UNIVERSITY OF CALIFORNIA SAN DIEGO

The Design and Implementation of a Stereo Camera and Image Processing Pipeline to Resolve the Real World Position of Lighting in the Built Environment

A Thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in

Computer Science

by

Anthony Xavier Quiroga

Committee in charge:

Assistant Professor Pat Pannuto, Chair
Professor Manmohan Chandraker
Professor Henrik Christensen

2024

The Thesis of Anthony Xavier Quiroga is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

VITA

2022 Bachelor of Science, University of California San Diego

2024 Masters of Science, University of California San Diego

ABSTRACT OF THE THESIS

The Design and Implementation of a Stereo Camera and Image Processing Pipeline to Resolve the Real World Position of Lighting in the Built Environment

by

Anthony Xavier Quiroga

Master of Science in Computer Science

University of California San Diego, 2024

Assistant Professor Pat Pannuto, Chair

This thesis investigates the feasibility of automated localization of light sources in the built environment. The aim is to enable at-scale geolocation of light sources with low-cost systems. The primary motivation is in support of a larger objective to later extract key parameters of electric grid performance from the discovered lights for wide-area, fine-grained grid health monitoring. Our findings suggest that it is possible to locate light sources in images and estimate the location reliably in the relative scenario, i.e., when the camera position is known a priori. However, we find that out-of-the-box supporting sensors—GPS for position and IMU for orientation—provide data with insufficient precision and accuracy for the final mapping task. We

finish, then, with a discussion of how further innovations in image processing can help overcome these limitations to accurately resolve the real world position of lighting in the built environment.

Chapter 1. Introduction

Deploying high-fidelity sensors across electric grids is high-cost due to the intensive labor required to install and maintain the sensors for every node connected to the electric grid. An infrastructure-less device for electric grid monitoring would be a step in the right direction for a cost-effective solution, and it is the purpose of larger project, LightsCameraGrid, that inspires and drives the work in this thesis.

Preliminary work shows that, in controlled conditions, cameras can accurately measure grid voltage and frequency just by taking images of light sources [7]. This information measured from light sources can directly tell us the state of the electric grid for a given region. The goal for a low-cost system for reliable electric grid quality detection is to utilize cameras from ubiquitous, commodity devices. To accurately estimate the phase and frequency of light sources outside of controlled, laboratory settings requires knowledge of the distance from the camera and light source. We need to estimate light sources that would be typically 15-35 meters(m) from the camera, but existing work has used computer vision for estimating distances at minimum 0.2m and maximum of 9.7m from the camera with 6% and 11% errors to ground truth respectively in the Global Positioning System (GPS) coordinate system [2]. However, grid measurement demands sub meter accuracy.

The goal of this project, then, is to move one step closer to autonomous, wide-area grid health measurement. We explore the viability of computer vision to solve the geolocation problem of lighting in the built environment. We will ultimately find that commodity cameras are sufficient to find and localize light sources relative to themselves, but commodity position and orientation sensors are insufficient to resolve the position of the camera itself.

1.1 Why Grid Health?

Developing countries undergoing electrification face many unique challenges in thier push for universal electrification due to low consumption and high-cost grid connections [10]. Low consumption occurs in these newly connected areas due to poor reliability of electricity For example, some cases in Kenya, power outages can occur for months, which results with low confidence in energy-required appliances [10]. High-income countries have instrumented electric grids that are equipped with numerous, high-fidelity sensors, for quality detection. Lower-resourced grids, however, often monitor only high, and some medium, voltage distribution systems. As a result, they have limited insight into performance issues such as outages or voltage sags [4]. In one grid study in Kenya, about 20% of the 150 low-voltage transformers studied had long-term outages, and the average outage was a four month duration [10]. To respond promptly, or to better predict outages, operators require the ability to understand grid health. Grids in developing countries do not have infrastructure to provide this information due to budget constraints [4]. What is needed is a cost-effective alternative approach to electric grid quality measurement to improve the reliability of electricity across developing countries, as without access to reliable electricity the rate of development will continue to be suppressed [10].

Both developing and high-income countries use Smart Meters to augment their electric grid, but Smart Meter deployment is costly and time consuming. Indeed, the US has only 73% penetration of Smart Meter deployment across its grid, expecting 85% by 2025 [3]. However, 11 states have only 15-50% penetration, and high population density states, such as New Jersey and Hawaii, have less than 15% [3]. In Accra, Ghana, the electric grid is only equipped with monitoring at high and medium voltage distribution points. As a result, these utilities significantly under detect low-voltage power outages [4]. To understand, predict, detect, and address these low-voltage power outages, a more cost-effective system is necessary to properly treat this reliability issue in grids with limited monitoring infrastructure.

1.2 What part of the project does this thesis solve?

The goal of this work is to solve the geolocation problem. To solve this problem, we need to build a system that that enables computer vision techniques, such as imaging geometry, to properly estimate where lights exist from 2D image space to 3D GPS space. This involves building a specific camera system called a stereo camera. A stereo camera system is a type of camera that has two or more camera sensors that enables human binocular vision, allowing imaging geometry to estimate the depth of objects and their location in 3D space. When we build this system, we take note of the range of distance we want to accurately estimate, which is between 15m-30m from the camera center, and build the stereo system to enable sub meter accuracy for this from our camera sensors. To enable an accurate estimation from the stereo camera, we utilize a camera calibration procedure for estimating objects at this distance as well.

We will also need sensors to estimate the absolute orientation of the camera platform to properly estimate the 3D location of the light sources in the images to the GPS coordinate system, which requires the use of orientation estimating sensors of a platform. Finally, we apply image processing and computer vision techniques to automate the process of detecting light sources in images, projecting them to the GPS coordinate system, and feature tracking for more robust absolute orientation estimation. Since sensors can be noisy, we also apply camera orientation optimization techniques to improve the sensor readings for a more robust orientation estimation.

1.3 Thesis Statement

Building a cost-effective system for detecting the voltage reception of light sources is important for understanding the health of the electric grid, but knowing the location of these light sources will enable action when necessary when voltage sags in areas where the grid monitoring system is deployed. *The goal of this project is to answer whether commodity, consumer-grade cameras and sensing systems can resolve the real-world position of lights in the built environment with sufficient fidelity for automated mapping of infrastructure at-scale.*

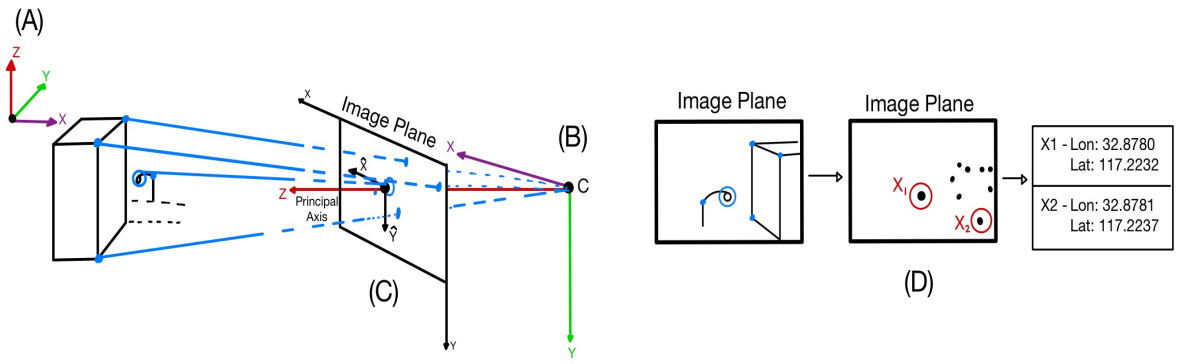


Figure 1.1. (A) World coordinate frame, (B) Camera coordinate frame, (C) image plane pixel coordinates, (D) ideal pipeline of light source detection to GPS estimation of said light sources

1.4 Pipeline of solution

The goal of this thesis is to solve the geolocation problem in the larger grid health monitoring project. To solve this problem, there needs to be an understanding of imaging geometry and how we utilize stereo cameras to estimate the location of objects in image pairs to 3D space.

In this solution, we need to measure the camera sensors parameters through camera calibration, which generates the camera's intrinsic and extrinsic parameters. The camera intrinsic parameters are the physical properties of the camera and the extrinsic parameters allow the mapping of the physical space of the world to the camera coordinate frame. These parameters then allow us to map 3D scene points onto the imaging plane of the camera sensor through imaging geometry. The important aspect of this are the extrinsic parameters of the camera, which generates the mapping of the 3D scene points in the world coordinate frame, which we see labeled as (A) in 1.1, onto the camera coordinate frame (B). This mapping is important as we can utilize these parameters to map to the GPS coordinate system to generate 3D scene points in the GPS system to localize the light sources in images, as we see in the final parts of our figure

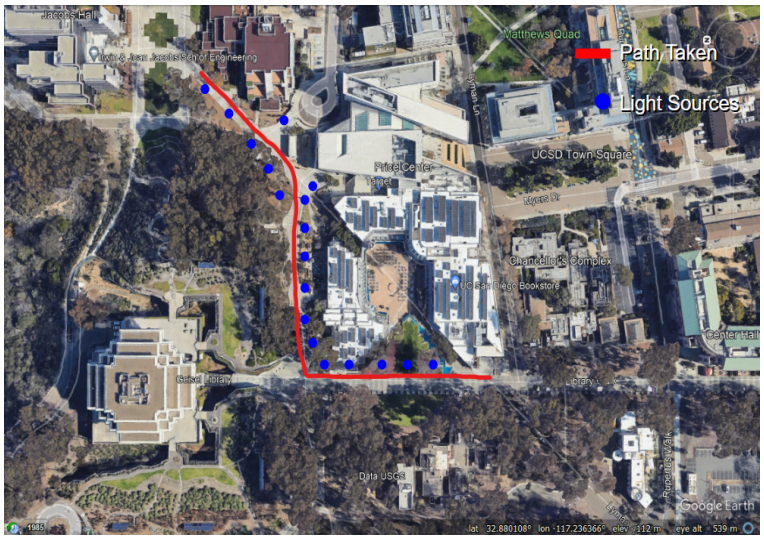


Figure 1.2. Example of an ideal path traversed with camera and estimated light sources we would want to see when running our algorithm.

(D). The goal of this project is to enable the pipeline of capturing an image of a light source, locate the light source in the image, estimate the 3D scene point of the light source, and map that estimation into the GPS coordinate frame.

1.5 Process of evaluation

For evaluation, we construct a dataset following a path that contains many light sources across the University of California, San Diego (UCSD) campus. The goal here is to evaluate how well our device can estimate the 3D location of the light sources across the images we take in terms of the relative projection space and their ground truth position on the map 1.2.

As we walk along the designated path, we note the GPS location of each of the light sources we designate to track in the dataset. Then we create a mapping of the light sources, similar to 1.2, in which we will plot their true GPS location, and create a relative 3D map of the GPS coordinate designating the original starting GPS location of the camera's as the origin.

Finally, we construct the relative mapping from our algorithm as well as the GPS mapping and compare the estimated points to the ground truth point by both euclidean distance of the individual points, and by the relative distance from each projected point to the camera starting

center point.

