Single-View 3D Object Shape Estimation

Yiming QIAN

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Graduate Program in
Computer Science
York University
Toronto, Ontario

November 2020

Abstract

An accurate single-view 3D reconstruction algorithm would allow researchers to develop better systems in robotics and VR/AR applications. However, single-view 3D reconstruction is an ill-posed problem and thus solvable only for scenes that satisfy strong regularity conditions. Features such as texture variations, haze, colour, shading, known object size and occlusion can provide information about 3D structures. Further, the built environment has repeated patterns and orthogonal straight lines that contain structured information with strong regularities. In this report, I will introduce the research I have done for my PhD on the single-view reconstruction problem in two parts. Part 1 will detail the three research projects I have completed on single-view 3D reconstruction of the built environment: 1) line segment detection [4], 2) single-view geometry-driven road segmentation [5], and 3) single-view 3D reconstruction of Manhattan buildings [158]. Part 2 will detail the research I have done on reconstructing the 3D shape of more general objects from their bounding contours.

Acknowledgement

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor James H. Elder, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like to thank Karen Englander and Laura Muntean, for their valuable help on proofreading my dissertation.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, I could not have completed this dissertation without the support of my friends who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Table of Contents

List of Tables

List of Figures

A List of Acronyms

3DBM	3D Manhattan Building Dataset
CRF	Conditional Random Field
CNN	Convolutional Neural Network
DNN	Deep Neural Network
DORN	Deep Ordinal Regression Network
FCRN	Fully Convolutional Residual Network
FOV	Field of View
LS3D	Line-Segment-to-3D
LSD	Line Segment Detector
MAE	Mean Absolute Error
MAP	Maximum A Posteriori
MCMLSD	Markov Chain Marginal Line Segment Detector
MT	Manhattan Tree
PPHT	Progressive Probabilistic Hough Transform
RMS	Rooted Mean Square
ROI	Region of Interest
SIRFS	Shape, Illumination and Reflectance from Shading
SSWMS	Slice Sampling Weighted Mean Shift
SVM	Support Vector Machine
VP	Vanishing Point

Chapter 1

Introduction

When you are wandering around an art gallery, you can see beautiful landscapes that are created from brushes and paint on a canvas. For example you can see in Figure 1.1 mountains in the background, and a farm field that connects you to the mountain range. Tall trees grow on the side of the field. It is so peaceful in the arms of nature away from the city. You may lose yourself in thought while admiring the painting and imagine yourself walking around in this beautiful landscape.

Have you ever thought about how you perceive that the mountains are far away in such a simple 2D painting?

Centuries of art work (some perspective painting examples are shown in Figure 1.2) reveal that human minds have the ability to see things in 3D from a 2D image.

Inspired by human vision, computer scientists have started to explore the possibilities of reconstructing a 3D world from a single 2D image. Single-view reconstruction has significant cost/range advantages over direct depth sensors such as LiDAR, Kinect or stereo systems. It has a wide range of applications such as:

1. Self-driving cars
2. Unmanned aerial vehicles
3. Driver assistance systems
4. Road surveys

Figure 1.1: View from Robin Hood's Hut, Somerset (A3), drawing pen, water wash, Conte crayons and pencils by Sara Waterer [173].

5. Building surveys
6. VR/AR
7. 3D movie production
8. Assistive devices for the visually impaired
9. Indoor/outdoor surveillance

My PhD thesis consists of two parts. The first part focuses on the 3D reconstruction of rectilinear, piecewise planar surfaces in a built environment. The second part focuses on 3D reconstruction of more general shapes from their bounding contours. I will detail my specific contributions to each project in subsequent chapters.

In the first part, I have completed three projects: 1) MCMLSD line segment detection [4, 48], 2) single-view geometry driven road segmentation [27, 5, 157], and 3) single-view 3D reconstruction of Manhattan buildings [158].

Figure 1.2: (Above, left to right) Renaissance examples of perspective, including Leonardo da Vinci's "The Last Supper"(1494) and Pietro Perugino's "Delivery of the Keys"(1482). (Below, left to right) More recent examples including Andrew Wyeth's "Wind from the Sea"(1947), Camille Pissarro's "Avenue de L'Opera, Paris"(1898), and Mary Cassatt's "The Child's Bath"(1893).

The first project is titled "*MCMLSD: A Dynamic Programming Approach to Line Segment Detection*". In this project, I was involved in the development of a line segment detection algorithm that provided the foundation for later projects. Projection to the image distorts properties such as angles, distance, and the ratios of distance. However, one property that is preserved during the projective transformation is straightness. The straight line in 3D is always straight in 2D. Since the 3D built environment contains many straight lines, detection of the projections of these lines in the image is an important early step in 3D reconstruction.

The second and third projects are applications that took advantage of this line segment detector. The second project is titled "*Single-View Geometry-Driven Road Segmentation*", uses line segments to estimate the road vanishing point and horizon line.

The third project is titled "*LS3D: Single-view 3D Reconstruction of Manhattan Buildings*". In the built environment, 3D scenes are often dominated by lines in three mutually orthogonal directions [33]. This regularity can be used to transform the 2D line segments

into a 3D CAD model from a single image.

In the second part of my thesis, I have completed a project titled "3D Object Rim Reconstruction from 2D Occluding Contour". Surface cues such as shading and texture provide local constraints on shape, but recovery of global object pose and depth is notoriously difficult. The visual boundary of the object provides a potentially important constraint on this global computation. While the image contour bounding the projection of the 3D object is planar, the back-projection of this contour to the surface of the object (the 3D 'rim') is a more general space curve - non-planar and oblique to the line of sight. Here we explore a novel method for estimating the 3D rim given only the 2D bounding contour of the object.

A list of related publications:

- G. Cheng, Y. Wang, **Y. Qian**, and J. H. Elder. Geometry-guided adaptation for road segmentation. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pages 46–53, 2020

- **Y. Qian**, S. Ramalingham, and J. Elder. LS3D: Single-view Gestalt 3D surface reconstruction from manhattan line segments. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 399–416, 2018

- E. J. Almazàn, R. Tal, **Y. Qian**, and J. H. Elder. MCMLSD: A dynamic programming approach to line segment detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5854–5862, July 2017

- J. H. Elder, E. J. Almazàn, **Y. Qian**, and R. Tal. MCMLSD: A probabilistic algorithm and evaluation framework for line segment detection, 2020 (In preparation)

- G. Cheng, **Qian, Y.**, and J. H. Elder. Fusing geometry and appearance for road segmentation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 166–173, 2017

- E. J. Almazan, **Qian, Y.**, and J. H. Elder. Road segmentation for classification of road weather conditions. In G. Hua and H. Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 96–108, Cham, 2016. Springer International Publishing

- **Qian, Y.**, E. J. Almazan, and J. H. Elder. Evaluating features and classifiers for road weather condition analysis. In *International Conference on Image Processing (ICIP), 2016 IEEE*. IEEE, 2016

Part I

Single-View 3D Estimation of Piecewise-Planar Shapes

Chapter 2

Literature Review

2.1 Introduction

To reconstruct 3D representations, there are two popular approaches: geometric methods and machine learning methods. Geometric methods study the relationship between lines, junctions and surfaces in an image, then use such relations to model 3D scenes through, for example, a line model [130], box model [70, 169, 135, 131], or Photo Pop-up [72, 73, 66, 124, 10]. The line model consists only of line segments. The box model targets simple indoor room scene environments in which a maximum of 5 major planar surfaces (typically floor, ceiling and three walls) are visible to the observer. The Photo Pop-up model categorizes image pixels into simple geometric classes (eg. horizontal, vertical) to form a Photo Pop-up visualization. Machine learning methods learn statistical models between the 2D image and 3D world to produce a range map [140, 141, 143, 107, 102, 171, 178, 46, 45, 26, 57, 187], or CAD model [78] representation. This latter approach is used by the real estate industry to build virtual show rooms [78].

Different types of single-view 3D reconstruction algorithms are built for different purposes. The line and box models are designed to reconstruct man-made environments. The Photo Pop-up model is more general and can be applied to man-made as well as natural scenes. Machine learning methods generate a range map. Unlike geometric methods machine learning methods, which are used primarily for the built environment, can be applied to any environment as long as there is sufficient data to train the system.

This chapter is organized as follows. In section 2.2, I will review the fundamental background knowledge of geometry required for single-view 3D reconstruction algorithms. In section 2.3, I will present a list of 3D datasets that can be used for 3D reconstruction training and evaluation. Finally, in sections 2.4 and 2.5, I will review the single-view 3D reconstruction algorithms that use geometric methods and machine learning methods.

2.2 The Geometry of Single-View 3D Reconstruction

An image captured by a camera is a two-dimensional representation of the three-dimensional world. In this section, I will first review the basic pinhole camera model [67]. Then I will review 3D world assumptions which may be imposed to allow for reconstruction of the 3D world from a single image.

2.2.1 Camera Model

The pinhole camera model (Figure 2.1) maps a 3D point C in the world to a 2D point c in the image via a 3×4 projection matrix P as:

$$c = PX = K[R|t]C \tag{2.1}$$

where c and C are homogeneous vectors in the form: $c = (x, y, w)^\intercal$, $C = (X, Y, Z, W)^\intercal$. P is the projection matrix, which can be decomposed into an intrinsic matrix K, a rotation matrix $R = R_x(\alpha)R_y(\beta)R_z(\gamma)$ and a translation vector $t = [v_x, v_y, v_z]^T$, where

$$K = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & u_0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix} \quad R_y(\beta) = \begin{bmatrix} cos\beta & 0 & -sin\beta \\ 0 & 1 & 0 \\ sin\beta & 0 & cos\beta \end{bmatrix} \quad R_z(\gamma) = \begin{bmatrix} cos\gamma & sin\gamma & 0 \\ -sin\gamma & cos\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.3}$$

Here f is the focal length, and k_u, k_v are scale factors relating to pixel size. $R_x(\alpha)$, $R_y(\beta)$, $R_z(\gamma)$ are the rotation matrices that rotate around x, y and z axis by α, β and γ angle. The principle point $[u_0, v_0]$ is the location of the camera centre projected to the image plane.

Homogeneous coordinates, introduced by August Ferdinan Möbius in 1827, are a way to represent N-dimensional coordinates with N+1 numbers. They have three benefits:

1. Affine transformations (including translation) can be represented by linear matrix multiplications.

2. Points at ∞ can be represented by setting $W = 0$ and $w = 0$.

3. 3D to 2D projection can be represented as linear matrix multiplications.

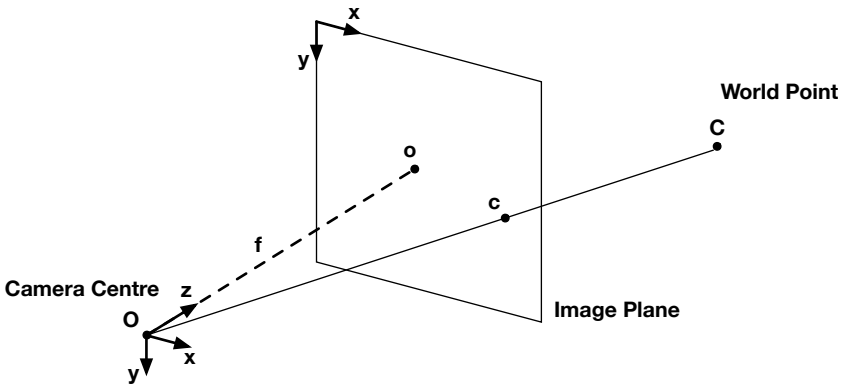


Figure 2.1: A pinhole camera where a 3D point C is projected onto image plane c. O is the camera centre and o is the principle point on the image plane. f is the focal length of the camera.

2.2.2 3D Worlds

Single-view 3D reconstruction is ill-posed and only solvable for scenes that satisfy strong regularity conditions. In the traditional 3D reconstruction literature, there are many world assumptions that provide additional constraints to help in the 3D reconstruction process. In chronological order they are:

1. **Blocks World**(1963) [136]: A world that is built of cubes, wedges, and hexagonal prisms (Figure 2.2 (a)).

2. **Polyhedral World**(1968) [65]: A world where all the objects are made with three-dimensional solid bodies bounded by a finite number of planar faces (Figure 2.2 (b)).

3. **Trihedral World**(1971) [76]: A world that contains only plane-bounded solid objects where every vertex is shared by exactly three faces (Figure 2.2 (c)).

4. **Origami World**(1980) [83]: A world built of stand-alone 2D piecewise planar surfaces in 3D space (Figure 2.2 (d)).

5. **Manhattan World**(1999) [33]: A world consisting of three mutually orthogonal directions of lines and surfaces. (Figure 2.2 (e)).

6. **Atlanta World**(2004) [144]: A mixture of Manhattan worlds (Figure 2.2 (f)).

7. **Box World**(2009) [70]: A world in an indoor room environment which could be approximated as a box made with five walls - left wall, middle wall, right wall, floor and ceiling (Figure 2.2 (g)).

The relationship between these worlds is illustrated in Figure 2.3. A polyhedral world assumes all the objects are three-dimensional solid polyhedrons. A trihedral world is a polyhedral world in which each vertex is shared by exactly three faces. A blocks world is a world that is built of cubes, wedges, and hexagonal prisms. An origami world assumes a three-dimensional world that is built from connected 2D piecewise planar surfaces. Atlanta world can be considered a subset of a block world and an origami world in which each object consists of three families of mutually orthogonal surfaces. In a Manhattan world, all objects are aligned to a common 3D orthogonal coordinate frame. The box world is a special case of Manhattan world that assumes a single box viewed from inside with five visible walls.

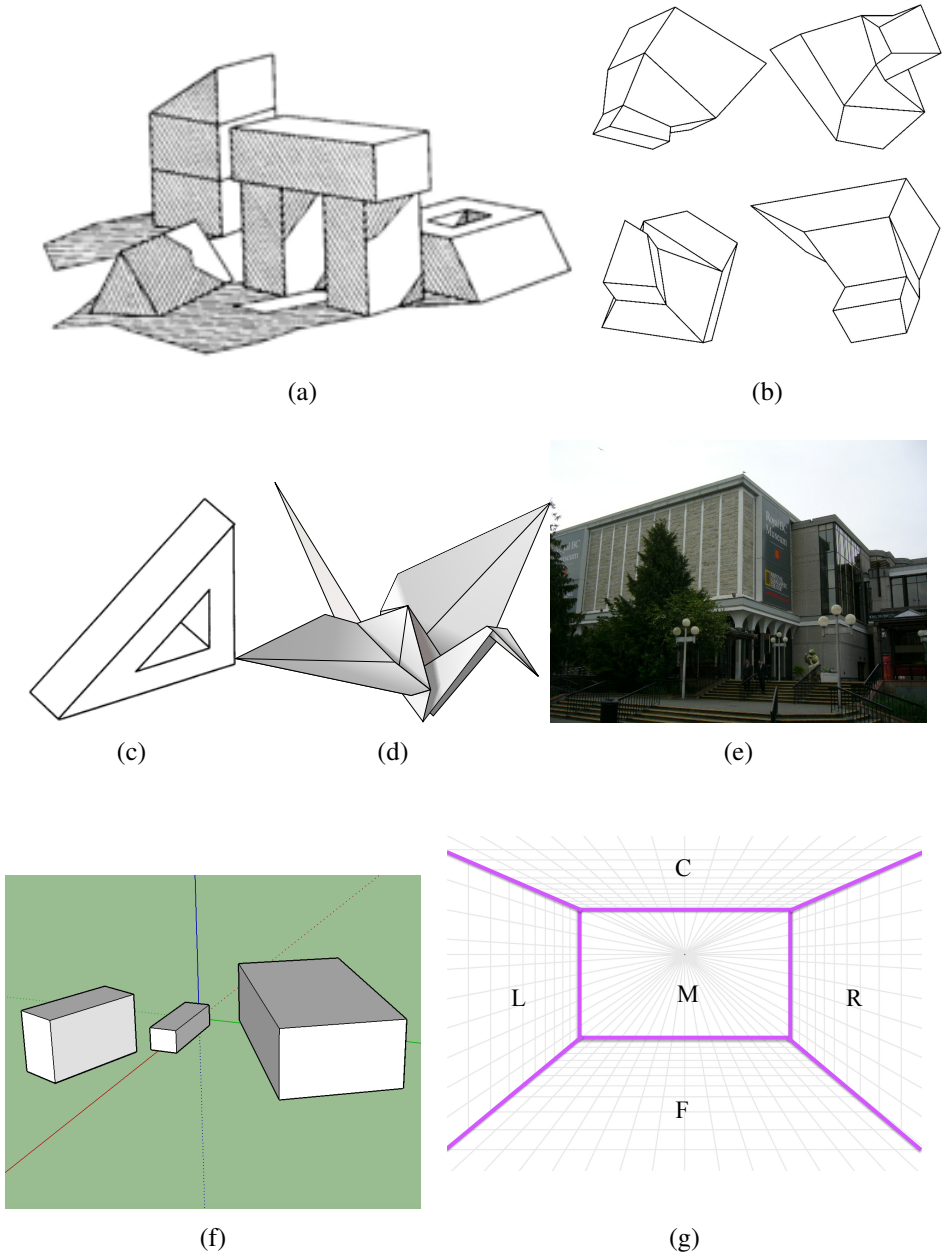


*Figure 2.2: (a) A blocks world. Figure taken from [136]. (b) A polyhedral world. Figure taken from [139]. (c) A trihedral object. Figure taken from [152]. (d) An origami crane. Figure taken from [15]. (e) A Manhattan world scene. Figure taken from [40]. (f) An illustration of Atlanta world scene. (g) The box world in which each pixel is assigned to one of five faces(**L**eft wall, **M**iddle wall, **R**ight wall, **F**loor and **C**eiling) in a box.*

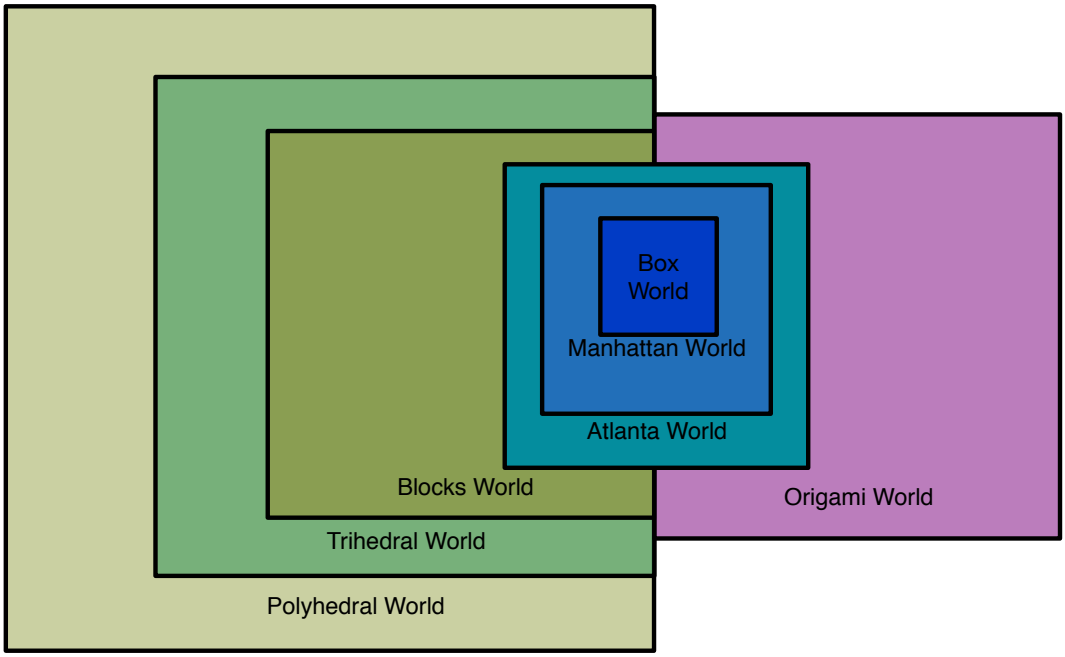


Figure 2.3: A Venn diagram shows the relationships between different world assumptions.

2.2.2.1 Blocks World

Roberts [136] assumed a blocks world built of cubes, wedges, and hexagonal prisms. His algorithm was constructive, working from edges to line segments, to 2D polygons and then to 3D blocks models. He was able to demonstrate a system that worked for simple combinations of model blocks under ideal laboratory imaging conditions.

2.2.2.2 Trihedral World

A trihedral scene [76] is an environment that contains an assortment of solid polyhedra which have exactly three planar surfaces at each of the vertices and two surfaces associated with each edge. All junctions in trihedral scenes can be labelled as either L, Y, W or T [31, 168]. The L junction consists of 2 non-collinear lines. The Y junction consists of three non-collinear lines where the three mutual angles are all less than π (Figure 2.4). The W junction consists of three non-collinear lines where one of the three mutual angles is greater than π (Figure 2.4). The T junction is made with three lines: two of them are collinear. These labelled junctions provide some information about the trihedral scenes. The Y and W junctions appear at the corners of scenes where the 3 surfaces meeting at the junction are visible to the observer. The L junction appears when only one of these surface is visible.

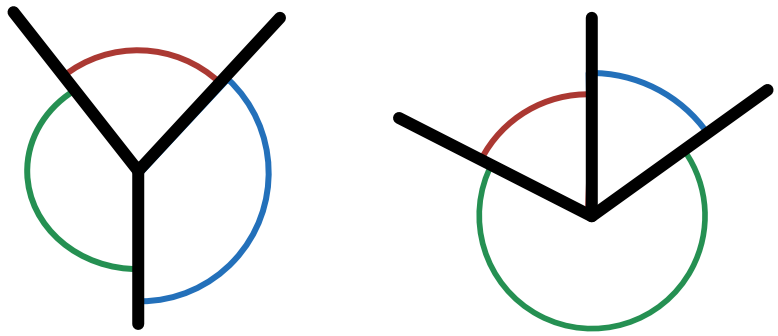


Figure 2.4: Y junction(left) and W junction(right). In a Y junction, three angles (red, green, and blue) are all less than π. In a W junction, one of the three angles (green) is greater than π.

2.2.2.3 Polyhedral World

The polyhedral world generalized trihedral world to arbitrary polyhedra[152]. The polyhedral objects are made with three-dimensional solid bodies bounded by a finite number of planar faces. To identify the polyhedral scene elements for spatial interpretation, two assumptions are applied:

1. For every face, one side is occupied with material, the other side is an empty space and each edge is shared by exactly two faces.

2. The observer is not coplanar with any face or any pair of non-collinear edges.

The line segments in the polyhedral scenes can be classified into four labels: the plus label $+$, the minus label $-$, and the arrow label $\uparrow$ in two directions. The plus label indicates convex lines (Figure 2.5 (a,e)), and the minus label indicates concave lines (Figure 2.5 (d,h)). The arrow label indicates that the plane on the right side of the arrow is the occluding plane (Figure 2.5 (b,c,f,g)). These line segment labels (Figure 2.6) can be used in the 3D

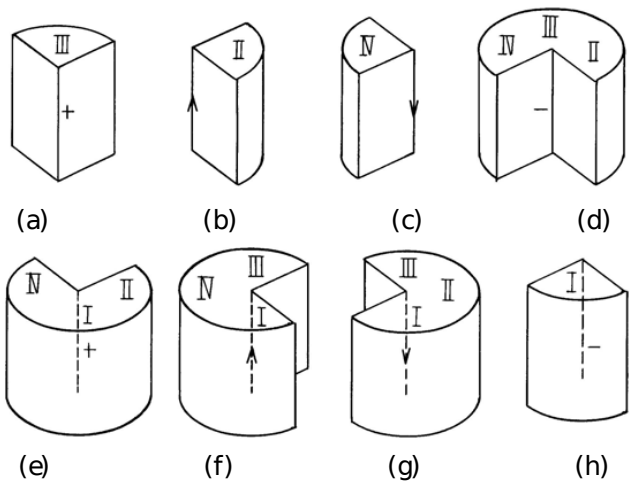


Figure 2.5: Different types of line labelling. Visible lines are drawn as solid lines, invisible lines are drawn as dashed lines. Figure taken from [152].

reconstruction process.

2.2.2.4 Origami World

In a polyhedral world, objects are assumed to be solid, but Kanade[83] suggested objects can also be made of a collection of open surfaces similar to origami. In this world, each object can be expressed as a graph structure where each vertex is a surface, and graph edges represent the boundary edges between pairs of surfaces. An example of an object and its origami world surface connection graph is shown in Figure 2.7.

2.2.2.5 Manhattan World

In the built environment, 3D scenes are often dominated by lines in three mutually orthogonal directions[33]. This is known as the Manhattan Assumption. By estimating these three mutually orthogonal directions, we can estimate the rotation of the camera relative to the world. (I will review this process in more detail in Section 2.4.2.) This Manhattan Assumption is widely used in 3D reconstruction tasks on man-made structures[39, 70, 169, 131, 135, 130]. The accuracy of estimated Manhattan line directions can be evaluated on the York Urban Database[40].

2.2.2.6 Atlanta World

In urban scenes, structures in the scene might contain multiple groups of orthogonal vanishing directions. In such scenes, instead of recovering one Manhattan world, multiple Manhattan worlds can be recovered. This is known as Atlanta world [144].

2.2.2.7 Box World

Hedau et al. [70] suggested that many indoor Manhattan scenes could be approximated as boxes made of five visible walls - left wall, middle wall, right wall, floor and ceiling.

2.2.3 Discussion

An image is a perspective projection of the 3D world. This perspective projection distorts many 3D properties such as angles, distance and ratios of distance, but one property is always preserved - line straightness. This invariance has been utilized in 3D world assump-

tions [136, 65, 76, 83, 33, 144, 70] as a constraint in the single-view 3D reconstruction task.

Among all of these world assumptions, the Manhattan world assumption [33] is one of the strongest. It assumes a world consisting of three mutually orthogonal directions of lines and surfaces, which is applicable to a large portion of modern architecture and indoor room environments [70]. I will review Manhattan world 3D reconstruction research in more detail in Section 4. However, before I review the geometric 3D reconstruction methods, I will review the 3D datasets that have been widely used for this task.

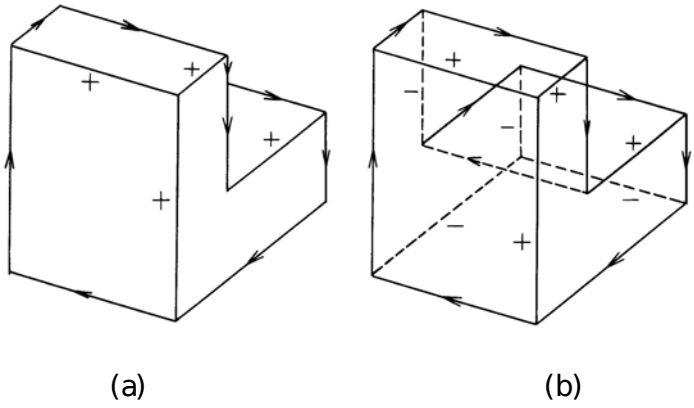


Figure 2.6: Labelling in (a) natural and (b) hidden-part-drawn line drawings. Figure taken from [152].

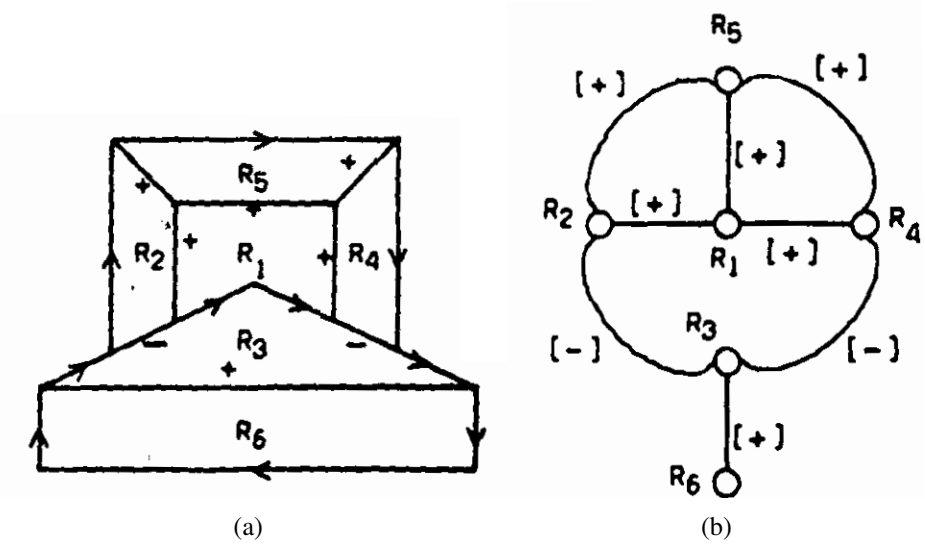


Figure 2.7: An object and its origami world surface connection graph. (a) A 3D projection to a 2D image. (b) The origami world surface connection graph (a). R1-R6 are surfaces, and edges are labelled line segments shared by surfaces. Figure taken from [83].

