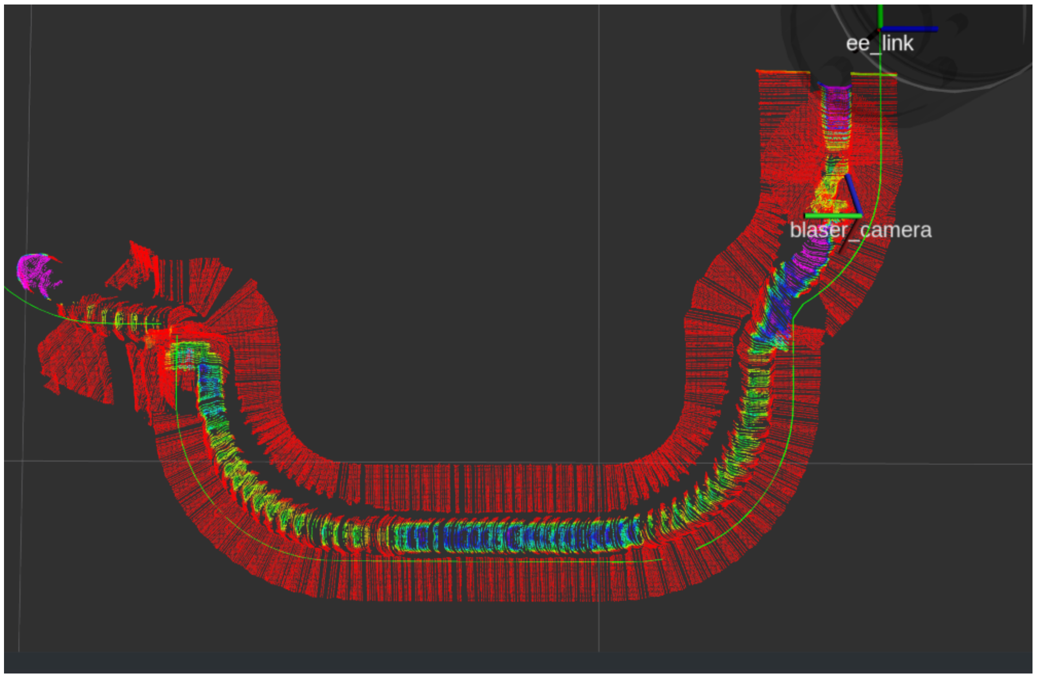
turn the caster wheel, eventually making the wheel parallel to the orientation of the moving force.

Compared to regular cartesian 3D printers where the state is only the xyz positions, robot arms usually have 6 Degrees of Freedom (DOF) on its end-effector. Simply asking the robot arm to follow the G-Code waypoint positions with a fixed orientation causes bad scan coverage especially when the arm motion is parallel to the laser stripe. A good rule of thumb for good scan coverage is to always have the sensor placed perpendicular to the extruded material, so that each sectional sample could cover as much of the printed material as possible when moving to the next waypoint.

One naive approach is to always set the orientation of the waypoints to the direction of the vector connecting two waypoints. One problem with this approach is that sudden angle changes in printing trajectory will cause a large turning rate and a jittery printing motion. We take inspiration from the physical caster wheel and added smoothing to the orientation calculation by taking the average of orientations over a sliding window. This approach ensures good scan coverage for additive trajectories while making sure the end-effector do not turn too quickly. Fig. 6.2 shows our virtual caster orientation planning strategy at work, generating an efficient coverage of the printed material as shown in Fig. 6.3.



(a) UE4 simulation



(b) Reconstructed material profile

Figure 6.3: The resulting in-situ scan coverage during a printing trajectory without revisitation.

6.3 Alternating Shutter Colorized Point Cloud

A good implementation of the alternating shutter approach should produce distinct bright and dim frames, never causing the laser stripe to be visible in the bright frame. We test our hardware synchronized implementation by running a hand-held scan on an industrial part and observing the color of the registered point cloud. From Fig. 6.4 we can see there is no bleeding of laser pixels into the bright image frame used for colorization.