1.6 Solution pipeline

In this project, our solution for a cost effective system is a stereo camera with a camera orientation sensor and GPS tracker for absolute orientation of the camera in the GPS coordinate system. This combination of sensors allow us to estimate the light sources we see in our 2D images to 3D space in GPS coordinates, but this requires understanding our camera sensor parameters with our orientation sensor and GPS tracking device. To understand the parameters of our camera sensor, I speak in section 2 about how we conduct the process of camera calibration, which estimates the intrinsic and extrinsic parameters of the camera that enable the 2D to 3D point estimation we need. Once we have our calibrated system, in section 3.1 to 3.3, I speak in detail how we automate the detection of light sources in images through color space filtering and use point correspondence detection between the left and right image for matching points to conduct the 2D to 3D point mapping in our stereo system. In section 3.4, I speak thoroughly how we use the combination of our GPS tracking device and camera orientation sensor to estimate the direction our camera's facing in the GPS coordinate system through computer vision optimization techniques. We evaluate how our system performs with this pipeline through a short paths on the UCSD campus on GPS light source detection and comparing to the ground truth.

Chapter 2. Building an Accurate, Economical, Stereo Camera

Our solution to the low-cost grid monitoring system is to utilize the stereo camera device as a means to monitor the electric grid. There are several methods that could have been used to solve the geolocation problem this project faced. For example, we could have chosen a monocular camera approach with satellite information to conduct triangulation for 3D point estimation in this manner through few assumption choices and optimization algorithms, but it defeats the purpose of our goal for this project. We want a low-cost system for electric grid monitoring, where our ideal end-to-end system would be smartphones since they are ubiquitous, commodity, devices that contain all the sensors we need. Therefore, as a prototype device to evaluate the effectiveness of our pipeline, we opted for a stereo camera system, two cameras fixed in a horizontally aligned setup, with imaging geometry as our chosen solution as it's low-cost and a proven system for 2D to 3D point mapping that we can directly implement ourselves. However, with this chosen system in mind, we need to account for what we need to properly build the system for our specific problem.

We aim to estimate light sources between 15-25m from the camera, so we need to accurately measure depth at that range. Then, we need to properly build the stereo camera system to handle estimating objects at that depth accurately to sub-meter accuracy. Once we have our system built to solve our problem, we need to estimate the parameters of our cameras to conduct imaging geometry with reasonable accuracy, which is through camera calibration, where we can effectively estimate our cameras intrinsic and extrinsic parameters for 2D to 3D point estimation.

2.1 Camera Parameters

2.1.1 Intrinsic and Extrinsic Parameters

A camera has two sets of parameters, defined as the camera intrinsic and extrinsic parameters. Intrinsic parameters are the physical properties of the camera, specifically focal length f, principal point offset, and axis skew. The focal length is the distance between the pinhole of the camera and the image plane, or camera sensor. The mapping of this distance onto the image plane is denoted in the x and y direction, or f_x and f_y. When an image is taken, the light captured in the scene is then reflected onto the physical sensors, and discretized into pixel coordinates of the image, which is then centralized based on the origin of the image coordinate system. This converts any light captured from the sensors into the pixel coordinate system, and the location of the pixels is based on the principal point of the camera, which is the line perpendicular to the image plane that passes through the pinhole, denoted as x_0 and y_0. Lastly, axis skew is the cause of sheer distortion in the image, denoted as s. Finally, the mathematical representation of the intrinsic parameters is defined as K:

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

We can describe this as the mapping from the camera coordinates from 3D space to the 2D image plane.

Extrinsic parameters are what allow the mapping of the physical space of the world to the camera coordinate frame. More precisely, this is the manipulation of coordinate frames between the camera and the world. The camera itself maps the physical scene onto the image plane by defining a coordinate frame, called the camera coordinate frame, as we see in figure 2.1.

The camera will see the point X in the scene, project it onto the image plane, which is then mapped by the intrinsic parameters of the physical camera into the 2D image plane as in

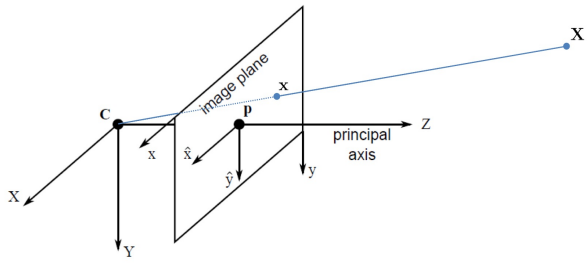


Figure 2.1. Example of Camera Projection, where point X maps onto image plane at point x through camera intrinsic parameters. Image from [1]

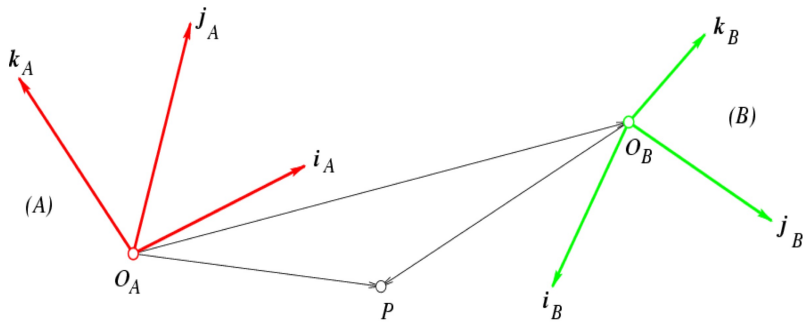


Figure 2.2. Example of coordinate frame (A) and coordinate frame (B) having different orientation and origin points, where a possible rotation and translation can map between frames (A) and (B) to correspond to the same point P

figure 2.1. In the real world, the scene is not mapped perfectly for the camera to build the scene as the camera coordinate frame in 2.1. Instead the world exists in the relative world coordinate frame, which will be different than the camera coordinate frame due to the different origin point of each frame. The camera extrinsic parameters are then the mapping of the world frame to the camera coordinate frame. It is primarily a 3D to 3D transformation, where it conducts a rotation of the world frame to align it with the orientation of the camera frame, and then translates the origin of the world coordinate frame to the camera coordinate frame so the scene points are properly mapped to the image plane, as seen in figure 2.2.

The mathematical representation of the extrinsic parameters is defined as R for rotation

of the coordinate frames and T for the translation between the origin of the coordinate frames. From equation 2.2, the rotation matrix R represents the rotation needed to align the world-axes in the camera coordinate frame. The equation 2.3 is the translation of the origin points from the world coordinate frame to the camera coordinate frame in 3D space.

$$R = \begin{bmatrix} R_1 & R_2 & R_3 \\ R_4 & R_5 & R_6 \\ R_7 & R_8 & R_9 \end{bmatrix} \tag{2.2}$$

$$t = \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T \tag{2.3}$$

The full mathematical representation of the extrinsic matrix being defined as such:

$$M_{4\times4} = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{bmatrix} \tag{2.4}$$

The camera matrix, what we use for imaging geometry algorithms, is then:

$$P = K \cdot M = K \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{bmatrix} \tag{2.5}$$

2.1.2 What is Needed for Triangulation for Real World Positioning

We can utilize the properties of the camera to construct a two camera setup, defined as stereo cameras, which can allow us to conduct triangulation with the image pairs we capture. The imaging geometry algorithm, triangulation, enables the estimation of 3D scene points in an image by choosing two corresponding points between the image pair we capture from the stereo camera, as seen from figure 2.3.

Then, the first step of the algorithm is to detect corresponding points between the image

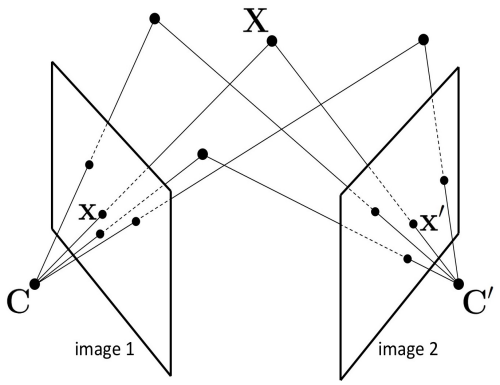


Figure 2.3. Example of two-view geometry using triangulation for 3D point estimation. Image from [1]

pairs we take from the stereo camera. Once we find the corresponding points, particularly light sources, we can conduct triangulation. However, when we do triangulation with just the extrinsic and intrinsic camera parameters, we estimate the 3D scene points from the image onto a relative world coordinate frame. We want to estimate the light sources in the GPS coordinate frame, and to do this, we need to have a mapping between the GPS coordinate frame, world coordinate frame, then to the camera coordinate frame. We can do this by using Euler angles of the platform with respect to the magnetic north pole, which enables us to get the absolute orientation of the camera in the GPS coordinate frame. We then create a rotation matrix, similar to 2.2, by using Euler angles, which are a set of 3 elemental rotations that is following the Tait-Bryan angles of rotation. We create rotation matrices based on our device's convention, which is one of X-Y-Z, Z-X-Y, Y-X-Z, Y-Z-X, Z-Y-X, or X-Z-Y.

Euler angles are the specific rotations about three principle axes, specifically the X-axis, Y-axis, and Z-axis. We utilize the angles of rotations about these axes to create the rotation matrix for the mapping between the GPS coordinate frame to the camera's world coordinate frame. We get this mapping from the equation 2.6:

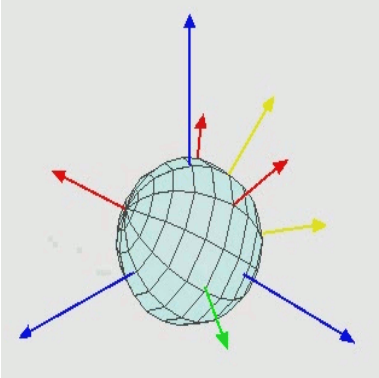


Figure 2.4. Example of a possible rotation of the origin frame (Blue) across 3 elemental euler rotations to the new frame (Red). Image from Wikipedia on Euler Angles.

$$R_X(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix}, R_Y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}, R_Z(\phi) = \begin{bmatrix} cos\phi & -sin\phi & 0 \\ sin\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

Where the resulting rotation mapping from the Euler angles to the coordinate frame of interest is represented in equation 2.7:

$$R = R_X(\psi)R_Y(\theta)R_Z(\phi) \quad (2.7)$$

We get rough rotation estimation from an Inertial Measurement Unit (IMU) sensor, but due to the sensitivity of the sensor, we must add vision-based optimization techniques to correct as much of the error as we can. We utilize camera motion from a video of a scene, select keyframes from the frames we capture, and implement a computer vision optimization algorithm called bundle adjustment. This optimizes the 3D estimation error we capture across all of the frames of interested scene points, corrects the re-projection error of those points to minimize it, and in turn gives us more accurate estimation of light sources in the GPS coordinate frame.

