Mobile Real-Time Single Image 3D Corridor Reconstruction Using J-Linkage

Greg Olmschenk[1] Zhigang Zhu[2]
The Graduate Center and the City College
Of The City University of New York
[1]golmschenk@gradcenter.cuny.edu [2]zhu@cs.ccny.cuny.edu

Abstract

We present a real-time algorithm that reconstructs 3D models of corridors using a mobile device. Contrary to previous approaches, our approach uses a non-iterative, simultaneous model reconstruction method called J-Linkage, which is both accurate in parameter estimation and efficient in computation. We first use J-Linkage to find the vanishing points in a scene, and then show how the detected vanishing points along with corresponding straight line edges can be used to accurately determine the important features of a corridor, such as doors. Finally, we show how this approach has advantages in both speed and accuracy when compared with previous solutions. Implementation in a mobile computing device is carried out and experimental results are provided. The ability to model 3D representations of corridors in real-time on a mobile device can be used in a range of applications including mobile apps to help the visually impaired navigate, robotic navigation, building inspection, and virtual tours.

1 Introduction

In the world built by humans, corridors are everywhere. As such, extracting corridor information is an important step toward understanding many common environments. Furthermore, mobile devices have become ubiquitous. The ability to model 3D representations of corridors in real-time on a mobile device makes possible applications ranging from mobile apps to help the visually impaired navigate [1] to robots understanding their environment on the fly [2] to building inspections [3].

The most common way to extract the information about a corridor in computer vision is to use a range sensor and perform the computations necessary to find the walls of the corridor. These methods require relatively expensive equipment, heavy processing, and high energy consumption. This makes the method impractical for a robot that needs to move quickly, a wearable system that needs to be built cheaply, or a visually impaired person who does not want to carry a laptop on their back processing data all day. The most recent relatively low-cost 3D sensors are the RGB-D sensors, such as Microsoft Kinect and Asus Xtion Pro, however their sensing ranges are quite limited (0.5 to 4 meters) and are not particularly suitable for corridor detection. We propose a high speed, low resource, long range, inexpensive solution to these needs. Effective portable corridor detection capabilities with nothing more than a single consumer camera such as a webcam or the cameras found on a smartphone.

The method proposed in this paper uses a variation of the non-iterative J-Linkage solution. In other solutions, such as with Hough transformations, the accuracy is limited by the parameterized of the voting space and higher accuracies directly translate to more cumbersome computation. Our solution does not use a parameterized voting space and provides a fast and accurate algorithm.

We have made the following three contributions in this work. First, we leverage the J-Linkage approach to be used in the detection and reconstruction of corridors. To our knowledge, our algorithm is the first such J-Linkage based corridor detection. Second, we provide a method in which the vanishing points and corresponding straight line edges can be used to create a 3D reconstruction of the corridor. Finally, we implement the above methods as an iOS application to provide a mobile real-time algorithm for corridor reconstruction from a single image.

2 Related Work

Past research has been done on allowing a robot to quickly move through a corridor [4]. However, these methods rely on existing 3D maps of the corridors which the robot matches its currently surroundings with. The method described in this paper expands on this by requiring no previous knowledge. Other fast corridor detection options include those with powerful laser range sensors [2] leading to cumbersome and expensive equipment being required.

The most detailed indoor modeling methods use point cloud data gathered by 3D range sensors [5]. This 3D data can often be processed more quickly by reducing the point clouds to surfaces [6]. Other range methods use 2D laser range sensors to provide a faster, less resource intensive approach [7]. These range sensor methods use powerful hardware to extract precise position data without the need to interpret information based on a RGB image.

The use of a single RGB camera to detect doors has been used extensively in existing research [8]. The means used to find the frames of the doors in these methods is similar to the extraction of corridors in our approach.

Detection of corridor vanishing points to determine the direction of the hallway has been extensively studied [9]. We build on these methods by utilizing and creating a model of the hallway and its features such as doors, turns, and corners of the hallways. Moreover we offer a new method in vanishing point detection based on a relatively recent approach called J-Linkage [10]. The J-Linkage approach is used to find multiple instances of a model, and neither prior knowledge to the number of models nor iterating across a parameter space is needed.

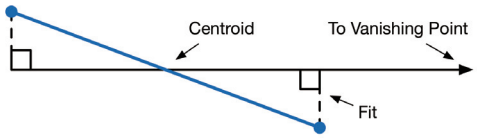

Figure 1. The approximation used for estimating an edge's preference set.

