Monocular Vision SLAM using Line Segment Landmarks and Manhattan Grid Estimation

Tim Straubinger

Fall 2020

Abstract

In Simultaneous Localization and Mapping (SLAM), a robot agent senses its surroundings and maintains a geometric representation of its environment while computing its own orientation relative to that environment. Many successful applications of SLAM rely upon advanced sensing hardware which is expensive or difficult to obtain for hobby roboticists and which produces a volume of data that is too great for many embedded systems to meaningfully process in real time. However as digital cameras become more commonplace and processing power on small devices increases, basic computer vision on simple hardware becomes feasible. Monocular cameras provide rich information about the environment, but in order to apply this information to SLAM in a 3D environment, the missing depth information must somehow be accounted for, and simplifying assumptions and reductions must still be made to allow interactive computation.

This work explores the application of straight line segments extracted from visible edges in monocular camera images captured by a small indoor robot, as proposed by Lee *et al.* [LKC19], as sensory input to a SLAM system. Line segments are used both as landmarks and as indicators of the principal geometric axes of the building. A simple robot with a monocular camera is constructed, and a dataset of camera images and robot odometry is collected. A novel algorithm for estimating a Manhattan grid orientation from line segments is proposed and evaluated. The limitations and shortcomings of this work due to high variance in vanishing point estimation and due to unreliable landmark correspondences are discussed.

Source Code

The software implementation of this work, as well as the acquired dataset, is available at https://www.github.com/timstr/MonoCamLegoSLAM

1 Introduction

Effective and robust SLAM techniques frequently rely upon sophisticated sensing hardware such as LIDAR systems, laser range finders, or depth cameras. However, meaningfully processing the high volume of information provided by such sensors is computationally expensive, and this becomes infeasible for robots with limited computational resources and low-budget hardware. The implementation of a SLAM system that is both computationally inexpensive and reliant on only simple forms of sensor data poses a unique challenge, and the compromises between rich semantic information, simple sensors, and efficiency are a broad area for exploration.

Digital cameras are becoming increasingly common in everyday hardware, and typical consumer webcams are able to perform a large amount of on-board processing such as colour space encoding, down-sampling, and even video compression, which simplifies the workload for a micro-controller that uses such a camera as a sensor. Even after down-sampling and conversion to grey-scale, an image that is tiny by everyday standards still contains a prohibitively large amount of information for a simple embedded system. Camera images observe light that is affected by a vast number of confounding phenomena such as the physical geometry of the scene, its optical properties, lighting and occlusions, and camera-specific effects such as field-of-view, exposure, focal length, and others. Decoupling all of these parameters is currently at the cutting edge of computer vision research, and is far out of of the reach of simple micro-controllers. Thus, simplifications are needed in order to categorize the information in observed camera images in a format suitable for mapping and localization.

To extract features from camera images, various local feature descriptors have been proposed [Low04; BTV06; Cal+10; Rub+11]. These feature descriptors aim to identify distinct sections of an image with labels such that the extracted labels suffice for robust correspondence-finding of observed objects in images when seen at different scales, viewing angles, and other conditions.

As an alternative to these feature descriptors, line segments detected from contiguous straight edges provide an alternative image feature that is easier to reason about [Gio+12]. Additionally, whereas local feature descriptors may be extracted from visually complex objects, straight line segments are matched to basic but nonetheless descriptive object contours. Lee *et al.* [LKC19] observe that in the context of indoor home environments, whose structures comprise primarily of right-angled walls, floors, ceilings, and furniture items, these line segments are preferentially matched to static structural elements rather than to dynamic obstacles like humans and pets.

While local feature descriptors have been applied to SLAM [MMT15], line-segment-based SLAM [LKC19] provides an alternative which facilitates easier geometric reasoning and, under simple assumptions about the building layout, may be used both as landmarks and as near-direct indicators of the robot's angular orientation. Specifically, assuming that the building structure conforms to an orthogonal or Manhattan grid, the directions of the major axes of this grid may be estimated by the vanishing points to which observed line segments are aligned [Baz+12; Zha+16]. The combination of Manhattan grid orientation and line segment landmarks provide geometric descriptions of the environment that may be applied to maintaining a map database and correcting estimated robot poses [LKC19].

2 Related Works

In order to reduce digital camera images to simple feature labels that suffice for correspondence matching, several image descriptors have been proposed. These feature descriptors automatically identify points of interested and label them in a deterministic manner. Scale Invariant Feature Transform (SIFT) [Low04] is a local image descriptor which is invariant to changes in scale and rotation and is robust to noise and differences in lighting. Speeded-Up Robust Features (SURF) [BTV06] is a closely related technique which offers greater efficiency and improved robustness in some cases. While SIFT and SURF produce continuous feature descriptions, Binary Robust Independent Elementary Features (BRIEF) [Cal+10] instead produces bit strings from image locations which simplify reasoning about feature similarity, and is claimed to yield similar performance. The closely-related Oriented FAST and Rotated BRIEF (ORB) image descriptor [Rub+11] offers better rotation invariance and noise robustness compared to BRIEF. For a detailed evaluation of these feature descriptors and their behaviours under various optical effects, readers are referred to Karami *et al.* [KPS17].