2.3 List of 3D Datasets

Public datasets have been a crucial resource in the 3D reconstruction field. They allow researchers to estimate parameters, and train, evaluate and compare algorithm performance. Table 2.1 lists three publicly available line segment datasets. The York Urban DB [40] is a dataset that contains line segment labels and their Manhattan orientation label. Later, Cho [29] enhanced this dataset by annotating much finer line segment labelling which increased the average number of line segment labels per image from its original 118.8 to 676.7. Unlike the original York Urban DB that only labels the Manhattan lines, this dataset labels both Manhattan and non-Manhattan lines. Furthermore, the curves in the image are labelled as multiple short straight line segments.

Table 2.2 lists the publicly available 3D scene datasets. The NYU v2 [149], Make3D [143], ScanNet [36], ETH 3D [146], KITTI [164], SYNS [1] and SUN RGB-D [151] are range map datasets constructed using 3D scanners. LS3D [158] is constructed using 3D CAD models of buildings.

Table 2.1: Line segment datasets

Dataset	Num. of Images	Ave. Seg. Per Image	Manhattan Label	Junction Label
YorkUDB [40]	102	118.8	Yes	No
YorkUDB+ [29]	102	676.7	No	No
Wireframe [75]	5462	104.8	No	Yes

Table 2.2: 3D scene datasets

Dataset	Num. of Images	Resolution	Semantic Label	Indoor	Outdoor
Make3D [143]	534	55x305	No	No	Yes
NYU v2 [149]	1449	640x480	Yes	Yes	No
ScanNet [36]	2.5M	640x480	Yes	Yes	No
ETH 3D [146]	454	6048x4032 752x480	No	Yes	Yes
KITTI [164]	92,750	375x1242	No	No	Yes
SYNS [1]	88 Panoramic	Panoramic 10,054x3,771	No	Yes	Yes
SUN RGB-D [151]	10,335	628x468 640x480 512x424	Yes	Yes	No
LS3D [158]	118	4912x3264	No	No	Yes

2.4 Geometric Methods for Single-View 3D Reconstruction

World assumptions provide constraints to estimate 3D structures from 2D images. One of the strongest constraints is perhaps the Manhattan world assumption, which is applicable in many built environments. Given the Manhattan world assumption, we can use a linear perspective to classify the lines or surfaces into three mutually orthogonal directions, and this classification can then be used to estimate 3D structures in the scene.

2.4.1 Edge and Line Segment Detection

The first stage of a geometric single-view algorithm is typically to detect the edges and lines in the image. In this section (Part 1), I will review three popular edge detection algorithms. Part 2 will review how edge information can be used to find straight lines in the image. Part 3 will focus on how line segments can be detected.

2.4.1.1 Edge Detection

Edges are points in the image where intensity changes abruptly. Processing edges instead of the whole image reduces the size of the data while preserving the important structural information. There are many edge detection methods such as Canny [22] and Elder& Zucker [50] that have been widely applied. More recently, deep learning edge detection algorithms (eg. [176]) have emerged.

Canny Edge Detector [22]: Early edge detection algorithms suffered many problems such as inaccurate edge point localization, duplicate edge marking, and false edges. To solve this problem, Canny [22] developed an algorithm with four stages:

1. Noise reduction with Gaussian filters
2. Intensity gradient identification
3. Non-maximum suppression
4. Hysteresis threshold

In the first stage, a Gaussian filter is applied to smooth the image which reduces the noise in the image (Figure 2.8 (b)). This reduces the number of false edges. Then this smoothed image is filtered with a local intensity gradient filter to find the luminance gradient magnitude and direction at each pixel (Figure 2.8 (c)). The next stage is to scan through the edge map to remove any unwanted edges. This is done by a non-maximum suppression which suppresses all the gradient values in the direction normal to the edge except the local maximum (Figure 2.8 (d)), resulting in a thin version of the edge map. The last hysteresis threshold stage decides which edges are high quality edges. Two thresholds minVal and maxVal are used in this process. Any edges with an intensity gradient more than maxVal are labelled as high quality edges and those below minVal are labelled as non-edges. Edges that lie between the two thresholds are classified as edge or non-edge based on their connectivity. If connected to high quality edge pixels, they are labelled as high quality edges. Otherwise, they are removed (see Figure 2.8 (e)).

(a) (b) (c) (d) (e)

Figure 2.8: Canny edge detector process. (a) Original image. (b) Smoothed with Gaussian filter. (c) Map of intensity gradient. (d) Non-maximum suppressed "thin" version of edge map. (e) Edge map after hysteresis threshold.

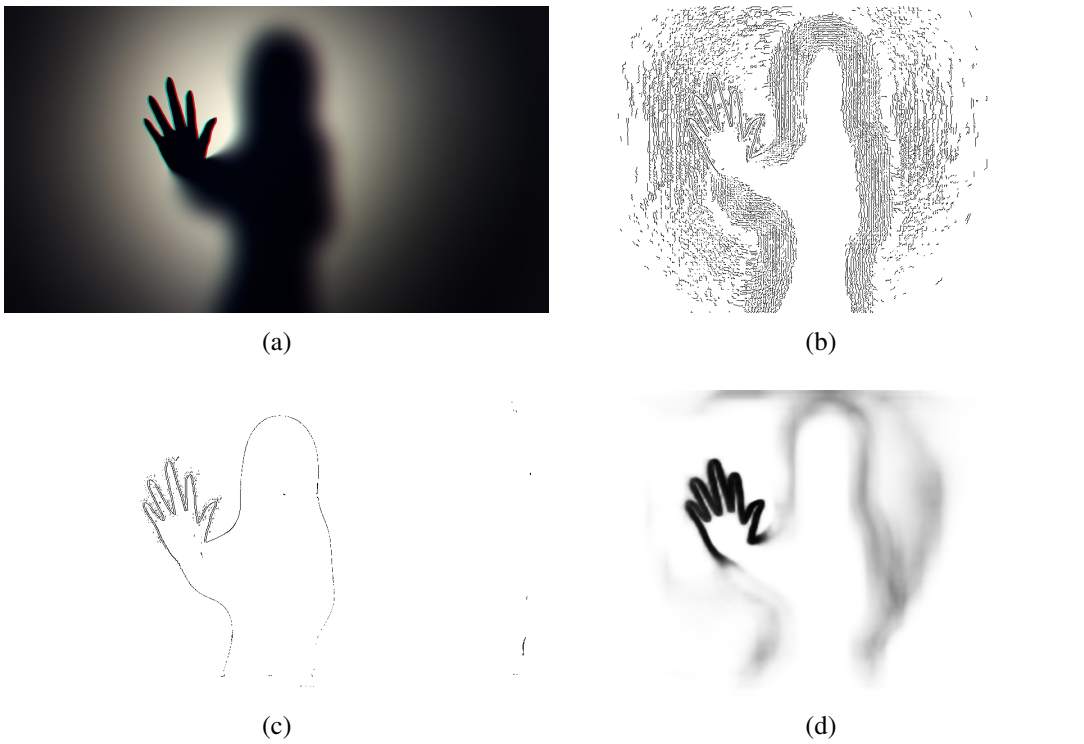
Elder & Zucker Edge Detector [50]: Discriminating true edges from noise is difficult for blurred or low contrast edges, which generate smaller gradients. Elder & Zucker [50] solved this problem by applying a multi-scale approach and finding the minimum reliable scale for gradient and second derivative estimation. The edges are localized using both gradient and second derivative information. Local estimation at the minimum reliable scale guarantees that the sign of the second derivative is reliable. It was used to localize the edge estimated from the gradient map. An example of the gradient and second derivative computation using local scale control for a test image is shown in Figure 2.9.

(a) (b) (c)

Figure 2.9: An example of local scale control. (a) Map of minimum reliable scale for gradient estimation. (b) Map of minimum reliable scale for second derivative estimation. (c) Detected edges. Six scales were used (0.5,1,2,4,8,16) and the shade of grey indicates the smallest scale at which gradient estimates are reliable. Black indicates scale at 0.5 pixel, lighter shades indicates higher scales, and white indicates that no reliable estimates could be made.

Holistically-Nested Edge Detector [176]: The recent development of deep learning algorithms enables more data driven methods for edge detection. Xie & Tu [176] introduced an end-to-end convolutional neural network to detect edges. An example output from the Holistically-Nested Edge Detector is shown in Figure 2.10(c).

Discussion: In terms of run time, the Canny edge detector (Figure 2.10(b)) is fast, taking 0.01 seconds to process a 512×512 image on an i7 CPU. It is one of the best candidates for real-time applications. On the other hand, the Elder & Zucker edge detector (Figure 2.10(c)) takes 0.5 seconds to process the same image on the same CPU. It delivers a finer edge detail for shadow and blurry parts, making it a good candidate for applications that do not require real-time performance. The holistically-nested edge detector (Figure 2.10(d)) is the most computationally intensive algorithm of the three, taking 12 seconds on an i7 CPU and 0.4 seconds on an Nvidia K40 GPU. It utilizes a data driven deep learning approach to deliver high quality edge estimation. Due to its high computational requirements, it is more feasible for high performance computers or cloud clusters.



(a) (b)

(c) (d)

Figure 2.10: Examples of edge detection algorithms (a) Original image (b) Canny Edge Detector [22] (c) Elder and Zucker Edge Detector [50] (d) Holistically-Nested Edge Detector [176].

2.4.1.2 Line Detection

Once edges are detected, they can be used to detect lines in the image. A common approach is the Hough Transform [74, 44], which uses the normal parametrization of a line in polar coordinates (Figure 2.11),

$$\rho = xcos(\theta) + ysin(\theta) \tag{2.4}$$

where ρ denotes the signed distance of the line from the origin, and $\theta \in [-\pi/2, \pi/2)$ denotes the angle of the orthogonal projection from the origin to the line. Each edge has position and orientation information which can be written as a point (ρ, θ) in polar coordinates. The map $H(\rho, \theta)$ that counts the occurrences of these points is called the Hough map. It represents the image evidence for all possible lines passing through the image.

Figure 2.11: Hough parameters of a line.

Ideally, all edges associated with a line would map to the same bin in the Hough map $H(\rho, \theta)$. However, due to noise in the image, that is not generally the case, leading to missing peaks or false peaks in the Hough map, causing missing lines or false lines. One way to mitigate the problem is to accurately model the uncertainty in the location and orientation of each edge, mapping this uncertainty to a smooth local distribution over the Hough parameters. Tal. et al. [154] modelled the edge position and orientation information through a bivariate normal (BVN) kernel (Figure 2.12) in the Hough map using image statistics. Lines in the Hough map can then be extracted in a greedy fashion and probabilistically subtracted from the Hough map. At each iteration, the global maximum of $H(\rho, \theta)$ is detected to identify the position and orientation of the dominant line not yet extracted. All edges are identified that fall within a 3σ boundary in terms of both distance from the line and angular deviation. These identified edges are then subtracted from the Hough map $H(\rho, \theta)$. Using statistical measurements of edges in the YorkUrbanDB dataset [40], Gaussian likelihood models of the edge uncertainties were estimated: σ_x, $\sigma_y = 0.49$ pixels,

error σ_θ = 5.3 degrees.

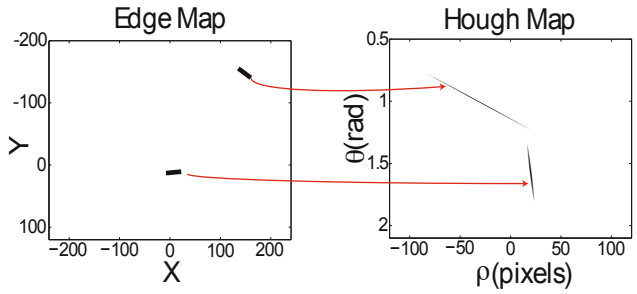


Figure 2.12: Each edge votes according to a BVN kernel. Figure taken from [154].

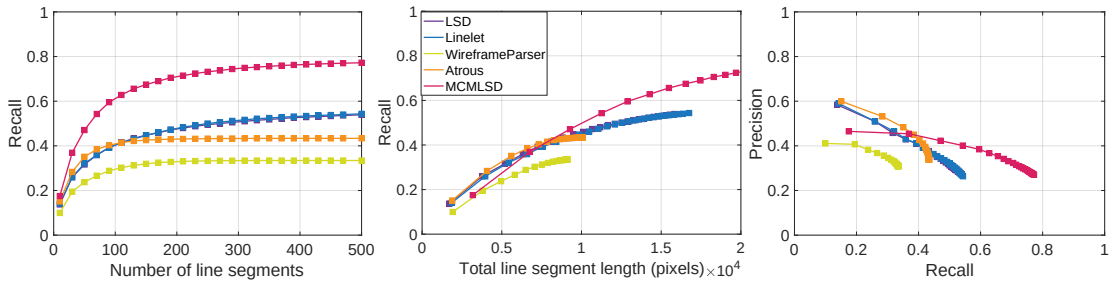
2.4.1.3 Line Segment Detection

Peaks in the Hough map can only identify lines of infinite extent; they do not identify the endpoints of the line segments. The endpoints of line segments can be estimated by line segment detection algorithms [166, 4, 29, 182, 75]. There are three main line segment detection methods:

1. Perceptual grouping methods
2. Hough transform methods
3. Deep learning methods

Perceptual grouping methods [166, 29]: These use geometric grouping cues to cluster edges that are close to each other and roughly collinear. Each cluster is cleaned and then analysed [42, 166] to obtain high quality line segments.
Hough transform methods [4]: The edge map of an image identifies the lines of infinite extent. Then each detected line in the image domain is analysed to localize the line segments in it. The Hough transform method has the advantage of accumulating the global edge information in the image domain.
Deep learning methods [75, 182]: Deep learning methods use images labelled with major line segments to train a network to extract line segments directly from image pixels.

A comparison of these five algorithms is shown below. LSD [166] and linelet [29] are perceptual grouping methods. MCMLSD [4] is a Hough transform method. Wireframe [75] and atrous [182] are deep learning methods. Quantitative evaluation is shown in Figure 2.13, and a visual comparison is shown in Figure 2.14. The MCMLSD algorithm tends to identify long continuous line segments in the image, while the LSD and linelet algorithms generate more disconnected line fragments and fail to identify the faint line segments on the ground. The Wireframe and Atrous algorithms are deep learning based methods generate very fragmented segments and tend to miss a lot of texture details.



(a) (b) (c)

Figure 2.13: Performance of many line segment detection algorithms. (a) Recall as a function of the number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

Discussion: From the qualitative and quantitative evaluation above, we can see that the MCMLSD algorithm outperformed the other four algorithms on the York Urban Database. The MCMLSD algorithm is able to detect a greater number of connected segments and its ability to extract long continuous segments may be an advantage for single-view 3D reconstruction.

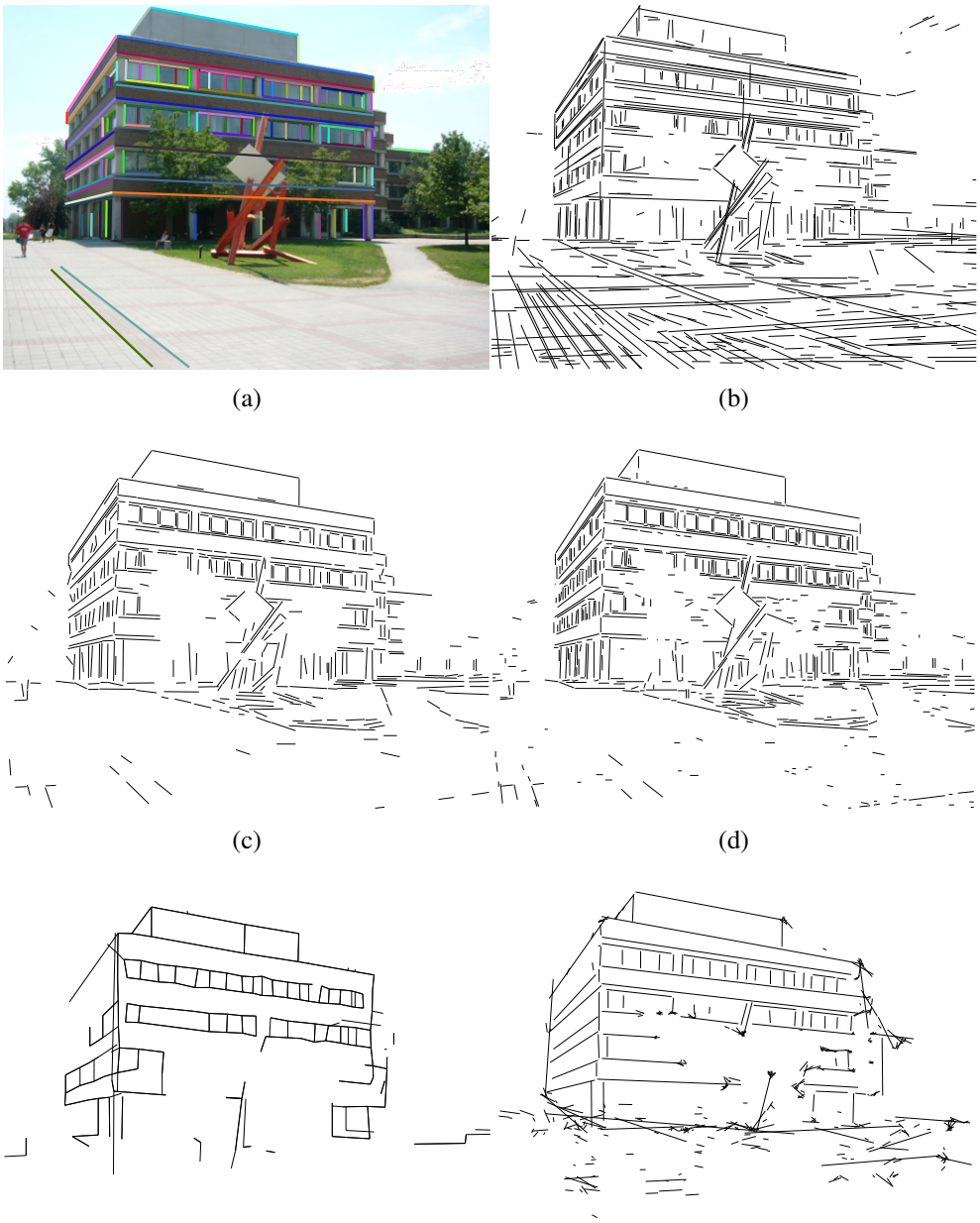


Figure 2.14: Examples of line segment detection algorithm output (a) Original image (b)MCMLSD [4] (c) LSD [166] (d)Linelet [29] (e)Wireframe [75] (f)Atrous [182]

2.4.2 Manhattan Frame Estimation

A line in the image together with the camera centre forms an interpretation plane that can be represented by its normal vector l (Figure 2.15) [154]. Lines with the same 3D orientation will produce interpretation plane normals that are coplanar and orthogonal to the 3D orientation of the lines. The error $\Delta\theta_i$ of a line with respect to a hypothesized 3D Manhattan orientation vector is measured by the angular deviation between the line interpretation plane normal and the plane π perpendicular to the hypothesized 3D Manhattan orientation.

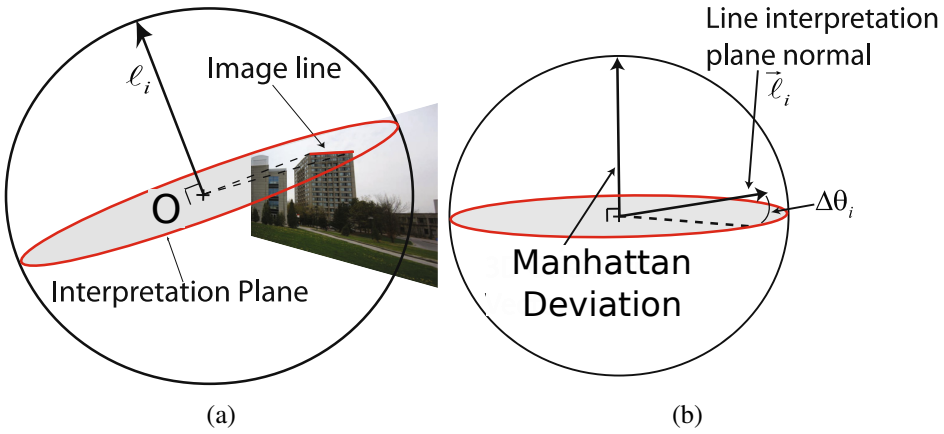


Figure 2.15: (a) A line in the image can be represented in the Gauss sphere by its interpretation plane normal. (b) Error model for a line in the Gauss sphere. Figures taken from [154].

In [154] each observed line l_i is assumed to be generated by a latent Manhattan variable m_{l_i} that belongs to one of four classes: horizontal(1), horizontal(2), vertical and background. The Euler angles Ψ describe the rotation of the camera relative to the Manhattan frame. The probability of each observed line is a mixture of the four models.

$$p(l_i|\Psi) = \sum_{m_{l_i}} p(l_i|\Psi, m_{l_i}) p(m_{l_i}) \tag{2.5}$$

The Euler angles Ψ can be estimated by maximizing the probability using all lines.

$$\hat{\Psi} = \arg\max_{\Psi} \sum_{i} \log p(l_i|\Psi) \tag{2.6}$$

The likelihood $p(l_i|\Psi, m_{l_i})$ and prior probability $p(m_{l_i})$ are estimated from hand-labelled ground-truth lines in the YorkUrbanDB Dataset [40].

$$p(l_i|\Psi, m_{l_i}) = \frac{1}{2b} e^{\frac{-|\Delta\theta_{m_{l_i}}|}{b}} \tag{2.7}$$

where b=0.80 for horizontal lines and b=0.57 for vertical lines. The likelihood for background $p(l_i|m_{l_i} = B)$ is uniform, independent of orientation.

A comparison between three Manhattan frame estimation algorithms on the YorkUrbanDB Dataset [40] is shown in Table 2.3. The line based estimation algorithm achieved the best result with an average error of 1.7 degrees.

Table 2.3: Manhattan frame estimation algorithm comparison

Method	Mean Error (degrees)	Run Time (Sec)
Line based estimation [154]	1.7	8
Gradient based method [33]	9.6	22
Edge based method [40]	4	5

Discussion: Manhattan frame estimation identifies the orientation of three mutually orthogonal directions of the 3D world. Accurate estimation of the Manhattan frame is important for 3D reconstruction. Line based estimation as proposed by Tal and Elder [154] delivers the best accuracy at 8 seconds runtime.

2.4.3 Box Room Estimation

Hedau et al. [70] suggested that many indoor Manhattan scenes could be approximated as boxes (Section 2.2). Under this assumption, each pixel in the indoor scene is classified into one of five faces - left wall, middle wall, right wall, floor and ceiling. Using this pixel assignment, a box is generated to represent the room layout. This simple box room layout representation provides an easy solution for VR/AR applications such as placing virtual furniture in the room [78].

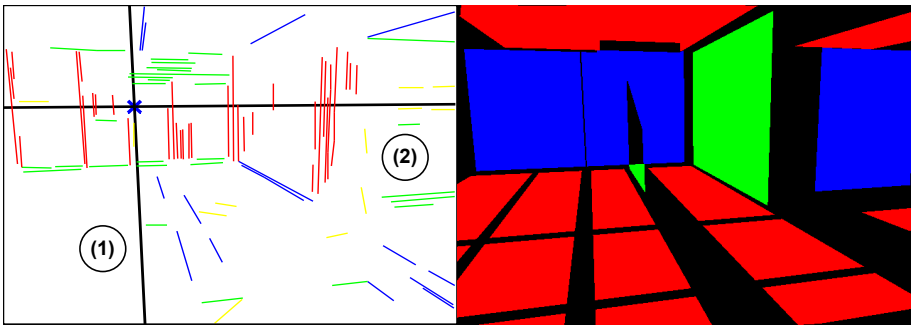
The box world assumption is a strong constraint. There are many attempts [70, 169, 101, 147, 131] to use this constraint to generate the 3D layout of indoor room scenes. Features such as line segments, orientation maps (OM) [100], and geometric context (GC) [73] are captured and a classifier is then applied to pick the best layout hypothesis based on these features.

Orientation Maps(OM)[100]: The orientation map (Figure 2.16 (b)) is a map that indicates the Manhattan orientation of pixels in the image. Each pair of nearby line segments of orthogonal orientation defines a quadrilateral in the image. Pixels within the quadrilateral are labelled with its 3D orientation. For example in Figure 2.16 (a), pixel (1) is supported by the green line above and a blue line to the right which indicates pixel (1) is in a horizontal plane. Similarly, pixel (2) is supported by the green lines above and below, and the red line to the left. So this pixel (2) is on a vertical surface with a normal vector perpendicular to the red and green lines.

Geometric context (GC)[73]: The geometric context feature defines each pixel into one of two classes: vertical and support. The input image is first broken up into super-pixels, then surface cues are extracted from each super-pixel to form feature vectors. A binary classifier is trained to categorise each super-pixel into one of these two classes.

A quantitative comparison of state of the art room layout estimation algorithms is shown in Table 2.4. From the comparison, we can see Ramalingam et al.'s[131] system using junctions, GC and OM under the Conditional Random Field (CRF) framework delivered the best performance.

The box assumption is not perfect. It only works for enclosed box room environments and is not applicable to irregular rooms. Irregularly shaped furniture will also have a negative impact on layout estimation.



(a) (b)

Figure 2.16: (a) The red, green, blue are Manhattan line segments, the yellow line segments are background. The black lines are vanishing lines. The circles with number 1 and 2 in the centre are two sample points. (b) An orientation map where red represents horizontal planes, and blue and green represent vertical planes. Figure taken from [100].

Table 2.4: Algorithm evaluation comparison on Hedau et al's dataset[70]. OM is orientation maps[100], GC is geometric context[73].

Method	Features	Pixel Error
Hedau[70]	Line Segments+VP	21.2%
Wang[169]	OM	22.2%
Lee[101]	OM+GC	16.2%
Schwing [147]	OM+GC	13.6%
Ramalingam [131]	Junction+GC+OM+Pairwise+Triple	13.3%

2.4.4 Manhattan 3D Reconstruction with Junctions

In Section 2.2.2.2, we learned that intersections of line segments form various types of junctions. These junctions can be used to infer 3D geometry from a 2D image. Ramalingam et al. [131] used this approach to reconstruct 3D line drawings of Manhattan buildings. Visualizations of this junction based Manhattan line 3D reconstruction are shown in Figure 2.17. His approach consists of three steps:

1. Detecting Manhattan line segments
2. Connecting the Manhattan line segments into a graph
3. Lifting the line graph into 3D

The connectivities between two line segments are of two types:

1. Two collinear lines meet at a point (incidence)
2. Two or three orthogonal lines intersect at a point to form a junction

As I explained in section 2.2.2.2, in polyhedral scenes, junctions can be classified into Y, W, L, X, T categories [76, 31, 168] (Figure 2.18 and 2.20(b)). The Y and W junctions indicate convex or concave corners; L indicates corners; X usually indicates translucency; T junctions provide evidence of occlusion between two surfaces.

Ramalingam et al. [131] proposed an algorithm that uses Manhattan lines to categorize junctions in the image into L,T,W,Y, and X types. Each point p_i in the image has 6 possible orientation regions with respect to the vanishing point (Figure 2.19) denoted as $r \in \{\overrightarrow{x}, \overleftarrow{x}, \overrightarrow{y}, \overleftarrow{y}, \overrightarrow{z}, \overleftarrow{z}\}$.

These orientations represent the direction toward or away from vanishing points in x, y, and z directions. Inside each orientation region from point p_i, there may be several line segments. The total length of the line segments in each orientation region within a 60 pixel radius of point p_i is stored as a 6-cell feature vector array $V_j(p_i)$ where each cell represents an orientation. $V_j(p_i)$ is then converted into a binary vector by a threshold h. This binary vector is matched with one of five binary templates (see illustration of 5 binary templates in Figure 2.20) to classify point p into one of L, T, W, Y, X or null junction types.