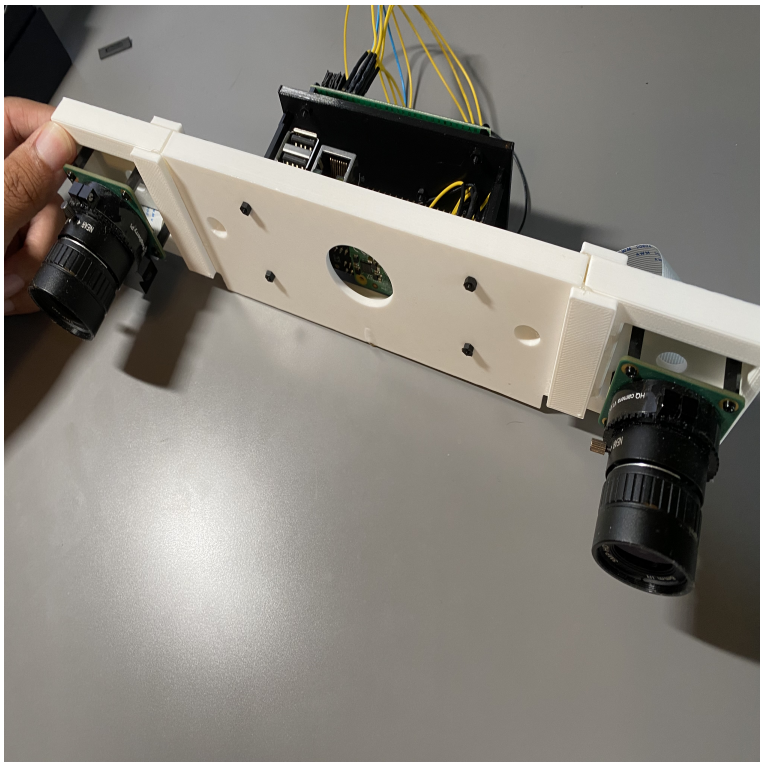


Figure 2.5. Stereo-pi stereo camera system built for project. GPS and IMU sensor (discussed in 2.3) attached to the back of the camera

2.2 Camera System Calibration

We need our end-to-end system to be able to acquire an image of light sources within 15m-25m distance from the camera center, detect those light sources in an image, and estimate the 3D GPS location of those light sources. The step that requires the most attention is the first and third step for building our system. We must have our camera system to be able to estimate the 3D position of light sources 15-25m from the camera. This requires building a stereo camera rig with sufficient fidelity to estimate objects at a depth of 10-20. To do so requires an accurate measurement of the intrinsic and extrinsic parameters of the camera for optimal imaging geometry, and accurate absolute orientation readings for the camera platform with respect to the GPS coordinate system.

2.2.1 Stereo System Construction

Our envisioned final application deployment will place our camera system 15m-25m from light sources we wish to analyze, and will require estimates of depth and location with sub meter accuracy. To do this, we consider multi-view imaging geometry with a stereo camera system containing very specific dimensions to minimize the error of our specific problem at hand.

An off the shelf stereo-pi camera system, features cameras with 7.53mm focal length and camera sensors allowing up to 5MP resolution 2.5. With this, we can utilize image geometry properties to build a better understanding of what errors we will expect to have with these specific stereo camera dimensions, and build our system to have the expected sub-meter error for depth calculation.

With imaging geometry, depth estimation improves based on the pixel disparity of corresponding objects in image pairs, and we can increase or decrease disparity based on the physical configuration of our stereo camera system. With stereo cameras, two cameras are horizontally aligned with one another, and the distance between these cameras, defined as the stereo baseline and measured from camera center, is tuned to increase or decrease pixel disparity in an image at certain depths.

Since we want to estimate light sources that will typically be 15m-20m from the camera, we want our camera baseline to have the error of the estimated depths at those distances to be minimized. Based on the works of [8], as the stereo camera's baseline increases, the distance at which the system is able to project the object's depth reduces significantly, as seen from figure 2.6.

This work coincides with the analytical approach to estimate the stereo camera baseline that minimizes the resulting error of depth estimation at specific distances. We implement a baseline of 22cm (b), with our camera's focal length of 7.54mm (f), and aim for a pixel disparity error of 0.5 (D_d) pixels through calibration. From equation 2.8, this configuration results in

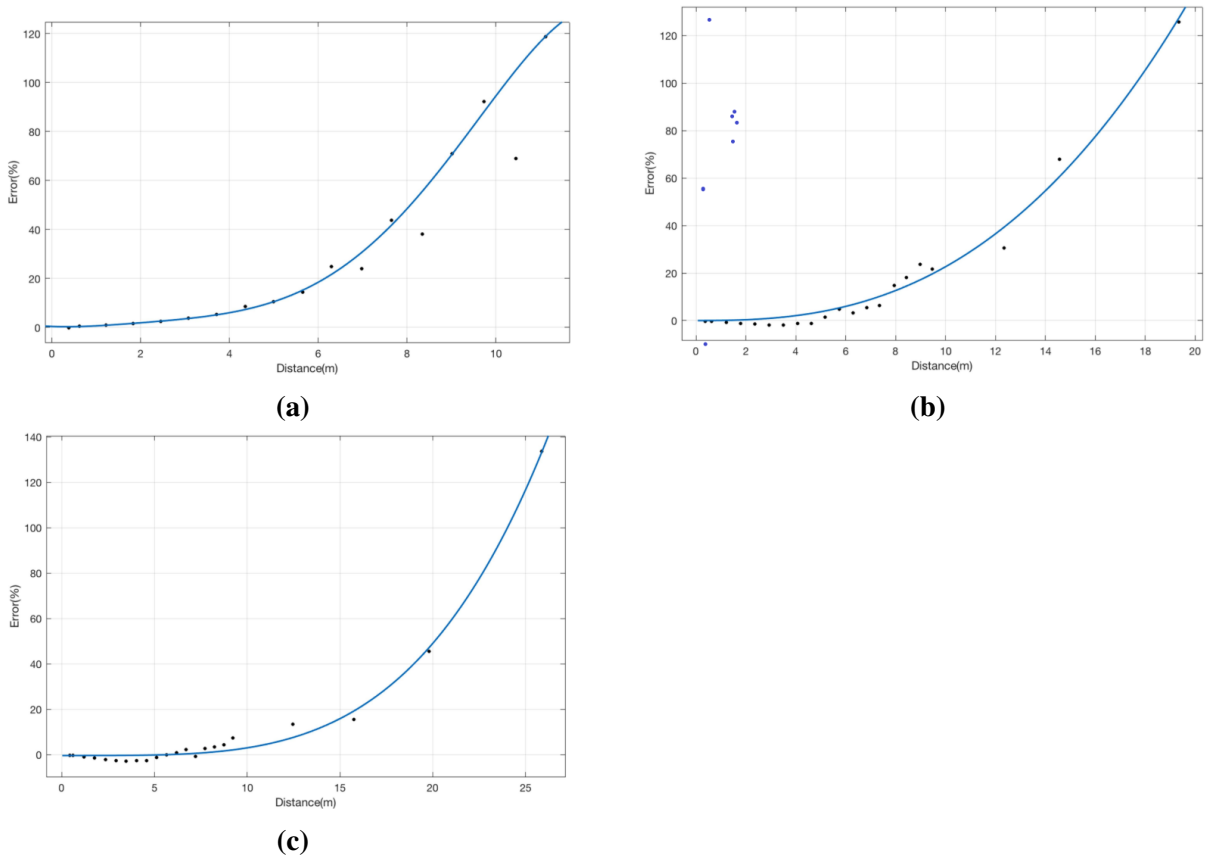


Figure 2.6. Error against distance of object being estimated with stereo baselines (a) 19.81, (b) 29.92, and (c) 39.84cm. Figures from [8]

the figure of our theoretical errors. With this system, we can expect sub-meter accuracy within 15m-20m, but objects beyond this depth the error increases exponentially to above sub-meter accuracy (Fig. 2.7).

$$D_d = \frac{z^2}{b * f} \cdot D_p \tag{2.8}$$

Given our application, we would want a stereo baseline configured between 35cm-40cm to ensure the error of points estimated at a depth of 10m-20m only results in errors below 20%, or as close to sub-meter error as possible. Due to the constraints of available ground truth and current system setup, we chose a baseline of 23 cm since light sources will typically be 10m from the stereo camera with this specific dataset.

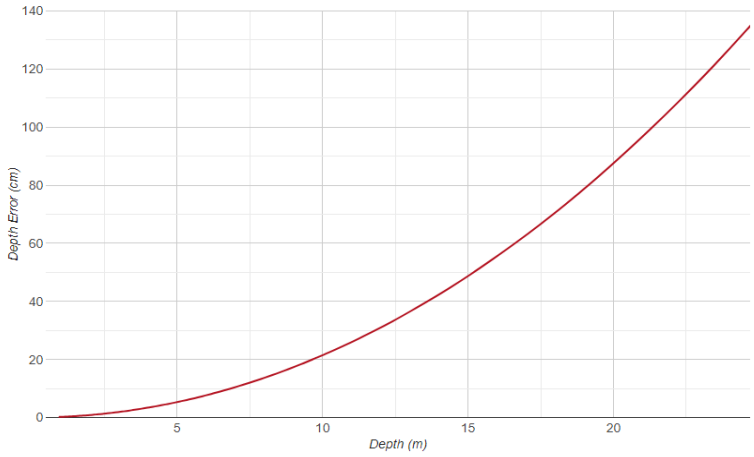


Figure 2.7. Estimated depth error of a stereo system against objects at certain distances from camera, with baseline of 22cm, focal length of 7.54mm, and pixel disparity of 0.5. Graph taken from Nerian - Lens Focal Length and Stereo Baseline Calculator

2.2.2 Camera calibration process

After designing the stereo camera system, we need to conduct camera calibration to allow us to extract the intrinsic and extrinsic parameters from our camera. Knowing these parameters will allow us to conduct imaging geometry techniques. The intrinsic parameters of a camera tells us the mapping of objects in the scene to the pixels of an image. The extrinsic parameters of a camera describes the orientation of the camera platform and the location of the camera in a relative coordinate system. These parameters are used to conduct imaging geometry, which is what we utilize to estimate the 3D location of objects in the 2D image using triangulation.

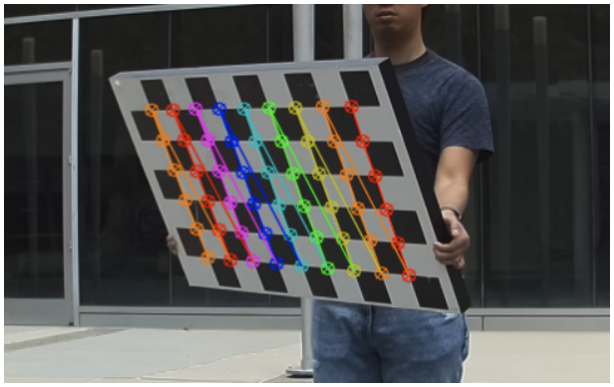


Figure 2.8. Example of calibration process, specifically checkerboard detection algorithm using OpenCV.

We conduct a simple calibration procedure for our cameras, as seen in figure 2.8. We estimate the intrinsic and extrinsic parameters with the planar pattern recognition procedure, which uses nonlinear refinement method to estimate these parameters [12]. We first use a planar object that is precisely measured and easily detectable from the camera's perspective, which is typically a checkerboard pattern on a flat planar surface. The procedure detects the corners of the checkerboard squares and assigns relative world coordinate parameters. We then use the nonlinear method to robustly estimate the 3D location of these chosen pixel coordinates into the relative world coordinate frame, and fine tune these estimates across the images to reduce the root mean squared error (RMSE) of the re-projection error (sub-pixel) from the 3D scene point mapping to the 2D image points in pixel coordinates.

$$\mathbf{B} = \mathbf{A}^T\mathbf{A}^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{2.9}$$

Intrinsic and extrinsic parameters of a camera are estimated is by solving a closed form solution, that was proven in [12], from the equation 2.9. This resolves in the vector $\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]$. Then, with this vector, we have the equation 2.12, where h_i are the ith column of the homography matrix $\mathbf{H}$. The matrix $\mathbf{H}$ is derived from our model plane, or checkerboard, where we have $\mathbf{H} = [h_1, h_2, h_3]$, and is equivalent to 2.10, to some scalar λ.

$$\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_1 & \mathbf{h}_3 \end{bmatrix} = \lambda\mathbf{A}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \tag{2.10}$$

With r_1, r_2, r_3 being vectors of the rotation matrix R, and that r_1 and r_2 are orthonormal, we have equation 2.11, where , which is how we get equation 2.12.

$$\mathbf{h}_1\mathbf{A}^T\mathbf{A}^{-1}\mathbf{h}_2 = 0 \tag{2.11}$$

Table 2.1. Root Mean Squared Error (RMSE) of the re-projection error through the calibration process. These results imply our 3D to 2D mapping from scene point to image pixel coordinate point is sub-pixel accurate.