Unlike local image descriptors, Line Segment Detector [Gio+12] extracts image features that are simultaneously low-dimensional and visually apparent to a human observer, and which may span much larger portions of an image. The detected line segments follow visual contours and straight edges, and are iteratively grown from simple co-linear edge portions until large contiguous line segments are yielded. In urban and indoor environments, whose structures often align to 3 orthogonal grid axes, also referred to as a Manhattan grid, the observed line segments tend to associate with one of these three principal directions. By clustering these aligned segments and extracting the implied set of vanishing points, the Manhattan grid orientation relative to the camera may be estimated. However, clustering and classifying each detected line segment while fitting them to an orthogonal grid and detecting outliers is a challenging problem in combinatorial optimization. Bazin *et al.* [Baz+12] assume a perspective camera model and propose a branch-and-bound algorithm which searches for the globally optimal clustering of line segments into vanishing points and outliers. Zhang *et al.* generalize on the perspective assumption by proposing a camera model which permits non-linear curvature (for example, due to fisheye lenses) and which contains the perspective camera model as a special case. Their method is able to simultaneously estimate both the vanishing point and some camera parameters if they are not explicitly provided.

SLAM with visual or range-finding inputs requires some method for aligning geometric measurements in order to identify landmarks and to localize the robotic agent relative to those landmarks. If depth or distance information is unknown, as is the case with monocular camera images, some manner of accounting for perspective effects and possibly occlusions is required. Random Sample Consensus (RANSAC) [FB81] is a stochastic technique for simultaneously classifying data points as inliers and outliers while fitting a model to the data, which has successfully been applied to the problem of estimating an observer's position from landmark measurements. Bundle Adjustment [Tri+99] is a closely-related technique which replaces the stochastic selection process of RANSAC with a simple least-squares minimization that optimizes over landmark locations and robot poses subject to the observed landmark locations after camera projection. A more general stochastic optimization process is Simulated Annealing [Rut89], which solves combinatorial optimization problems by applying random modifications to a candidate solution with a gradually increasing preference towards solutions producing a better score, where the choice of score function is domain-specific.

Mur-Artal *et al.* [MMT15] propose to use the ORB image feature descriptor [Rub+11] to extract landmarks and find correspondences between landmark observations as part of a SLAM system. The technique is applicable to outdoor environments and fairly general static scenery. However, specifically targeting indoor environments which may contain moving people, Lee *et al.* [LKC19] use line segments extracted using LSD [Gio+12]. In their work, detected line segments are used both to estimate the orientation of Manhattan grid axes under the orthogonal structure assumption and as landmarks for SLAM. Using the vanishing point labels assigned to each line segment obtained using Zhang *et al.*'s method [Zha+16], as well as local image patches for correspondence finding, Lee *et al.* [LKC19] align landmarks and correct robot poses using least-squares optimization. Although the camera model used for vanishing point estimation permits non-linear camera models, a perspective camera is nonetheless assumed in their work to simplify the projective geometry of landmark observations during optimization.

3 Method

Closely following the work of Lee *et al.* [LKC19], a simple robot equipped with a differential drive and a forward-facing monocular webcam is constructed. A dataset is acquired by driving the robot around an indoor environment while gathering images and recording odometrical information. While the method proposed by Lee *et al.* [LKC19] is designed to be efficient enough to enable real-time usage, their dataset is collected offline and the robot motion is treated as an input. Due to this and to a much slower micro-controller, most processing is performed offline in the proposed method as well. Similarly to Lee *et al.* [LKC19], no path planning is attempted because the available line segment information is believed to be insufficient for collision avoidance. An alternative algorithm for vanishing point estimation is proposed that uses simulated annealing and a cost function that favours orthogonal view points. Similarly to Lee

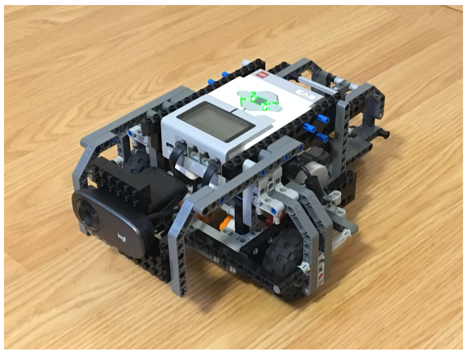

Figure 1: The robot used in this work.

et al. [LKC19], landmarks are extracted from line segments and tagged using local image patches for correspondence finding. The extracted landmarks are simultaneously aligned with robot poses according to their observed orientations as seen by the camera.

3.1 Robotic Hardware

A basic robot was constructed using Lego® with an EV3™ micro-controller, two NXT™ servo motors, and a Logitech® C310™ commodity webcam. This robot is depicted in Figure 1. The micro-controller runs the custom ev3dev operating system [ev3] to allow for more ergonomic software development. Like the robot used by Lee *et al.* [LKC19], the motion of this robot is restricted to a plane. For dataset acquisition, the robot was controlled remotely using a Bluetooth® connection from a laptop. Software for the robot was written in C++ and is available at https://www.github.com/timstr/MonoCamLegoSLAM.

The robot is driven by two coaxial wheels to which the servo motors are attached and is supported by a freely-moving castor wheel. This robot is best described by the differential drive model. The wheel radii were measured by driving the robot forwards for a period of time and comparing the distance travelled with the recorded angular travel of the servo motors. Similarly, to measure the wheel displacement, the robot was driven to rotate in place for several complete turns, and the resulting angular travel of the servo motors was used in combination with the previously-measured wheel radii to determine the wheel displacement. With the wheel radii and displacement known, the robot's change in position can be estimated from odometry using dead reckoning.