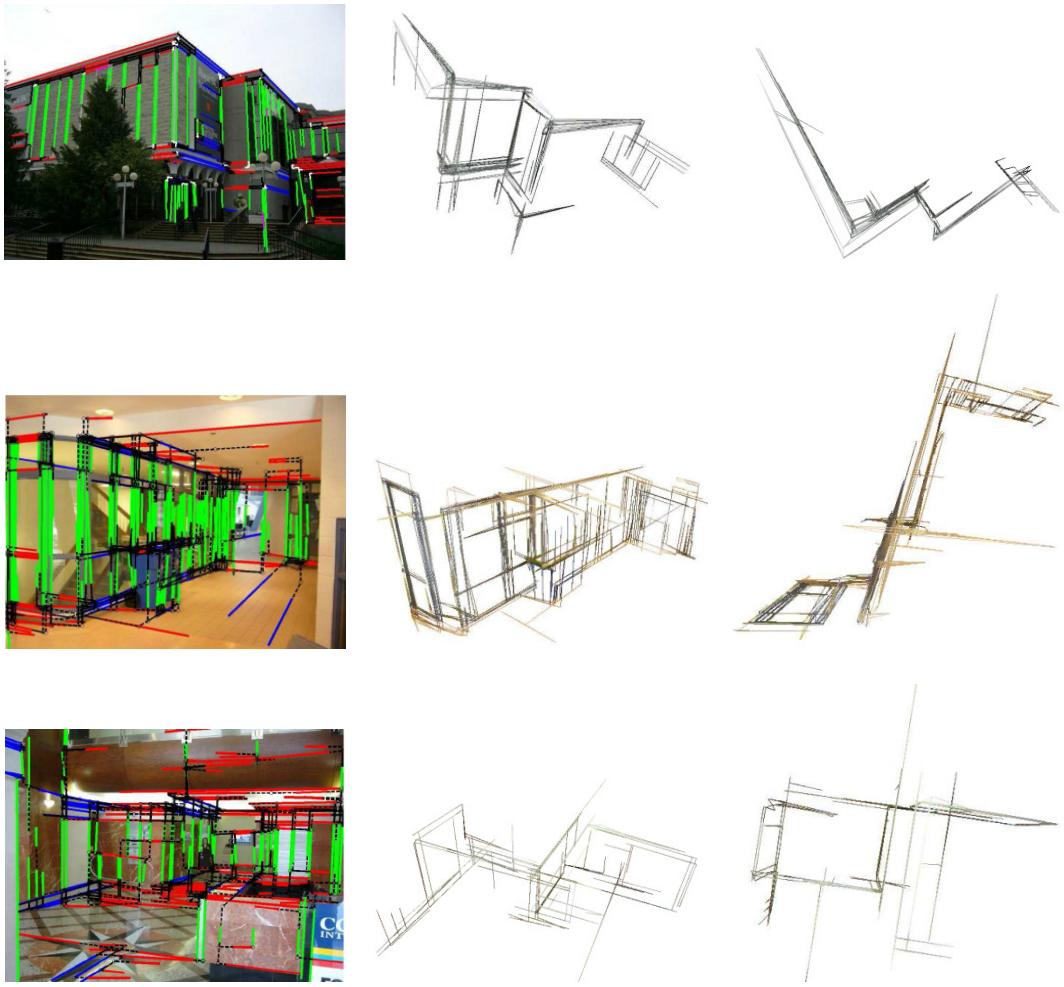


Figure 2.17: Line reconstruction results. The first column shows the Manhattan lines overlaying the original images. The second and third columns show perspective 3D views of the 3D line structures. Figure taken from [130].

The incidences and junctions connect lines together into a line graph structure $G = (V, \varepsilon)$, where the vertices $V = 1, 2, \ldots, n$ represents the lines, and edges $\varepsilon \in (i, j)$ represent intersections or incidences between lines l_i and l_j. An example of a line graph is shown in Figure 2.21.

In the graph $G = (V, \varepsilon)$, each vertex is a Manhattan line segment. Let the 3D coordinate of one of the end points on line V_i be $p_i = \lambda_i d_i$, where λ_i is an unknown depth

Figure 2.18: A living room with several junctions of types L, T, Y, X, W. Figure taken from [131].

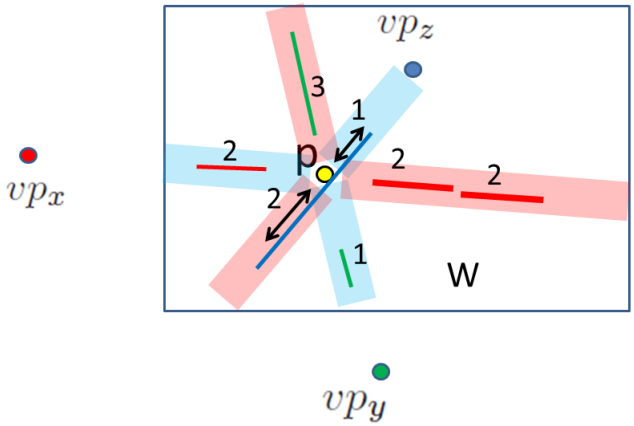


Figure 2.19: Lines at a junction point p. From p there are 6 orientation regions with respect to the vanishing point. Inside each orientation region, there may be several Manhattan line segments of different lengths. The total line segment length inside each orientation region is used to classify the junction type. Figure taken from [131].

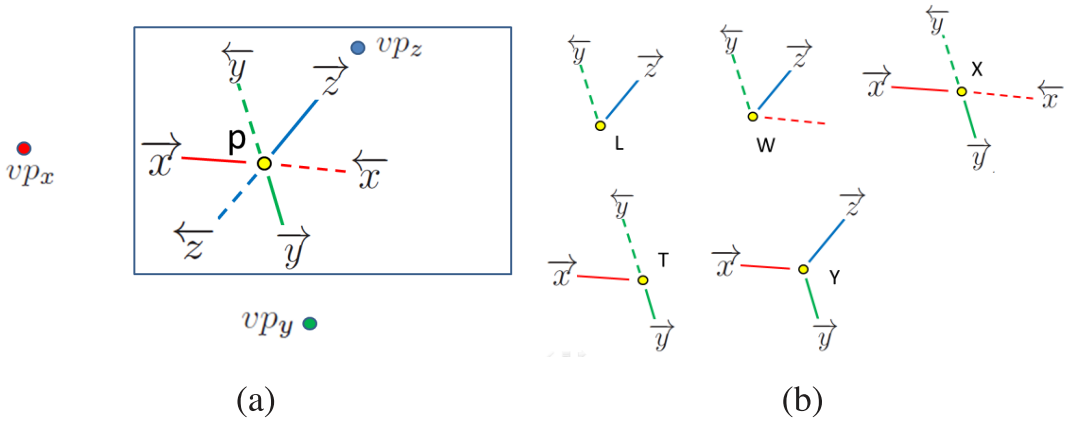


Figure 2.20: (a) There are 6 possible directions at point p based on the three vanishing points. (b) Binary templates for 5 junction types. Figure taken from [131].

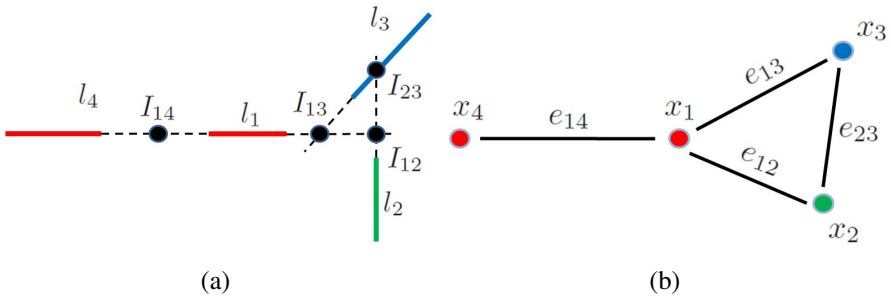


Figure 2.21: (a) 4 line segments (l_1, l_2, l_3, l_4) connected by incidences and junctions where l_{14} is an incidence connection node and I_{12}, I_{13} and I_{23} are junction nodes. (b) A graph structure representing connections in (a). In this graph each vertex is a line segment and each edge is an incidence or junction. [130]

parameter, $d_i = (x, y, f)$ represents the 3D coordinate of point p_i in the image domain, and f is the focal length of the camera. Under the Manhattan assumption, recovering the 3D location of one end node on the line is sufficient to recover the 3D coordinates for the whole line segment. d_i is a constant vector which leaves the vertex V_i with 1 degree of freedom (a linear model) in the 3D space. By optimizing the parameter λ_i for all the lines, a 3D line model can be obtained. The optimization algorithm is based on the assumption that when two lines form a junction or incidence, the 3D distance of these two lines should

be very small. For example, in Figure 2.22, two lines (l_i and l_j) in the image form a junction. By adjusting the scale parameter λ_i and λ_j, the 3D distance between these two lines S_{ij} can be minimized. This optimization process can be handled by a linear programming formulation that computes:

$$\begin{aligned} &\min_{\lambda_i} \sum_{(i,j)\in\varepsilon} (\|\omega_{ij} s_{ij}\|_1) \\ &s.t \quad |\lambda_i d_{ia} - \lambda_j d_{ja}| \leq s_{ij}, a \in \{x, y, z\} \backslash \{D_i, D_j\}, \lambda_i \geq 1, i \in \nu \end{aligned} \tag{2.8}$$

where s_{ij} is a slack variable minimizing the distance between two lines l_i and l_j. ν is the collection of line segments in the image. $D_i \in \{x, y, z\}$ is a Manhattan line direction for line l_i. The weight parameters ω_{ij} are set to 10 for Y and W junctions, 5 for X junctions, and 1 for L junctions and incidence junctions. T junction weights are set to 0 because T junctions are unreliable, as they can be generated by both planar surfaces and occluding boundaries (Figure 2.23). The optimal fully-connected 3D structure is extracted by computing the minimum spanning tree (MST) using the slack value s_{ij} as weight.

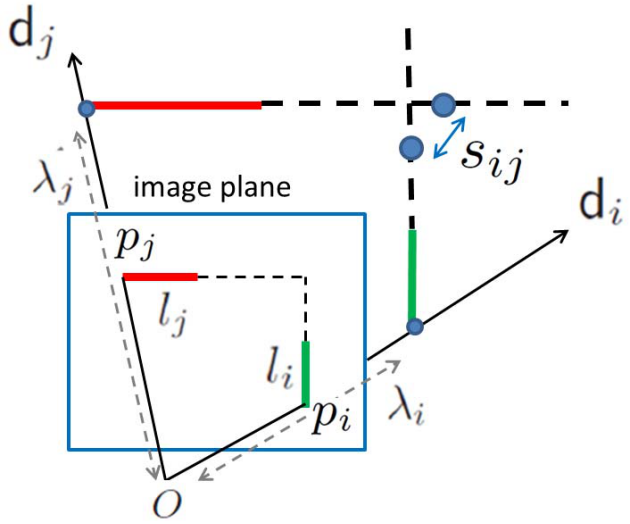


Figure 2.22: Line segments l_i and l_j intersect in the image domain. s_{ij} indicates the 3D distance between line l_i and l_j. λ_i and λ_j are adjusted to minimize s_{ij} and thus to recover the 3D coordinates of l_i and l_j. [130]

Figure 2.23: T junctions in a built environment. The T junction on the left is caused by occlusion. The T junction on the right is caused by a planar reflectance pattern.

Discussion: This approach is limited to connected Manhattan structures and will fail for a structure that is not Manhattan or not connected. It uses LSD [166] to detect line segments, which sometimes fails to identify continuous segments. Using a more advanced line segment detection algorithm [4, 29, 182, 75] might improve performance.

2.4.5 Photo Pop-up

A Photo Pop-up model [73, 62, 124] is similar to a pop-up illustration in a children's book (Figure 2.24). It is a 3D reconstruction that is made from 2D vertical objects on a supporting platform.

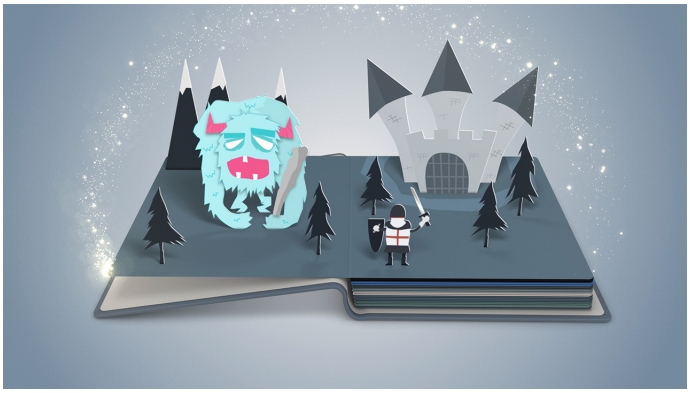


Figure 2.24: A pop-up illustration in a children's book [93]

In this model, each image pixel is categorized into one of three major classes: *support*, *vertical* and *sky*. The *support* consists of surfaces parallel to the ground such as road surfaces, table tops and lakes. The *vertical* class consists of surfaces such as walls, cliffs, trees or people. The *sky* is simply the sky region in the image. The *vertical* class can be further divided into *planar surfaces* and *non-planar surfaces*. Planar surfaces include building walls, and cliff faces. Image pixels on vertical *planar surfaces* are classified into *left*, *centre* and *right* categories. The *non-planar surfaces* includes trees, people, and cars and are further subdivided into two sub-categories: *porous surfaces* and *solid surfaces*. *Porous surfaces* are surfaces that do not have a solid continuous surface. *Solid surfaces* are continuous surfaces. Supports form a platform that supports vertical surfaces. An example of a surface Photo Pop-up visualization is shown in Figure 2.25.

There are numerous algorithms that attempt to infer such a model from a single image [73, 62, 124]. A quantitative comparison of four algorithms evaluated on the Geometric Context dataset [73] is shown in Table 2.5 and a qualitative comparison is shown in Figure 2.26 . The Pan [124] algorithm achieved the best performance. In this algorithm, semantic segmentation [117], geometric context [73], and orientation maps [100] are uti-

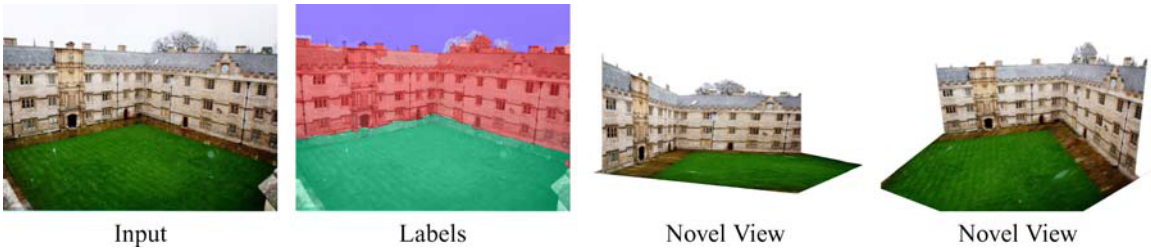


Figure 2.25: Pop-up 3D reconstruction. Pixels in the input image are labelled as support, vertical or sky classes. The second column shows the labelling result. The third and fourth columns show the 3D reconstruction from two different view angles. Figure taken from [73].

Table 2.5: Comparison of surface layout estimation algorithms on Geometric Context dataset [73].

	Hoiem [73]	Gupta [62]	Pan [124]	OM [100]
Layout Accuracy	72.87%	73.59%	74.82%	71.2%

lized to detect and decompose the building region into a set of distinctive facade planes. A distinctive facade plane is a facade plane whose orientation is different from its adjacent facade planes. This set of distinctive facade planes together with assumptions on building height and camera height are used to create a conditional random field (CRF) model to estimate the surface layout.

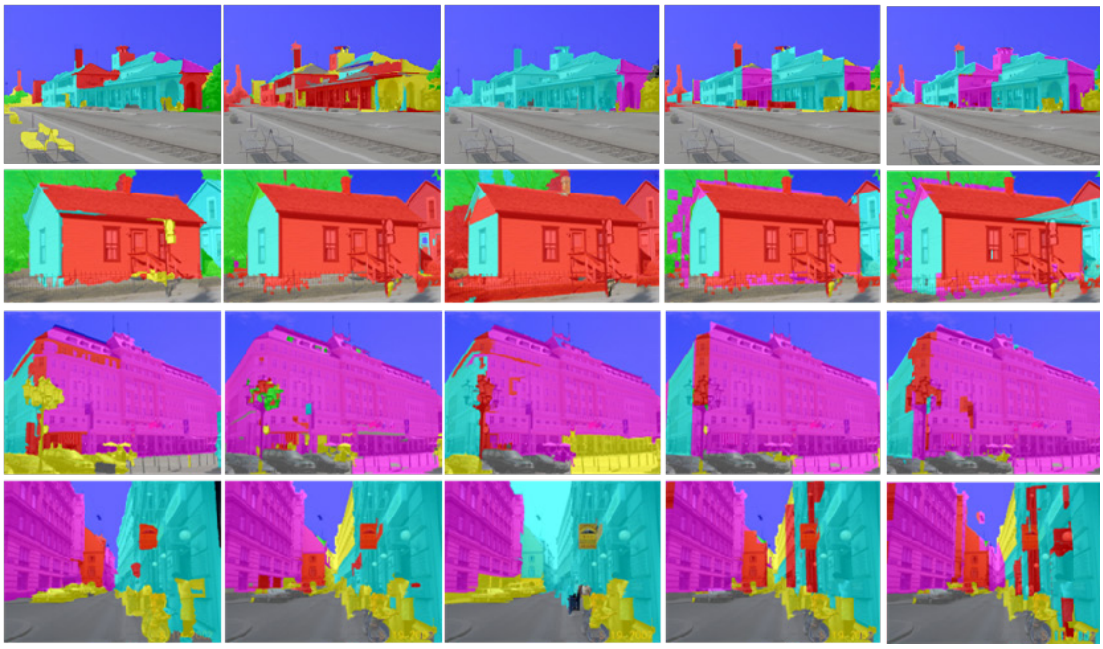


Figure 2.26: Qualitative comparisons of surface layout estimation. From left to right: Ground truth; Hoiem et al. [73]; Gupta et al. [62]; Pan CRF [124]; Pan without CRF [124]. Surface layout colour code: Magenta - planar right; Cyan - planar left; Red - planar centre; Green - nonplanar porous; Yellow - nonplanar solid; Blue - sky; Grey - support. Figure taken from [124].

2.4.6 Discussion

3D Geometry can be inferred from a 2D image through primitive features such as line segments and appearance cues. A line segment is a powerful cue that remains a line segment after perspective transformation. It can be used to estimate advanced geometrical features such as vanishing points, Manhattan frame orientation, orientation maps, geometric contexts, and junctions. These advanced features can be used to infer a 3D box room layout [70, 169, 101, 147, 131] or build a 3D line drawing of Manhattan buildings [130]. Under non-Manhattan environments where line segments alone cannot provide sufficient information for the reconstruction, appearance cues can provide additional information to build a Photo Pop-up 3D reconstruction [73, 62, 124]. A diagram illustrating the relation between primitive features, advanced features and 3D reconstruction is shown in Figure 2.27.

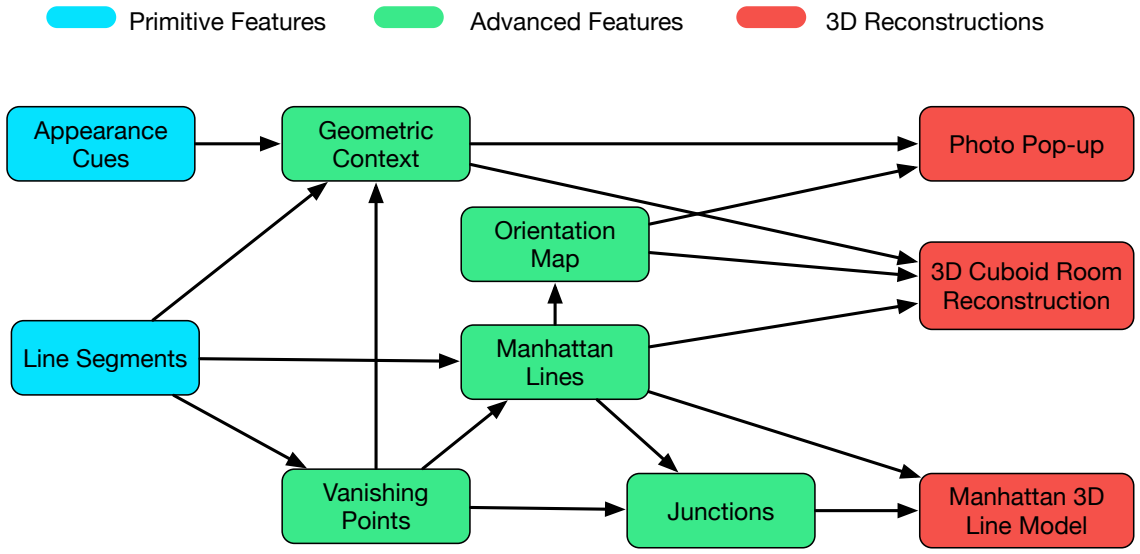


Figure 2.27: Relationships between geometry based reconstruction methods.

2.5 Machine Learning Methods for Single-View 3D Reconstruction

Constrained piecewise planar models are appropriate for some scenes. For more general scenes, a range map can be estimated using supervised machine learning techniques. Make3D [140, 141, 143] was an early attempt to recover a range map from a single image. It utilizes local edge, texture, and colour features and models these features in a Markov Random Field (MRF). Later, this approach was improved by extracting deep learning features [107, 102, 171, 178] as feature vectors to train graph models.

Another way to estimate depth is through an end-to-end deep neural network [46, 45, 26, 99, 55]. These deep learning approaches need a large amount of labelled data which require a lot of resources to obtain. Recent work demonstrates that deep networks for single-view range map estimation can be trained from multi-view data such as calibrated stereo cameras [57] or un-calibrated video sequences [187] using reprojection error as the supervisory signal. Details of these methods are described below.

2.5.1 Graph Models

Graph models specify a factorization of the joint distribution over a set of variables into a product of local conditional distributions, as well as a set of conditional independence relations [17]. Graph models have been widely applied to image de-noising, classification and segmentation tasks. Saxena et al. [140, 141, 143] modelled the depth map estimation problem with a multi-scale Markov Random Field (Figure 2.28). In the algorithm, hand-crafted local and global image features were extracted to create a depth prediction model. They showed that the depth information in an unstructured scene can be estimated by using local edge, texture and colour features from a single image.

Local features are extracted from a set of small homogeneous regions called "super-pixels" [51]. Each super-pixel represents a coherent region in the image in which all pixels have similar properties(Figure 2.29). Using super-pixels reduces the computational requirements of the algorithm.

The depth value of each super-pixel is inferred from maximum a posteriori(MAP)

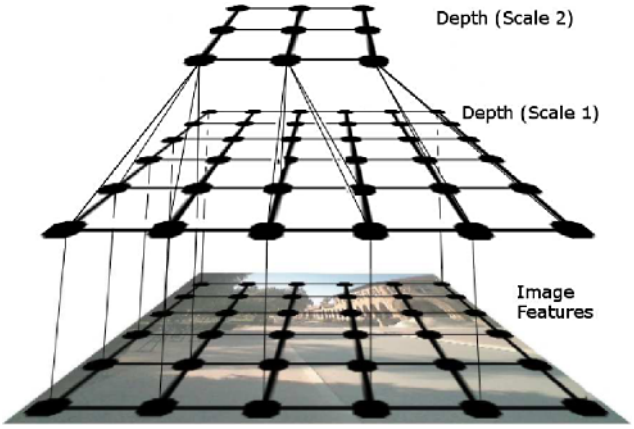


Figure 2.28: The multiscale MRF model for modelling relations between features and depths, the relation between depths at the same scale, and the relation between depths at different scales. [141]

Figure 2.29: An image segmented into super-pixels. Figure taken from [142].

estimation of a Gaussian model:

$$P_G(d|X;\theta,\sigma) = \frac{1}{Z_G} exp(-\sum_{i=1}^{M} \frac{(d_i(1) - x_i^T\theta_r)^2}{2\sigma_{1r}^2} - \sum_{s=1}^{3}\sum_{i=1}^{M}\sum_{j\in N_s(i)} \frac{(d_i(s) - d_j(s))^2}{2\sigma_{2rs}^2}) \tag{2.9}$$

where x_i is the feature vector from patch i, and i and j are index numbers of two nearby image patches. $d_i(s)$ and $d_j(s)$ are the estimated depth values at patch i and patch j at scale $s = 1, 2, 3$. θ and σ are parameters of the model learned from training data. Different parameters $(\theta_r, \sigma_{1r}, \sigma_{2rs})$ are used for each image height r. M is the total number of super-pixel patches in the image, and Z_G is the normalization constant for the model. The param-

eter θ_r is estimated by maximizing the conditional log likelihood $l(d) = logP(d|X;\theta_r)$. σ_{1r} and σ_{2rs} are estimated by quadratic programming fitting to the expected value of variance in the training data. The algorithm was trained and tested on a dataset acquired using a camera of fixed height, focal length, and horizontal optic axis. Consequently, height in the image is a good predictor of depth.

More recently, hand-crafted features have been replaced with deep-learning features [102], significantly improving the model accuracy. Furthermore, deep networks [107, 171, 178] can be directly applied to estimate the unary potential and pairwise potential in the graph model. A performance comparison of recent graph based depth prediction methods is shown in Table 2.6.

	Feature	Model	Training Methods	Scale	RMS Error
Make3D [143]	Hand Crafted	MRF	Max. Likelihood + QP	Multi-Scale	1.214
Liu [107]	CNN	CRF	CNN	Single-Scale	0.824
Li [102]	CNN	CRF	Least Squares	Multi-Scale	0.821
Wang [171]	CNN	CRF	CNN	Multi-Scale	0.745
Xu [178]	CNN	CRF	CNN	Multi-Scale	0.655

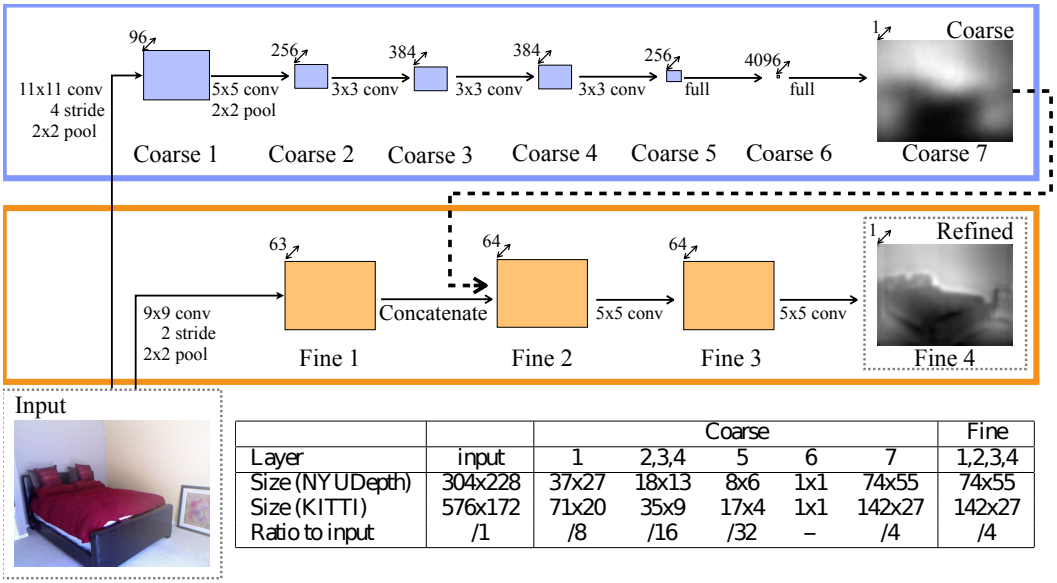
Table 2.6: Performance comparison of graph based depth prediction methods evaluated on the NYU v2 dataset [149]. The Make3D algorithm was trained on the Make3D dataset and evaluated on the NYU v2.

Discussion: The graph model is capable of processing a wide range of scenes. One weakness of the graph model approach is that it requires a lot of training data. A second problem is that it tends to overfit to the training dataset, resulting in poor generalization.

2.5.2 End-to-End Deep Networks

Another way to reconstruct a range map from a single image is to train an end-to-end deep network. Eigen et al. [46] proposed a structure that uses two deep convolutional neural networks: a global coarse-scale network and a local fine-scale network. The global network predicts the overall global depth map. The upper layers are fully connected while the lower and middle layers are convolutional layers. The global depth map is refined by a local fine-scale network that codes more local details such as objects and wall edges

(Figure 2.30). The loss function of the algorithm is scale-invariant mean squared error



		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Figure 2.30: The network structure used in [46] consists of a global coarse-scale network and a local fine-scale network. In the global network, the first five layers are convolutional layers with max-pooling and rectified linear units as activation functions. The latter two layers are fully connected layers. The local network is a relatively shallow convolutional network. Its features are concatenated with the global depth map followed by two convolutional layers. Figure taken from [46].

(Equation 2.10).

$$
\begin{aligned}
D(y, y^*) &= \frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}(\sum_i d_i)^2 \\
&= E[(d_i - E[d_i])^2] \\
&= E[d_i^2] - (E[d_i])^2 \\
&= VAR(d_i)
\end{aligned}
\tag{2.10}
$$

where y and y^* are the predicted and ground truth depth maps respectively, n is the number of pixels indexed by i, and $d_i = \log \frac{y_i}{y_i^*}$.