Metric	Left Camera	Right Camera	Stereo Calibration
RMSE	0.1285	0.1227	0.2265

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{B} \tag{2.12}$$

We then have constraints on the intrinsic matrix, where we can rewrite the linear equation as such:

$$\mathbf{V}\mathbf{b} = 0 \tag{2.13}$$

The vector $\mathbf{V}$ is a $2 \times n$ matrix, where if $n \geq 3$, the linear equation 2.13 has a unique solution defined up to a scale factor. We solve for the left null space of 2.13, which the solution is the Eigen vector associated with the smallest eigenvalue of $\mathbf{V}^T\mathbf{V}$, giving us the vector $\mathbf{b}$. We can then map these values to the camera intrinsic parameters, as in equation 2.9, which to optimize, we apply a Maximum-Likelihood Estimation and minimizing the function 2.14.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} ||m_{ij} - \hat{m}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j) \tag{2.14}$$

This equation is then minimized using a nonlinear optimization function, called the Levenburg-Marquardt algortihm [1]. In this equation, m_{ij} is the actual point in the image, $\hat{m}$ is the estimated 2D point using the estimated 3D point M_j with the camera projection parameter for the mapping of 3D to 2D image points.

After obtaining the optimized intrinsic and extrinsic estimation, we want an RMSE value below 0.5 which tells us the average pixel error for the correct 3D estimation. During our calibration process, we calibrated the cameras to this RMSE value when conducting experiments the following experiments, with are results in 2.1.

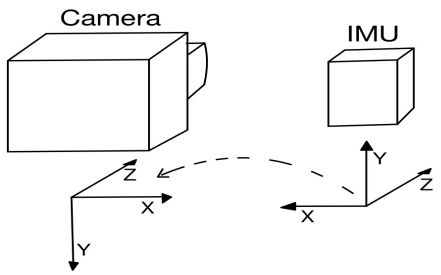


Figure 2.9. Orientation of the Camera coordinate frame and the Adafruit BNO055 coordinate frame.

2.3 Absolute Position and Orientation of the System in Real Space

Generating the mapping between the GPS coordinate frame to the World coordinate frame of the camera is very difficult due to the sensitivity of absolute orientation sensors. The specific sensor we are using the Adafruit BNO055, which is a 9 Degree of Freedom (9 DoF) inertial measurement unit (IMU) sensor, which contains an accelerometer, gyroscope, and magnetometer to apply sensor fusion and capture Euler angles of the platform with respect to the magnetic north. The readings are described in 2.9, where yaw is the angle difference between the platform's z axis and reference z-axis, pitch is the angle of the x-axis to the reference frame, and roll is the angle of the x-axis to the reference frame. Having this data, we can create a rotation to capture the orientation of the camera platform with respect to the magentic north, which is the mapping between the GPS coordinate frame to the camera's world coordinate frame. In order for the system to attain absolute position and orientation in real space, we would need accurate measurement of the platform's euler angles from our IMU device and accurate position of the camera's location from the GPS tracking device.

This would be a simple addition to the stereo camera system, but unfortunately, IMU sensors are extremely sensitive to any magnetic noise in the given area. Due to this, using an isolated IMU sensor leads to many inaccuracies in 3D point estimation. Instead, then, we separate the questions of position and orientation for the camera setup.

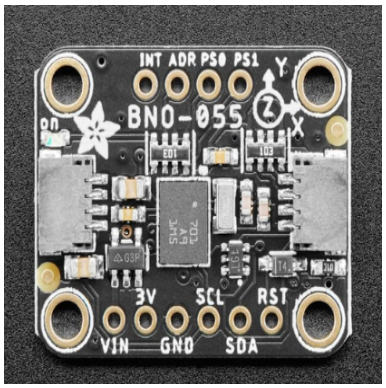


Figure 2.10. Image of Adafruit BNO055. Image taken from Adafruit website

2.3.1 Position

It is essential to know the location of the camera in GPS coordinates as it enables 3D point estimation to accurately detect light sources at reasonable locations of the world. For absolute positioning of the camera, we simply use a GPS tracking device, called the SparkFun Quiic GPS Breakout Lite. With this device, we receive an approximate location of the GPS coordinates of our camera, which allows for localization of lightsources to be in a reasonable location when estimating their 3D location relative to the camera.

Figure 2.11. Image of SparkFun Quiic GPS Breakout Lite. Taken from Sparkfun website

2.3.2 Orientation

Now that we have absolute position of the camera in GPS coordinates, our next step is to gather the absolute orientation of our camera with respect to the magnetic north. This is to

enable the understanding of where our camera is facing in the GPS coordinate frame. We tested two separate devices with a 9 DoF IMU sensor, where our results showcased that we needed more than just an IMU sensor for absolute orientation

Android Phone as Absolute Orientation Reader

The first device we investigated was an Android phone IMU sensor for absolute orientation readings, specifically Google Pixel 1. This involved understanding the default coordinate frame of the android device, how absolute orientation is calculated with the device, and the convention at which the Euler Angles are read to develop the proper rotation transition.

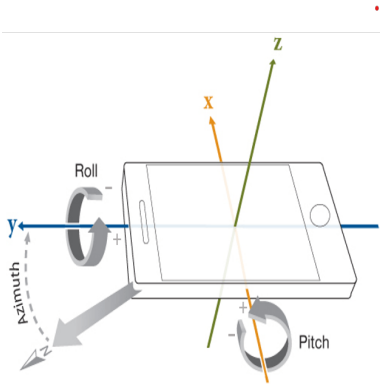


Figure 2.12. Image of android coordinate frame for absolute orientation readings. Coordinate frame is in Roll, Pitch, Azimuth. Figure is taken from Mathworks.

With the android phone having a frame defined in 2.12, it's defined, in documentation, that the absolute orientation is calculated by the direction of the Azimuth axis, and in this coordinate frame, Azimuth is only equal to Yaw when the Roll is set to 0, or in our case, when the phone is laid flat horizontally. Therefore, to use the Android phone absolute orientation estimation, we constructed the system as in 2.13, and calculated the rotation of coordinate frame between the two devices to develop a proper rotation mapping of the absolute orientation of the camera using Euler angles. For this experiment, we tested on the Intel Realsense D455 to remove any errors from calibration and isolate the testing on the IMU device only. After running several experiments, we experienced an inconsistent amount of errors due to the sensitivity of the phone's readings 2.2. After several trials of attempting to work with the android phone, we

acquired a designated IMU sensor to further conduct analysis on how accurate an IMU sensor can be for our experiment.

Table 2.2. Distance error between ground truth point and the estimated point from the stereo camera with IMU sensor at each position from 2.13

Camera Position	Distance Error	Angle Difference
1	0.396270	2.293977
2	2.534764	41.405446
3	5.048347	90.638874

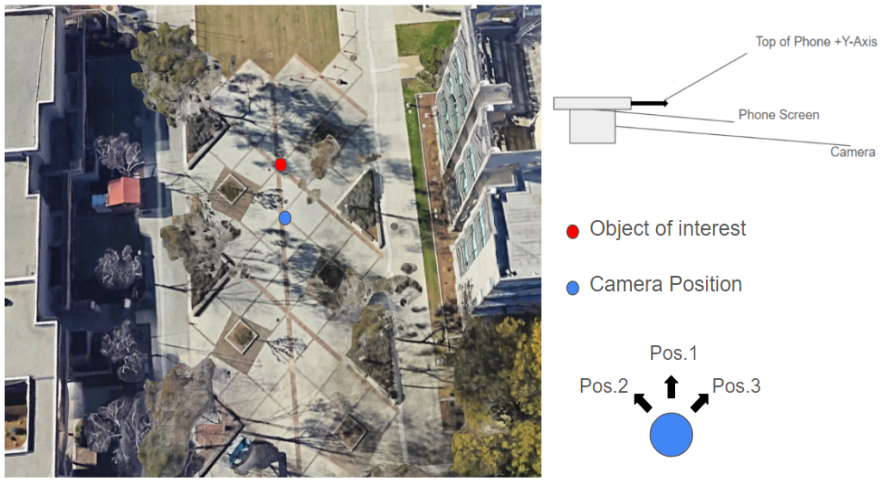
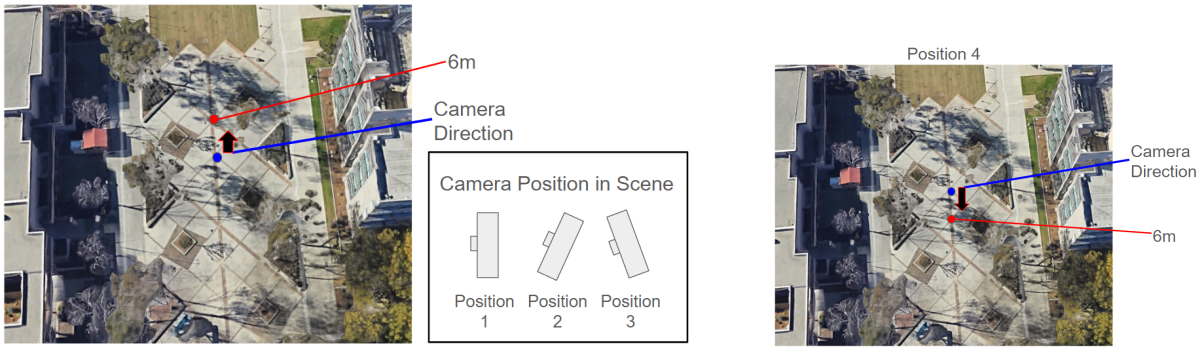


Figure 2.13. Top down overview of experiment, where camera is oriented in 3 different positions (Pos.), with object of interest in frame, and estimated the GPS location of the object in each image. Phone is position facing down horizontally on camera, with Z-axis down and Y-axis pointing forward.

Utilizing the Adafruit BNO055

The IMU sensor we used in our second set of experiments was the Adafruit BNO055, which was a 9 DoF sensor that recorded the absolute orientation of the device directly in Euler Angles. With the coordinate frame defined in 2.9, I created the rotation matrix to convert the coordinate frame of the sensor to align with the camera coordinate frame, and utilize the Euler angle readings to construct the absolute orientation estimation. An example of the results from one set of our experiments, shown in table 2.3, shows that the primary issue was the drag error



(a) **(b)**

Figure 2.14. Overview of one set of experiments testing the Adafruit BNO055 IMU device with the Intel Realsense D455 stereo camera to ignore the calibration step. (A) shows the first three orientations of the static camera and (B) shows the position of camera and with orientation of position 1.

that resulted from the device. The experiment set up is shown in figure 2.14 After calibrating the IMU sensor, there were errors in the readings, but it was consistent in the initial error estimates. However, when testing different orientations of the camera in a scene, there was an obvious drag error of the IMU sensor, where rotating the camera when the IMU sensor was active resulted in much more inaccurate readings. When the camera would project an object facing one direction, and we rotated the camera by 180 degrees in the opposite direction, the resulting projected point would only be projected with a less than 180 degree change in position, with an average of about 30 degrees less. This was a severe issue that was not solved by just using the IMU sensor as an isolated device.