In order to estimate vanishing points from line segments, the camera's field of view is required. The vertical and horizontal fields of view were obtained by measuring the extents of a visible portion of a flat surface parallel to the camera at known distance.

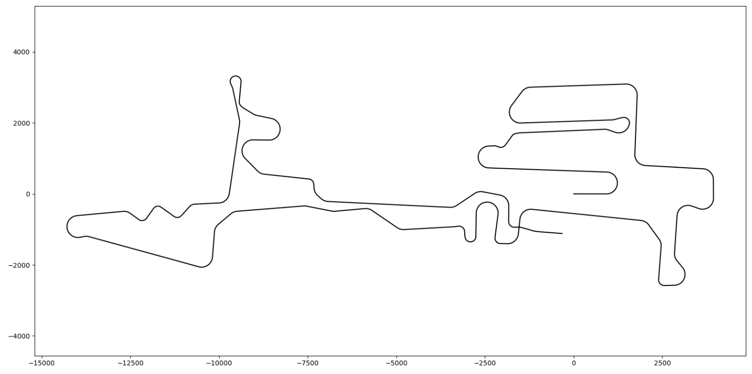

Figure 2: Estimated robot trajectories using dead reckoning. Units are in millimeters. Note the disconnected start and end locations.

3.2 Dataset

A dataset of odometry and camera images was acquired by manually steering the robot through two rooms of an indoor living suite. While most furniture was aligned to right angles, several items seen in the dataset have pronounced straight edges but do not conform to a Manhattan grid. These items include chairs with slanted backrests, hanging coats and sheets, and various circular and cylindrical items. Additionally, while no humans appear directly in the dataset, two cats fulfill the role of moving obstacles to help test the robustness of the various downstream methods. Odometry data was collected at a rate of 10 samples per second while camera images were captured once per second. The robot was driven for approximately 13 minutes, during which it covered a total distance of about 73 meters and captured 493 images. At the end of the route, the robot was driven back to its starting position to allow accumulated drift to be quantified simply by the difference between the start and end positions, which is a metric also employed by Lee *et al.* [LKC19]. Using dead reckoning, as visualized in Figure 2, the final drift by this metric is reported as a linear displacement of 1164 millimeters and an angular displacement of 4 degrees over the course of the route.

3.3 Line Segment Detection

Once camera images are obtained, Line Segment Detector [Gio+12] is used to extract straight line segment features. LSD proved to be a bottleneck when run synchronously on the micro-controller, and so this step is performed offline after the dataset is fully acquired. The extracted line segments are described by the vertical and horizontal coordinates of both endpoints of each line in image space, and these are persisted and associated with each image in the dataset.

3.4 Vanishing Point Estimation

In order estimate the Manhattan grid orientation relative to the camera from line segments which possibly contain outliers, an algorithm based on simulated annealing is proposed. This algorithm is simpler to reason about than Bazin *et al.*'s [Baz+12] branch-and-bound approach that relies on defining an algebra for high-dimensional non-linear intervals. Like Zhang *et al.*'s [Zha+16] method, which relies upon RANSAC, the proposed algorithm uses stochastic optimization in the form of simulated annealing. By incorporating the constraints of orthogonality between vanishing points and penalties for outliers into the energy function, and by upholding the requirements of simulated annealing for eventual convergence, the algorithm converges to a feasible clustering of line segments into vanishing points and outliers. Additionally, while the full cost function is specified in terms of all pairs of line segments and has a computational cost that is quadratic in the number of line segments, the algorithm is made more efficient by only computing the change in cost due to single label changes, which is linear. The details of the proposed algorithm are given in the following sections.

3.4.1 Camera Assumptions

Because the camera has already been assumed to a perspective camera in the line segment projections given by Lee *et al.* [LKC19], and because the robot motion is assumed to be limited to within the ground plane, several simplifications can be made to the vanishing point estimation compared the sophisticated and more general-purpose algorithms proposed by Bazin *et al.* [Baz+12] and by Zhang *et al.* [Zha+16]. These assumptions are stated where relevant to the algorithm.

3.4.2 Discarded Segments

As in Lee *et al.*, line segments with length less than 15 pixels are discarded because they are more likely to correspond to dynamic and non-structural parts of the environment.

Because the robot motion is restricted to the ground plane, the vertical axis of the Manhattan grid is already known. Furthermore, because the camera has been assumed to be a perspective camera, and because the camera faces directly forwards and thus exactly towards the horizon, vertical lines in the environment are expected to always appear vertical to the camera. Because these vertical lines offer no new information about the Manhattan grid orientation, they are discarded. This may remove false positives in the event that lines which are horizontal in the environment are viewed from below and along the same angle, but these are nonetheless ambiguous with vertical line segments and these are not frequently observed in practice.

3.4.3 Candidate Vanishing Point Directions

Because the camera is always aimed at the horizon, the remaining pair of horizontal Manhattan grid axes result in vanishing points that are always expected to lie along the middle of the camera image. In order to determine candidate vanishing points, the line passing through each line segment is intersected with the visual horizon, and the point of intersection is described using angles via a gnomonic projection. The gnomonic projection is chosen because it represents viewing angles uniformly and because it gives meaning to lines that are parallel to the horizon, effectively extending the real number line. The resulting angles measure the candidate vanishing point's location relative to the robot, such that horizontal lines point to vanishing points left and right of the robot. While it may seem counter-intuitive to find the intersections of line segments that are arbitrarily close to being parallel, the gnomonic projection allows these to be represented effectively and with numerical stability. Each remaining line segment is associated with a candidate vanishing point direction in this manner.