Specifically, we use the J-Linkage approach to determine vanishing points and the corresponding perspective features in the corridor scene. Our approach for finding the vanishing points and corresponding straight line edges is similar to that described in [11], but we offer a real-time implementation on a mobile device.

3 Approach

In the following, we will describe the steps we use for corridor modeling: vanishing point detection, corridor feature detection, and their real-time implementation in a mobile device. The 3D estimation and corridor modeling is also described. For brevity, the initial detection of 1-pixel wide straight line segments will not be explained in detail, but the following is a summary. Edge pixels are found using a standard Canny edge detection followed by a non-maximum suppression resulting in an edge map with edges 1 pixel thick. This map then undergoes a contour extraction process to determine connected components. We walk along each contour and continuous sets of edge pixels on a contour form candidates for straight line edges.

3.1 Vanishing Points From J-Linkage

Given a set of straight edges, we begin by generating a set of possible mathematical models for vanishing points. Specifically, two semi-randomly chosen straight edges are used to generate a vanishing point model. An edge $\boldsymbol{e}_i$ is randomly selected as the first edge, then the second edge $\boldsymbol{e}_j$ will be selected based on proximity to the first with the probability given by:

$$P(\boldsymbol{e}_i|\boldsymbol{e}_j) = \begin{cases} \frac{1}{Z} exp - \frac{\|\boldsymbol{e}_j - \boldsymbol{e}_i\|^2}{\sigma^2}, & \text{if } \boldsymbol{e}_i \neq \boldsymbol{e}_j \\ 0, & \text{if } \boldsymbol{e}_i = \boldsymbol{e}_j \end{cases} \quad (1)$$

where Z is a normalization constant and σ is heuristically determined. The vanishing point model is determined by calculating where the two edges would intersect if they were infinite lines. A large number of vanishing point models are generated in this way (in our experiments, it was 500 models).

Next, the conceptual space is initialized. Every vanishing point model is compared with every straight edge to test if the vanishing point is in the preference set of that edge. Specifically, a line is considered which runs through the centroid of the edge to the vanishing point as is shown in Figure 1. The vanishing point is said to be in the edge's preference set if:

$$d < l \cdot \epsilon_f \quad (2)$$

where d is the distance from the edge's end point to the line, l is the length of the edge, and ϵ_f is a threshold constant. The preferences of each edge is then stored in a binary matrix indexed by X and Y, with the Y-axis representing the edges and the X-axis representing the vanishing point models where a 1 corresponds with a edge preference or a 0 for an edge non-preference.

Each row in the conceptual space is now taken to be a cluster of edges, all of which are initially singletons. The clusters are then merged based on their Jaccard distance, given by:

$$d_J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \quad (3)$$

where an element is in a set if it is in the preference set of the cluster. The two clusters which are closest via their Jaccard distance will be merged and the resulting cluster will be given a preference set that is the intersection of the original two clusters. This process is continued until all clusters have disjoint preference sets (i.e. the Jaccard distance is 1).

After the merging, there will be many clusters which contain only a few edges, corresponding to outliers, and a few clusters which contain many edges, corresponding to real vanishing points. These vanishing points are then calculated based on the best fit point corresponding to the edges in the related cluster. Tardif [11] sums maximal edge end point distances to determine the best fit vanishing point for a given cluster of edges. For efficiency in mobile device computation, we instead preform a RANSAC computation gathering a number of vanishing point models proportional to the number edges in the cluster. The models are then averaged to determine the vanishing point for the cluster.

3.2 Manhattan Direction Vanishing Points

Features in the 3 Manhattan directions are strongly represented and will be among the larger vanishing point clusters. Denoting K as the 3×3 intrinsic camera parameter matrix and ω as the image of absolute conic given by $K^{-T}K-1$, a commonly used approach for measuring K is:

$$\boldsymbol{v}_1^T \omega \boldsymbol{v}_2 = 0 \quad (4)$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are orthogonal vanishing points. However, when K is known, this equation can instead be used to find orthogonal vanishing points. From this we can determine the 3 Manhattan direction vanishing points: edges running in the corridor direction, vertical edges (such as door frames and other wall features), and edges perpendicular to both (such as tiles on the floor or light panels on the ceiling). The use of them will be described in more detail below.

3.3 Corridor Features

We use the vanishing points obtained using J-Linkage and corresponding straight edges we have obtained to determine the important features of the hallway. The Manhattan direction vanishing point which is closest to the center of the image is taken to be the vanishing point whose parallel lines run along the axis of the corridor. This assumption should always be true given an image that actually presents itself as being of a corridor.