Later, Eigen et al. [45] improved their algorithm by adding more layers, a new loss function (Equation 2.11), and joint estimation of depth with surface orientation and se-

	Eigen [46]	FCRN [99]	DORN [55]
Number of Layers	23	50	20
Number of Parameters	2M	63M	108M
Loss Function	Error Variance	berHu	Ordinary Regression Loss
Framework	Theano	Tensorflow	Caffe
RMS Error	0.871	0.573	0.509

Table 2.7: A comparison of different end-to-end depth prediction networks.The RMS errors were evaluated on the NYU v2 dataset [149].

mantic category. They employed a new loss function given by:

$$D(y, y^*) = \frac{1}{n}\sum_i d_i^2 - \frac{1}{2n^2}(\sum_i d_i)^2 + \frac{1}{n}\sum_i[(\bigtriangledown_x d_i)^2 + (\bigtriangledown_y d_i)^2] \tag{2.11}$$

where $\bigtriangledown_x d_i$ and $\bigtriangledown_y d_i$ are the horizontal and vertical image gradients of the error.

In 2016, Laina et al. [99] trained a deep ResNet [68] concatenated with up-sampling layers on the single-view range estimation problem, showing that more layers and using reverse Huber(berHu) as a loss function achieves better accuracy. The reverse Huber(berHu) is formulated as:

$$berHu(x) = \begin{cases} |x| & \text{if } |x| \leq= c \\ \frac{x^2+c^2}{2c} & \text{if } |x| > c \end{cases} \tag{2.12}$$

Later, Fu et. al [55] introduced an ordinary regression loss function and a multi-scale network structure. The new loss function improved the convergence speed and achieved higher accuracy. The multi-scale structure avoided unnecessary spatial pooling and allowed multi-scale information to be processed in parallel. A comparison of different end-to-end depth prediction networks is shown in Table 2.7.

Discussion: End-to-end deep networks achieve better performance than the graph based approach on the NYU v2 [149] dataset. This is due to the powerful architecture and millions of free parameters in the network model. Similar to graph based models, one of the limitations is over fitting on the training dataset which leads to poor generalization. Another disadvantage is that, due to computational constraints, the resolution of reconstruction is limited(eg. 147×109 pixels).

2.5.3 Multi-View Supervised Learning

Collecting RGB-D data is a long and expensive process. It requires a specialized depth sensor and a lot of post-processing to obtain high quality depths data. There are unsupervised alternatives that use stereo cameras or video stream.

2.5.3.1 Learning Single-View Estimation with a Stereo Camera

Garg et al. [57] suggested that low cost stereo camera data could be used to train a deep network to perform depth estimation from single images. They proposed a stereopsis-based auto-encoder to train the depth model with stereo image pairs (Figure 2.31). The left image I_1 and right image I_2 are used in training, and the output is the depth map d. This depth map is used to predict the left image I_W from the right image I_2. The loss function E of this CNN model is the $L2$ loss between I_1 and I_w with $L2$ regularization:

$$E = \sum_{i=1}^{N} E^i_{recons} + \gamma E^i_{smooth} \tag{2.13}$$

$$E^i_{recons} = \sum \left\| I^i_w(x) - I^i_1(x) \right\|^2 dx \tag{2.14}$$

where E^i_{recons} is the $L2$ error between left image I^i_1 and the reconstructed image I^i_w. γ is the weight for the regularization term. x is the index of pixels in the left image. Assuming epipolar lines fall along horizontal scan lines, the reconstructed image I^i_w can be synthesized as $I^i_2(x + fb/d_i(x))$, where $d^i(x)$ is the predicted depth value for pixel x, f is the focal length of the cameras and b is the distance between the two cameras. The smoothing term of the loss function is defined as $L2$ regularization on the disparity gradient:

$$E^i_{smooth} = \left\| \bigtriangledown D^i(x) \right\|^2 \tag{2.15}$$

Discussion: Unlike range map data, stereopsis data is easy to obtain from low cost stereo cameras. With nearly unlimited data, a large variation of data can be obtained easily to train a depth estimation model that leads to better generalizability. The disadvantage of this method is that its accuracy is limited by the hardware to only work for nearby surfaces. The theoretical depth resolution of a stereo camera dZ_c as a function of distance Z [98] is:

$$dZ_c = \frac{Z^2}{fb} dp_x \tag{2.16}$$

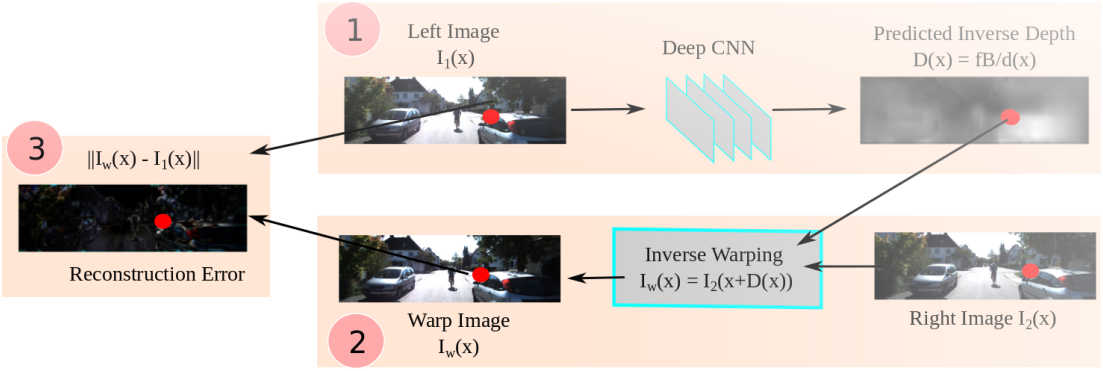


Figure 2.31: The stereopsis based auto-encoder: Component 1 is a convolutional neural network encoder, which generates an inverse range map $D(x)$. Component 2 uses the inverse range map and the right image to generate a synthesized version of the left image. Component 3 calculates the reconstruction error between the real left image and the synthesized image. Figure taken from [57].

where f is focal length, b is baseline and dp_x is disparity accuracy.

As the distance Z increases, the depth resolution of a stereo camera increases as the square of the distance. Thus stereo cameras are only recommended for close-range vision (e.g. 1-20m for ZED stereo camera [122]). Due to this limitation on stereo camera hardware, the single-view 3D reconstruction model trained from stereo images only works well for close-range environments. Also note that the method will fail for featureless or highly reflective surfaces.

2.5.3.2 Depth Estimation Model using Video Streams

Video streams can also be used to train a depth prediction network. Zhou et al. [187] proposed two jointly trained CNNs (Figure 2.32) that consist of a depth CNN and a pose CNN. Training data consists of triplets of consecutive video frames (I_{t-1}, I_t, I_{t+1}). The depth CNN model takes the middle frame (I_t) as input, and all three frames are input into the pose CNN model to generate two relative camera poses ($\hat{T}_{t-1}$, $\hat{T}_{t+1}$). Each of these poses together with estimated depth ($\hat{D}_t$) is used to generate the synthesized estimation of I_{t-1} and I_{t+1} from I_t. Let p_t denote the homogeneous coordinates of a pixel in the target view and K denote the camera intrinsic matrix. The projection of target view point p_t onto

source view point $p_s$is:

$$p_s \sim K\hat{T}_{t\to s}\hat{D}_t(p_t)K^{-1}p_t \tag{2.17}$$

where $\hat{T}_{t\to s}$ is the relative camera pose matrix estimated from the pose CNN model. $\hat{D}_t(p_t)$ is the predicted depth value at pixel coordinate p_t in the input image. p_s is the projected coordinate. The depth and pose CNN networks are trained jointly by minimizing the error between the real image and the synthesized image.

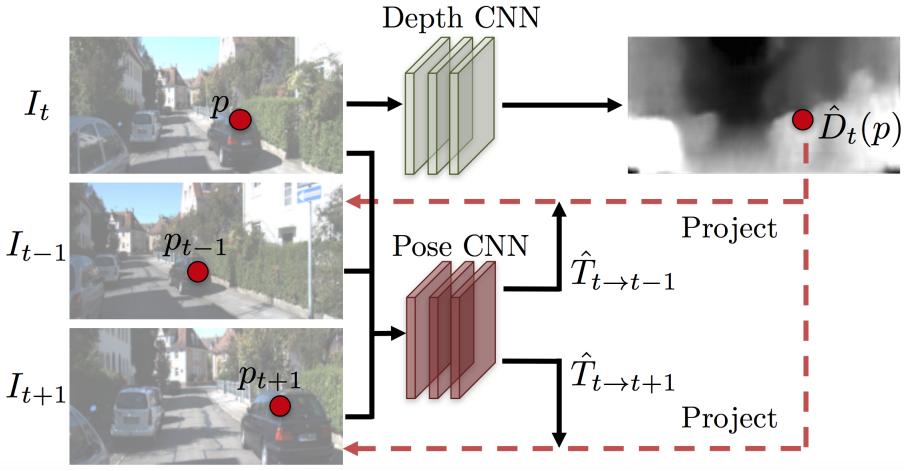


Figure 2.32: Overview of the view synthesis algorithm. The depth network takes I_t as input to generate a depth map $\hat{D}_t$. The pose network generates two camera poses ($\hat{T}_{t-1}$ and $\hat{T}_{t+1}$) from I_{t-1} and I_{t+1} relative to I_t. The image I_t is used to generate a synthesized version of I_{t-1} and I_{t+1} using camera poses and the depth map. [187]

Discussion: Video streams are much easier to collect than range data. A performance comparison of algorithms on the KITTI dataset [54] is shown in Table 2.8, where Eigen [46] and Liu [107] are deep learning algorithms trained on range data.

Methods	Eigen [46]	Liu [107]	Garg [57]	Zhou [187]
Data Source	Range	Range	Stereo	Video
RMSE	6.307	6.523	5.104	6.565

Table 2.8: Performance comparison of depth estimation algorithms on the KITTI dataset [54].

A motion-based method can only be accurate if the camera translation is reasonably

large relative to the distance of the surface. It will fail for featureless or highly reflective surfaces.

2.5.4 Discussion

Machine learning approaches have advantages and disadvantages. They are able to learn 3D reconstruction models from an enormous amount of range map, stereo imaging, or video data. Given a single 2D image, these methods can generate a dense range map which is suitable for applications such as robotics and self-driving cars. These methods work on a large variety of scenes as long as there are sufficient data to train.

On the other hand, due to computational constraints, these methods fail to create crisp clear representations and can only generate low resolution range maps. In these methods, networks are less explainable. Furthermore, these methods tend to over-fit the training data and lead to poor generalization.

2.6 Conclusion

A critical difference between the methods of Section 2.4 and 2.5 is the nature of the model they produce. In section 2.4, a geometrical model is generated. In section 2.5, a range map is generated. The geometrical model is useful for built environments, especially where the Manhattan world assumption holds. The range maps are collections of points that are more suitable for unstructured scenes. In terms of application, geometrical methods have less expensive hardware costs. The geometrical methods can be deployed onto embedded machines or micro-processors that have limited computational power and battery life. The machine learning approach is generally more expensive. It requires an expensive GPU to run and a large amount of disk space to store their networks. One possible solution for the GPU requirement is to migrate the data processing onto cloud servers.

Single-view 3D reconstruction has many unsolved problems. For the geometric approaches, the reconstruction either generates a 3D line drawing [131], box layout [70, 169, 101, 147, 131, 70, 78], or Photo Pop-up [73, 62, 124]. There are limitations for those three methods. The 3D line drawing does not have surface information. The box layout has surfaces, but it is only applicable to simple indoor room scenes. The Photo Pop-up only identifies rough categorical surface types. The next step for the geometric approach is to develop a new way to represent precise 3D surface models for more general Manhattan scenes.

The machine learning approaches generate low resolution range maps which fail to create crisp clear representations. These deep networks lack explainability. Furthermore, these methods tend to over-fit the training data which leads to poor generalization. The future directions for machine learning based approaches are:

1. Create a way to generate high resolution crisp clear range maps.
2. Improve the explainability of the networks.
3. Improve generalization.
4. Compress the network for greater efficiency.
5. Since the geometric and deep learning approaches are complementary, a promising research direction is to study how the two approaches can be combined.

Chapter 3

MCMLSD: Line Segment Detection

3.1 Project Description

The goal of this project was to develop a high quality line segment detector which forms the foundation for later projects. When a piecewise planar scene is perspectively projected onto an image plane, properties such as angles, distances and ratios of distance are distorted. However, one property that is preserved over the projective transformation is straightness. The straight line in 3D remains straight when projected to 2D. These line segments in an image provide valuable information about the 3D geometry of the scene. In this work, we have evaluated our algorithm on YorkUrbanDB[40] and Wireframe[75] datasets which are , to my knowledge, the only available Manhattan line datasets at the time of writing.

My contribution to this project consists of:

1. Improving the runtime efficiency of the algorithm from 105 seconds per image to 2.8 seconds per image, on average.
2. Extending the algorithm to work on any image resolution.
3. Performing additional experiments to compare with recent deep neural network algorithms.
4. A more extensive evaluation and comparison with state-of-the-art algorithms on YorkUrbanDB[40] and Wireframe[75] datasets.

5. Devising an improved line segment ranking strategy.

6. Evaluating new line segment evaluation methods.

Related Publications:

- E. J. Almazàn, R. Tal, **Y. Qian**, and J. H. Elder. MCMLSD: A dynamic programming approach to line segment detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5854–5862, July 2017

- J. H. Elder, E. J. Almazàn, **Y. Qian**, and R. Tal. MCMLSD: A probabilistic algorithm and evaluation framework for line segment detection, 2020 (In preparation)

3.2 Introduction

Much of our visual world can be approximated as piecewise planar, particularly in built environments. The boundaries and creases of these piecewise planar surfaces project to the image as line segments, and as a consequence the accurate detection of line segments continues to be one of the most important low-level problems in the field of computer vision. Line segments are important features for many tasks, including feature matching across views [145], vanishing point detection [92] and 3D reconstruction [125, 185, 71].

Two frameworks have been popular for line segment detection: perceptual grouping and global Hough analysis.

3.2.1 The Perceptual Grouping Approach

In the perceptual grouping framework, a set of heuristics typically based upon geometric grouping cues (e.g., proximity, good continuation) is used to group roughly collinear local features (e.g., edges or vectors tangent to isophotes) into extended line segments, which are evaluated according to some quality of fit measure. An early example is the hierarchical heuristic framework developed by Boldt and colleagues [19]. More recent multi-stage grouping efforts include the SSWMS approach of Nieto et al. [119], which involves an iterative selection of image points with a strongly oriented gradient structure, followed by

an iterative growing process, the approach of Lu et al. [110], which involves both linking and splitting, and the biologically inspired approach of Liu et al. [109], which employs 'simple cell' filters to detect local oriented structure, 'complex cell' mechanisms that locally integrate these responses and 'hyper-complex' mechanisms to detect endpoints.

An alternative to this multi-stage grouping approach is to analyze the covariance matrix of image locations in a set of connected edges and label a set as a line segment if the smallest eigenvalue falls below a threshold [63, 106]. While beautifully simple, these methods are not robust to gaps or intersections in the edge map.

Another issue in this perceptual grouping framework is that some threshold on the quality of fit measure must be applied in order to discriminate 'true' line segments from false conjunctions that might arise by chance. This issue was addressed in the LSD framework introduced by von Gioi et al. [166] and based on earlier work by Desolneux et al. [41]. In this framework the so-called *a-contrario* approach is used to explicitly compute the probability that inferred line segments might have occurred by chance, given a maximum entropy model of the edge map. (This is related to the minimum reliable scale null hypothesis testing framework for edge detection developed by Elder & Zucker [50].) While this approach does not eliminate the need for a threshold, it transfers the threshold to a quantity (e.g., expected number of false positives per image) that is much easier to set rationally. A much faster version of this method dubbed EDLines was later introduced by Akinlar & Topal [2].

Recent work in this area has focused on trying to discriminate salient or important line segments from less important 'background' segments. Kim et al. [87] used a combination of luminance and geometric features to select the most significant edges, reporting superior performance to LSD on two test images. Brown et al. [20] used a measure of divergence between colour statistics on either side of a hypothesized line segment to favour salient segments. The method outperformed LSD and Hough methods using quantitative measures of repeatability and registration accuracy on image pairs (see Section 3.5 below).

3.2.2 The Hough Approach

A drawback of the perceptual grouping approach is that local decisions are made before potentially relevant global information can be brought to bear. The Hough approach avoids

this problem by accumulating edges over the entire image into a histogram of potential line positions and orientations. Accuracy can be improved by modeling uncertainty in local edges and propagating that uncertainty to the Hough map [154].

While the Hough approach to line detection has the advantage of integrating information globally, identifying the endpoints that define the extent of the line segment in the image is not necessarily straightforward. A number of methods scan the detected lines in the image space looking for a maximal chain of connected or nearly-connected edges [61, 113]. Others have attempted to identify the endpoints of each line segment by analyzing the exact shape of a characteristic 'butterfly' pattern around the associated peak in the Hough map [82, 56, 180, 181, 179]. One major limitation of this approach is that only one segment can be found per line, whereas in built environments it is quite common to find multiple co-linear segments.

3.3 Our Approach

The advantage of the Hough approach is that it can integrate all evidence for line hypotheses prior to inference. The perceptual grouping approach, on the other hand, allows endpoints to be detected more directly, and permits the identification of multiple segments per line.

Our two-stage method, an early version of which was published at CVPR 2017 [4], combines the advantages of these two approaches. In the first stage we employ the probabilistic Hough method of Tal & Elder [154] to identify globally optimal lines. In the second stage we search each of these lines in the image for the segment(s) that gave rise to it.

The key observation that recommends this approach is that narrowing the search for segments from the 2D image to 1D lines allows the problem to be modeled as the labelling of hidden states in a linear Markov chain model. The problem of determining the maximum probability (MAP) assignment of segments can then be shown to have an optimal substructure property that leads to an exact dynamic programming solution in linear time.

The benefits of this approach are several:

1. Each of the lines identified by a peak in the Hough map results from careful accu-

mulation of the global evidence for the line, and thus will more accurately identify the position and orientation (ρ, θ) parameters of the line segments than will a few local edges.

2. The lines identified by the probabilistic Hough method have a natural order according to their significance in the Hough map, allowing the line segment search to be limited to the most significant lines.

3. In urban scenes, co-linear line segments are common, arising from architectural repetition seen in cladding, windows, etc. Unlike many Hough methods, our approach allows multiple segments to be recovered for each line.

4. Limiting search to a line allows the problem of determining maximum probability segments to be solved exactly, using dynamic programming, in linear time.

3.4 The Deep Learning Approach

Most recently, two deep neural network (DNN) algorithms for line segment detection have been reported. The Wireframe algorithm [75] is based upon a stacked hourglass network [118] that takes as input a 320×320 pixel RGB image and produces as output a 320×320 pixel map encoding the estimated locations and lengths of the line segments in the image. In particular, if a pixel is judged to lie on a segment, the pixel value indicates the estimated length of that segment, while a pixel not lying on any segment should have a value of 0. The network is trained to minimize an L^2 loss and the scalar output is thresholded to filter out shorter or lower-confidence segments.

Note that the Wireframe network delivers a raster map - essentially an edge map where edges are constrained to lie on straight lines - rather than a vectorized description of the locations, lengths and orientations of the line segments in the image. To obtain the latter, a parallel network is trained to detect junctions in the image, and then a somewhat complex process is followed to segment the edge map into line segments between junctions.

The Attraction Field algorithm [182] also employs a deep network, but in a rather different way. The key insight is that it is easier for a deep network to map the input image to a dense pixel grid of values than to a sparse boundary map. Thus to adapt the

problem of line segment detection to deep networks, each sparse ground truth line segment map is used to generate a dense ground truth Attraction Field Map (AFM) that represents the vector displacement to the nearest line segment point at every pixel in the image. A network is then trained to estimate this dense AFM. At inference, the estimated dense AFM is reduced to a sparse line segment map using a 'squeeze module' that accumulates votes for line segment pixels by summing discretized displacement vectors over all pixels in the AFM.

The authors experiment with two network architectures: A U-Net [137] and a modified U-Net, referred to by the authors as *a-trous*, that uses the ASSP module of DeepLab v3+ [25] and the skip connections of ResNet [68]. An L^1 loss on the AFM is employed.

As for Wireframe, the AFM approach produces a raster map of edge pixels that must then somehow be grouped into line segments. In the case of AFM, a heuristic, iterative, greedy region-growing approach similar to that used in LSD[166] is employed.

Both Wireframe and Attraction Field algorithms are trained on the Wireframe training dataset.

Both Wireframe and AFM are claimed to outperform all prior non-DNN approaches to line segment detection, including our MCMLSD approach [4]. However, we argue in this paper that the evaluation performed in these two DNN papers is limited, and that a more careful analysis reveals that for the problem of line segment detection (not edge detection), MCMLSD and indeed some older non-DNN algorithms substantially outperform both of these DNN approaches on a number of metrics. Given that these networks also involve tens of millions of free parameters, we argue that more explainable methods such as MCMLSD may be preferable for many applications.

3.5 Prior Evaluation Methodology

Due in part to a lack of high quality labelled ground truth, most traditional line segment detection methods were evaluated only qualitatively on real imagery [19, 56, 166, 2, 106]. More recently, quantitative evaluations have been conducted based on datasets consisting of pairs of images related by a known homography [20]. This is a promising method, but it does suffer from two potential drawbacks. First, it is restricted to an analysis of co-planar line segments. Second, the evaluation presupposes that the goal of line segment detection

is for the association of these segments across images for the purposes of homography or disparity estimation. However there are many other possible applications - single view reconstruction, for example.

While task-specific evaluation methodologies may be appropriate in some cases, it would be preferable to have an evaluation method that is more general. In this work we present a new methodology for quantitative evaluation of line segment detectors on real images that does not assume a specific task, using images from the YorkUrbanDB [40] (`www.elderlab.yorku.ca/YorkUrbanDB/`) and Wireframe (`https://github.com/huangkuns/wireframe`) datasets.

3.6 Algorithm

3.6.1 Line Detection

One problem with traditional Houghing methods is that noise in the observations tends to cause each line to generate multiple peaks in the Hough map. To address this issue we employ a probabilistic Hough transform method [154] (code available from `elderlab.yorku.ca/resources`). The method uses edges detected by the multi-scale Elder & Zucker edge detector [50], models uncertainty in the location and orientation of the detected edges and propagates this uncertainty to the Hough map. This propagation of uncertainty produces a smooth Hough map that is roughly resolution invariant and greatly reduces the multiple response problem. The problem is mitigated further by a sequential line extraction step in which each peak in the Hough map is visited in descending order of significance, and edges contributing to the peak are subtracted from the Hough map when it is visited.

3.6.2 Line Segment Detection

Each selected peak in the Hough map identifies a line that extends from one of the image borders to another. In general, this line is only partially occupied by line segments in the image. The goal is now to find these segments, based on the location and orientation of nearby edges.

Prior work [40] suggests that most edges generated by a line and detected by the Elder & Zucker edge detector lie within one pixel of the line. To ensure we capture all edges related to a line we extend our search to all pixels within two pixels of the line (Fig. 3.1). The orthogonal projections of these pixel locations onto the line then define an ordinal sampling $i \in [1, \ldots, N]$ of the line. We let x_i represent the binary hidden segment state (ON or OFF) indicating whether a visible segment is present at position i on the line, d_i the distance from the line to the associated pixel and y_i the associated image observation at that pixel. Each observation y_i consists of 1-2 features:

1. A binary variable e_i indicating whether an edge exists at this pixel.
2. The angular deviation θ_i of the edge from the line, if the edge exists.

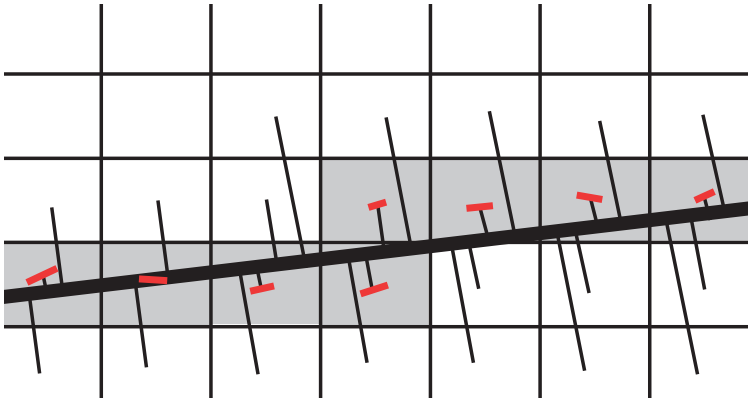


Figure 3.1: Orthogonal projections (thin black lines) of all pixels within two pixels of a detected line (thick black line) define an ordinal sampling of the line $i \in [1 \ldots N]$. Pixels within this band occupied by edges (shown red on grey) with orientations similar to the line support the assignment of the ON state for the associated segment variable x_i at sampled line locations.

These features provide information about the probable state of the line at the associated position:
$p(y_i|x_i) \propto p(e_i = 1|x_i, d_i)p(\theta_i|x_i, e_i = 1)$ for edge pixels.
$p(y_i|x_i) \propto p(e_i = 0|x_i, d_i)$ for non-edge pixels.
(Note that we have assumed that the angular deviation θ_i is independent of the distance d_i of the pixel from the line.)

We learned these distributions from the 640×480 pixel images and hand-labelled ground truth lines from the YorkUrbanDB training dataset [40]. Figs. 3.2(a-b) show the likelihoods $p(e_i = 1|x_i = \mathrm{ON}, d_i)$ and $p(e_i = 1|x_i = \mathrm{OFF}, d_i)$ as functions of d_i for ON and OFF states respectively. We represent these distributions as histograms. (The likelihoods for non-edge observations $p(e_i = 0|x_i = \mathrm{ON}, d_i)$ and $p(e_i = 0|x_i = \mathrm{OFF}, d_i)$ are the complements of the edge likelihoods.)

Figs. 3.2(c-d) show the probability $p(\theta_i|x_i, e_i = 1)$ as a function of the angular deviation θ_i for ON ($x_i = 1$) and OFF ($x_i = 0$) states, respectively. For the ON state we approximate the heavy-tailed distribution as a mixture of a uniform and a Gaussian distribution (shown in red). For the OFF state we employ a histogram representation.

Given these observations, we wish to determine the sequence of hidden states $x_1, ..., x_N$ that maximizes

$$p(x_1, \ldots, x_N|y_1, \ldots, y_N) \propto p(y_1, \ldots, y_N|x_1, \ldots, x_N)p(x_1, \ldots, x_N) \tag{3.1}$$

We assume that, when conditioned on the hidden states x_i, the observations y_i are mutually independent and independent of all x_j, $j \neq i$. We further assume that the hidden states are first order Markov so that Eqn. 3.1 becomes

$$p(x_1, \ldots, x_N|y_1, \ldots, y_N) \propto p(y_1|x_1)p(x_1)\prod_{i=2}^{N} p(y_i|x_i)p(x_i|x_{i-1}) \tag{3.2}$$

The Markov assumption implies an exponential distribution of segment lengths; for the YorkUrbanDB training dataset we have verified that this distribution is indeed very close to exponential for segments down to $\sim$ 15 pixels in length. (For smaller segments the density falls off, possibly due to difficulties in hand-labelling shorter segments.)

Table 3.1 shows values for the priors $p(x_1)$ and $p(x_i|x_{i-1})$, estimated from the 51 640×480 pixel images from the YorkUrbanDB labeled training dataset [40]. (Note that since the probabilities for ON and OFF states sum to 1 there are only 3 free parameters.) We make the approximation that $p(x_i|x_{i-1})$ is independent of the variation in spacing between points on the line. Since the average segment in the YorkUrbanDB generates more than 500 point samples, errors due to this approximation tend to average out.