Given a line segment spanning from (x_0, y_0) to (x_1, y_1) in pixel space, its corresponding vanishing point angle θ is given as follows. Let c_w, c_h be the camera image's width and height, respectively, and let f_x, f_y denote the horizontal and vertical components of the field of view of the camera. The segment endpoints are first projected from image space to a plane P at unit distance from the camera as

$$\begin{aligned}
x_0^P &= \tan\left(\frac{f_x}{2}\right)\left(\frac{2x_0}{c_w} - 1\right),\\
y_0^P &= \tan\left(\frac{f_y}{2}\right)\left(1 - \frac{2y_0}{c_h}\right),\\
x_1^P &= \tan\left(\frac{f_x}{2}\right)\left(\frac{2x_1}{c_w} - 1\right),\\
y_1^P &= \tan\left(\frac{f_y}{2}\right)\left(1 - \frac{2y_1}{c_h}\right).
\end{aligned}$$

where the y direction has been flipped as per screen-space convention. Then, the horizontal angle to the vanishing point implied by the line segment is given by

$$\theta = \text{atan2}\left(\frac{x_0^P y_1^P - x_1^P y_0^P}{y_1^P - y_0^P}\right)$$

where $\text{atan}(\alpha/\beta)$ has been replaced with the otherwise equivalent $\text{atan2}(\alpha, \beta)$ for numeric stability when the denominator is close or equal to zero, which occurs when line segments appear horizontal.

3.4.4 Segment Labeling and Cost Function

After computing vanishing point angles, each observed line segment is assigned one of three labels A_1, A_2, O, the first two of which correspond to the two horizontal Manhattan grid axes and the third of which denotes outliers that are not well-aligned to either axis. In order to find the pair of vanishing points that is in agreement with as many observed line segments as possible, a cost function C is defined that is minimized when line segments are clustered to a pair of cohesive and orthogonal vanishing points. Outliers are assigned a fixed cost which is defined simply in terms of an angular threshold. Let N be the total number of observed line segments. Let the label of the i-th line segment be denoted as l_i where $i \in \{0, ..., N-1\}$ and $l_i \in \{A_1, A_2, O\}$, and let $\theta_{\min}$ denote the angular threshold beyond which an inlier would be preferentially considered to be an outlier. Let $d(\theta_i, \theta_j)$ be defined as the distance between angles modulo π, which is used to allow opposite angles to be considered equal. d may be defined as

$$d(\theta_i, \theta_j) = \pi \left| \frac{\theta_1 - \theta_2}{\pi} - \left\lfloor \frac{\theta_1 - \theta_2}{\pi} + 0.5 \right\rfloor \right|$$

which is minimized when θ_1 and θ_2 differ by a multiple of π, or in other words, are colinear. Note that $\pi/2 - d(\theta_1, \theta_2)$ analogously measures how orthogonal θ_1 and θ_2 are.

Considering two line segments i and j, their pairwise cost given as

$$c_{pair}(i,j) = \begin{cases} d(\theta_i, \theta_j) & \text{if } l_i = A_1 \text{ and } l_j = A_1 \\ \pi/2 - d(\theta_i, \theta_j) & \text{if } l_i = A_1 \text{ and } l_j = A_2 \\ \pi/2 - d(\theta_i, \theta_j) & \text{if } l_i = A_2 \text{ and } l_j = A_1 \\ d(\theta_i, \theta_j) & \text{if } l_i = A_2 \text{ and } l_j = A_2 \\ 0 & \text{otherwise} \end{cases}$$

which is minized when labels of the same vanishing point are parallel or when labels of opposite vanishing points are orthogonal. Outliers are accounted for by a separate function, defined as

$$c_{outlier}(i) = \begin{cases} \theta_{\min} & \text{if } l_i = O \\ 0 & \text{otherwise} \end{cases}.$$

In the implementation, $\theta_{\min}$ has been chosen to be $\pi/5$, which has the effect that segments with angles more than $\pi/5$ radians from the center of their associated vanishing point would yield a lower but fixed cost of they were labeled as outliers. Now, the full cost function can be defined as

$$C = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c_{pair}(i,j) + \sum_{i=0}^{N-1} c_{outlier}(i).$$

The computational cost of evaluating this function explicitly is quadratic, but for the purposes of simulated annealing, the change in cost ΔC due to a single label change suffices, and this requires only a single pass over the segment labels and can be computed in linear time.

3.4.5 Simulated Annealing Loop

To perform simulated annealing, each segment is assigned a random initial label. A maximum number of iterations $k_{\max}$ is chosen, which in the implementation has been set to $k_{\max} = 100N$. During the k-th iteration, a random segment is chosen and random new label is chosen for that segment. Given a linearly decreasing temperature $T = 1 - k/k_{\max}$, the probability P of accepting the new label is defined to be

$$P = \frac{\left(1 - T^3\right) e^{-\alpha \Delta C}}{1 + e^{-\alpha \Delta C}}$$

where α is a scaling constant, chosen in the implementation to be $\alpha = 7.5$. As the temperature tends towards zero, the probability P tends to zero if the cost increases, and to a positive value otherwise, as is required for simulated annealing to converge optimally. Finally, having computed P, a uniform random number $r \in [0, 1]$ is generated and if $r < P$, the new label is chosen, otherwise the old label is kept. This completes a single iteration of the simulated annealing loop.