Table 2.3. Results from experiment 2.14, with distance error, and angle, between the estimated GPS location of object and ground truth position.

Camera Position	Distance Error	Angle Difference
1	7.375311	64.667558
2	12.056049	140.187042
3	8.118174	71.554735
4	11.129661	151.343633

As a result of these experiments, we concluded that using the IMU sensor as the sole device for absolute orientation estimation was not the solution. We need a more robust optimiza-

tion algorithm involving other sensor information to properly estimate the camera's orientation in the GPS coordinate frame. The solution we arrived at was using camera motion and bundle adjustment for an improved orientation estimation.

Chapter 3. Extracting Light Locations From a Series of Stereo Images

Now that we have our image acquisition system in place, we turn to our image processing pipeline. First, to enable 2D to 3D point estimation, we conduct triangulation for scene point estimation, which requires finding point correspondences in our image pairs we capture with our stereo camera system. Typically, point correspondences would be the key points in images, which would often be detected corners that exist in the scene, but for our case the points of interest will be light sources that exist in the images. Therefore, in our pipeline, we have to implement a light detection algorithm to isolate the pixel location of light sources as our key points, and conduct a point correspondence algorithm to match the detected keypoints across the image pairs we capture in our stereo system. Finally, because of the IMU and GPS tracking sensors are noisy, we apply an optimization algorithm on our captured dataset to improve the estimation of our camera's orientation in the GPS coordinate system for improved accuracy of light source point estimation.

3.1 Triangulation (3D point estimation)

For 3D point estimation with a stereo camera system, we have to apply two-view imaging geometry techniques. Specifically, we use triangulation as our method to estimate 3D scene points from our point correspondences in the 2D image pairs we capture. We select two points in pixel coordinates in the image pair, which we denote as x_1 and x_2 for the respective images, and conduct a linear triangulation algorithm to estimate the 3D scene point that was captured in the scene. The linear triangulation method is a direct analogue to the Direct Linear Transform (DLT) method [1]. The idea behind this method is that for each image in the stereo camera pair,

we have a scene point X that will map to points x_1 and x_2 for the respective cameras. Given the camera matrix P_i for both cameras $i = 1,2$, we have the formula:

$$\mathbf{x}_i = P_i \mathbf{X} \tag{3.1}$$

Then for each image point, we can write three equations, of which two are linearly independent, due to the fact that the cross product eliminates the homogeneous scale factor of image points. Normally, when we represent points, such as $x_1 = \tilde{x}_i \begin{bmatrix} x & y \end{bmatrix}^T$, it is in the inhomogeneous representation, and for this algorithm, we use the homogeneous representation, which is simply $x_1 = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$, which can be in different scale then the inhomogeneous representation. Therefore, because we remove the scale factor, we able to use the following equations:

$$x(\mathbf{p}^{3T}\mathbf{x}) - (\mathbf{p}^{1T}\mathbf{x}) = 0 \tag{3.2}$$

$$y(\mathbf{p}^{3T}\mathbf{x}) - (\mathbf{p}^{2T}\mathbf{x}) = 0 \tag{3.3}$$

$$x(\mathbf{p}^{2T}\mathbf{x}) - y(\mathbf{p}^{1T}\mathbf{x}) = 0 \tag{3.4}$$

Where $\mathbf{p}^{iT}$ are the rows of the camera matrix P, and these equations are linear, where we can create the relationship $A\mathbf{X} = 0$, in which A is:

$$A = \begin{bmatrix} x_1\mathbf{p}_1^{3T} - \mathbf{p}_1^{1T} \\ y_1\mathbf{p}_1^{3T} - \mathbf{p}_1^{2T} \\ x_2\mathbf{p}_2^{3T} - \mathbf{p}_2^{1T} \\ y_2\mathbf{p}_2^{3T} - \mathbf{p}_2^{2T} \end{bmatrix} \tag{3.5}$$

We solve A through the homogeneous method, specifically DLT, which is finding the smallest singular value of A through Singular Value Decomposition (SVD), and using the singular vector corresponding to that singular value will be our solution, or estimation, of the 3D scene

point from the corresponding points x_1 and x_2.

For our use case of triangulation, it is a little more complex as our keypoints that we intend to match across image pairs will not simply be Scale-Invariant Feature Transform (SIFT) features or corner points. Instead, we will want to detect light sources in images, which requires a two-fold process. First, we must detect light sources in an image reliably. Then, we must find the corresponding light source in the following image pair to properly conduct the trinangulation method.

3.2 Light Detection Algorithm

For detecting light sources in our images, we took inspiration from research in traffic light detection algorithms, and applied the concept of converting our images to different color spaces to extract the pixel location of possible light blobs in an image through color analysis. Our process was to test three different color spaces of traffic light detection papers that shared similar datasets to our problem, conduct pixel analysis on a set of images, and empirically choose the best color space for our problem that reliably detects the light sources that we expect to see. The three color spaces we attempted to test where the Hue, Saturation, and Value (HSV) color space [6], a modified HSV color spaces only using the SV channels [11], and the YCbCr color space [5].

3.2.1 Color Space Conversion

The first method we attempted was to convert our images to the HSV color space. This is done though the manipulation of the RGB values of our pixels. First, we convert the 0-255 range of values from the RGB pixels of the image to a normalized value of R'G'B' by $R' = R/255$, $G' = G/255$, and $B' = B/255$, then take the single minimum and maximum value that exist in any of the three channels $C_{max} = max(R', G', B')$ and $C_{min} = min(R', G', B')$. Then, for each channel, we conduct the calculation:

$$H = \begin{cases} 0 & \Delta = 0 \\ 60 \times (\frac{G'-B'}{\Delta} mod 6) & C_{max} = R' \\ 60 \times (\frac{B'-R'}{\Delta} + 2) & C_{max} = G' \\ 60 \times (\frac{R'-G'}{\Delta} + 4) & C_{max} = B' \end{cases} \tag{3.6}$$

$$S = \begin{cases} 0 & C_{max} = 0 \\ \frac{\Delta}{C_{max}} & C_{max} \neq 0 \end{cases} \tag{3.7}$$

$$V = C_{max} \tag{3.8}$$

Where $\Delta = C_{max} - C_{min}$ and H,S, and V corresponds to each channel respectively in HSV color space. Our second method is converting the images to the HSV color space, but instead we create a binary image using the S and V channels of the color space. This is done through the following equation, where we selected the scalar values of $\alpha = 0.35$ and $\beta = 0.65$ through an empirical analysis on our dataset images:

$$C = \alpha \cdot S + \beta \cdot V \tag{3.9}$$

Finally, our third method was to convert the RGB image into the YCbCr color space. This is completed by first normalizing the RGB pixel values to R'G'B' once again, and performing the following three equations on each R'G'B' pixel for the respective channels, which involves using constants K_R, K_G, and K_B:

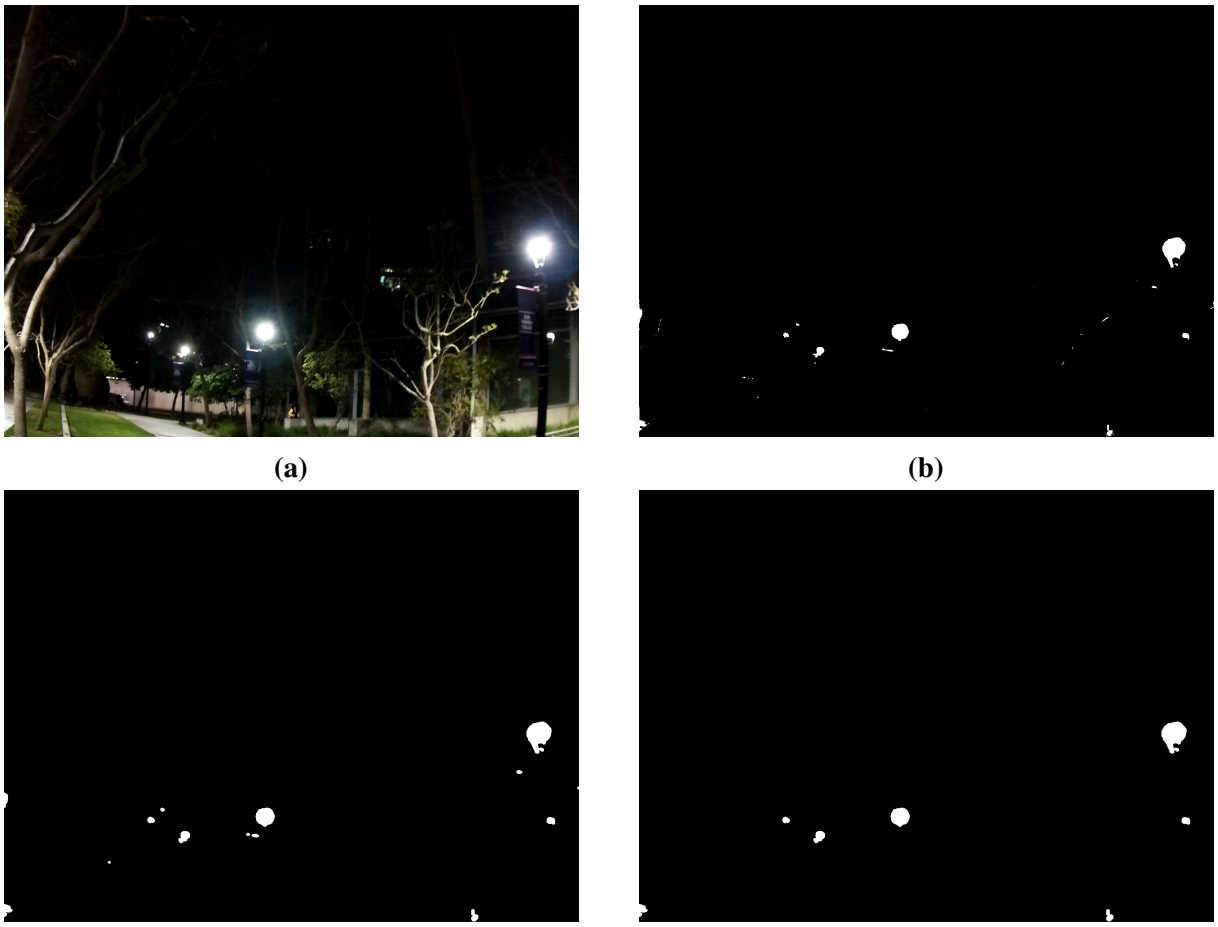
$$Y' = K_R \cdot R' + K_G \cdot G' + K_B \cdot B' \tag{3.10}$$

$$C_B = \frac{1}{2} \cdot \frac{B' - Y'}{1 - K_B} \tag{3.11}$$

$$C_R = \frac{1}{2} \cdot \frac{R' - Y'}{1 - K_B} \tag{3.12}$$

Where K_R, K_G, and K_B are derived from the corresponding RGB space with the constraint that $K_R + K_G + K_B = 1$.

3.2.2 Filtering Algorithm



(a) **(b)** **(c)** **(d)**

Figure 3.1. (A) Original image, (B) Image after applying converting to the YCbCr color space and applying pixel value filtering, (C) Image after applying dilation and erosion, and (D) image after applying connected component analysis.