The standard errors for these parameter estimates are relatively small, and we have verified that variation within this range has negligible effect on results. While these parameters are specific to the YorkUrbanDB dataset and may therefore be sub-optimal for

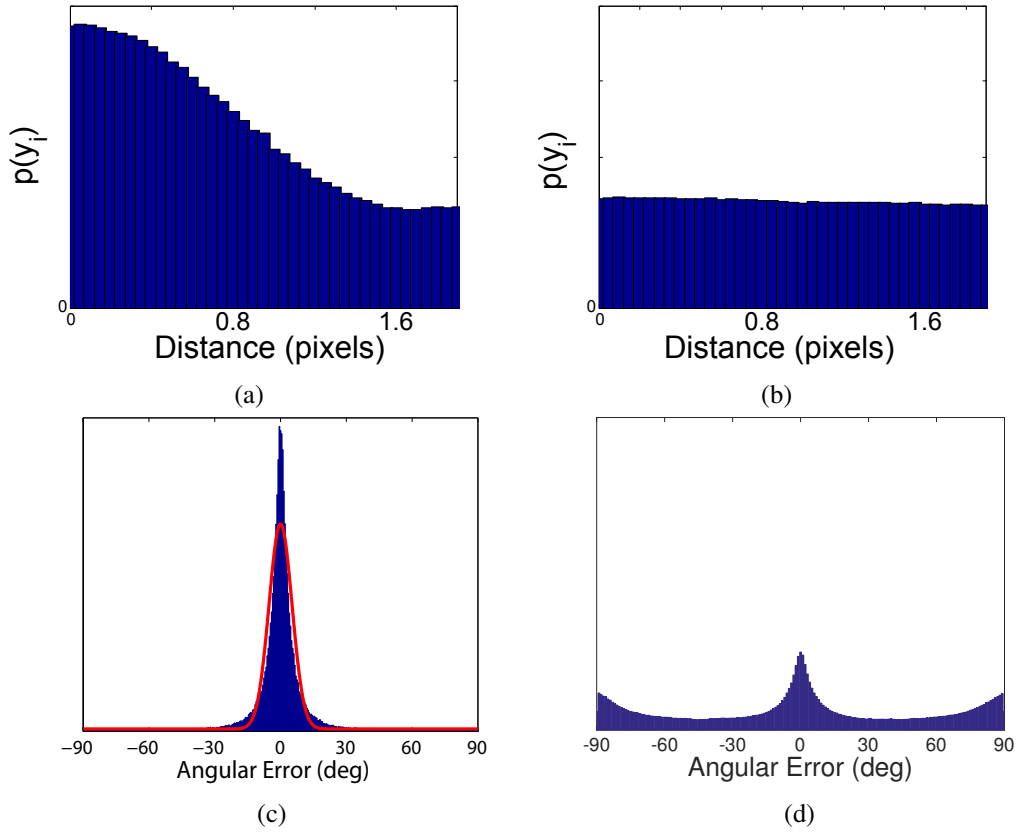


Figure 3.2: Likelihoods for line segment extraction, learned from the YorkUrbanDB training dataset [40]. (a-b) Likelihood $p\left(e_i|x_i, d_i\right)$ for distance d_i of observations from line for (a) ON ($x_i = 1$) and (b) OFF ($x_i = 0$) states. (c-d) Probability $p\left(\theta_i|x_i, e_i\right)$ for the angular deviation θ_i of observed edges from the line for (c) ON ($x_i = 1$) and (d) OFF ($x_i = 0$) states.

other kinds of imagery, they *can* be generalized to other image resolutions. Assuming that the number of segments per line and their relative length are functions of the scene and not the sensor, $p(OFF)$ and $p(ON)$ will be resolution-invariant and the probability of state changes will vary inversely with resolution. For example, doubling the resolution to 1280×960 pixels will halve the probability of transition from OFF to ON or ON to OFF.

Table 3.1: Prior marginal probabilities $p(x_i)$ and conditional transition probabilities $p(x_i|x_{i-1})$ for the hidden segment state x_i, derived from the YorkUrbanDB training dataset.

Parameter	**Mean**	**Std. Err.**
$p(OFF)$	0.75	0.0079
$p(ON)$	0.25	0.0079
$p(OFF\|OFF)$	0.9986	0.0001
$p(ON\|OFF)$	0.0014	0.0001
$p(ON\|ON)$	0.9949	0.0004
$p(OFF\|ON)$	0.0051	0.0004

The factoring of the global probability of the line segment configuration along the line confers an optimal substructure property that allows a dynamic programming solution to the problem of finding the maximum a posteriori configuration. In particular, let the cost function $C_i(j)$ represent the minimum negative log probability of all sequences $\{x_1,\ldots,x_i\}$ ending in state $x_i = j$. Then the maximum probability sequence of states over the whole line is the sequence that minimizes $\min_j C_N(j)$.

Defining the cost of transitioning from state j at location $i-1$ to state k at location i as

$$c_i(j,k) = -\log\left(p(y_i|x_i=k)p(x_i=k|x_{i-1}=j)\right), i=2,\ldots,N \tag{3.3}$$

$$C_1(k) = -\log\left(p(y_1|x_1=k)p(x_1=k)\right) \tag{3.4}$$

$$C_i(k) = \min_j\left(C_{i-1}(j)+c_i(j,k)\right), i=2,\ldots,N \tag{3.5}$$

Thus the cost function $C_i(k)$ can be computed sequentially from $i=1$ to $i=N$ in $O(N)$ time (Fig. 3.3). In order to recover the maximum probability configuration, an

auxiliary data structure containing

$$\hat{s}_i(k) = \arg\min_j (C_{i-1}(j) + c_i(j,k)) \tag{3.6}$$

is maintained, allowing the maximum probability configuration to be unwound from x_N back to x_1.

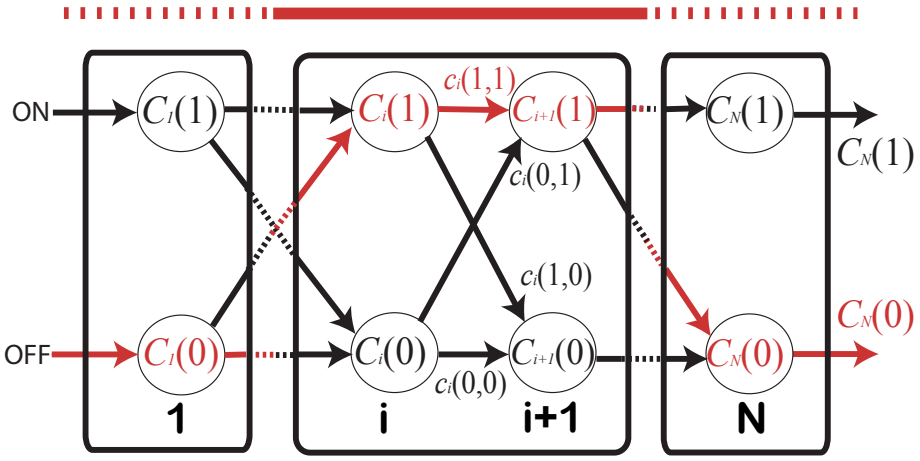


Figure 3.3: The sequence of segment state variables x_i are assumed to form a Markov chain. To compute the MAP solution we build a trellis table from the first line position $i = 1$ to the last line position $i = N$ that identifies the minimum cost (negative log probability) to reach either possible state (ON or OFF) at each position i. The selected MAP path is shown in red, and the resulting ON/OFF states are indicated by the solid/dashed line above the trellis.

Once a line segment is detected all associated edges (i.e., edges within two pixels of the segment) are removed from the image. This serves to reduce the incidence of multiple detections for the same segment.

3.7 Ranking

Having extracted MAP segments for each line in the image, we would like to rank their significance. This will allow downstream applications to select only the number of segments needed to support their application, and can serve to eliminate low-ranked noise segments. Our Markov chain model allows us to approach the ranking problem from a probabilistic

perspective. In particular, we evaluate the following four probabilistic methods for ranking a segment of length M extending from position i to position $i + M$:

Ranking Method 1. Posterior probability of line segment.

$$p(x_{i...i+M} = ON|y_{i...i+M})$$

This ranking criterion will maximize the expected number of segments with no false alarms.

Ranking Method 2. Posterior probability of line segment multiplied by length.

$$p(x_{i...i+M} = ON|y_{i...i+M}) * M$$

This criterion will maximize the expected total length of segments with no false alarms.

Ranking Method 3. Posterior odds for fully ON vs fully OFF configurations.

$$\frac{p(x_{i...i+M} = ON|y_{i...i+M})}{p(x_{i...i+M} = OFF|y_{i...i+M})}$$

Ranking Method 4. Sum of marginal posterior probabilities for ON states. The forward-backward algorithm is used to compute the posterior probability at each location.

$$\sum_{j=i}^{i+M} p(x_j = ON|y_{i:i+M})$$

This measure reflects the expected number of ON samples on the segment, and thus will maximize the expected number of correctly labelled locations within the segment.

3.8 Evaluation Methodology

It is important to evaluate line segment detection algorithms on real, complex images. Prior evaluations have generally been qualitative (i.e., visual). Recent efforts to quantify the evaluation require pairs of images related by a known homography, and are perhaps thus best suited for matching tasks [20]. Here we propose an alternative quantitative evaluation methodology that does not assume the existence of image pairs or known homographies and thus could be applicable for a broader range of tasks.

Our proposed evaluation method does require an image dataset in which important segments have been labelled. Here we employ two. The YorkUrbanDB dataset [40], which consists of 102 images of urban scenes, randomly divided into training and test subsets of 51 images each. In each image, major line segments that conform to one of the three so-called Manhattan directions [34] (i.e., vertical or horizontal and conforming to the main directions of orthogonal walls, streets, etc.) have been identified and labelled by hand. This database has been used widely to train and evaluate algorithms for vanishing point detection [156], line detection [11] and Manhattan frame estimation [40, 154]. We also evaluate the more recent and much larger Wireframe dataset [75], which consists of 5,462 images (5,000 for training, 462 for test) of man-made scenes.

We assume that the line segment detector under evaluation returns a list of line segments in ranked order. We sample each ground truth and detector segment uniformly with a sample spacing of one pixel and use these point samples to evaluate the detector as a function of the number k of top-ranked segments selected, varying k from 10 to 500.

For each value of k we first identify potential point matches as those (ground truth, detector) point pairs lying within a threshold distance of $2\sqrt{2}$ pixels of each other. This threshold was selected to associate any pair of lines that could potentially appear in the image with less than a one-pixel intervening gap. We then sort these candidate matches by Euclidean distance and accept matches in greedy fashion starting with the smallest distance, matching each point at most once, and thus arriving at a near-optimal bipartite match. Having associated ground truth and detector points, we employ the Hungarian algorithm [96] to identify the optimal bipartite match between ground truth and detector segments that maximizes the total number of points matched.

Now that we have a 1:1 association between ground truth and detector segments, it remains to evaluate the quality of this association. We propose three evaluative measures.

3.8.1 Recall as a Function of the Number of Segments

We can compute a measure of recall as the number of ground truth point samples matched to detector samples, divided by the total number of ground truth point samples. This measure of recall is problematic if we allow matches without regard to the segments on which the points lie, as it does not penalize under-segmentation (joining multiple short

segments into a single long segment) or over-segmentation (breaking up a long segment into multiple short segments).

However, constraining matches to lie on 1:1 associated segments solves both of these problems. In the case of under-segmentation, only one of the shorter ground truth segments is matched, leading to a high penalty. In the case of over-segmentation, only one of the detector segments is matched, again generating a high penalty.

Without additional constraints, using recall by itself is still problematic, as it is biased toward detectors that report a larger number of segments, thereby maximizing the probability of detecting ground truth points. We address this by comparing recall as a function of the same number k of segments reported.

3.8.2 Recall as a Function of Total Segment Length

There is still a potential bias in this recall-vs-k measure. Neglecting co-linear ground truth segments, the method can be biased toward detectors that report segments of maximal length (i.e., global lines) as this minimizes the risk of missing ground truth points. To address this potential bias, our second performance measure reports recall as a function of the sum L of the lengths of detected segments. This severely penalizes detectors that report over-long segments.

3.8.3 Precision-Recall

Our third and final performance measure is conventional precision-recall. We can take as a measure of precision the number of ground truth point samples matched to detector samples, divided by the total number of detector point samples. Again, by enforcing a 1:1 matching at the segment level, both under-segmentation and over-segmentation are penalized.

To facilitate future comparisons, the code that performs these evaluations as well as the code for the MCMLSD algorithm is available online at `elderlab.yorku.ca/resources`.

3.8.4 Limitations of Precision-Based Measures of Performance

Since the YorkUrbanDB dataset does not provide a complete labelling of all segments in an image, detection of a segment that is not in the dataset does not necessarily represent an error. For this reason, the absolute precision values reported here should be interpreted with caution. Nevertheless, since the segments labelled in the YorkUrbanDB dataset are highly-visible Manhattan features projecting from prominent structures in the scene, it is reasonable to expect a superior detector to rank these highly, and therefore attain higher *relative* precision values compared to inferior detectors.

The creators of the Wireframe dataset attempted to label all the line segments associated with the scene structures. Unlike the YorkUrbanDB dataset, these are not restricted to Manhattan lines, and so one expects the dataset to contain a more complete labelling, potentially allowing for higher precision. However, the authors also avoided labelling line segments in what they considered 'texture'. This includes straight line segments projecting from regular tiling and cladding patterns on horizontal and vertical surfaces, which are very common in the built environment. Since these can be quite useful in establishing surface orientation for both human and computer vision systems, for many applications one would want a line segment detector to detect these, yet such detectors will tend to generate lower precision on the Wireframe dataset.

Given the limitations of precision measures for these two datasets, we feel it is important to consider multiple different measures of performance when evaluating and comparing algorithms, and so we report performance using all three measures in what follows.

3.9 Results

3.9.1 Ranking

Our first goal is to evaluate the four candidate ranking methods discussed in Section 3.7. Using the default Hough resolution recommended by Tal & Elder [154] ($\Delta\rho = 0.2$ pixels, $\Delta\theta = 0.1$ deg), we find that our MCMLSD algorithm generates an average of 414 lines and 488 line segments for each 640×480-pixel image from the YorkUrbanDB training dataset. Note that not all lines generate a segment and some generate several segments.

Fig. 3.4 shows the 10 top-ranked segments produced by each of our four ranking methods on an example image. We find that the multiplicative nature of the first criterion favours short high-confidence segments. This problem can be addressed by multiplying by segment length (Method 2), forming a contrast between purely ON and purely OFF configurations (Method 3), or summing the ON point marginals (Method 4) to estimate the number of correctly labeled points.

(a) Method 1 (b) Method 2 (c) Method 3 (d) Method 4

Figure 3.4: 10 top-ranked segments for four ranking methods on example image.

Figure 3.5 shows the recall for each of these ranking methods as a function of the number of segments returned, on the YorkUrbanDB training dataset. Note that ranking methods 1-3 are all based upon the probability that the segment is exactly correct, i.e., all points on the segment have the ON state. However, our method of evaluation measures the recall, which is the proportion of ground truth segment points successfully detected. Thus what we really care about are the marginal probabilities, not the configuration probability, since the expected recall is the sum of the marginal probabilities of the ON state over the segment, which is exactly the quantity computed by Method 4, which we adopt as our ranking method of choice. We call the resulting algorithm the Markov Chain Marginal Line Segment Detector (MCMLSD) to capture the importance of the Markov chain model of the line as well as the probabilistic ranking that maximizes the expected number of correctly labelled points on the segment.

3.9.2 Hough Resolution

Having selected the ranking method, we fine-tune the Hough resolution parameters ($\Delta\rho$, $\Delta\theta$) on the YorkUrbanDB training data, computing recall for the top 100 lines over a range of parameter values and then using kernel regression with bandwidths selected by leave-

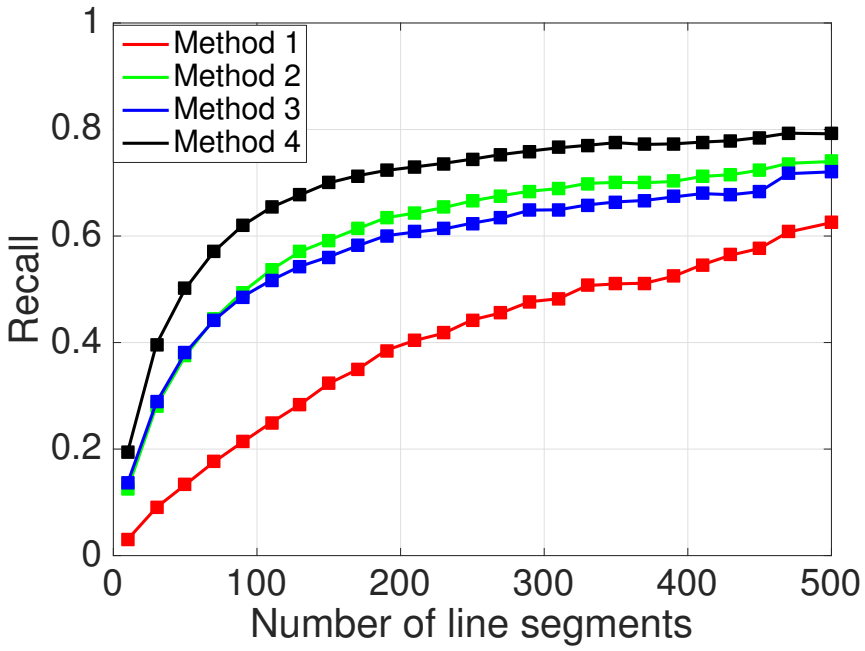
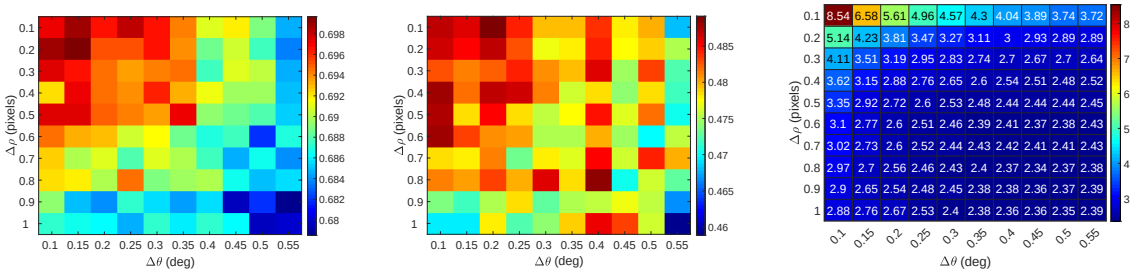


Figure 3.5: Performance of the four ranking methods described in section 3.7, as measured by recall vs number of segments returned, on the YorkUrbanDB training dataset.

one-out cross-validation to generate a smooth objective surface (Figure 3.6). The optimal parameter values using this approach were found to be $\Delta\rho$= 0.4 pixels and $\Delta\theta$= 0.46 deg. We adopt these parameter values for all subsequent experiments on both the YorkUrbanDB and Wireframe datasets.



(a) (b) (c)

Figure 3.6: Performance and run-time analysis of Hough map resolution. (a) Mean recall over number of segments $k = 1, \ldots, 500$ returned. (b) Mean precision over number of segments $k = 1, \ldots, 30$ returned. (c) Corresponding run time of MCMLSD algorithm per image (sec).

3.9.3 Algorithms Evaluated

We compare the proposed MCMLSD method against five leading methods:

1. The Slice Sampling Weighted Mean Shift (SSWMS) method of Nieto et al. [119]
2. The widely-used Line Segment Detector (LSD) method of Grompone von Gioi et al. [166]
3. The linelet-based method (linelet) of Nam-Gyu et al. [29]
4. The deep-learning Wireframe Parser method of Huang et al. [75]
5. The deep-learning Attraction Field method of Xue et al. [182], with *a-trous* architecture.

SSWMS. We obtained the code for the SSWMS method from `sourceforge.net/projects/lswms`. (The authors renamed the method LSWMS there.) There are two parameters and we used the author-recommended default values for both:

- orientation threshold $\Delta\theta = 22.5$ deg
- mean shift bandwidth $= 3$ pixels

The SSWMS algorithm is designed to output segments in descending order of salience - we therefore use this order to rank the segments.

LSD. We obtained the code for LSD from `www.ipol.im/pub/art/2012/gjmr-lsd/`. We rank segments using the criterion recommended by the authors and employed in later work [20], namely in increasing order of the number of expected false alarms, which is one of the outputs of the LSD detector.

Linelet. We obtained the code for the Linelet[29] algorithm from `https://github.com/NamgyuCho/Linelet-code-and-YorkUrban-LineSegment-DB`. We rank segments using the criterion recommended by the authors.

Wireframe Parser. Xue et al. [182] provide the segments generated by the Wireframe Parser for both YorkUrbanDB and Wireframe test sets; we use these to compute performance. Since the authors do not specify a ranking method, we rank the segments in descending order of segment length.

Attraction Field. As for the Wireframe Parser, Xue et al. [182] provide the segments generated by the Attraction Field method for both YorkUrbanDB and Wireframe test sets, so we use these to compute performance. We rank the segments using the criterion recommended by the authors.

3.9.4 Qualitative Results

Fig. 3.7 shows the top-ranked 90 segments returned by each algorithm on four example images from the YorkUrbanDB test dataset. To our eyes, the Attraction Field and MCMLSD results look strongest, but in complementary ways. While the Attraction Field method appears more adept at picking out short segments (e.g., the windows in the first example), MCMLSD is more successful at recovering the longer segments (e.g., the lines on the ground plane in the first three examples).

Interestingly, in the second example the Attraction Field method succeeds in detecting some of the line segments projecting from the tiling pattern on the floor, despite being trained (on the Wireframe dataset) to ignore these.

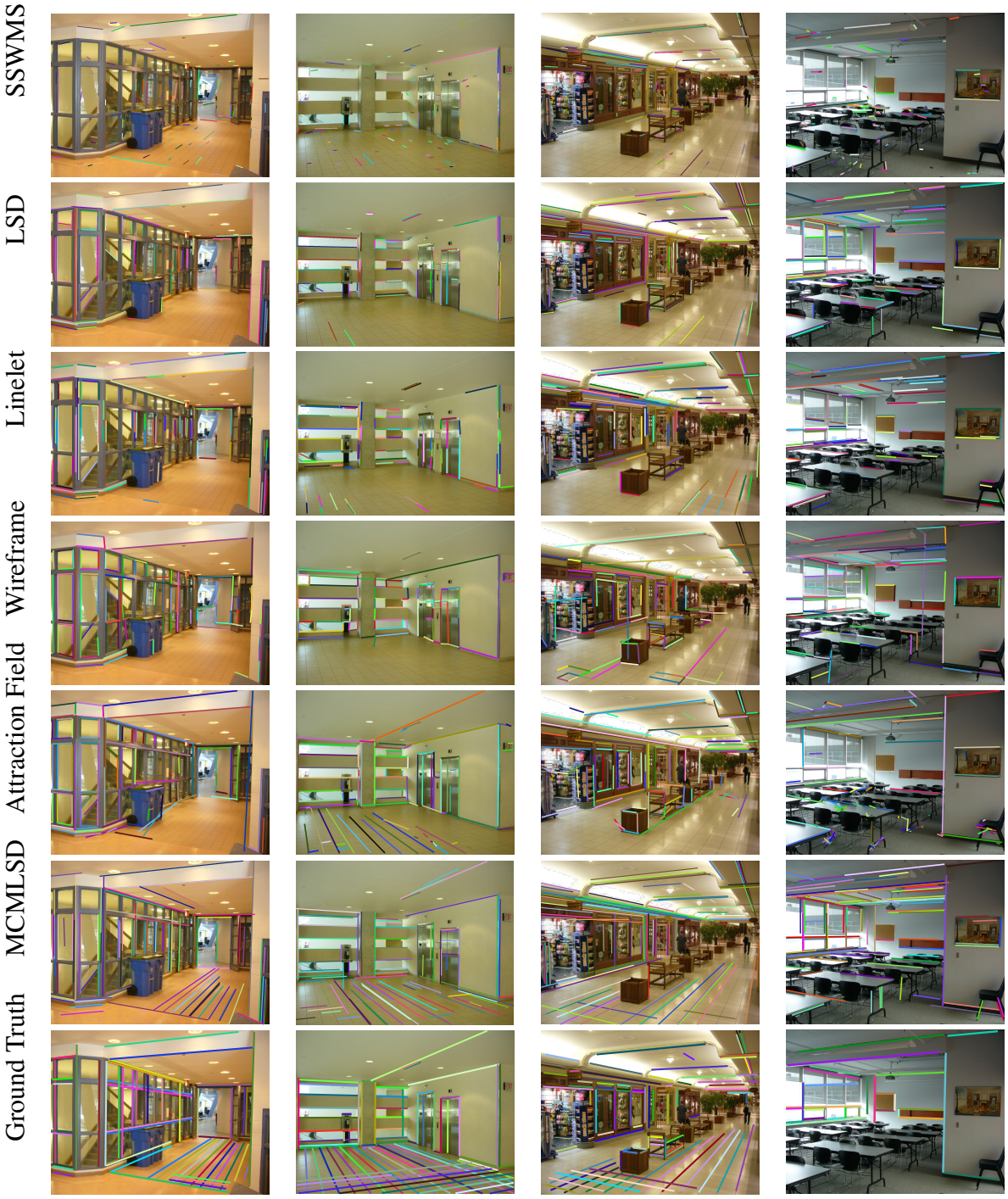


Figure 3.7: Top 90 segments returned by the six algorithms under evaluation, together with hand-labelled ground truth, for four example test images drawn from the YorkUrbanDB dataset.

3.9.5 Quantitative Results

3.9.5.1 YorkUrbanDB Test Set

Fig. 3.8(a) shows the mean length of ranked line segments returned by each of the six algorithms, compared to the ground truth line segment lengths. Although the algorithms generally rank longer segments higher, all ultimately return segments that are on average shorter than the ground truth segments. Consistent with the qualitative observations above, MCMLSD tends to return longer segments than the other approaches.

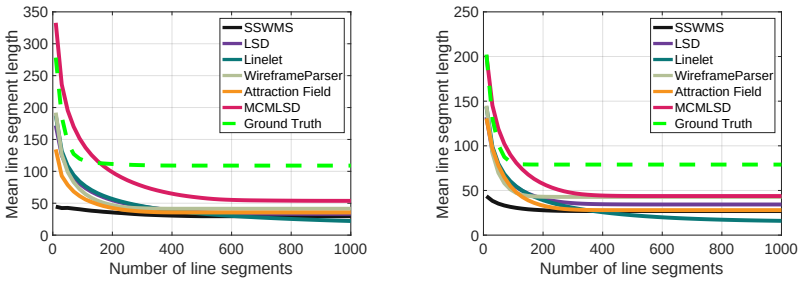
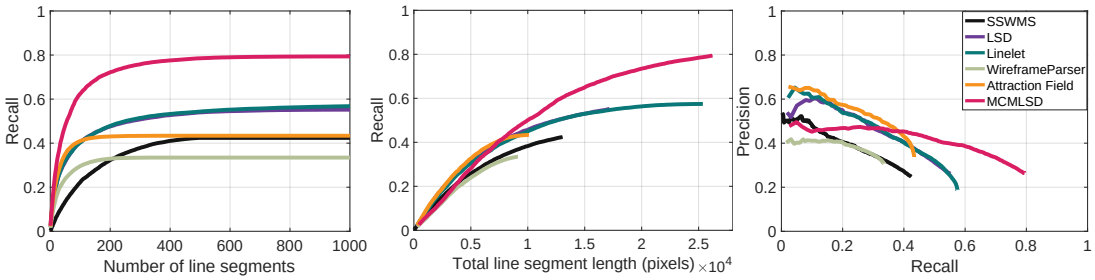


Figure 3.8: Mean length of ranked line segments returned by each algorithm for (a) YorkUrbanDB and (b) Wireframe test sets, as a function of the number of segments returned. Ground truth line segments are ranked from longest to shortest.

Fig. 3.9 provides a quantitative comparison of all six algorithms on the YorkUrbanDB test set. MCMLSD achieves a maximum recall of 0.8, roughly 45% better than the LSD and Linelet methods. Interestingly, MCMLSD outperforms the more recent deep learning algorithms by an even larger margin - maximum recall for MCMLSD is roughly 140% higher than for the Wireframe Parser algorithm and 90% higher than for the Attraction Field algorithm.

Analysis of each of the three performance measures yields additional insights. Fig. 3.9(a) shows that if a constraint is placed on the number of segments returned, e.g., to limit complexity for downstream analysis, MCMLSD consistently achieves higher recall. While the deep Attraction Field algorithm is competitive with the traditional LSD and Linelet algorithms for very tight constraints (fewer than 100 segments), it falls behind as this constraint is relaxed.



(a) (b) (c)

Figure 3.9: Performance of the six algorithms under evaluation on the YorkUrban Dataset. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

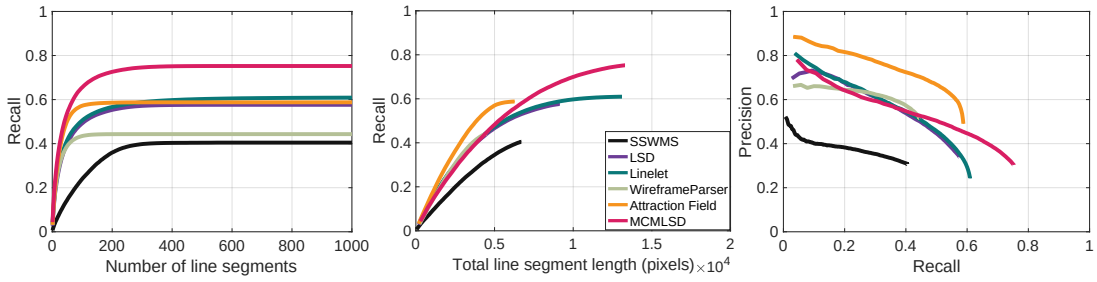
The story is a little different if the constraint is placed on the total segment length rather than the total number of segments (Fig. 3.9(b)). Here we see that while MCMLSD vastly outperforms the other methods for more relaxed constraints (more than 10^4 pixels), for tighter constraints, the Attraction Field, LSD and Linelet algorithms become slightly superior. This can be accounted for by the tendency of MCMLSD to extract longer segments than the Attraction Field, LSD and Linelet algorithms.