When $k_{\max}$ iterations have been performed, the average viewing angle of the viewpoint with the most associated segments is returned. In the event that too few segments are associated with any vanishing point, an error status is returned instead. The minimum number of segments in the implementation has been chosen as 3.

3.5 Landmark Extraction

Line segments that are observed from several different poses are candidates for landmarks. Given the camera parameters and initial estimates of the robot positions, the 3D orientations of the landmarks may be recovered. However, in order to correctly associate and distinguish landmarks from one another, additional information is needed for correspondence finding. While it may feel appropriate to use image descriptors such as ORB [Rub+11] for this purpose, image descriptors are designed to determine locations of interest independently and thus cannot easily be applied at a desired location, such as that of an observed line segment. Lee *et al.* [LKC19] propose to simply use a square 11×11-pixel image patch taken from the visual center of the line segment, and to use such a patch extracted from every line segment observation to find correspondences. The same approach is adopted in this work, although it is far from ideal, as documented in the results section.

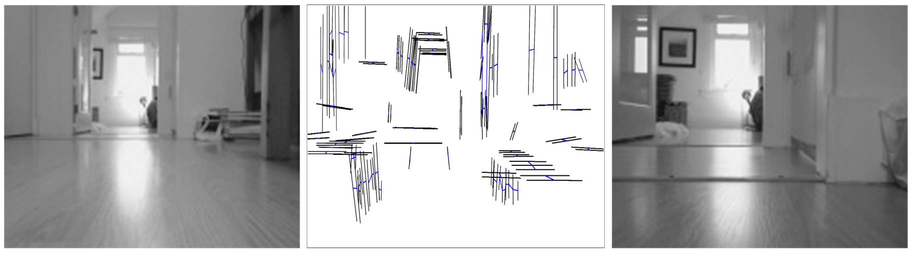

Figure 3: Relative motion of extracted landmarks between frames during continuous forward travel. On the left is the first in a series of camera images. In the center, the landmarks visible between subsequent frames are shown in black, and the relative motions of their centers are drawn in blue. On the right is the final camera image. Best viewed in colour.

3.6 Landmark Alignment and Robot Pose Correction

Lee *et al.* [LKC19] use the labels assigned to each line segment by their vanishing point estimation method to categorize landmarks as being either aligned to the vertical axis or to one of the two horizontal axes of the underlying Manhattan grid. This simplifies some of the math for projecting 3D line segment end points onto the camera plane. However, their equations serve to align landmark endpoints directly, rather than considering the colinearity of observed landmarks. This difference is important, as the line segments produced by LSD [Gio+12] are observed in practice to drift back and forth along their corresponding edges as well as change size between subsequent pictures of the same environment from very similar viewpoints. The effect of this is that the image patches selected at the line segment centers are taken from different locations, which disrupts correspondence finding. This is illustrated in Figure 3, which shows varying amounts of unwanted drift between subsequent landmark observations. In particular, note the jagged blue lines at the bottom left and bottom right as well as the very long pair of vertical blue lines near the top center, which indicates that landmark segments either drifted vertically by a great distance along an edge or were falsely identified. Only adjacent camera images are considered in this visualization for simplicity and because they are expected to have the most visual similarities.

While the difficulties of landmark correspondence as proposed undermine further efforts to perform landmark pose estimation and robot orientation correction, a technique for doing so assuming correct landmark correspondences is nonetheless proposed for the sake of completeness. The proposed technique seeks to maximize the visual alignment of observed landmarks while optimizing over landmark locations and estimated robot poses, and is thus a form of bundle adjustment. Unlike traditional bundle adjustment and the method of Lee *et al.* [LKC19], the proposed landmark alignment technique is designed to maximize the colinearity of line segments, rather than the displacement between their end points, in order to permit the drift and possible truncation of large line segments that are typically observed in the indoor setting. Evaluation of this technique and a robust method for finding line segment landmark correspondences is unfortunately deferred to future work.

3.6.1 Camera Projection

In order to relate landmark poses in 3D space to their observed 2D locations in image space at some estimated robot pose, the projection of a single 3D point onto the camera image is used as first step for further derivations.

Given the robot's estimated location and heading in the ground plain R_x, R_y, R_θ, and the location P_x, P_y, P_z of a point P in 3D space, we first translate and rotate P to place the robot at the origin, looking along the positive X direction, which yields a new point W^R in the robot's coordinate frame defined as

$$\begin{aligned}
W^R_x &= \cos(-R_\theta)\,(P_x - R_x)\ \ - \sin(-R_\theta)\,(P_y - R_y)\,,\\
W^R_y &= \sin(-R_\theta)\,(P_x - R_x)\ \ + \cos(-R_\theta)\,(P_y - R_y)\,,\\
W^R_z &= P_z.
\end{aligned}$$