After we apply our color space filtering to detect possible light sources in the image, there will be salt and pepper noise that will lead to many false detections. Because of this, we need to apply image filtering algorithms to reduce the amount of possible false detection in the image to ensure we reliably detect light sources only. We do this through a pipeline of image

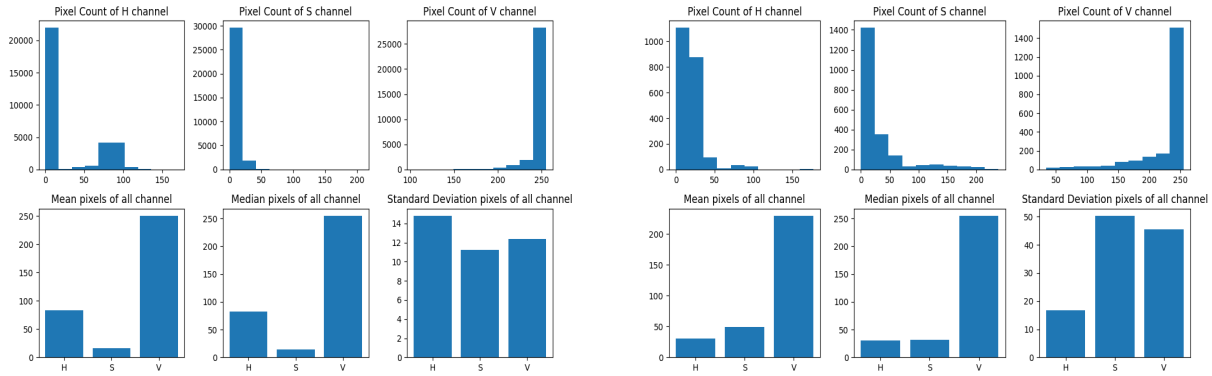
erosion, to eliminate salt and pepper noise, image dilation, which is to increase the size of the detected light sources to their original shape after erosion, connected component analysis to apply filtering on the blobs we detect based on the shape and area, and finally blob detection to detect the remaining light source blobs in the image and get the center of the blobs to the sub-pixel level as our key points. The process of this pipeline is shown in the figure 3.1.

3.2.3 Selecting the Right Color Space

Figure 3.2. Example image of selecting ROI light sources for our training data.

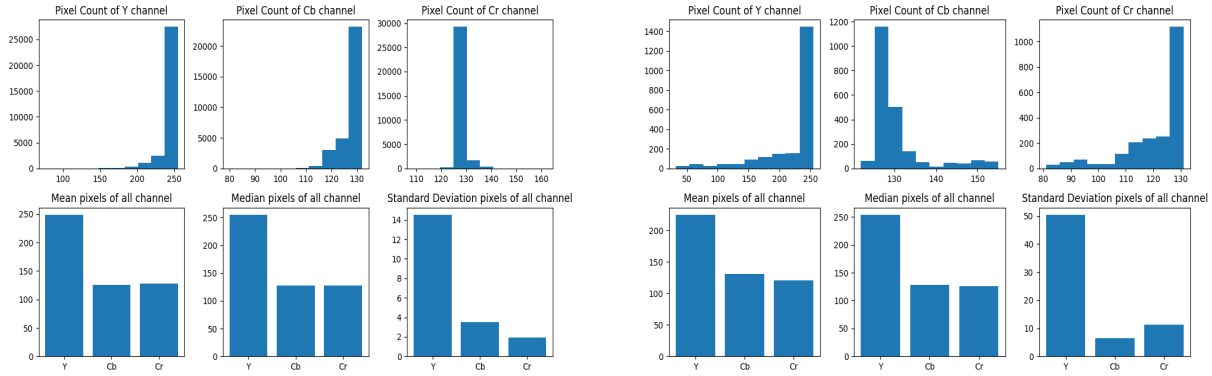
Our methodology for evaluating the effectiveness of each color space for detecting light sources involves using a subset of our evaluation dataset as training images, where we convert the set of images to the corresponding color space, analyze the average and standard deviation of the pixel value of the region of interest (ROI) of our light sources, as we see in figure 3.2 and create a filtering algorithm that filters the image based on the range of values we chose to filter for each corresponding color space. The range of values we decide to filter with is the range of $Avg \pm Std\_Dev$, or average and standard deviation, that we encounter when conducting the color space analysis.

We do a color study analysis of light sources of two different colors we typically see at the UCSD campus, which is denoted as Amber lights and White lights. We analyze the both the HSV and YCbCr color space by viewing the mean, median, and standard deviation of the pixel values of light sources we expect to see during our dataset. Our process was to select regions



(a) **(b)**

Figure 3.3. (A) evaluation of pixel values in HSV color space for existing white light sources in the images. (B) evaluation of pixel values in HSV color space for existing amber light source in the image.



(a) **(b)**

Figure 3.4. (A) evaluation of pixel values in YCbCr color space for existing white light sources in the images. (B) evaluation of pixel values in YCbCr color space for existing amber light source in the image.

of interest (ROI) of our images that is a small subset of our dataset for evaluation that contain light sources to analyze the pixel values of all ROI's we detected across all subset of images for their mean, median, and standard deviation, and to use these values to build the range of pixels we will use for binarization of our images for light source detection by the mean and standard deviation of those pixel values. We can see from figures 3.3 and 3.4 how we analyzed each color space, which we then used to build the table of the range of pixel values to use for light source detection. Our next step was to conclude the effectiveness of each method we discussed

in section 3.2.1 through an empirical study of their performance on test datasets.

In the figures 3.6, 3.7 and ,3.5, we can see the performance of each method. Specifically, we evaluate HSV, HSV with two channels only, and YCbCr respectively on a test set of images on our dataset where we did not conduct analysis on. From both an empirical study, and as we can see from the examples, the color space YCbCr generated the best results which is the method we continued with through the rest of the project.

(a) **(b)**

Figure 3.5. (A) Base Image (B) Detected Light sources after filtering with YCbCr color space

(a) **(b)**

Figure 3.6. (A) Base Image (B) Detected Light sources after filtering with HSV color space

(a) **(b)**

Figure 3.7. (A) Base Image (B) Detected Light sources after filtering with SV color space

3.3 Point Correspondence Detection

Point correspondence between the image pairs for detected light sources is quite simple due to the light source detection algorithm developed in the previous section. When we compute the light source detection algorithm on the image pairs captured from our stereo camera, we have detected blobs where the light sources exist in the images that contain the sub-pixel centroid of those blobs and the size of the blobs in pixel area. For point correspondence matching across the image pairs, the information contained of the detected blobs makes it very simple as we conduct an exhaustive comparison across every blob detected in the left image to the right image with specific constraints to our system to determine whether the lights ”match” or not. Our conditions for whether lights match or not are based on the size of the detected light blobs and the centroid. For our specific conditions, we first check if the size of the blobs between possible matching lights differ between a value of no more than 100 pixels, which was an empirical value that was decided on through testing with the worst case scenarios. In our dataset, the centroid pixel difference of the detected blobs in the X-axis values only differ by 30 pixels and the pixel difference of the Y-axis values differ by 50 pixels for the non-rectified image case and 3 pixels for the rectified image case. Finally, we can see our results in the following figure 3.8.

Figure 3.8. (A) Left Base Image (B) Right Base Image. (C) and (D) are the left, and right, images after applying the YCbCr filtering method and point correspondence detection.

3.4 Relative localization estimation

Our last step of the pipeline for light source geolocation is optimizing the absolute orientation estimation from our sensor readings of the stereo camera system. For this optimization, we take inspiration from Strasdat et al. in their paper [9]. We apply a keyframe and bundle adjustment approach to optimize the camera orientation for 3D point estimation for a given scene. Our data set consists of walking along a path on the campus of UCSD, where we are able to easily track features consistently project 3D points across frames. With this in mind, for computation optimization, we take a keyframe approach by only using bundle adjustment on keyframes of our data that have a certain amount of overlap in repeating features. In our case, we define a keyframe in our data if the current frame has 40%, or less, overlap of key features from the previous denoted keyframe. This requires feature tracking across frames. We employ a two-view feature tracking method across all of our data 3.9, and we keep track of our defined keyframe's features, and compare the features to the following frames to determine the overlap, where once it crosses below the threshold of 40% overlap, we take that frame as a keyframe, save all of those features, and repeat the process. Our process for selecting features is using SIFT to determine feature points across the image pair, run a k-nearest-neighbor matcher to find point correspondences, then use RANdom SAmple Consensus (RANSAC) to remove outliers by finding the fundamental matrix across frames. One thing to notice is in the figure, we can see that despite running outlier rejection, outliers still exist in the matching frames, which perpetuate across the entire dataset when doing keyframe detection due to using night time images only.

After we collect our keyframes, we take note of all of the keypoints that were tracked for all of our keyframes, designate 3D scene points that correspond to matching key points across all frames, and finally triangulate the corresponding 3D scene point based on the first keyframe and the stereo image pair. Finally, once we have this dataset of all the 3D scene points, all of the 2D points that correspond to that scene point, and all the camera extrinsic parameters used to estimate the 3D scene point for all observations, we apply bundle adjustment.

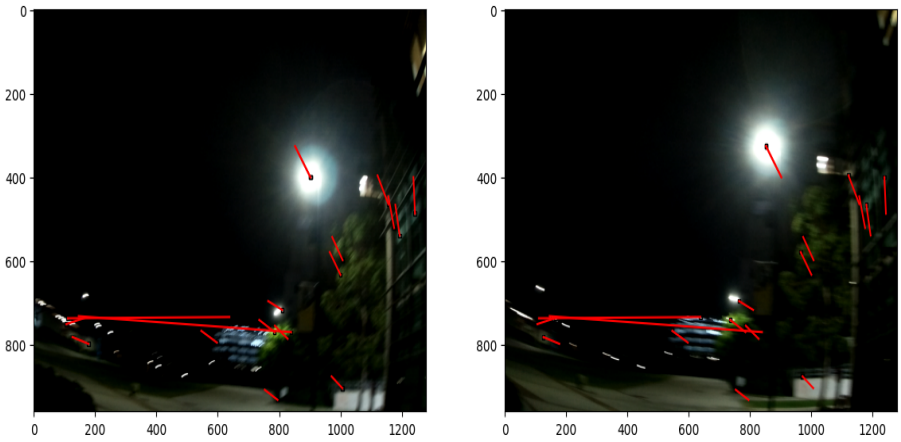


Figure 3.9. Example of tracking features across two views in successive frames. This is using the left camera only as the basis.