Finally, Fig. 3.9(c) shows that the algorithm of choice very much depends upon the relative value of precision and recall for a particular application. If recall greater than 0.4 is required, MCMLSD is clearly preferred. However, if precision greater than 0.45 is required, then recall must be sacrificed and the Attraction Field or Linelet algorithms are preferred. It should be remembered, however, that since the YorkUrbanDB ground truth is incomplete, lower precision may be due to detection of useful segments that just do not happen to be labelled in the ground truth.

3.9.5.2 Wireframe Test Set

Fig. 3.10 shows the same evaluation on the Wireframe test set. The maximum recall achieved by MCMLSD is 0.75, almost as high as for YorkUrbanDB, even without fine-tuning of parameters or distributions, indicating good generalizability. The performance advantage is smaller than for YorkUrbanDB, but MCMLSD is still 26% better than its

closest competitors. In terms of maximum recall, the Attraction Field algorithm is now competitive with the LSD and Linelet algorithms. MCMLSD still leads the pack when the number of line segments is constrained (Fig. 3.10(a)). As for the YorkUrbanDB dataset, when total line segment length is constrained, MCMLSD dominates in the high-recall regime. However, for the Wireframe dataset, the Attraction Field algorithm now dominates in the low-recall regime. Fig. 3.10(c) tells a similar story for precision-recall: MCMLSD dominates in the high-recall regime, but the Attraction Field method is superior in the low-recall (high-precision) regime. The improved performance of the Attraction Field method for the Wireframe dataset relative to the YorkUrbanDB dataset is not surprising, as it was trained on the Wireframe training partition.



(a) (b) (c)

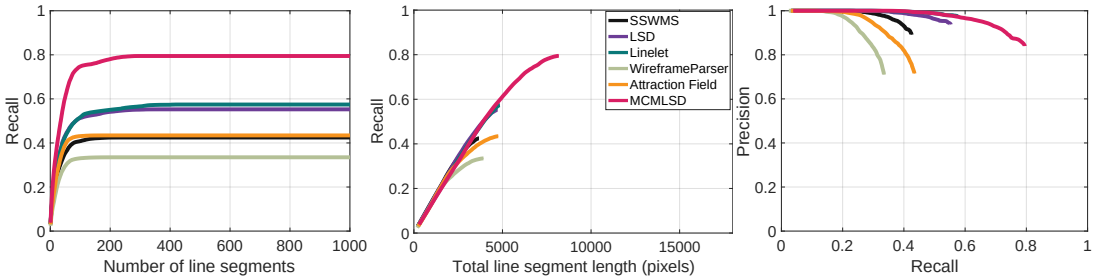
Figure 3.10: Performance of the proposed MCMLSD methods compared with the state of the art on the Wireframe dataset[75]. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

3.9.5.3 Ranking Revisited

What accounts for the superiority of MCMLSD in the high-recall regime, and the superiority of the Attraction Field algorithm (and, for YorkUrbanDB, the LSD and Linelet algorithms) in the low-recall regime? One possible factor is the quality of the line segments they return. But another possible factor is the ranking employed by each method. To dissociate these two factors, we employed an oracle to rank the segments returned by each algorithm for the YorkUrbanDB dataset according to ground truth precision. Specifically, after 1:1 association with ground truth segments, the algorithm segments are ranked

in terms of the proportion of their points that have a 1:1 ground truth match. Ties are resolved by length, with longer segments ranked first.

The results are illuminating (Fig. 3.11). While MCMLSD necessarily still achieves highest recall, and still dominates when the number of line segments is constrained, the precision advantage of the Attraction Field method in the low-recall regime has evaporated. This indicates that the advantage of the Attraction Field algorithm in the low-recall regime derives not from superior line segments but from superior ranking. This in turn suggests that the performance of other methods such as MCMLSD in the low-recall regime might be improved by adopting a revised ranking strategy.



(a) (b) (c)

Figure 3.11: Performance on the YorkUrbanDB dataset when an oracle is used to rank the segments by their precision. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

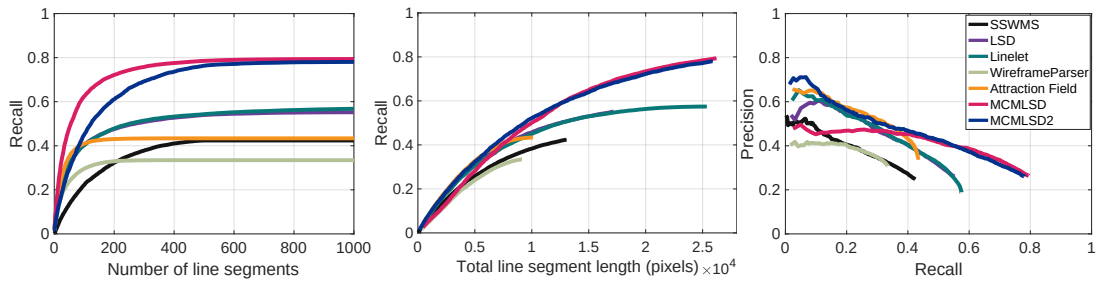
One limitation of the ranking strategy adopted in our original CVPR paper [4] is that it considers only the location and orientation of edges detected by the Elder & Zucker multiscale edge detector [50], which employs a signal detection approach based only on the local luminance contrast. This ignores local colour and texture cues that can signal the relative importance of these edges.

To incorporate this appearance information, we employ the structured forests edge detector of Dollár and Lawrence [43] (code obtained from `https://github.com/pdollar/edges`), which was trained on the BSDS 500 dataset to use the local pattern of colours and textures to identify edges delineating “distinguished things", as judged by human observers [112]. Our hypothesis is that the output of the structured forests

edge detector will thus carry appearance cues complementary to our probabilistic ranking measure, which is based solely on the geometry of the edges.

To test this hypothesis, we construct a logistic regressor that takes both of these cues as input to predict the precision of each segment, train the regressor on the YorkUrbanDB training set, and then use it to rank segments in the test set. Since we are interested in improving the precision of the detector, we employ a modified version of Ranking Method 4 (Section 3.7), using the mean of the marginal probabilities along the segment instead of the sum. To form the appearance cue we average the scalar responses of the structured forests edge detector at the locations of the Elder & Zucker edges within a 2-pixel distance of the line segment.

Fig. 3.12 shows results of the MCMLSD algorithm using this revised ranking strategy (dubbed MCMLSD2), alongside the original MCMLSD algorithm and the five competitors. We see that with this revised ranking strategy, MCMLSD2 loses some recall performance when the number of line segments is constrained (Fig. 3.12(a)), but is still vastly superior to the other methods. At the same time, the precision of MCMLSD2 matches that of the Attraction Field, LSD and Linelet algorithms in the low-recall regime, and is far superior in the high-recall regime (Fig. 3.12(c)).



(a) (b) (c)

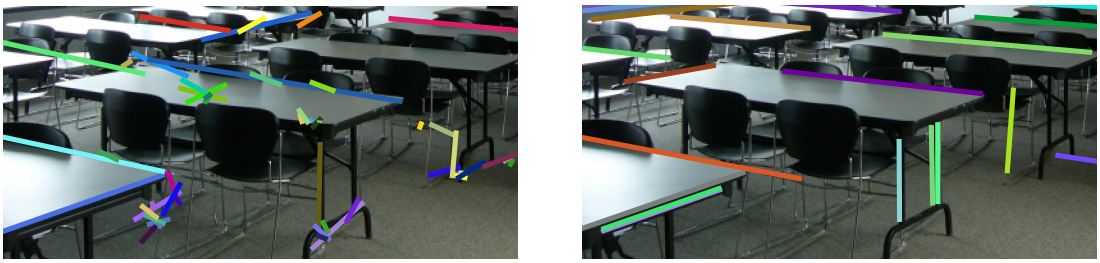
Figure 3.12: Results including MCMLSD2, which uses the structured forests edge detector [43] to incorporate local appearance cues when ranking segments. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

3.9.6 Reconciling with Recent Evaluations

The results above may at first seem puzzling, since they seem to contradict the evaluations reported in recent papers that claim superiority of the deep Wireframe Parser and Attraction Field methods [75, 182]. This contradiction is due to differences in how algorithms were evaluated. We laid out our evaluation methods in Section 3.8 of this paper and in our original CVPR paper [4]. There are two key deviations between our evaluation approach and the approach employed in the Wireframe and Attraction Field papers that account for this contradiction.

3.9.6.1 Distance Threshold

In our evaluations, we employ a distance threshold of $2\sqrt{2}$ pixels, to associate any pair of lines that could potentially appear in the image with less than a one-pixel intervening gap. This seems like a reasonable threshold, able to account for small localization errors in edge detection due to pixel discretization. However, in the deep network papers, a threshold of 1% of diagonal image size was employed, which for the YorkUrbanDB results in a threshold of 8 pixels, 2.8 times the threshold we employed. This looser threshold is convenient for the deep networks, which by necessity use sub-sampled images and struggle to localize segments with precision. Fig. 3.13 shows an example.

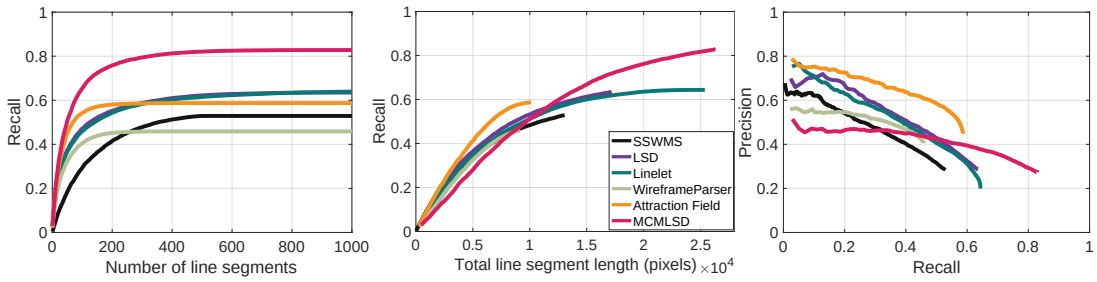


(a) (b)

Figure 3.13: Crop of example YorkUrbanDB test result for the (a) Attraction Field and (b) MCMLSD algorithms. Observe the alignment errors of some of the segments returned by the Attraction Field algorithm.

To assess the importance of this threshold, we re-evaluated all algorithms using the looser threshold of 8 pixels. Fig. 3.14 shows the results for the YorkUrbanDB dataset. We

see that as we loosen the threshold, performance rises for all algorithms, but the performance of the deep algorithms (Attraction Field and Wireframe Parser) rises disproportionately, confirming the poorer localization performance of these methods.



(a) (b) (c)

Figure 3.14: Evaluation on the YorkUrbanDB dataset using the looser distance threshold of 8 pixels employed in the Wireframe [75] and Attraction Field [182] papers. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

3.9.6.2 Enforcing 1:1 Matches

While the looser distance threshold clearly helps the deep algorithms, Fig. 3.14 makes clear that this alone cannot fully account for the claim that deep networks uniformly perform better than MCMLSD and earlier algorithms such as LSD and Linelet. Here we address a more serious issue that gets to the heart of what we mean by line segment detection.

In both the Wireframe and Attraction Field papers, a very simple method is employed to match algorithm segments and ground truth: points on detected segments that lie within 8 pixels of a ground truth segment are identified as hits. Normalizing by the total length of the detected segments and the ground truth segments forms the precision and recall measures, respectively.

In Section 3.8 of this paper and in our original CVPR paper [4], we were careful to articulate the problems with this simplistic approach. First, when matching points on detected segments with points on ground truth segments, it is important that these matches

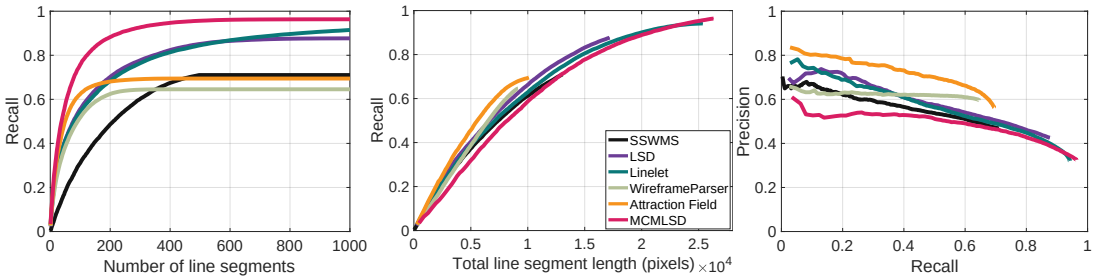
be 1:1. In other words, the same ground truth point should not be used to generate hits for multiple points on detected segments. Similarly, the same point on a detected segment should not be matched to multiple ground truth points. Importantly, this constraint penalizes algorithms that generate multiple detections for a single ground truth segment, or that confuse two neighbouring ground truth segments as a single segment. Note that not enforcing this constraint leads to pathological results. For example, an algorithm that generates dense, 16-pixel wide regions of filled pixels centred on the ground truth segments will be evaluated to have perfect precision and perfect recall.

However, enforcing 1:1 matches between points is not enough. The problem of line segment detection is not to detect isolated edges but to recover the continuous line segments present in an image. The output line segments can be coded in various ways, e.g., by the 2D locations of their endpoints. Critically, the output is more than an edge map: each line segment is a higher-level organization of edge points into a more global representation.

This means that to fairly evaluate a line segment detector, the 1:1 constraint must be applied at the segment level, not at the pixel level. As articulated in Section 3.8 of this paper and in our original CVPR paper [4], this is critical in order to penalize under- and over-segmentation. Again, not imposing this constraint will lead to pathological results. For example, an algorithm that returns a scatter of tiny line segments that are all only one pixel long but lie within the distance threshold of ground truth and account for all ground truth points will generate perfect precision and recall scores.

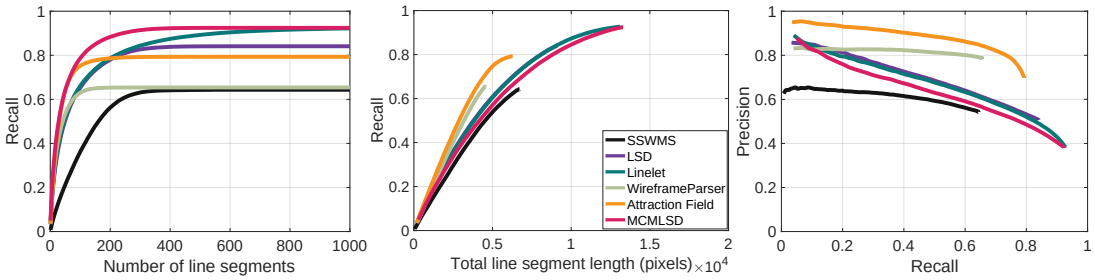
To assess the importance of this segment-level 1:1 matching constraint, we re-evaluated all algorithms *without* this constraint, i.e., using the simple matching method employed in the Wireframe Parser [75] and Attraction Field [182] papers, and also using the looser distance threshold employed in these papers. As shown in Figs. 3.15 and 3.16, despite this relaxation in the evaluation criteria, MCMLSD still outperforms the deep learning algorithms in terms of maximum recall and recall as a function of the number of line segments returned. However, the authors of these deep learning papers did not report these measures of performance but only the precision-recall curves shown in Figs. 3.15(c) and 3.16(c). Here we see that removing the 1:1 matching constraint particularly advantages the deep Wireframe and Attraction Field algorithms, leading to clear superiority of the Attraction Field method in the low-recall regime, although MCMLSD and Linelet methods still

achieve much higher recall. But again, we remind the reader of the limitations of precision measures for these incomplete datasets (Section 3.8).



(a) (b) (c)

Figure 3.15: Performance of MCMLSD compared with the state of the art on the YorkUrban Dataset using pixel level evaluation. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.



(a) (b) (c)

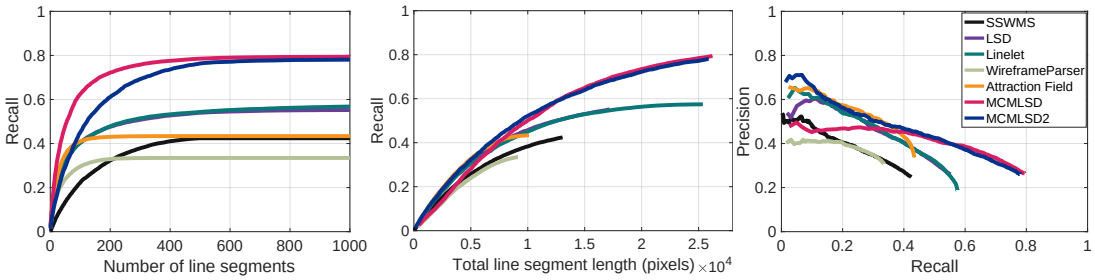
Figure 3.16: Performance of MCMLSD methods compared with the state of the art on the Wireframe dataset using pixel level evaluation. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

3.9.7 Summary of Quantitative Results

To summarize, the relative performance of line segment detection algorithms very much depends on how performance is measured. Recent deep learning papers have loosened

distance thresholds and not enforced 1:1 matching constraints, and under these conditions they achieve higher precision, although still inferior recall. It is possible that there are some applications for which this measure of performance is appropriate. For example, one may attempt to use only a pixel-level Hausdorff distance to register two images or to register an image to a CAD model. However, for most downstream applications, e.g., single-view 3D reconstruction [130, 158], an *organization* of points into accurate line segments is desirable, and to evaluate this one must impose 1:1 matching constraints.

Fig. 3.17 (copied from Fig. 3.12 for convenience) summarizes performance relevant to these requirements, specifically for a distance threshold of $2\sqrt{2}$ to ensure accuracy, and a 1:1 matching constraint imposed at the segment level. Here we see that by any of the three measures of performance, one of the two versions of MCMLSD is recommended. If the number of output lines is to be restricted and recall is the priority, the original MCMLSD algorithm vastly outperforms other methods. However, if precision-recall performance is the priority, then MCMLSD2 is recommended, as it matches or surpasses the performance of all other methods in the low-recall regime while vastly outperforming in the high-recall regime.



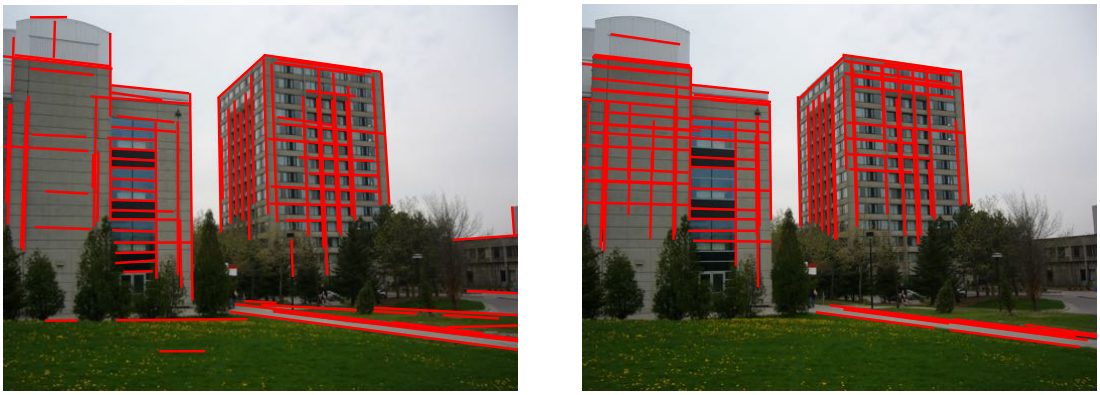
(a) (b) (c)

Figure 3.17: Summary of results. (a) Recall as a function of number of segments returned. (b) Recall as a function of the total length of segments returned. (c) Precision-Recall.

3.10 Image Resolution

One limitation of current deep learning methods is that the computational load for learning and inference may become untenable for higher image resolutions. In contrast, the MCMLSD algorithm adapts well to different image resolutions without fine-tuning as long as the transition probabilities are scaled appropriately (Section 3.6.2). For example, doubling the resolution requires that the transition probabilities from OFF to ON and from ON to OFF be halved.

As an example, Fig. 3.18 shows the top 90 segments returned for an image from the York UrbanDB dataset at normal (640×480 pixel) and high (1280×960 pixel) resolutions. Note that the algorithm is able to take advantage of the higher resolution to deliver more complete and accurate segments.



(a) 640×480 pixels (b) 1280×960 pixels

Figure 3.18: Top 90 segments for MCMLSD on an example image at low and high resolutions.

3.11 Run Time

The dynamic programming solution for line segment detection runs in $O(N) = O(\sqrt{n})$ time, where N is the number of point samples on the line and n is the number of pixels in the image. Given a set of m detected lines, the total time complexity of line segment extraction is $O(m\sqrt{n})$.

Table 3.2 shows the average run time for the six algorithms tested here on the 640×480 pixel images from the YorkUrbanDB training dataset. The SSWMS, LSD, Linelet and MCMLSD algorithm were tested using a 3.4 GHz Intel Core i7 with 8GB RAM. The deep network Wireframe and Attraction Field algorithms were tested using an NVIDIA Titan X GPU with Xeon E5-2620 2.10GHz CPU.

The MATLAB implementation of our MCMLSD algorithm has an average run time of 2.81 sec per image. Aside from the Linelet algorithm, which is very slow, the other algorithms are optimized and implemented in C++, returning results within a few hundred milliseconds.

About 63% of our run time is taken by the probabilistic Hough method for line extraction [154], which we believe could be sped up considerably with more efficient coding practices and implementation in C or C++. There are also many opportunities for mapping to parallel hardware, as edge detection is dominated by convolutions and in the dynamic programming line segment detection stage, lines separated by more than 4 pixels are processed independently.

Table 3.2: Average number of segments returned and run time per image for the six systems evaluated.

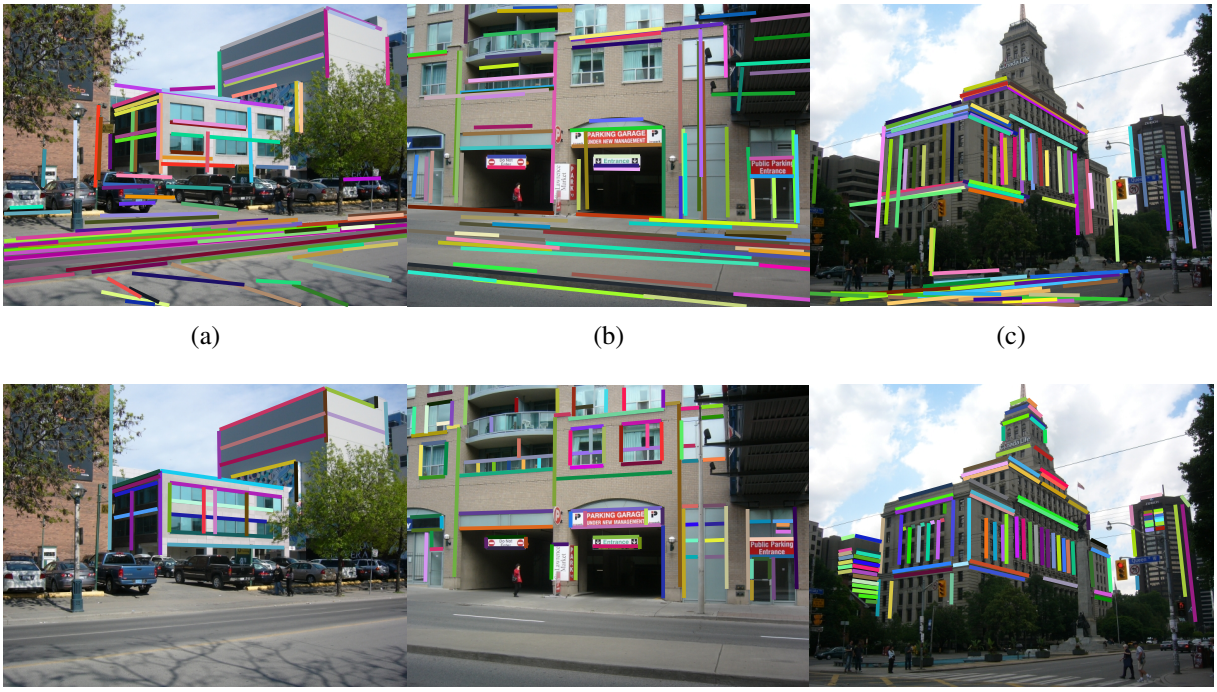
Algorithm	# Segments	Run Time (sec)
SSWMS	391	0.30
LSD	537	0.27
Linelet	967	34.5
Wireframe	228	0.446
Attraction Field	303	0.152
MCMLSD	488	2.81

3.12 Failure Mode Analysis

The Fig. 3.19 shows the three images in the YorkUDB test set with lowest f-scores. From these images we can make the following observations:

1. Our algorithm tend to detect line segments on the ground and these are often ranked pretty high by our ranking method. While these line segments are real, the YorkUrbanDB ground truth often does not include them.

2. Our algorithm detects segments on smaller objects like lamp posts and traffic signals, whereas the YorkUrbanDB tends to focus more on line segments on architecture.



(a) (b) (c)

(d) (e) (f)

Figure 3.19: The three images with the lowest f-scores in the test set of the YorkUDB. The first row shows the top 90 line segments from the MCMLSD algorithm. The second row is ground truth.

3.13 Conclusion

We have developed and evaluated a novel method for line segment detection called MCMLSD that combines the advantages of global probabilistic Hough methods for line detection with

spatial analysis in the image domain to identify segments. The key insight is that limiting segment search to Hough-detected lines leads naturally to a Markov chain formulation that allows maximum probability solutions to be computed exactly in linear time. Our method also has the advantage that it can detect multiple segments lying on the same line, a common scenario for images of the built environment. This formulation leads to a natural probabilistic measure for ranking segments based upon the sum over point marginals, which maximizes the expected number of correctly labelled points on detected lines.

A second contribution is our new methodology for evaluating line segment detectors on an incomplete labelled dataset. By constraining matches between ground truth and detector output to be 1:1 at the segment level, we show that under- and over-segmentation are penalized appropriately. Using this new evaluation methodology we find that MCMLSD outperforms the state-of-the-art by a substantial margin. The code for MCMLSD and our evaluation method is available at `www.elderlab.yorku.ca/resources`.

Chapter 4

Road Segmentation From Geometry

4.1 Project Description

The goal of this project was to apply a geometry-driven approach to identify a road region in a dash cam image given various view angles and different lighting conditions. This algorithm utilized the line segment detector from my first project to estimate road vanishing points and horizon lines. These vanishing points and horizon lines were our prime indicators of the location of the road region. The dataset we used in the project was designed to include extremely challenging weather conditions. At the time of writing, at least, there were no existing public datasets that provided this diversity.

My contribution to this project was:

1. Improving the vanishing point reliability evaluation algorithm.
2. Developing a new horizon estimation algorithm.
3. Implementing a texton-based texture classification algorithm.
4. Re-labelling the dataset.
5. Developing new algorithms to generate region of interest (ROI) masks.
6. Cross-validation experiments to find optimal parameters.

Related Publications:

- G. Cheng, Y. Wang, **Y. Qian**, and J. H. Elder. Geometry-guided adaptation for road segmentation. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pages 46–53, 2020
- G. Cheng, **Qian, Y.**, and J. H. Elder. Fusing geometry and appearance for road segmentation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 166–173, 2017
- E. J. Almazan, **Qian, Y.**, and J. H. Elder. Road segmentation for classification of road weather conditions. In G. Hua and H. Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 96–108, Cham, 2016. Springer International Publishing
- **Qian, Y.**, E. J. Almazan, and J. H. Elder. Evaluating features and classifiers for road weather condition analysis. In *International Conference on Image Processing (ICIP), 2016 IEEE*. IEEE, 2016

4.2 Introduction

Automatic assessment of road weather conditions using vehicle camera data can be used to inform the human driver, driver-assist controls and autonomous control systems. Moreover, the information can be shared across connected vehicles, alerting following vehicles to conditions ahead. Another application is automatic dispatch and verification of snow ploughs and service vehicles. Given their typically wide geographic distribution, these service vehicles can provide real-time data on road conditions to central management, which can then use the data to verify maintenance and optimize dispatch.