A second translation places the robot's camera at the origin, assuming the camera is displaced C^R_x units forward and C^R_y units to the right in the robot's coordinate frame, which results in a 3D point in the camera's coordinate frame, given by

$$\begin{aligned}
W^C_x &= W^R_x - C^R_x,\\
W^C_y &= W^R_y - C^R_y,\\
W^C_z &= W^R_z.
\end{aligned}$$

Using similar triangles, the Y and Z components can be mapped to an imaging plane located at unit distance in front of the robot camera by dividing by their X component, which achieves a perspective projection. The case where the point being projected lies behind the camera is not considered, because the point is expected to belong to an observed line segment which, by construction, is in front of the camera. This point is then scaled according to the field of view and mapped to pixel space. The vertical component is inverted by convention. Note that the horizontal and vertical axes are given by $+Y$ and $+Z$ in the robot's coordinate frame and by $+X$ and $-Y$ in pixel space. Given horizontal and vertical fields of view f_x, f_y and pixel resolutions w, h, the new point V in pixel space is given by

$$
\begin{aligned}
V_x &= (w-1)\left(\frac{1 + W^C_y / W^C_x}{2\tan(f_x/2)}\right), \\
V_y &= (h-1)\left(\frac{1 + W^C_z / W^C_x}{2\tan(f_y/2)}\right).
\end{aligned}
$$

3.6.2 Projected Line Segment Alignment

Given the projection from a point P in world space to a point V in pixel space, a line segment in world space can be related to a line segment seen by the camera simply by projecting both endpoints P_0, P_1 of the estimated 3D position as described above to produce $\hat{V_0}, \hat{V_1}$. Given an observation of the same line segment at a particular pose, whose endpoints in pixel space are given by V_0, V_1, the positional error of the estimated segment with regard the observed segment is given by the distance of both $\hat{V_0}$ and $\hat{V_1}$ from the line passing through V_0 and V_1. This permits segments that are along the same line but which have drifted or been truncated, which is presumed to be a better model for the behaviour of line segments observed in practise. The line-distance error may defined as

$$
\begin{aligned}
d &= \left|\hat{V_0} - (V_0 + h_0(V_1 - V0))\right| + \left|\hat{V_1} - (V_0 + h_1(V_1 - V0))\right| \\
\text{where } h_0 &= (\hat{V_0} - V_0) \cdot (V_1 - V_0) \\
\text{and } h_1 &= (\hat{V_1} - V_0) \cdot (V_1 - V_0)
\end{aligned}
$$

where $|\cdots|$ denotes the Euclidean norm and $\cdot$ denotes the vector inner project. The above formulae for line segment projection and alignment were implemented in Python using the PyTorch automatic differentiation library [Pas+19]. Given a set of initial landmark positions, estimated robot poses, and landmark observations, a global error function is defined as a summation of line-distance errors over all landmark observations. This error term is differentiated with respect to both estimated robot poses and estimated landmark observations, and is optimized iteratively using the Adam optimizer [KB17]. Results are humbly presented in the next section.

4 Results and Discussion

Broadly speaking, the proposed algorithm for vanishing point estimation performed poorly and was not suitable for correcting robot poses. Additionally,

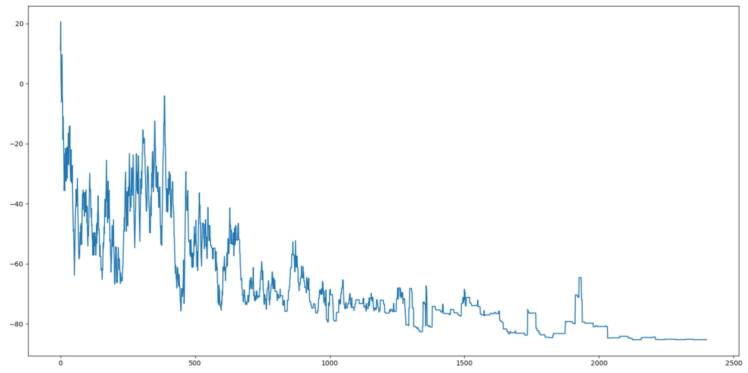

Figure 4: The evolution of the relative cost, in radians, over the course of a typical run of simulated-annealing-based vanishing point estimation

the proposed algorithm for jointly estimating landmark positions and correcting robot poses was not applied successfully, and this is believed to be due to the poor correspondence information provided by simple image tags that frequently move out of position.

4.1 Vanishing Point Estimation

After finding a suitable combination of hyper-parameters for the proposed vanishing point estimation algorithm, simulated annealing was successfully achieved and line segments were meaningfully clustered into vanishing point directions. As demonstration of the convergence of the simulated annealing algorithm, the change over time in the total relative cost starting at 0 for a typical input is shown in Figure 4.

Example line segment labels yielded by the algorithm are illustrated in Figure 5. Note that while the algorithm generally performs intuitively, some negative results are also observed, particularly when the contours of a cat or patterns on a draped blanket are falsely considered to be part of the Manhattan grid.