We specifically apply large-scale bundle adjustment based on the parameters we collected above. Our task is to refine the 3D coordinates and the extrinsic parameters of the stereo camera for each frame by minimizing the re-projection error of the 3D scene points onto the 2D images. In our case, let $\mathbf{X} = (X,Y,Z)^T$ for a given 3D scene point, R and T be the rotation and translation matrix of our left camera, and $R_{l,r}$ and $T_{l,r}$ be the rotation and translation matrix from our left camera to the right camera of our stereo system. Finally, we set up our parameter vector for bundle adjustment as $\mathbf{P} = ((R_1,T_1),(R_2,T_2),\ldots,(R_n,T_n),X_1,X_2,\ldots,X_m)$ for n observations and m 3D scene points estimated from our dataset, which would be in GPS coordinates. Our measurement vector is the 2D image points we used to project to the given scene point, or $\mathbf{M} = ((x_{1,1},x_{1,2})^T,\ldots,(x_{k,1},x_{k,2})^T)$ for the k pixel coordinates tracked for each estimated scene point. Our re-projection is done by the process of $x_{m,1} = P_{n,L}\mathbf{X}_k$ and $x_{m,2} = P_{n,R}\mathbf{X}_k$, where $P_{n,L} = K_L \cdot [R|T]$ and $P_{n,R} = K_R \cdot [R_{l,r}R|R_{l,r}T + T_{l,r}]$, where we calculate the sum of squares of the re-projection error for both pixel points $x_{m,1}$ and $x_{m,2}$ for each point m corresponding to the observation n and the scene point k. With bundle adjustment optimizing for the scene reconstruction, our original estimation of the camera absolute orientation from our sensors will be refined and the estimated GPS location of light sources will be more accurate than previous experiments prior to bundle adjustment.

Chapter 4. Evaluation

To evaluate the effectiveness of our system, we run an experiment with a static deployment to evaluate the accuracy of our sensors without optimization and an experiment walking a short path around a subsection of the UCSD campus to employ bundle adjustment for absolute orientation refinement. We first evaluate the individual components of measuring IMU accuracy, stereo depth estimation accuracy, and fixed position accuracy. We then consider the complete system, and evaluate the ability to reconstruct light positions along a walking path.

4.1 Static Position Experiments

To evaluate the accuracy of the IMU sensor and test the depth accuracy of our stereo camera system, we design a controlled experiment where we place the camera at a fixed easily identifiable landmark. We use Google Earth to estimate the GPS of the landmark, and remove the GPS device from this experiment. We then place a camera calibration board as a robust landmark at known positions in the captured scene and manually select points in the image to have a more accurate 3D point estimation. Figure 4.1 shows the 10 positions we evaluate.

Overall Performance

For every position mentioned in figure 4.1, we took images of holding the calibration board at each of those positions as shown in figure 4.2. We manually select corresponding

Table 4.1. Average distance error of projected GPS location of calibration board and the ground truth location of the board in GPS coordinates in meters (m).

Average Error (m)	Minimum Error (m)	Maximum Error (m)
35.525	15.151	50.520

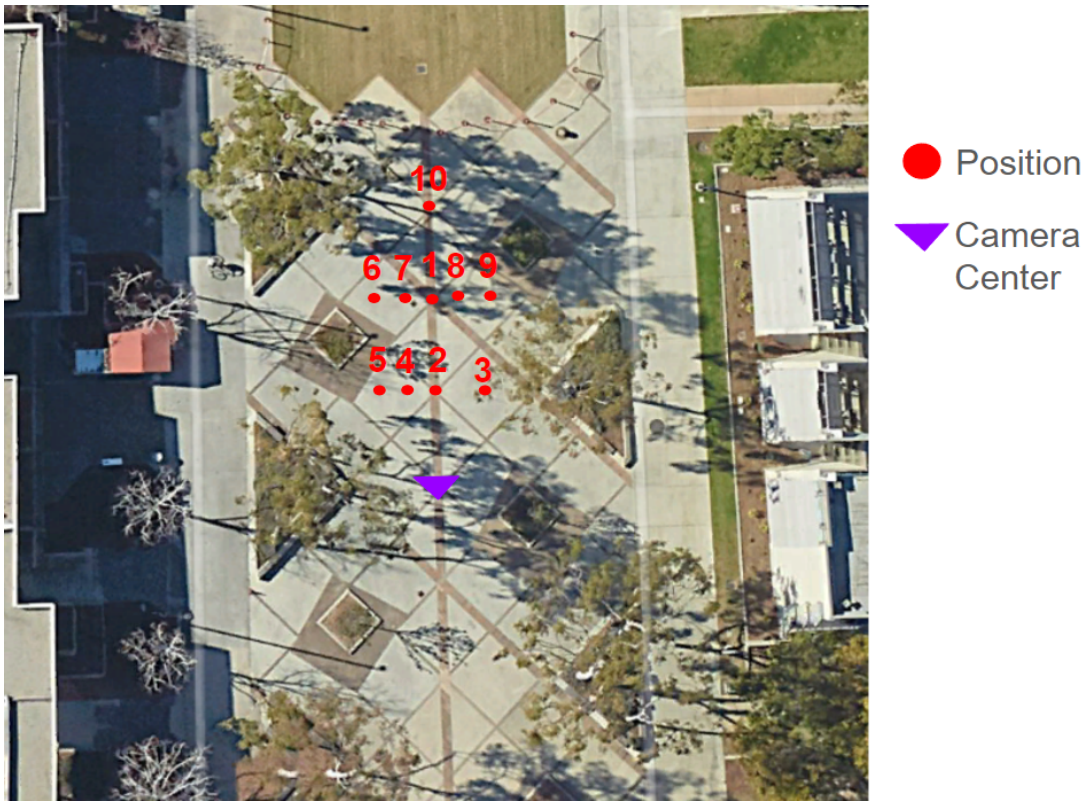


Figure 4.1. Experiment of testing the accuracy of IMU sensor absolute orientation readings and camera calibration parameters. Held a calibration board at each of 10 positions (Red Circles) while keeping camera still (Purple Triangle).

Figure 4.2. First 3 checkerboard positions to evaluate stereo camera calibration parameters and IMU orientation readings.

points in the image pairs for a more accurate 3D point estimation to ensure we are projecting the object of interest correctly. We evaluate our camera's depth estimation and absolute orientation estimation. As we can see in table 4.1, our results can be improved, where the average distance error, in meters (m), between the estimated point and ground truth point was 35 m, minimum distance error was 15 m, and the maximum was 50 m.

In table 4.2, the estimated depth of the calibration board at each position is much larger than the ground truth distance between the camera center and position of the calibration board in GPS coordinates. This result was a behavior we did not expect considering the accuracy of our calibration parameters, which we should have a re-projection error of 0.12 for each camera and a stereo calibration re-projection error of 0.22 in terms of pixel error. This depth error and the error we see from our IMU device for absolute orientation reading from the estimated 3D GPS location and ground truth location is also a factor in the very large discrepancy of error distance we see. We also see that the depth estimation error is not consistent for every location in the image, where the error seems to increase towards the sides of the image, where radial distortion of the image increases, which is another factor we have to account for in following experiments.

Table 4.2. Estimated depth (in meters) of stereo camera for each checkerboard position and the ground truth distance between camera center and checkerboard location.

Object Position	Depth (m) (Camera)	Depth (m) (Ground Truth)
Pos. 1	14.842	12.916
Pos. 2	10.276	6.461
Pos. 3	10.495	6.641
Pos. 4	9.539	6.906
Pos. 5	11.153	7.288
Pos. 6	18.743	12.942
Pos. 7	17.467	12.928
Pos. 8	18.771	13.288
Pos. 9	19.526	14.054
Pos. 10	26.606	19.279

4.2 Putting It All Together: Walking with GPS, IMU, and Bundle Adjustment

To evaluate our system's absolute orientation estimation, we walked along the path shown in figure 4.3. We collected a total of over 180 frames, where each frame contains a corresponding IMU Euler angle reading for system orientation and GPS reading for system positioning. We evaluated our system's accuracy against the light sources in our dataset at the locations seen in the figure on both the raw sensor outputs of the system's orientation and position, and the refined orientation and position of our system using bundle adjustment as the optimization algorithm.

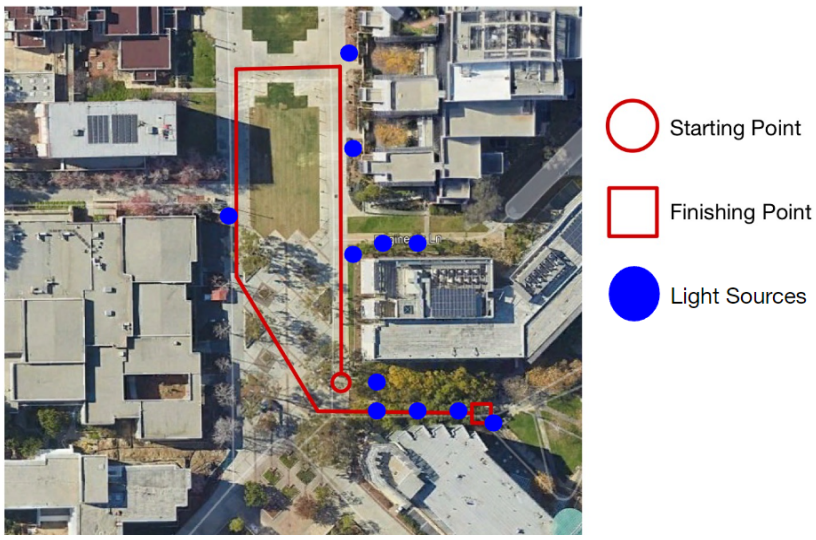
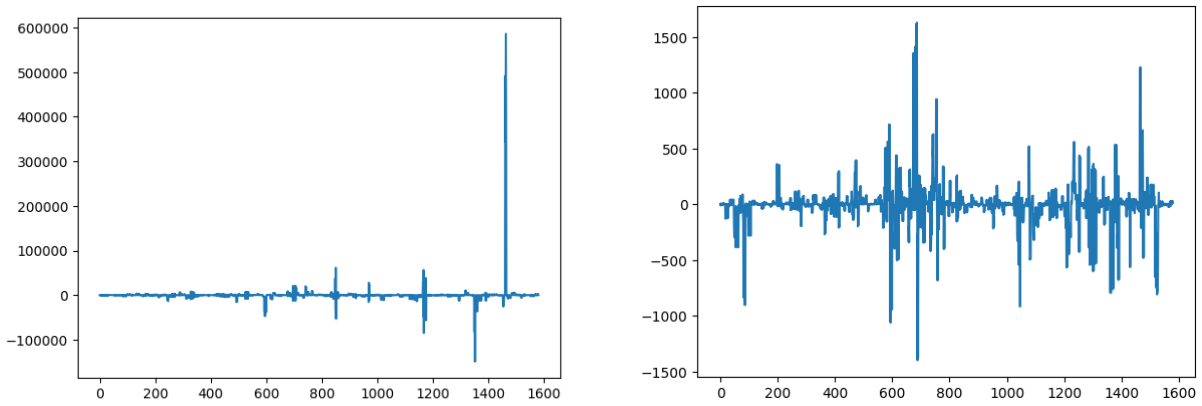


Figure 4.3. Experiment for testing camera orientation optimization through camera movement with keyframe detection and bundle adjustment. Red trail is the path walked, blue circles are light sources we evaluated, and hollow red shapes are the starting, and end, point of the experiment

4.2.1 Evaluating the Effectiveness of Bundle Adjustment

For the 180 frames from figure 4.3, we apply key frame detection for two-view image pairs and then apply bundle adjustment on the remaining frames based on estimated 3D GPS scene points of the scene to minimize the sum of squares re-projection errors of the 3D scene points by optimizing both the camera orientation and estimated 3D scene points simultaneously. We detect keypoints in the image pair through SIFT features once again for bundle adjustment to have more observations for optimizing the camera pose estimates. After applying bundle



(a) **(b)**

Figure 4.4. The sum of square re-projection errors of the 3D estimated GPS scene points to the 2D image points for each observation in our dataset prior to bundle adjustment (A) and after bundle adjustment (B).

adjustment, our camera absolute orientation readings will be refined for the given scenes, which we hypothesized to lead to better accuracy. Based on the figure 4.4, we can see that bundle adjustment minimized the cost function, the sum of squares of the reprojecting errors, and refined the camera poses to adjust to the scene we collected with our image frames.