While future generations of service vehicles may be manufactured with appropriate built-in cameras, in the meantime there is interest in retrofitting existing vehicles with removable dash cams that can be used for multiple purposes. This poses a challenge for video analytics, as the pose of the camera relative to the road surface may vary considerably. Since the cameras are mounted inside the vehicle, imagery may be partially occluded

by the hood of the vehicle. For snow ploughs, the road surface may also be occluded further by the plough, depending on its position (Fig. 4.1).

Bare		Covered		
Bare dry $n = 32$	Bare wet $n = 14$	Iced covered $n = 8$	Snow covered $n = 16$	Snow packed $n = 30$

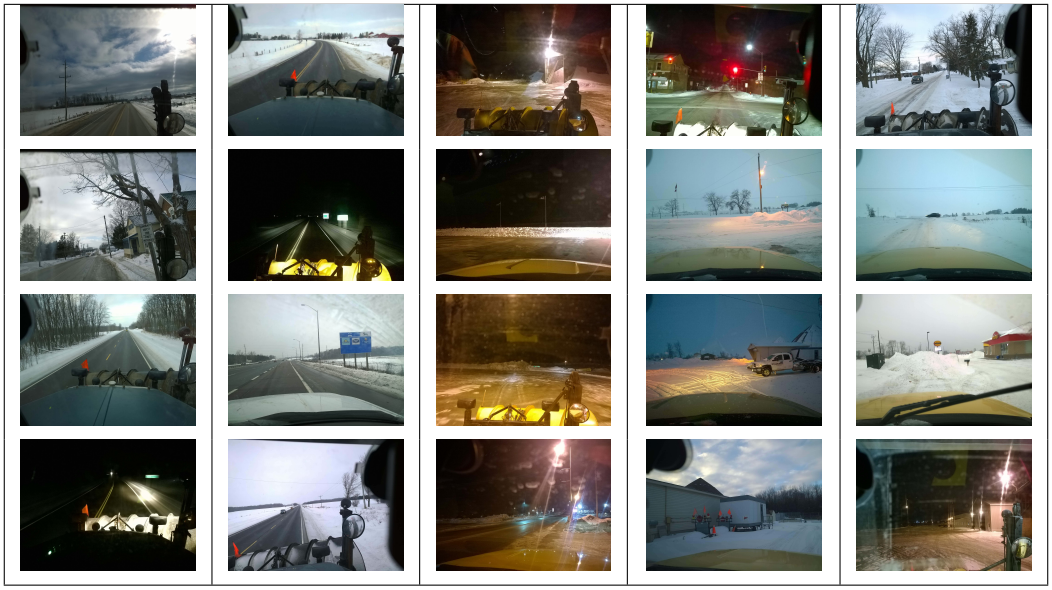


Figure 4.1: Example images from training dataset, and the number n in each class.

To address these challenges a reliable algorithm for segmenting the road surface from the imagery is required. This method must be able to handle variations in the position and pose of the camera as well as geometry of the road surface. Using appearance features of the road surface (e.g., texture) for segmentation is unlikely to be reliable across diverse weather conditions, since the road appearance will vary considerably and sometimes may strongly resemble other surfaces in the scene. For these reasons, we focus here on geometric methods for identifying the road surface and show that by fusing a combination of these methods we can significantly improve road weather classification performance.

In particular we develop a novel method for estimating the road vanishing point, which yields a triangular road segmentation hypothesis. This vanishing point method also delivers a measure of reliability, which can be used to identify when the vanishing point is

ill-defined (in a parking lot, for example). Under these conditions we revert to a weaker segmentation based upon detection of the horizon line. Combining these with a spatial prior then delivers an estimated road segmentation tailored to each image.

This paper is organized as follows: Section 4.3 reviews prior work, Section 4.4 details our road segmentation algorithm, Section 4.5 describes our classification process, Section 4.6 reports results and finally Section 4.7 presents our conclusions and plans for future work.

4.3 Prior Work

4.3.1 Road Segmentation

One might initially imagine using surface appearance properties to distinguish the road pavement surface from other surfaces in the scene [8, 7]. However, this approach is problematic here for at least three reasons: 1) pavement appearance varies depending upon the exact road materials employed and age of the road; 2) other nearby surfaces (e.g., sidewalks, driveways, buildings) may be constructed from similar materials; 3) the road surface may be partially or completed covered by snow and/or ice.

For these reasons, we use geometric methods to estimate the road segmentation. This is still non-trivial due to diversity in camera pose and road geometry. Roads vary in shape, are sometimes relatively unstructured, non-homogeneous and vary in appearance under varying weather and illumination conditions.

Previous approaches have estimated the vanishing point [133, 90, 134], horizon [6] and/or border lines of the road [91]. The vanishing point is typically detected using texture information from Gabor wavelet filters [133, 90, 6], using a Hough transform [172] or with a line segment voting scheme [153]. In this work we adapt recent work on Hough-based vanishing point detection [154] that has proven effective for Manhattan frame estimation.

The horizon line is often estimated as the line that partitions the image into two regions that differs maximally in appearance [38, 35], however more elaborate approaches based on gist descriptors [120, 6] have also been employed. Here we identify the horizon as the vertical location that maximizes the RMS first derivative in the vertical direction across all horizontal locations and colour channels.

4.3.2 Road Weather Classification

Since the focus of this paper is on road segmentation, we will review the literature on road weather classification only briefly.

Much of the prior work on road condition classification has focused on the use of polarization and infrared cameras [104, 184, 79, 23, 80, 81], which can be expensive and installation can be complex. However, there are also a number of efforts employing standard RGB cameras. Omer & Fu [121] used an SVM with RGB and gradient histogram features to classify conditions as bare, covered or covered with bare tire tracks. However their approach required manual cropping of each image to extract the image region projecting from the ground surface, which is impractical for a real system. Kauai et al [86] used colour cues to add some degree of illumination invariance, however their approach depends on detecting white line markings to identify the road area, which will fail under snowy conditions or for roads that are poorly marked. In a very recent paper, Amthor et al. [9] proposed a spatiotemporal approach that integrates over many frames to detect specular reflections indicative of wet conditions. While their method improves on prior approaches, it requires integration over many frames, increasing computational load and delay.

For the present paper we employ the classifier reported recently by Qian et al [157], which uses a naive Bayes classifier over texton and luminance features. Please see Section 4.5 for more details.

4.3.3 Dataset

To train and evaluate our algorithm, we employ the challenging dataset of 100 2048×1536 pixel images (Fig. 4.1) introduced by Qian et al. [157], obtained directly from the authors. The dataset contains roads under different weather conditions, from bare dry to snow packed. Each of the five classes was randomly and evenly split between training and test datasets, each consisting of 50 images.

The pictures were taken at different times of the day, thus covering a wide range of illumination conditions and the camera pose varied considerably. The road condition class was identified manually by our industrial partner. For training and evaluation purposes, we manually segmented the road surface from the background. Running our classifier

on the manually segmented imagery allows us to estimate the potential for increasing classification performance through further improvements in segmentation.

4.4 Road Segmentation

The main challenge for segmenting the road surface is the variability of the road appearance under different weather conditions, which limits the utility of appearance features such as luminance, colour, texture and detailed road markings. We therefore propose a method that relies on contextual information to define the vanishing point of the road and horizon line. The process consists of four stages: 1) estimation of the vanishing point of the road; 2) assessment of the reliability of this vanishing point estimate; 3) direct estimation of the horizon line, for vanishing points assessed to be unreliable; 4) fusion with a spatial prior to identify the region of the image corresponding to the road surface.

4.4.1 Vanishing Point Estimation

A vanishing point is defined as a point in the image plane where parallel lines converge. We use the line detector algorithm of Tal & Elder, 2012 [154] (code obtained directly from authors). The detector returns betwen 122 and 746 lines for each of the images in the training dataset. It also returns the estimated total length l of the line segments along each detected line. As revealed in Fig. 4.2, the geometry (position ρ, orientation θ, length l) of each line provides some information about the likelihood that it is generated by the vanishing point (ON) versus a background process (OFF). We therefore rerank the lines using a naive Bayes model to approximate the likelihood ratio L_i for each line i:

$$L_i = \frac{p(\rho_i, \theta_i, l_i|\mathrm{ON})}{p(\rho_i, \theta_i, l_i|\mathrm{OFF})} \approx \frac{p(\rho_i|\mathrm{ON})p(\theta_i|\mathrm{ON})p(l_i|\mathrm{ON})}{p(\rho_i|\mathrm{OFF})p(\theta_i|\mathrm{OFF})p(l_i|\mathrm{OFF})}. \tag{4.1}$$

We estimate the vanishing point as the point in the image that minimizes the distance to the top-ranked n lines. Specifically, we adopt a naive Bayes approach and choose the point v that maximizes

$$p(v|\mathrm{L}) \propto \prod_i^n p(d_i|v)p(v) \tag{4.2}$$

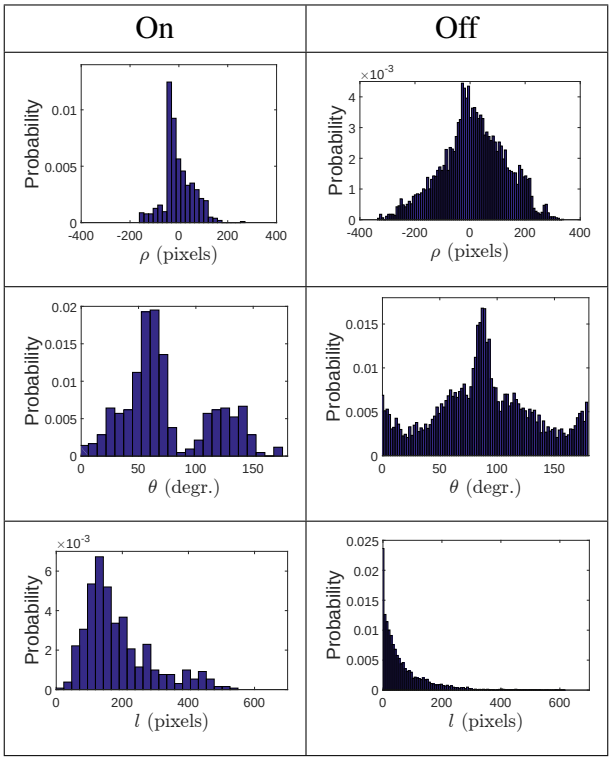
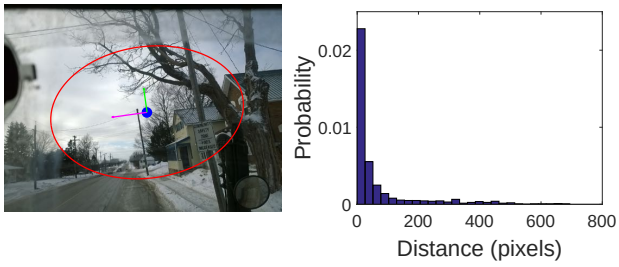


Figure 4.2: Likelihood distributions for the three line features (ρ, θ and l).

where $\mathrm{D} = (d_1, d_2, ..., d_n)$ are the distances to the detected lines. The spatial prior $p(v)$ is modelled as a Gaussian distribution, and the likelihoods $p(d_i|v)$ are determined from the training data as shown in Fig. 4.3.

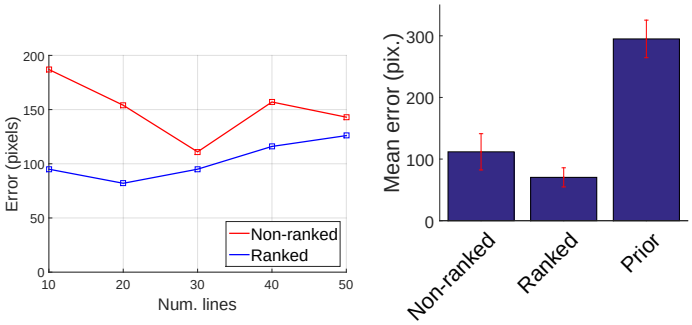
Eqn. 4.2 is not convex in general; to maximize we select the optimal solution from 50 gradient descent solutions (MATLAB fminsearch), initialized by randomly sampling from the prior $p(v)$.

Figure 4.4 compares performance of the vanishing point algorithm on the training dataset with and without the re-ranking step. We find that generally the re-ranking improves results and that error is minimized by using the top-ranked 20 lines. Note also that using the lines to estimate the vanishing point yields a much lower error than using the centroid of vanishing points from the training set (Prior model). Fig. 4.5 shows examples of automatically estimated vanishing points.



(a) Prior (b) Likelihood

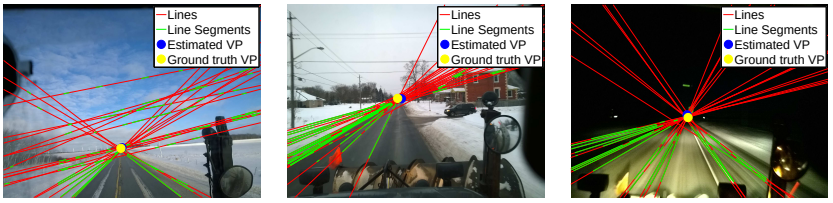
Figure 4.3: (a) Vanishing point prior distribution plotted on a sample image from the dataset. The red ellipse indicates the 95% confidence interval for the vanishing point. (b) Likelihood distribution for the distance of the top-ranked 20 detected lines from the vanishing point.



(a) Number of lines (b) Mean error

Figure 4.4: (a) Average Euclidean error of the estimated vanishing point as a function of the number n of lines employed. (b) Mean error and standard error of the mean, collapsing over n.

There are some situations where the vanishing point of the road is not readily apparent, such as in parking lots or intersections (Fig. 4.10 (c-f)). Our vanishing point algorithm will tend to produce large errors for these cases, which could in turn lead to large errors in the road segmentation. To prevent this, we assess the reliability of vanishing point estimates as the average distance $\bar{D}_k$ of the top k lines closest to the estimated vanishing point. If $\bar{D}_k$ exceeds a threshold t, we reject the vanishing point estimate and use an alternate method to determine the horizon line (see below). There are two free parameters for this reliability measure: the number k of lines and the threshold t. We optimize using the training data based on the ultimate error in estimating the horizon line, assessed as the average absolute



(a) (b) (c)

Figure 4.5: Examples of automatically estimated vanishing points.

error at left and right image boundaries (Fig. 4.6): values of $k = 12$ and $t = 21$ were found to be optimal.

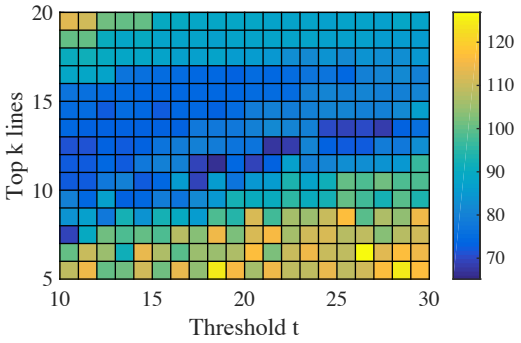


Figure 4.6: The error graph of different combinations of threshold t and top k lines.

4.4.2 Horizon Estimation

When vanishing point estimation fails it may still be possible to constrain the road location using the horizon line. The horizon line can be defined by two parameters, such as its vertical location and orientation for example. For our dataset, however, we found that orientation estimation could be quite unreliable. We therefore fixed the orientation to horizontal.

To estimate the vertical location of the horizon, we used the training images for which our vanishing point estimates were judged (automatically) to be unreliable as a horizon training dataset ($k = 23$ images in all). We then 1) normalized each image to have zero

mean and unit variance, 2) registered them vertically so that their horizons aligned, 3) extracted the luminance channel from each image, 4) averaged over the horizontal position (see Fig. 4.7a) and 5) cropped to extract a luminance vector $\mathbf{l}_i$ of length n centred at the horizon. Finally, we computed the first m principal components $\mathbf{u}_i$ (see Fig. 4.7b) of length n over these k vectors.

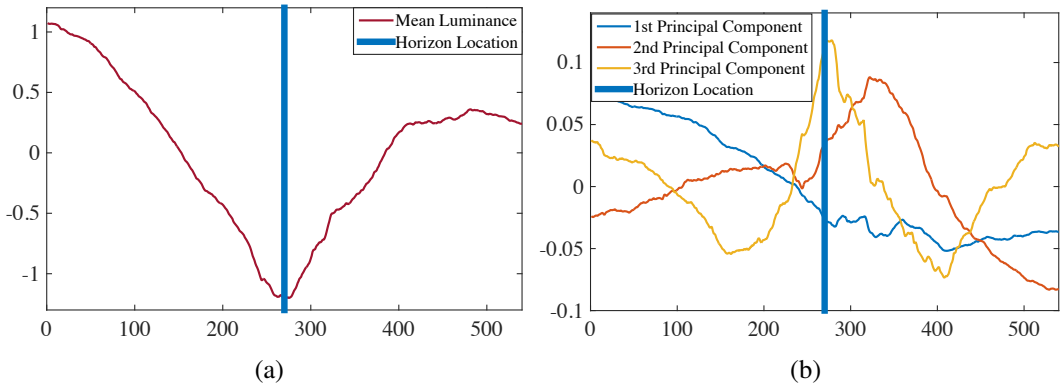


Figure 4.7: (a) Mean normalized luminance of horizon training images as a function of vertical displacement from horizon. (b) First three principal components of vertical luminance distribution around horizon location.

To estimate the vertical position y_h of the horizon for a target image, we 1) normalize the image to have zero mean and unit variance, 2) extract the luminance channel, 3) average over the horizontal position and 4) convolve the resulting vector $\mathbf{l}$ with each of the m principal component vectors to generate m projection vectors $\tilde{\mathbf{l}}_i$. Approximating the training data as multivariate normal, the log probability that the horizon lies at the vertical location y can be estimated as:

$$\log p\left(y_h = y\right) \propto -\sum_{i=1}^{m} \left(\tilde{l}_i(y) - \bar{\mathbf{l}}^T \mathbf{u}_i\right)^2 / \lambda_i, \tag{4.3}$$

where $\bar{\mathbf{l}}$ is the mean over the training vectors $\mathbf{l}_i$ and λ_i is the ith eigenvalue. The horizon is then estimated by maximizing this log probability over y.

The two free parameters of this method are the length n of the principal component filters and the number m of these filters to employ. We optimized these parameters by grid

search over the training data (Fig. 4.8), finding $n = 300$ pixels and $m = 6$ to be optimal. Examples of horizon lines estimated on the test dataset are shown in Fig. 4.10(c-f).

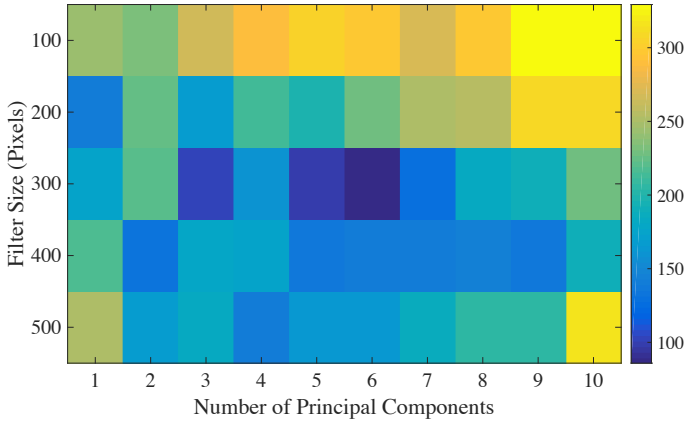


Figure 4.8: Mean vertical location error in horizon estimate over training data, as a function of the length n and number m of principal components filters.

4.4.3 Region of Interest

The road vanishing point and horizon line provide crucial geometric constraints on the location of the road surface. To turn these constraints into an approximate segmentation of the road, we fuse them with a spatial prior that has been conditioned on the estimated vanishing point or horizon. Fig. 4.9(a) shows the spatial prior, learned over the training dataset, without conditioning. The prior is quite diffused, due not only to the variability in road geometry but also to the variation in camera placement. Figs. 4.9(b-c) show the same prior, computed relative to the vanishing point location for images where a vanishing point can be identified (a) and relative to the horizon line for images where it cannot (c). Note that conditioning on the geometry leads to a more focused and accurate indicator of the road surface location.

In order to segment the road for a novel image, we first estimate the vanishing point. If the estimate is judged to be reliable, we register the associated prior (Fig. 4.9(b)) with the estimated vanishing point and label all pixels above a threshold probability p_0 to be road pixels. If the vanishing point is judged to be unreliable we follow the same procedure

for the estimated horizon line and prior. If our vanishing point and horizon line estimates were perfectly accurate, a threshold of $p_0 = 0.5$ would maximize the proportion of correct pixel labelling rate (road/non-road). Fig. 4.9(d) shows that our road segmentation algorithm performs substantially better than the method of Qian et al. [157], which involved simply thresholding the spatial prior, with no estimation of road geometry. This figure also confirms that a threshold of 0.5 works well in practice and is the value we adopt for road weather classification.

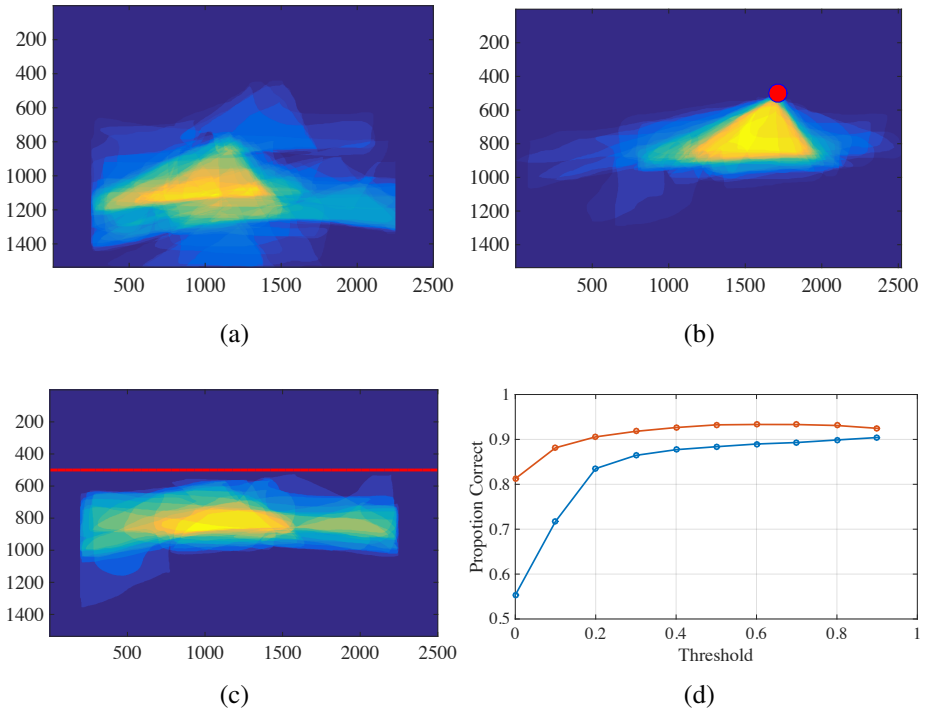


Figure 4.9: (a) Prior spatial distribution of road pixels, in absolute image coordinates. (b) Prior horizontally and vertically registered to vanishing point. (c) Prior vertically registered to horizon. (d) Proportion of correctly labelled image pixels for the test set, as a function of the probability threshold p_0.

Fig. 4.10 shows representative road segmentation results on the test dataset. There remain some failure modes for both road vanishing point and horizon detection, but generally speaking the results are good, as indicated by the median examples (b).

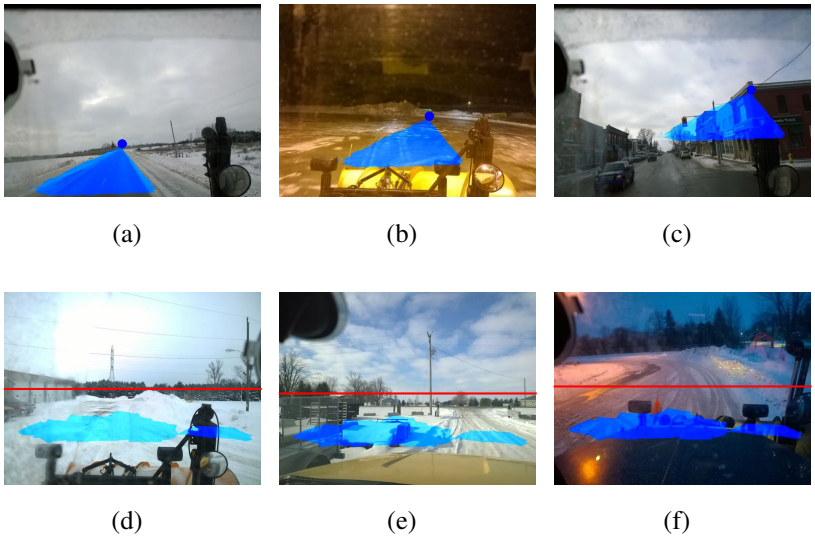


Figure 4.10: Example road segmentations on the test dataset. Results are evaluated based on proportion of correctly labelled pixels (road/non-road). (a)-(c) show best, median and worst-case (failure mode) examples for cases where a vanishing point could be estimated. (d)-(f) show best, median and worst-case examples for cases where the vanishing point was deemed unreliable and a horizon estimator was used.

4.5 Road Condition Classification

To assess the utility of the improved road segmentation, we use it to define the region of interest (ROI) for road weather classification. In particular, we use an adaptation of the method of Qian et al. [157], which employs scale- and orientation-invariant MR8 filters [165]. These filters produce an 8-dimensional feature vector at each pixel in the ROI. Each ground truth ROI in the training set is divided into 8×8 pixel non-overlapping patches, each of which is then represented by an $8 \times 8 \times 8 = 512$-dimensional feature vector. K-means is then used to cluster the feature vectors from the training data into $k = 74$ textons.

While MR8 is roughly luminance-invariant, luminance can carry information about weather (snow is bright, wet roads tend to be darker). Qian et al. therefore augmented this texton descriptor with a 20-bin histogram of grey level deviations from the mean image

luminance.

Qian et al based their classifier on a naive Bayes model of exponential-χ^2 distances of input vectors from their class-conditional means. Here we take a simpler approach, using an SVM with RBF kernel in a one-versus-all design (the SVM implementation provided in the MATLAB Statistics and Machine Learning Toolbox). We found this to yield very similar results.

4.6 Performance Evaluation

Classification results on the test set for two classes (bare vs snow/ice covered), three classes (dry, wet, snow/ice covered) and five classes (dry, wet, snow, ice, packed) are shown in Fig. 4.11. We evaluate results using three different methods to define the ROI: 1) manual segmentation; 2) the automatic segmentation method proposed here; 3) automatic segmentation using the fixed prior of Fig. 4.9(a)) [157]. As a baseline we also show performance for a classifier that simply selects the highest *a priori* probability. Note that the manual segmentation provides an upper bound on the possible payoff from further improvements to segmentation.

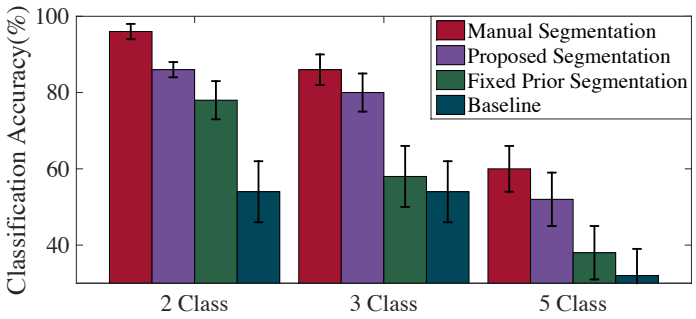


Figure 4.11: Results comparison for 2, 3 and 5 classification using manual, Proposed Segmentation, Fixed prior segmentation against random guesses.

Our proposed segmentation method achieved an accuracy of 86% for two-classes, 80% for three-classes, and 52% for 5 classes. Given the challenging nature of the dataset, these are promising results.

The proposed segmentation method improves classification accuracy over the fixed segmentation method used by Qian et al. [157] in all cases. Statistical significance of this improvement can be assessed by computing the posterior probability that the underlying probability of correct classification is greater for our proposed method, assuming a flat prior over the performance for the two methods. This yields a posterior probability of 0.81 for the 2-class case, 0.99 for the 3-class case and 0.90 for the 5-class case, corresponding to p-values of .19, .01 and .10 in the language of null-hypothesis testing.

4.7 Summary

In this paper we have proposed a novel algorithm for road segmentation from uncalibrated dash cameras, to support road weather analysis. The algorithm was designed to operate robustly over a diverse range of camera poses on both structured (highway/local road) and unstructured roads (parking lots). The approach consists of finding the vanish point, or horizon and fusing with a registered spatial prior. Classification performance on a challenging dataset was 86% and 80% for two- and three-class problems, respectively, representing a significant improvement over prior work [157]. Our analysis reveals that further improvements in performance will likely depend on improvements in both the segmentation and classification stages. The vanishing point from image could be used as a reference point to guild segmentation process. Currently, we are using simple Texton features, where deep learning algorithm could potentially deployed in the classification tasks.