However, as promising as some of the visualized clusterings may appear the Manhattan grid angles produced by the vanishing point are far less reliable than dead reckoning, and have a very large variance which renders them unsuitable for robot orientation correction. The high variance and failure rate of the vanishing point estimation are illustrated in Figure 6. Curiously, the algorithm appears to produce more reliable estimates that agree with odometry while the robot is rotating. One explanation for this is that the robot was preferentially driven parallel to the building structure, during which fewer line segments are observed, and those segments that are seen are likely at oblique angles, but during a turn, the robot is oriented against the Manhattan grid and is likely to see line segments

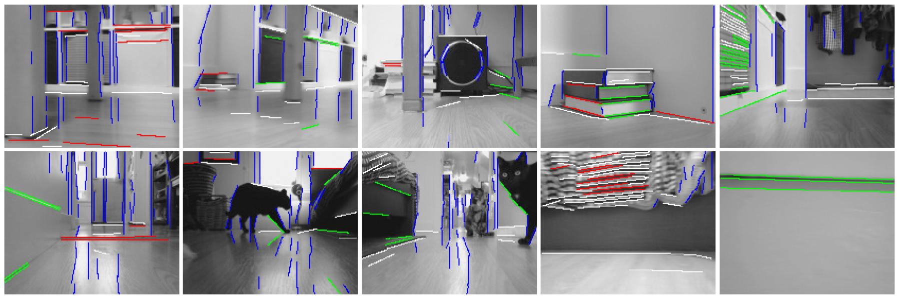

Figure 5: A selection of labeled line segments produced from the acquired dataset. Blue segments are vertical and are excluded from processing. Green and red segments point to the estimated orthogonal vanishing point directions. White segments were labelled as outliers. Best viewed in colour.

from both horizontal axes.

Over the course of the entire round trip, during which 493 camera images were captured, the average error between the estimated angle due to dead reckoning and that of detected vanishing point directions is 0.1232 radians or about 7 degrees. The algorithm explicitly failed to find a vanishing point direction a total of 42 times.

The poor performance is likely most attributable to the number of bold assumptions that were made about the camera and the robot's surroundings. Possibly the most suspicious assumption is that the horizon always appears at the exact vertical center of the camera plane, which may be intuitive for slanted line segments but which can lead to degenerate cases when nearly-horizontal line segments are observed close to the center of the image. A better understanding of the related techniques in vanishing point estimation would help to find improvements for the proposed algorithm, or alternatively, a reference implementation for those techniques, if one were made available, could be adapted to allow the rest of the SLAM system to be evaluated more meaningfully.

4.2 Landmark Position Estimation and Robot Pose Correction

While a complete software implementation of the proposed method for jointly optimizing landmark locations and correcting robot poses was produced and tested, a successful application of the approach resulting in realistic landmark positions and improved robot trajectories has yet to be observed. As previously mentioned, it is believed that the image patches from the centers of observed line segments are too unreliable for robust landscape correspondence finding, which is an important requirement for position estimation and correction.

As an illustration, Figure 7 shows the result of optimizing landmark locations and robot poses. In this experiment, the Euclidean norm was chosen for

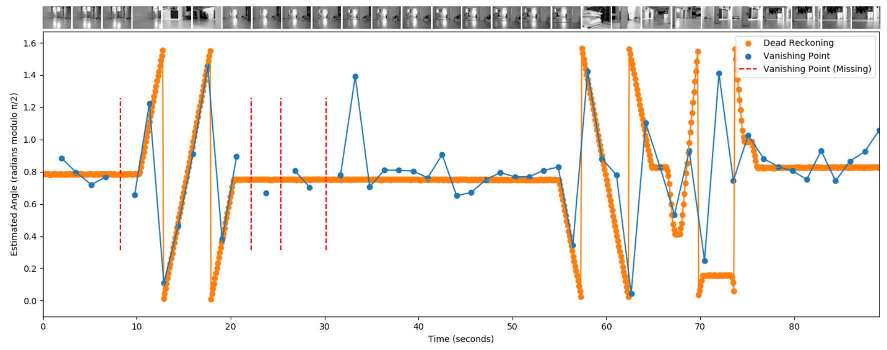

Figure 6: Comparison of estimated angles obtained by dead reckoning (shown in orange) and by vanishing point estimation (shown in blue, or in dashed red in case of explicit failure). Angles are shown modulo $\pi/2$ such that when the robot makes a half turn, the angle wraps around twice. Synchronized camera images are displayed on top in miniature. Best viewed in colour.

correspondence matching (as used in image feature descriptors, and because Lee *et al.* [LKC19] do not specify otherwise). A threshold value is chosen conservatively based on observed statistics. A sub-sequence of the dataset containing 79 images was taken, from which a total of 27 unique landmarks were extracted and 58 total landmark observations were recorded. Over the course of optimizations, the landmarks did not move from their default locations even though the optimizer was able to bring the error to close to zero. To rule out mistakes in the line-distance error formulation, the same experiment was performed using the end-point distance error as used by Lee *et al.* [LKC19], which resulted in a single landmark being brought to rest in a horizontal configuration while other landmarks similarly did not move. Different thresholds for image tag matching were used with no better results. It is thus believed that given the most accurate landmark correspondences by this method, there simply are not enough observations of the same landmarks from different angles, and that the resulting error metric thus cannot fully constrain the landmark locations in 3D space.

Naturally, a programmer error in the implementation cannot simply be ruled out. As a test, a fully-connected cube of line segments was constructed and projected onto the camera plane using the differentiable perspective camera model. The result is illustrated in Figure 8, which appears to be perfectly reasonable.