4.2.2 Performance of Light Localization with and without Camera Pose Refinement

We chose a set of 6 frames in our dataset that was used for bundle adjustment to conduct further analysis on how the camera poses were optimized by comparing the estimated GPS location of light sources to their ground truth location. We can see what frames we selected in figure 4.5.

However, as we can see in table 4.3, the outliers we see in our bundle adjustment figure heavily affected the accuracy of our estimated camera poses enough to worsen the accuracy overall. In every metric, the distance error between the estimated light source location and the ground truth light source location is larger after camera orientation refinement through bundle adjustment.

We can further see this behavior in our next two tables, where the average distance error after refinement is worse on all but one frame. However, we can also see the effect of bundle

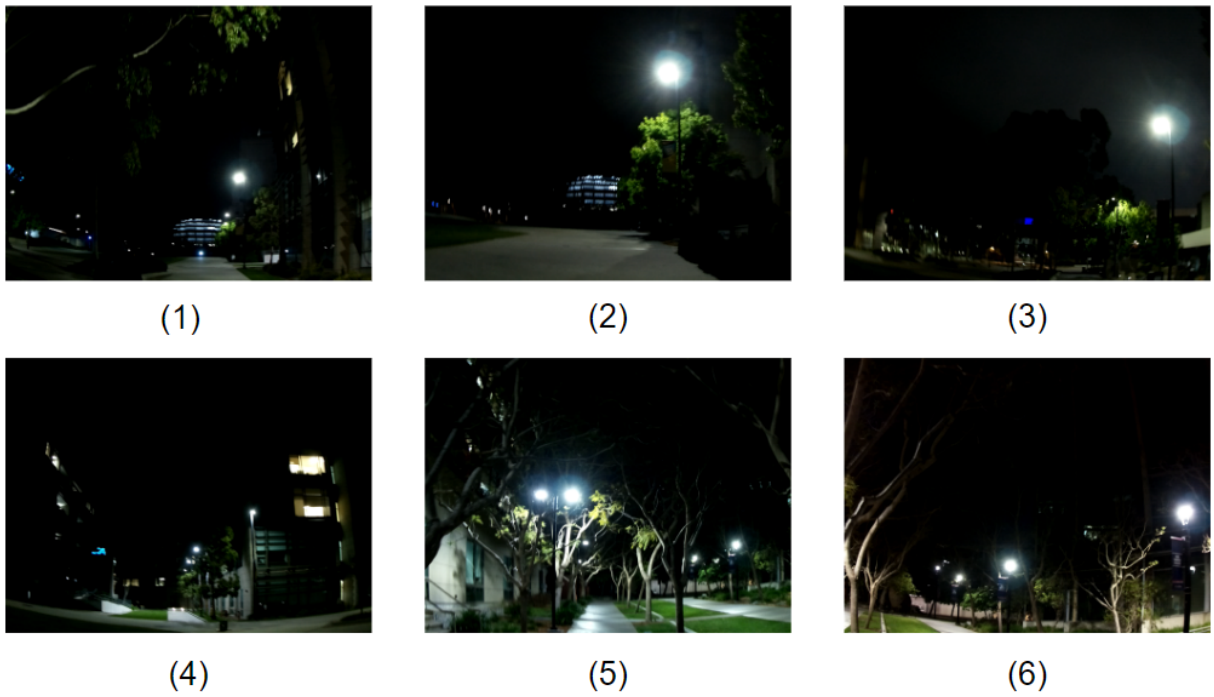


Figure 4.5. The 6 frames we chose and their corresponding labels for table 4.4 and table 4.5.

.

Table 4.3. Average, minimum, and maximum distance error, in meters (m), between the estimated light source GPS location of our system before and after refinement. Estimated light sources are shown in figure 4.3

Refinement	Average Error	Minimum Error	Maximum Error
Yes	694.550	15.592	2216.980
No	263.224	12.121	1191.391

Table 4.4. Average distance error and standard deviation between the estimated GPS coordinates of light sources from our system after camera orientation refinement from bundle adjustment and their ground truth GPS location (taken from Google Earth).

Frame	Average Error (m)	Standard Deviation (m)
1	2208.961	8.019
2	399.748	0.000
3	1508.082	0.000
4	430.712	7.278
5	747.039	15.813
6	32.133	14.600

Table 4.5. Average distance error and standard deviation between the estimated GPS coordinates of light sources from our system before camera orientation refinement from bundle adjustment and their ground truth GPS location (taken from Google Earth).

Frame	Average Error (m)	Standard Deviation (m)
1	852.471	338.920
2	72.356	0.000
3	188.500	0.000
4	184.909	73.195
5	305.364	122.087
6	81.710	44.150

adjustment in the case of camera orientation refinement based on the scene of the dataset, where the standard deviation of error per frame for the estimated GPS location of the light sources is much lower, and having more consistent GPS estimation compared to the case of the using the raw output from our IMU and GPS positioning sensors.

It is important to note that the using the raw output for absolute orientation and position of our IMU and GPS sensor leads to more accurate light source GPS estimation on average. Despite bundle adjustment refining the camera absolute orientation for our given scene, the outliers in our dataset from the sensor readings lead to incorrect optimization, causing this behavior. As a result, we would need more than just bundle adjustment and key frame detection for optimizing the absolute orientation and GPS positioning estimates of our system.

Chapter 5. Conclusion and Future Work

In this section, I discuss what do our results mean, what caused the bahvior we faced, and what we can do to further improve the absolute orientation, GPS positioning of our system, and object depth estimation.

5.1 Camera Calibration

From our results, it is evident that camera calibration still needs some improvement as our depth estimation is incorrect at each point of the camera, and even worse at positions at the side of the camera where radial distortion is more evident. Despite our seemingly good results in calibration, where we achieved a re-projection error of 0.12 for each camera, and a stereo calibration error of 0.22, the issue likely resides in the real world scaling of the camera parameters when converting scene points in the world coordinate frame to the camera coordinate frame. We know calibration is good based on the consistency of our depth estimation of objects across the experiment in 4.2.1, but the world scaling is incorrect that leads to this issue, where would need to further explore the exact cause through more sets of calibration.

5.2 Light Detection

Our results in automated light detection work extremely well, as we see in our examples in section 3.3. We can reliably detect light sources in the scene and match the corresponding light sources across our stereo image pair. However, there are cases we do receive some outliers in light source detection due to either lights reflecting on windows, or areas surrounding the light source causing strong illumination. As a result, our automated light source detection algorithm works well on the image filtering side, but can be improved by implementing voltage detection

in our pipeline to further filter the false positives and ensure we only detect light sources.

5.3 Positioning Lights in Real World Space

From our results, we can see that our pipeline for refining the initial readings of our IMU and GPS position sensor can be much improved due to the large light source GPS estimation errors we received between the distance of the ground truth point and our estimated point. We can see however that the issue arises in both the IMU sensor raw readings and in bundle adjustment, where the large outliers are causing the smoothing of our relative estimated 3D scene points to optimize in the wrong direction, and losing information of the real world location of our camera. This is still an area in the pipeline that needs large improvement, and requires rethinking of how we approached this problem due to the large noise of our raw sensor readings from the IMU and GPS devices.

5.4 Next Steps

The focus for improvement in our pipeline lies in both calibration and absolute orientation of our camera system. For calibration, we need to re-calibrate the camera by tuning our world scaling settings slightly to have a better understanding where in the calibration pipeline is our camera receiving these errors. This is a rather simple fix and requires time in tuning our calibration pipeline.

Improving absolute orientation of our camera system is more involved due to the errors we received. First focus of improvement would be to reduce the rate of outliers in our GPS positioning device, as the large outliers we receive in GPS location have an effect in how our bundle adjustment algorithm optimizes the camera's pose. This can be either including the camera center, or GPS location, during bundle adjustment, or implement a kalman filter using GPS location and camera motion to tune the GPS readings to reduce outliers. This can be a direct improvement to our pipeline.

Another opportunity for improvement is our process for building the scene for bundle

adjustment. As night time feature tracking is very sparse, this leads to a rather low amount of 3D scene point observations in our dataset. That leads to the large outliers we see having a stronger effect in optimization in the wrong direction. We can also improve how we conduct feature tracking, as our current implementation applies a two-view feature tracking method, and instead a more robust implementation would apply a trifocal tensor to track features across three frames and remove a large set of outliers. In this case, we would only include true point correspondences in our key frames to reduce the number of outlier 3D points we may have received with two-view tracking.

Including each of these components will lead to a direct improvement of our system as it is evident one of the main issues is the large outliers we see in our data set causing bundle adjustment to optimize poorly. Once implemented, we can conduct further evaluation on what areas of our system needs improvement, whether it is the sensors for the IMU or GPS positioning causing too much noise, or other optimization algorithms, such as using a Kalman filter for absolute orientation reading instead of a 9 DoF IMU sensor. that we can utilize to further improve the system.

5.5 Closing Remarks

This thesis set out to establish the visibility of developing an economical stereo imaging system to localize lights in the built environment. Our work to date establishes, that while this is indeed possible, to realize a system capable of autonomous mapping of infrastructure will require more advanced processing, in particular better utilization of information carried over time between frames, than is realized by this initial prototype system.

Bibliography

[1] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[2] Weifan Jiang, Vivek Kumar, Nikhil Mehta, Jack Bott, and Vijay Modi. Irrigation detection by car: Computer vision and sensing for the detection and geolocation of irrigated and non-irrigated farmland. In *2020 IEEE Global Humanitarian Technology Conference (GHTC)*, pages 1–4, 2020.

[3] Johnathan S. Jones. 128 million smart meters in the us in 2023, 2023.

[4] Noah Klugman, Joshua Adkins, Emily Paszkiewicz, Molly G. Hickman, Matthew Podolsky, Jay Taneja, and Prabal Dutta. Watching the grid: Utility-independent measurements of electricity reliability in accra, ghana. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)*, IPSN '21, page 341–356, New York, NY, USA, 2021. Association for Computing Machinery.

[5] Vladimír Kunštár, Štefan Hajdú, and Tibor Krajčovič. Traffic lights recognition using computer vision on the microcomputer platform. In *2023 International Conference on Applied Electronics (AE)*, pages 1–4, 2023.

[6] Mehdi Salarian, Andrea Manavella, and Rashid Ansari. A vision based system for traffic lights recognition. In *2015 SAI Intelligent Systems Conference (IntelliSys)*, pages 747–753, 2015.

[7] Zeal Shah, Alex Yen, Ajey Pandey, and Jay Taneja. Gridinsight: Monitoring electricity using visible lights. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '19, page 243–252, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Kexian Shen. Effect of baseline on stereo vision systems, 2018.

[9] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Real-time monocular slam: Why filter? In *2010 IEEE International Conference on Robotics and Automation*, pages 2657–2664, 2010.

[10] Jay Taneja. If you build it, will they consume? key challenges for universal, reliable, and low-cost electricity delivery in kenya. *Center for Global Development Working*, 2018.

[11] Tai Huu-Phuong Tran, Cuong Cao Pham, Tien Phuoc Nguyen, Tin Trung Duong, and Jae Wook Jeon. Real-time traffic light detection using color density. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4, 2016.

[12] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.