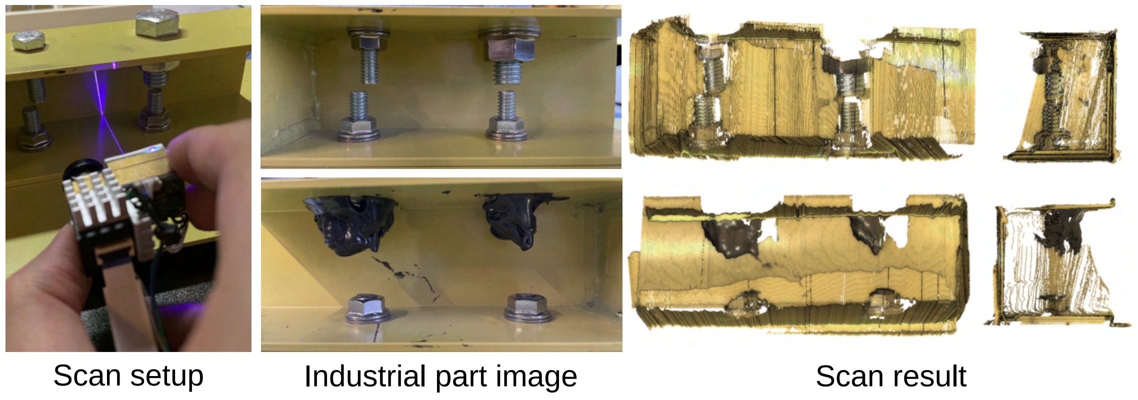


Figure 6.4: The hand-held scan results of an industrial part. There is no bleeding of laser stripes into the colorized point cloud.

Chapter 7

Conclusions

7.1 Contributions

In this work, we made the following three novel contributions:

- A design framework for a compact laser scanner that could be used for *in-situ* inspection in robotic additive manufacturing scenarios.
- A photorealistic simulation environment developed in Unreal Engine 4, designed for developing and testing additive manufacturing control algorithms, with sensor modeling, additive material fluid simulation and robot arm integration.
- An alternating shutter approach to use one camera as two, enabling depth and visual feedback, as well as scan point cloud colorization and vSLAM possible all within a compact sensor package.

We conducted a gap measurement test and determined our prototype sensor can measure gap widths down to 0.4mm within the sensing range (1-3.5 inches), and is able to detect gaps as small as 0.2mm. To test our robotic additive manufacturing simulation environment, we setup an example 3D print trajectory, developed a simple sensor placement strategy for maximizing scan coverage during the print, and demonstrated the *in-situ* reconstruction result of the printed material. We showed the effectiveness of our hardware synchronized alternating shutter implementation by benchmarking its framerate as well as showing a point cloud colorization result with no laser color bleeding.

7.2 Future Work

We hope that our sensor design and the integrated robotic manufacturing simulation environment can accelerate the development of depth and visual based closed-loop control algorithms and hence increase the level of autonomy in large scale robotics additive manufacturing projects. Additionally, the photorealistic simulation environment with material fluid simulation capability could be used for generating simulated sensor data and ground truth labels for defect detection learning algorithms, helping make future robotic manufacturing more intelligent.

Bibliography

[1] India Block. World's largest 3d-printed building completes in dubai, 2019. (document), 1.1

[2] Ai spacefactory wins nasa's 3d printed habitat challenge, 2021. (document), 1.1

[3] Torstein A. Myhre. Robot camera calibration, 2017. (document), 2.6

[4] Davide Sher. Ai build implements ai for error detection in 3d printing in new aimaker systems, 2019. (document), 5.1

[5] Izabela Hager, Anna Golonka, and Roman Putanowicz. 3d printing of buildings and building components as the future of sustainable construction? *Procedia Engineering*, 151:292–299, 2016. 1.1

[6] Jason Dunn. Reducing earth dependency for human spaceflight through robotic space manufacturing. In *65th International Astronautical Congress*, Toronto, Canada, 2014. IAC-14.15.3-B3.6.1.x27023. 1.1

[7] Nasa's centennial challenges: 3d-printed habitat challenge, 2019. 1.1

[8] Zhihua Lv and Zhiyi Zhang. Build 3d laser scanner based on binocular stereo vision. In *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 600–603. IEEE, 2011. 2.1.2

[9] Kenn Tornslev. *3D scanning using multibeam laser*. Informatik og Matematisk Modellering, Danmarks Tekniske Universitet, 2005. 2.1.2, 2.1.3

[10] Luigi Di Stefano and Andrea Bulgarelli. A simple and efficient connected components labeling algorithm. In *Proceedings 10th international conference on image analysis and processing*, pages 322–327. IEEE, 1999. 2.1.3

[11] Ken Young and Craig G Pickin. Accuracy assessment of the modern industrial robot. *Industrial Robot: An International Journal*, 2000. 2.1.4

[12] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaios. Fiducial markers for pose estimation. *Journal of Intelligent & Robotic Systems*, 101(4):1–26, 2021. 2.1.4

[13] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE*

and ACM International Workshop on Augmented Reality (IWAR'99), pages 85–94. IEEE, 1999. 2.1.4

[14] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005. 2.1.4

[15] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3400–3407. IEEE, 2011. 2.1.4

[16] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004. 2.1.4

[17] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015. 2.1.4

[18] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):1–11, 2017. 2.1.4

[19] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007. 2.1.4

[20] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014. 2.1.4

[21] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013. 2.1.4

[22] Agostino Martinelli. *Closed-form solutions for attitude, speed, absolute scale and bias determination by fusing vision and inertial measurements.* PhD thesis, INRIA, 2011. 2.1.4