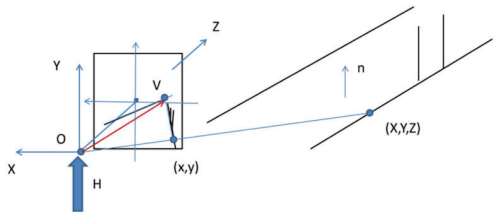

Figure 2. The geometry of the 3D modeling

We begin by distinguishing between the floor, walls, and ceiling in the image. The transitions between surfaces are prominent (usually among the most prominent) edges. However, they can not be recognized by their prominence alone. Therefore, we use the wall and floor/ceiling features to make the distinction. Specifically, wall features will tend to run straight up and down and will correspond to another of the 3 Manhattan-direction vanishing points. A similar case exists for the floor and ceiling features (such as floor tiles and light panels on the ceiling). Using these features, we can determine which of the straight edges corresponding to the corridor axis most accurately represents the end points of the wall and floor/ceiling features. Formally, each edge is voted on as the transition edge with a fitness given by:

$$F_{\boldsymbol{e}_i} = \sum_j \begin{cases} d_j \cdot w_j, & \text{if } p_j \text{ is surface side} \\ d_j \cdot w_j \cdot \gamma, & \text{if } p_j \text{ is counter-surface side} \end{cases} \tag{5}$$

where d_j is the distance from the point to the edge line being fitted, γ is a weighting constant, and the minimal fit is the accepted transition edge.

With the main surfaces defined, we determine the position of interesting features. Most obviously, door frames are both recognizable and user relevant. They show up as properly spaced wall features running along the wall's Manhattan direction axis.

With all the detected corridor features, we are able to obtain their 3D information using well established methods [12] based on some reasonable assumptions, such as the known camera height H (Figure 2). From the vanishing point V of the corridor, the plane equation of the floor and the direction of the corridor can be determined, and therefore the 3D coordinates (X, Y, Z) of any point on the floor can be calculated. With this 3D information, local 3D models of separate views can be integrated into a global corridor model.

4 Mobile Implementation

In our experiments, we used an iPhone 5s implementation developed in the Objective-C programming language. In the following, we will analyze the time complexity of each step, and provide some implementing details to make the entire system run in real-time.

In the edge detection step, with n_i and m_i denoting the image pixel dimensions, the Canny edge detection uses $O(n_i m_i log(n_i m_i))$ time. Luckily, this process uses OpenCV's highly optimized image processing libraries. The straight line extraction takes $O(n_c)$ time where n_c is the number of edge pixels. n_c is typically only a tiny fraction of $n_i \cdot m_i$ and so this processing takes very little time.

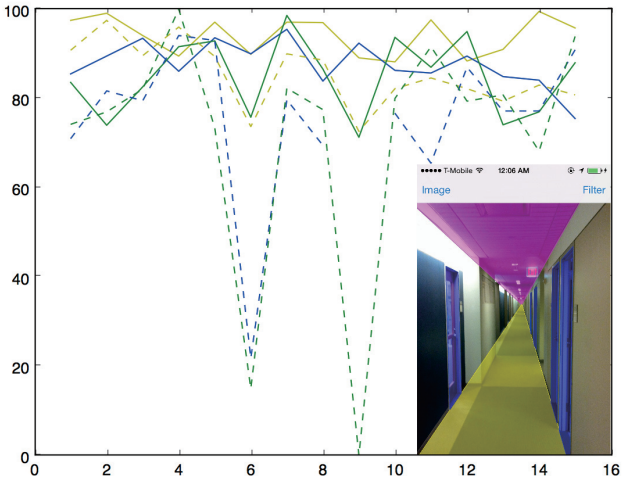

Figure 3. Comparison of algorithm results with ground truth data. Solid lines used trained images, dashed lines used untrained images. Yellow is the % of surface pixels correctly labeled, blue is the % of the true positive door pixels, and green is the % of the true negative door pixels

In the vanishing point detection and selection step, the J-Linkage clustering requires $O(v_m e_n^2)$, where v_m is the number of generated vanishing point models and e_n is the number of straight line edges. This is the most computational intensive part of the program taking up approximately 3/4 of the running time.

Determining which of the lines leading to the vanishing point in the direction of the corridor (i.e. parallel to the corridor axis), l_c, are the transition lines from the floor to the walls and from the walls to the ceiling requires that each of the wall, ceiling, and floor feature edges (i.e. parallel to the 2 remaining Manhattan direction vanishing points), e_f, give their weighted vote on each of them resulting in $O(l_c e_f)$. Both the number of lines (l_c) and number of features (e_f) are relatively low (usually less than 100), so this process is fast.