The failure of landmark position optimization is attributed to the poor landmark correspondence given by image patches, which did not identify enough observations of the same landmarks for meaningful pose correction. Given the rich literature of image feature descriptors, an alternative line segment descriptor could be sought which is invariant or at least robust to changes in perspective and translations along the line segment's edge. Presently, the perturbations in line segments given by Line Segment Detector [Gio+12] between different

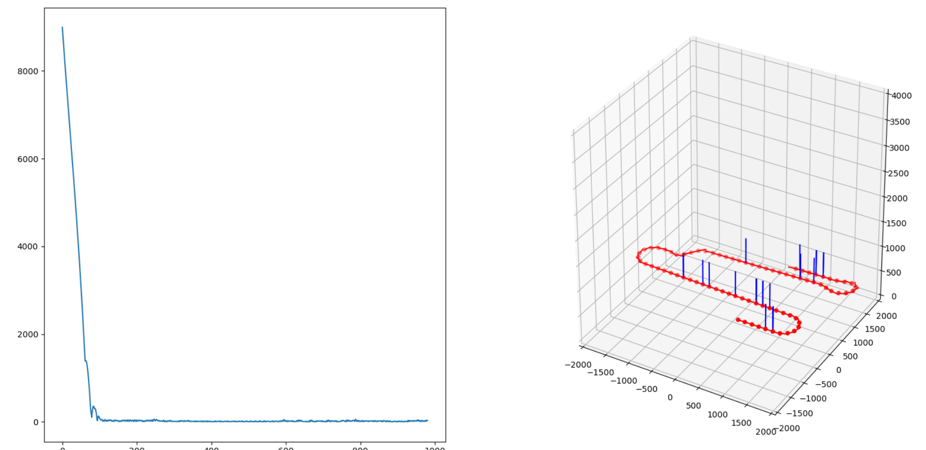

Figure 7: Landmark position optimization and robot pose correction. On the left is the evolution of the total line-distance error, and on the right is the resulting robot poses (in red) with the estimated 3D landmark positions (in blue). The landmarks have not moved perceptibly from their initial guesses. Best viewed in colour.

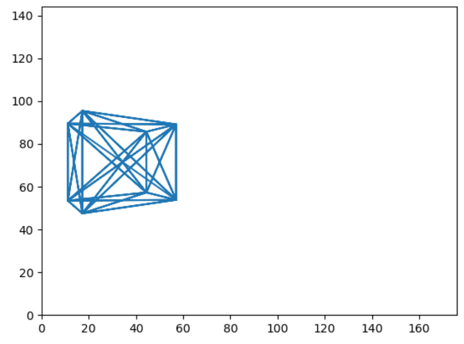

Figure 8: A cube of line segments as seen through the differentiable perspective camera model.

camera images is a major problem for line segment landmark correspondence finding as proposed by Lee *et al.* [LKC19] which undermines further successes in implementing SLAM using line segment landmarks.

5 Conclusion

A simple robot with a monocular camera was constructed and a dataset of indoor imagery was gathered. Line segments were extracted and used to estimate Manhattan grid directions and to establish landmarks. The proposed alternative algorithm for vanishing point estimation proved to be unreliable, and further work is needed for reliable correspondence finding between line segments. Despite these shortcomings, the proposed methods are presented for discussion, the implementation is made available for future explorations, and a number of distinct weaknesses in line-segment-based SLAM as proposed have been identified which, if addressed, may result in a robust and effective SLAM system.

6 Acknowledgements

I would like to thank Grompone von Gioi *et al.* for providing a well-documented open-source implementation of Line Segment Detector [Gio+12]. I additionally would like to thank my partner Dinah for gifting me the EV3™ micro-controller that was used in this work.

References

[FB81] Martin A. Fischler and Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[Rut89] R. A. Rutenbar. “Simulated annealing algorithms: an overview”. In: *IEEE Circuits and Devices Magazine* 5.1 (1989), pp. 19–26. DOI: 10.1109/101.17235.

[Tri+99] Bill Triggs et al. “Bundle Adjustment - A Modern Synthesis”. In: *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*. ICCV ’99. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 298–372. ISBN: 3540679731.

[Low04] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33833-8.

[Cal+10] Michael Calonder et al. “BRIEF: Binary Robust Independent Elementary Features”. In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15561-1.

[Rub+11] E. Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[Baz+12] J. Bazin et al. “Globally optimal line clustering and vanishing point estimation in Manhattan world”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 638–645.

[Gio+12] Rafael Gioi et al. “LSD: A line segment detector”. In: *Image Processing On Line* 2 (Mar. 2012), pp. 35–55. DOI: 10.5201/ipol.2012.gjmr-lsd.

[MMT15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[Zha+16] Lilian Zhang et al. “Vanishing Point Estimation and Line Classification in a Manhattan World with a Unifying Camera Model”. In: *Int. J. Comput. Vision* 117.2 (Apr. 2016), pp. 111–130. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0854-5. URL: https://doi.org/10.1007/s11263-015-0854-5.

[KPS17] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. *Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images*. 2017. arXiv: 1710.02726 [cs.CV].

[KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[LKC19] T. Lee, C. Kim, and D. D. Cho. “A Monocular Vision Sensor-Based Efficient SLAM Method for Indoor Service Robots”. In: *IEEE Transactions on Industrial Electronics* 66.1 (2019), pp. 318–328. DOI: 10.1109/TIE.2018.2826471.

[Pas+19] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019, pp. 8026–8037. URL: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[ev3] ev3dev.org. *ev3dev.org*. https://ev3dev.org/. Accessed: 2020-10-15.