[23] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *Journal of intelligent & robotic systems*, 61(1):287–299, 2011. 2.1.4

[24] Janne Mustaniemi, Juho Kannala, Simo Särkkä, Jiri Matas, and Janne Heikkilä. Inertial-based scale estimation for structure from motion on mobile devices. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4394–4401. IEEE, 2017. 2.1.4

[25] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-

manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016. 2.1.4

[26] Daqian Cheng, Haowen Shi, Albert Xu, Michael Schwerin, Michelle Crivella, Lu Li, and Howie Choset. Visual-laser-inertial slam using a compact 3d scanner for confined space. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2021. 2.1.4, 5.1.1

[27] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018. 2.1.4

[28] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE, 2006. 2.3.1

[29] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2.3.2

[30] Yiu Cheung Shiu and Shaheen Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb. *Purdue University Department of Electrical and Computer Engineering Technical Reports*, 1987. 2.3.3

[31] Jianfei Mao, Xianping Huang, and Li Jiang. A flexible solution to ax= xb for robot hand-eye calibration. In *Proceedings of the 10th WSEAS international conference on Robotics, control and manufacturing technology. World Scientific and Engineering Academy and Society (WSEAS)*, pages 118–122, 2010. 2.3.3

[32] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286. IEEE, 2013. 2.3.3

[33] RB Fisher and DK Naidu. A comparison of algorithms for subpixel peak detection. In *Image technology*, pages 385–404. Springer, 1996. 3

[34] Josep Forest, Joaquim Salvi, Enric Cabruja, and Carles Pous. Laser stripe peak detector for 3d scanners. a fir filter approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 646–649. IEEE, 2004. 3

[35] Li Qi, Yixin Zhang, Xuping Zhang, Shun Wang, and Fei Xie. Statistical behavior analysis and precision optimization for the laser stripe center detector based on steger's algorithm. *Optics express*, 21(11):13442–13449, 2013. 3, 4.2

[36] Karsten Haug and Guenter Pritschow. Robust laser-stripe sensor for automated

weld-seam-tracking in the shipbuilding industry. In *IECON'98. Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 98CH36200)*, volume 2, pages 1236–1241. IEEE, 1998. 3.1.1

[37] Qiucheng Sun, Jian Chen, and Chunjing Li. A robust method to extract a laser stripe centre based on grey level moment. *Optics and Lasers in Engineering*, 67:122–127, 2015. 3.1.1

[38] Yuehua Li, Jingbo Zhou, Fengshan Huang, and Lijian Liu. Sub-pixel extraction of laser stripe center using an improved gray-gravity method. *Sensors*, 17(4):814, 2017. 3.1.3

[39] Carsten Steger. An unbiased detector of curvilinear structures. *IEEE Transactions on pattern analysis and machine intelligence*, 20(2):113–125, 1998. 3.1.3, 3.1.3

[40] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004. 4.2

[41] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013. 4.2

[42] Epic Games. Unreal engine. 4.2

[43] Stephen James, Marc Freese, and Andrew J Davison. Pyrep: Bringing v-rep to deep robot learning. *arXiv preprint arXiv:1906.11176*, 2019. ??

[44] Patrick Mania and Michael Beetz. A framework for self-training perceptual agents in simulated photorealistic environments. In *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, 2019. ??

[45] Thomas Timm Andersen. Optimizing the universal robots ros driver. Technical report, Technical University of Denmark, Department of Electrical Engineering, 2015. 4.3

[46] Moldflow Plastics Labs. Moldflow material testing report. Technical report, NatureWorks PLA, 2007. 4.4

[47] Hongyao Shen, Weijun Sun, and Jianzhong Fu. Multi-view online vision detection based on robot fused deposit modeling 3d printing technology. *Rapid Prototyping Journal*, 2019. 5.1

[48] Mohammad Farhan Khan, Aftaab Alam, Mohammad Ateeb Siddiqui, Mohammad Saad Alam, Yasser Rafat, Nehal Salik, and Ibrahim Al-Saidan. Real-time defect detection in 3d printing using machine learning. *Materials Today: Proceedings*, 42:521–528, 2021. 5.1

[49] Guo Dong Goh, Swee Leong Sing, and Wai Yee Yeong. A review on machine learning in 3d printing: applications, potential, and challenges. *Artificial Intelligence Review*, 54(1):63–94, 2021. 5.1

[50] Daqian Cheng, Haowen Shi, Michael Schwerin, Michelle Crivella, Lu Li, and Howie Choset. A compact and infrastructure-free confined space sensor for 3d scanning and slam. In *2020 IEEE Sensors*, pages 1–4. IEEE, 2020. 5.1