The generation of a 3D model requires that each wall feature edge of interest (with interest usually meaning a strong edge) terminating at the ground plane, e_w as the total number, have its position along the corridor calculated. This calculation then occurs in $O(e_w)$ time.

In the corridor feature detection step, the processing time takes $O(e_m log e_m)$ time where e_m is the number of feature edges attributed to the currently being considered Manhattan direction vanishing point. This comes from the need to sort and then consider the edges in order. However, this process is very fast as the number of feature edges for a given Manhattan direction is small.

5 Experiments and Future Work

We have implemented the core algorithm of the corridor modeling on an iPhone 5s. In our current implementation, the average processing time on a 640x480 image is 0.18 second.

A genetic algorithm was used to train the detection algorithm's constants, such as the thresholds and conceptual space model limits. The training used manually annotated images as ground truth for comparison.

Figure 4. Results on trained images.

Figure 5. Results on untrained images. The last two show cases particularly difficult for our algorithm.

The comparison with the ground truth for trained and untrained images is shown in Figure 3. A snapshot of the iPhone implementation is inlaid in Figure 3, where the highlighted areas have been determined by the algorithm with purple labeling the ceiling, yellow the floor, and blue being doorways. Figure 4 shows the 3D measurement results for some trained cases while Figure 5 shows untrained cases. The results show that different transition regions are regularly accurate. The doors detected are only slightly less accurate. In particular, while many doors may be missed, the doors that are detected are usually reliable. This is the case for both the trained and untrained images. For the 15 trained images, the average percentage of surface pixels that are correctly labeled is 94.1% and, on average, 87.7% of doorway pixels are correctly detected, whereas only 15.2% of the pixels are incorrectly labeled as doorway pixels. In comparison, the percentage of surface pixel labeling for the untrained images decreases by about 8.8% on average, and due to the missing of some doors, the percentage of true door pixels drops 16.4%, but 71.1% of the labeled door pixels are still correct.

The last two of the images shown in Figure 5 are specifically displayed because they show the areas where the algorithm is weak. In one, the wall features are such that they cause the algorithm to assume a much higher floor than is real and end up being consistently considered as doors even when they are not. The other difficult image shows an image where a very large portion of the corridor is occluded by people. Nevertheless, the directions of the corridors are correctly detected, and in the latter case, an estimation of a narrower corridor because of the multiple people on both sides is a more safe estimation.

Future work will include using multiple 3D models generated from our algorithm in succession to increase the accuracy of the algorithms predictions and provide additional information. We would also like to explore the effectiveness of different feedback methods on a mobile device in relaying the information here to a visually impaired user.

6 Acknowledgment

This work has been supported by the National Science Foundation (Award # EFRI-1137172) and PSC/CUNY Cycle 44 Research Program (Award # 66786-00 44).

References

[1] R. Manduchi and J. Coughlan. (computer) vision without sight. *Commun. ACM*, 55:96–104, 2012.

[2] J. Forsberg, et al. Mobile robot navigation using the range-weighted hough transform. *Robotics Automation Magazine*, 2:18–26, 1995.

[3] A. Paterson, et al. Building inspection: can computer vision help? *Automation in Construction*, 7:13–20, 1997.

[4] A. Kosaka and J. Pan. Purdue experiments in model-based vision for hallway navigation. *IROS*, 1995.

[5] H. Du, et al. Interactive 3d modeling of indoor environments with a consumer depth camera. *UbiComp*, pp 75-84, 2011.

[6] M. M. Nevado, et al. Obtaining 3d models of indoor environments with a mobile robot by estimating local surface directions. *Robotics and Autonomous Systems*, 48:131–143, 2004.

[7] J. Larsson, et al. Laser based corridor detection for reactive navigation. *Industrial Robot*, 35:69–70, 2008.

[8] Y. Tian, et al. Computer vision-based door detection for accessibility of unfamiliar environments to blind persons. *Computers Helping People with Special Needs*, pp 263-270, 2010.

[9] R. Ebrahimpour, et al. Vanishing point detection in corridors: using hough transform and k-means clustering. *IET Computer Vision*, 6:40–51, 2012.

[10] R. Toldo and A. Fusiello. Robust multiple structures estimation with j-linkage. *ECCV*, 5302:537–547, 2008.

[11] J.-P. Tardif. Non-iterative approach for fast and accurate vanishing point detection. *ICCV*, pp 1250-1257, 2009.

[12] A. Criminisi, et al. Single view metrology. *International Journal of Computer Vision*, 40:123–148, 2000.