In recent years concern has been raised regarding the lack of generalization of traditional supervised road segmentation techniques, which tend to fail when facing situations not covered in the training set. To overcome this issue, attempts have been made to use online learning methods to adapt the parameters of the algorithm to the image data [155, 6]. These methods typically assume that the bottom of the image projects from the road surface [38, 7, 155, 6], which unfortunately is not the case for our dataset (Figure 4.1). Nevertheless, this is an important issue, and we hope to increase the adaptiveness of our method in the near future.

Chapter 5

LS3D: Building Reconstruction From A Single Image

5.1 Project Description

In the built environment, 3D scenes are often dominated by lines in three mutually orthogonal directions [33]. This regularity can be used to transform the 2D line segments into a 3D CAD model. We built our own dataset (3DBM) because there was no similar dataset available. We used depth maps for evaluation benchmarks because other 3D reconstruction algorithms use depth maps in their evaluation, and we wish to compare against these algorithms.

My contribution to this project is as follows:

1. With guidance from Professor James Elder and Professor Srikumar Ramalingam, I have developed a novel, explainable, single-view 3D reconstruction algorithm called LS3D that infers the 3D Euclidean surface layout of Manhattan buildings, up to an unknown scaling factor.

2. I built a new 3DBM ground truth dataset of 3D Manhattan building models and a novel evaluation framework that allows single-view methods for 3D Manhattan building reconstruction to be evaluated and compared.

3. I evaluated the LS3D algorithm against state-of-the-art deep learning algorithms on

the 3DBM dataset.

Related Publications:

- **Y. Qian**, S. Ramalingham, and J. Elder. LS3D: Single-view Gestalt 3D surface reconstruction from manhattan line segments. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 399–416, 2018

5.2 Introduction

Most 3D computer vision research focuses on multi-view algorithms or direct ranging methods (e.g., LiDAR, structured light). However, the human ability to appreciate the 3D layout of a scene from a photograph shows that our brains also make use of single-view cues, which complement multi-view analysis by providing instantaneous estimates, even for distant surfaces, where stereoscopic disparity signals are weak.

Recent work on single-view 3D reconstruction has focused on supervised deep learning to directly estimate depth from pixels. Here we take a different approach, focusing instead on identifying a small set of principles that together lead to a simple, unsupervised method for estimating 3D surface geometry for buildings that conform to a Manhattan constraint [33].

This approach has three advantages. First, it provides an interpretable scientific theory with a clear set of assumptions and domain of validity. Second, it leads to a well-specified hierarchical solid model 3D representation that may be more useful for downstream applications than a range map. Finally, as we will show, it generates results that are qualitatively and quantitatively superior to state-of-the-art deep learning methods for the domain of 3D Manhattan building reconstruction.

Single-view 3D reconstruction is ill-posed and thus solvable only for scenes that satisfy strong regularity conditions. Meanwhile, using the linear perspective - the assumption that features of the 3D scene are arranged over systems of parallel lines - is perhaps the most powerful of these constraints: its discovery is often cited as a defining achievement of the Early Renaissance. The linear perspective is a particularly valuable cue for urban environments, which abound with piecewise planar and often parallel surfaces that generate families of parallel 3D line segments in the scene.

A stronger constraint than the linear perspective is the so-called Manhattan constraint [34], which demands that there be three dominant and mutually orthogonal families of parallel line segments. Application of this additional regularity allows line segments to be labelled with their 3D orientation, but does not directly provide an estimate of the 3D surfaces or solid shapes in the scene.

To bridge this gap, we appeal to a long history of research in Gestalt psychology that identifies a principle of proximity and related cues of connectedness and common region as dominant factors in the perceptual organization of visual information [95, 94, 49, 167]. We use this principle of proximity repeatedly to construct successively more global and three-dimensional representations of the visual scene.

The proposed approach is anchored on a sparse line segment representation - Fig. 5.1 provides an overview of our Line-Segment-to-3D (LS3D) algorithm. Principles of proximity and good continuation are used to group local image edges into line segments, which are then labelled according to their Manhattan directions (Fig. 5.1(a)). A principle of proximity is then again employed to optimally group neighbouring orthogonal segments, forming local 2D minimal spanning Manhattan trees (MTs, Figs. 5.1(b-c)). Each of these local trees is then constructed into 3D using the Manhattan constraints. Note, however, that the relative depth of each 3D MT remains undetermined (Fig. 5.1(d)).

One of our main contributions is to show that each of these 3D Manhattan trees can be decomposed into a maximal set of non-subsuming 3D 3-junctions, 3-paths and L-junctions. Each of the 3-junctions and 3-paths defines a unique minimal covering cuboid, and each L-junction defines a unique minimal covering rectangle. The union of these cuboids and rectangles defines the 3D surface model for the tree (Fig. 5.1(e)).

The definition of these surfaces now allows the relative depth of these local 3D models to be resolved through a two-stage constrained optimization procedure. We first apply a principal of common region [167], minimizing the L_1 distance between parallel planes from different models that overlap in the image, forming sets of compound 3D models corresponding to connected regions in the image (Fig. 5.1(f)). Finally, we apply a principle of proximity to resolve the relative depth of these disjoint compound 3D models, minimizing the L_1 distance between parallel planes from distinct models, weighted by their inverse separation in the image. For both stages, occlusion constraints [62] play a crucial role in preventing physically unrealizable solutions. The resulting 3D scale model (Fig.

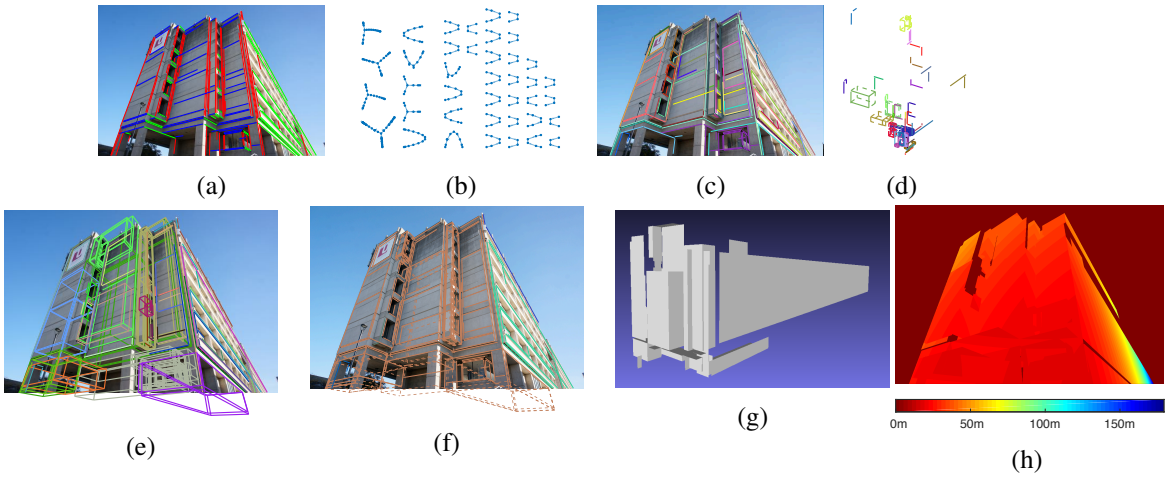


Figure 5.1: LS3D processing stages. (a) Detected Manhattan line segments. (b) Graphical structure of identified Manhattan spanning trees (MTs). Each vertex represents a line segment endpoint, and each edge represents either a real line segment or a junction between orthogonal segments. (c) MTs localized in the image (d) MTs lifted to 3D. Note that the relative depth of each MT remains unknown. (e) Minimal spanning cuboid/rectangle models. (f) Compound 3D models of connected structures. (g) Final model of visible surfaces. (h) Range map.

5.1(g)) can be used to generate a range map (Fig. 5.1(h)) for comparison with competing approaches.

Unlike deep learning methods, LS3D is not designed to recover an estimate of absolute depth for every pixel in the image, but rather an estimate of the Euclidean 3D layout of the Manhattan structures in the scene, up to a single unknown scaling factor. We therefore introduce a new 3D ground-truth dataset of solid massing building models and an evaluation framework suitable for the evaluation of such algorithms.

To summarize, our contributions are three: 1) We introduce a novel, explainable single-view 3D reconstruction algorithm called LS3D that infers the 3D Euclidean surface layout of Manhattan buildings, up to an unknown scaling factor, 2) We introduce a new 3DBM ground truth dataset of 3D Manhattan building models and a novel evaluation framework that allows single-view methods for 3D Manhattan building reconstruction to be evaluated and compared, and 3) Using this dataset and framework, we find that the LS3D method outperforms state-of-the-art deep-learning algorithms, both qualitatively

and quantitatively. The goal of this work is not to reconstruct general scenes. This is consistent with the computer vision tradition of focusing on important sub-problems, and making use of domain constraints. As we argue here, Manhattan structures are extremely common in our built environment and many non-Manhattan scenes can be modelled as a mixture of Manhattan frames. Thus it makes sense to have a specialized module for their reconstruction. Any system that does not take explicit advantage of Manhattan regularity will, we expect, fail to reconstruct a crisp orthogonal structure (See DNN output in Fig. 5.4)

5.3 Prior Work

Single-view 3D reconstruction is a classical computer vision problem that goes back to Roberts' PhD thesis [136, 64, 168, 83, 152]. More recent work has attempted to reconstruct piecewise planar 3D models of real scenes but under somewhat stronger assumptions. In their Photo Pop-up work, Hoiem *et al.* [73] modeled scenes as comprising three types of surfaces: ground, vertical and sky. Boosted decision tree classifiers were used to label superpixels from the image into one of these three semantic classes using a feature descriptor that includes appearance and geometric cues. The set of polylines defining the ground/vertical boundary was identified to estimate the 3D orientations of the vertical surfaces in the scene. Subsequent work globally optimizes the ground/vertical boundary [10] and generalizes to a larger range of camera poses and more fine-grained surface estimation [66].

While Hoiem *et al.* allowed vertical surfaces of arbitrary orientation, Coughlan and Yuille[33] observed that in the built environment, 3D scenes are often dominated by three mutually orthogonal directions (vertical + 2 horizontal) and developed a probabilistic approach to recover the rotation of this so-called Manhattan frame relative to the camera. Subsequent work [40, 154] refined this model to deliver more accurate Manhattan frames and to label the lines in the image according to their Manhattan direction.

The Manhattan constraint has been productively exploited by numerous subsequent 3D reconstruction algorithms. Delage *et al.*[39] developed a Bayes net model to identify the floor/wall boundary in indoor scenes and thus to recover the Euclidean floor/wall geometry. Hedau et al.[70] employed an even stronger constraint for indoor 3D room reconstruc-

tion, assuming that the room could be modeled as a single cuboid with intervening clutter. Subsequent improvements to indoor reconstruction based on this cuboid constraint has relied on novel features [101, 131, 111], physics-based constraints [62], Bayesian modeling [126], better inference machinery [52, 148], larger field-of-view [183], and supervised deep learning [37].

While these indoor room scenes are highly constrained, a more recent approach returns to the problem of reconstructing more general Manhattan scenes, indoors and outdoors [130]. Line segments are first detected and labelled with their Manhattan directions, and then a large set of potential 3D connectivities are identified between segment pairs. While many of these potential connectivities are false, an L1 minimization framework can identify the 3D solution that respects the maximal number of connection hypotheses, allowing the detected 3D line segments to be backprojected into 3D space.

As an alternative to the ground/vertical and Manhattan constraints one can assume that surfaces are linear 3D sweeps of lines or planar contours detected in the image. This constraint had led to interesting interactive systems, although fully automatic reconstruction is challenging [97].

The main competing fully automatic approach to constrained piecewise planar models attempts to recover an unconstrained range map, using supervised machine learning techniques. An early example is Make3D [143], which models range as a conditional random field (CRF) with naïve Bayes unary potentials over local edge, texture, and colour features and a data-dependent binary smoothness term.

More recent range map approaches tend to use deep neural networks [45, 108, 99, 107, 188, 55, 177, 129, 103]. For example, Eigen *et al.* train and evaluate a multi-scale CNN on their own NYU RGBD dataset of indoor scenes and the KITTI LiDAR dataset of road scenes [58, 45], while Laina *et al.* train and evaluate a single-scale but deeper ResNet concatenated with up-sampling layers [99] on Make3D [143] and NYU2 [45] datasets. Joint estimation of depth with surface orientation and/or a semantic category has been found to improve the accuracy of depth estimates [45, 177, 129].

One criticism of deep network approaches is the requirement for large amounts of labelled training data, but recent work demonstrates that deep networks for single-view range map estimation can be trained from calibrated stereo pairs [57, 103] or even uncalibrated video sequences [187] using reprojection error as the supervisory signal.

Recent research has also been exploring fusion of deep networks with more traditional computer vision approaches. The IM2CAD system [78], for example, focuses on the modeling of room interiors, optimizing configurations of 3D CAD models of furnishings and wall features by minimizing the projection error.

While deep networks have become the dominant approach to single-view 3D reconstruction, this approach has limitations. First, DNN models have millions of free parameters and are thus not easily interpretable. Second, while deep networks can provide an estimate of the rough scene layout, they typically fail to deliver the crisp and accurate geometry that is typical of urban environments. Third, most deep network approaches deliver a range map, which may be appropriate for some applications (e.g., navigation), but for applications such as construction, interior design and architecture a succinct CAD model is more useful.

We thus return in this paper to the classical geometry-driven approach. In particular, we ask, for the particular problem of single-view 3D Manhattan building reconstruction, how much can be achieved by a method that uses geometry alone, without relying upon any form of machine learning or appearance features. While the geometric approach has been criticized as unreliable [101], we show here that by integrating several key novel ideas with state-of-the-art line segment detection [4], reliable single-view 3D reconstruction of Manhattan objects can be achieved. By keeping the model simple we keep it interpretable, and by focusing on geometry, we deliver the crisp surfaces we experience in built environments, in a highly compact 3D CAD model form.

The focus on geometry and application of the Manhattan constraint links the proposed LS3D approach most directly to the line lifting algorithm of Ramalingam & Brand [130]. However, in this prior work there was no explicit grouping of line segments into larger structures, no inference of surfaces or solid models, and no quantitative evaluation of 3D geometric accuracy. LS3D thus goes far beyond this prior work in delivering quantitatively-evaluated 3D surface models. This is achieved through **three key contributions:**

1. While prior approaches [100, 101] use a 'line sweeping' heuristic to go from line segments to independent Manhattan rectangles, here we introduce a novel, principled approach to identify more complex 3D Manhattan trees, solving a series of three optimal bipartite matching problems to deliver spanning tree configurations

of orthogonal Manhattan line segments that together maximize proximity between grouped endpoints.

2. We introduce a novel method for converting these 3D Manhattan trees to surface models. The idea is based on decomposing each Manhattan tree into a maximal set of non-subsuming 3D 3-junctions, 3-paths and L-junctions. Each of the 3-junctions and 3-paths defines a unique minimal spanning cuboid, and each L-junction defines a unique minimal spanning rectangle. The union of these cuboids and rectangles defines the 3D surface model for the tree.

3. These 3D surface models contain multiple planes, providing stronger cues for estimating the relative depth of disconnected structures. We introduce a novel two-stage L_1 minimization approach that gives precedence to the Gestalt principle of common region [167] to first form compound 3D models of structures connected in the image, and later resolving distances between these disjoint structures.

5.4 The LS3D Algorithm

The LS3D algorithm is summarized in Fig. 5.1 and detailed below. Line segments are first detected and labelled according to Manhattan direction (Fig. 5.1(a), Section 5.4.1). From these, Manhattan spanning trees (MTs) are recovered (Fig. 5.1(b-c), Section 5.4.2) and then constructed into 3D models(Fig. 5.1(d), Section 5.4.3). A maximal set of minimal cuboids and rectangles that span each 3D MT is then identified (Fig. 5.1(e), Section 5.4.4) and their surfaces aligned in depth through a constrained L_1 optimization first for overlapping MTs (Fig. 5.1(f)) and finally for disjoint MTs (Section 5.4.5). The resulting 3D CAD model 5.1(g)) can be rendered as a range map (Fig. 5.1(h)) to compare with algorithms that only compute range maps.

5.4.1 Manhattan Line Segment Detection

We employ the method of Tal & Elder [154] to estimate Manhattan lines, based upon probabilistic Houghing and optimization on the Gauss sphere, and then the MCMLSD line segment detection algorithm [4], which employs an efficient dynamic programming

algorithm to estimate segment endpoints. MCMLSD produces line segment results that are quantitatively superior to prior approaches - Fig. 5.1(a) shows an example.

The MCMLSD algorithm identifies line segments that are co-linear: LS3D groups nearby co-linear segments into a single 'super-segment', but retains a record of the intermediate endpoints to support later surface creation (see below). We retain only segments over a threshold length.[1]

5.4.2 Manhattan Tree Construction

Prior line-based single-view 3D algorithms [100, 101] attempt to leap directly from line segments to 3D with no intermediate stages of perceptual organization. One of our main hypotheses is that the Gestalt principle of proximity coupled with sparsity constraints can yield a much stronger intermediate Manhattan tree representation that will subsequently facilitate global 3D model alignment.

First, a dense graph is formed by treating each segment as a vertex, and defining edges between pairs of vertices representing orthogonal segments with endpoints separated by less than a threshold distance.[2] (Note that an endpoint can lie on the interior of a super-segment.) To sparsify the graph we apply the constraint that each endpoint connects to at most one other endpoint in each of the two orthogonal Manhattan directions. This is achieved through a series of three optimal bipartite matchings, using a proximity-based objective function. Specifically, we seek the bipartite matching of all X segment endpoints to all Y segment endpoints that minimizes the total image distance between matched endpoints, and repeat for X and Z segments as well as Y and Z segments. These optimal bipartite matches are found in cubic time using the Hungarian algorithm [116]. We further sparsify the graph by computing the minimum spanning tree (MST) for each connected subgraph, generating what we will call local Manhattan trees (MTs, Fig. 5.1(b-c)).

[1] We use a minimum segment length of 100 pixels, and maximum gap between co-linear segments of 300 pixels. Sensitivity to these threshold is studied in Section 3.8.

[2] We use a threshold distance of 100 pixels - sensitivity to this threshold is studied in Section 3.8.

5.4.3 Lifting 2D MTs to 3D

Each of the MTs can be back-projected from 2D to 3D space using Manhattan direction constraints, up to an unknown distance/scaling constant λ. Assume a camera-centered world coordinate frame (X, Y, Z) in which the X and Y axes are aligned with the x and y axes of the image. Then any endpoint $\mathbf{x}_i = (x_i, y_i)^\top$ in the image back-projects to a 3D point $\mathbf{X}_i = \lambda(x_i, y_i, f)^\top$ in the scene, where f is the focal length of the camera. Note that while λ is unknown it must be the same for all endpoints in the MT.

Due to noise, Manhattan line segments will never be perfectly aligned with the Manhattan directions. Lifting an MT thus entails rectifying each segment to the exact Manhattan direction. We employ a sequential least-squares process. One of the endpoints $\mathbf{X}_0$ of the MT is first randomly selected as the 3D anchor of the tree: the 3D tree is assumed to pass exactly through $\mathbf{X}_0$. Then a depth-first search from $\mathbf{X}_0$ is executed, during which the Manhattan 3D location $\mathbf{X}'_j$ of each endpoint $\mathbf{X}_j$ is determined from the Manhattan location $\mathbf{X}'_i$ of its parent on the depth-first path by $\mathbf{X}'_j = \mathbf{X}'_i + \alpha\lambda\mathbf{V}_{ij}$, where $\mathbf{V}_{ij}$ is the 3D vanishing point direction for segment (i, j) and α is determined by minimizing $||\mathbf{X}'_i + \alpha\lambda\mathbf{V}_{ij} - \mathbf{X}_j||_2$. (Note that λ factors out of this minimization.) Figs. 5.1(c-d) show the MTs for an example image, each constructed into a 3D model up to a random scaling constant λ.

5.4.4 From Line Segments to Surfaces

A key contribution of our work is a novel method for inferring a surface structure from 3D MTs. We define a Manhattan three-junction as a triplet of orthogonal line segments that meet at a vertex of the MT, and a Manhattan three-path as a sequence of three orthogonal segments meeting end-to-end (Fig. 5.2). Since each segment can radiate from a junction in two ways, there are eight types of three-junctions, four that may be observed below the horizon and four that may be observed above (Fig. 5.2 Columns 2-3). Each three-path must begin at one of two endpoints of one of three segment types, and then continue to one of two endpoints of one of the remaining two segment types, and finally one of two endpoints of the remaining segment type. This leads to $2 \times 3 \times 2 \times 2 \times 2 = 48$ three-paths, however each of these has a metamer path that has been traversed in the opposite direction, so there are only 24 distinct three-paths, 12 that can be observed above the horizon and 12 that can be observed below (Fig. 5.2 Columns 4-9). Our main insight is that this collection

of 32 three-junctions and three-paths can be viewed as the the outcome of a generative process involving just four generic cuboid poses, two lying above the horizon and two below (Fig. 5.2 Columns 1).

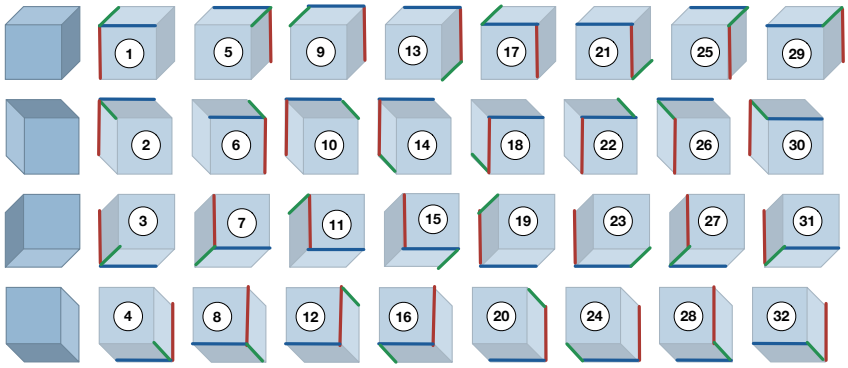


Figure 5.2: The 32 unique classes of Manhattan three-junctions and three-paths shown on the four classes of generic Manhattan cuboid poses.

This observation leads to a simple algorithm for bridging line segments to surfaces. We first decompose an MT into an exhaustive set of Manhattan three-junctions and three-paths. If any segments remain these are used to form two-paths with neighbouring orthogonal segments. This collection of three-junctions, three-paths and two-paths spans the MT. Three-junctions and three-paths are then used to spawn minimal spanning Manhattan cuboids as per Fig. 5.2, and two-paths, if they exist, span minimum spanning Manhattan rectangles. (Note that at an intermediate endpoint of a super-segment, the entire super-segment is considered to support the generated cuboid or rectangle - this serves to complete occluded surfaces.) Together, these cuboids and rectangles form a surface model for the MT.

It is important to distinguish our approach to inferring 3D surface models from prior work on recovering indoor scenes that 'sweeps' segments in orthogonal Manhattan directions [100, 101]. This sweeping approach estimates the 3D orientation of Manhattan rectangles, but not their relative depth, which must be resolved using strong constraints on the structure of the room (single floor and single ceiling connected by 'accordion' Manhattan walls).

By first connecting proximal orthogonal line segments into minimal Manhattan spanning trees, we provide the connectivity constraints necessary for producing more complex locally-connected 3D surface models, which generate much stronger constraints for

resolving relative depth (next section). This approach can be considered a quantitative expression of the 3D reasoning philosophy advocated by Gupta *et al.* [62], who argued for the use of simple solid models to make qualitative inferences about 3D scenes. While their goal was to compute qualitative spatial relationships between independent cuboids, we show that it is possible to recover *quantitative* 3D scene structures involving much more complex compound objects composed of many cuboids and rectangular surfaces.

5.4.5 Constrained L1-Minimization for Manhattan Building Reconstruction

A typical building generates many MTs and their relative distance/scaling must be determined. Our surface models allow us now to formulate a constrained L_1 optimization that identifies the scaling parameters minimizing separation between parallel planes while respecting occlusion constraints [62]. We partition the process into two stages based upon Gestalt principles of common region and proximity [167].

Stage 1 (Common Region): Let M represent the number of MTs in the model and let $\{\lambda_1, \ldots \lambda_M\}$ represent the unknown scaling parameters for these MTs. Visible rectangular facets from all MTs are projected to the image. The overlap in these projections defines an undirected common region graph $\mathcal{G}_{cr} = (\mathcal{V}_{cr}, \mathcal{E}_{cr})$ in which each vertex $i \in \mathcal{V}_{cr}$ represents a facet and each edge $(i, j) \in \mathcal{E}_{cr}$ represents overlap between parallel facets from different MTs. Fig. 5.1(f) shows the MTs within each connected component of this graph.

For each connected component $c \in [1, \ldots, C]$ of the graph we identify the MT m_c with the largest image projection and clamp its scaling parameter to $\lambda_{m_c} = 1$. Our goal is now to use linear programming (LP) to determine the remaining scaling parameters $\Lambda = \{\lambda_1, \ldots, \lambda_M\} \setminus \{\lambda_{m_1}, \ldots, \lambda_{m_C}\}$ that minimize the weighted distance between overlapping parallel planes from different MTs.

This minimization must, however, respect depth ordering constraints induced by the visibility of line segments. To code these constraints, for each MT i we identify all line segment endpoints $p_{ijk} \in P_{ij}$ that lie within a rectangular facet from another MT j. Letting d^-_{ijk} represent the depth of endpoint p_{ijk} when $\lambda_i = 1$ and d^+_{ijk} represent the distance to the overlapping facet from MT j along the ray from the camera centre to endpoint p_{ijk} when $\lambda_j = 1$, we have the depth ordering constraint $\lambda_i d^-_{ijk} \leq \lambda_j d^+_{ijk}$.

The resulting constrained optimization is thus:

$$\begin{aligned} \min_{\Lambda} \quad & \sum_{(i,j)\in\mathcal{E}_{cr}} |A_i \cap A_j| \cdot |\lambda_i d_i - \lambda_j d_j| \\ \text{s.t.} \quad & \lambda_i d_{ijk}^- \leq \lambda_j d_{ijk}^+, p_{ijk} \in P_{ij} \end{aligned} \tag{5.1}$$

Here $|\lambda_i d_i - \lambda_j d_j|$ is the distance between two parallel planes. We weigh this distance by the area of overlap $|A_i \cap A_j|$ of the two planar facets in the image.

Stage 2: (Proximity) Even after the scaling parameters of MTs within each connected component of the common region graph have been optimized to merge parallel planes in 3D, the relative scaling of each connected component will remain unknown.

To resolve these remaining degrees of freedom, we first identify a disjoint region graph $\mathcal{G}_{dr} = (\mathcal{V}_{dr}, \mathcal{E}_{dr})$ in which each vertex $i \in \mathcal{V}_{dr}$ represents a facet and each edge $(i,j) \in \mathcal{E}_{dr}$ represents two parallel facets from different MTs and different components that do *not* overlap in the image.

We then identify the connected component c with the largest image area and clamp its scaling parameter to $\lambda_{m_c} = 1$. We will now again use LP to determine the remaining scaling parameters $\Lambda_C = \{\lambda_{m_1}, \ldots, \lambda_{m_C}\} \setminus \lambda_{m_c}$ that minimize the weighted distance between non-overlapping parallel planes from different MTs and different components. We weigh this minimization by the sum of the areas $|A_i \cup A_j|$, and inversely by the minimum separation l_{ij} of the two planar facets in the image.

Note that although there is no overalap between the pairs of planar facets entered into the minimization, there may still be overlap between one or more visible line segments from one component and one or more facets from the other, and these must again be encoded as ordering constraints. The resulting constrained minimization is thus:

$$\begin{aligned} \min_{\Lambda_C} \quad & \sum_{(i,j)\in\mathcal{E}_{dr}} \frac{1}{l_{ij}} |A_i \cup A_j| \cdot |\lambda_i d_i - \lambda_j d_j| \\ \text{s.t.} \quad & \lambda_i d_{ijk}^- \leq \lambda_j d_{ijk}^+, p_{ijk} \in P_{ij} \end{aligned} \tag{5.2}$$

Fig. 5.1(g) shows the 3D surface model that results from this two-stage constrained minimization for an example image.

5.5 Evaluation Dataset

To evaluate the LS3D algorithm and compare it against state-of-the-art algorithms, we have created a new 3D ground truth dataset of 57 urban buildings that largely conform to the Manhattan constraint. The 3D building massing models (3DBMs) were obtained through the City of Toronto Open Data project from `www.toronto.ca/city-government/data-research-maps/open-data` and were simplified in MeshLab[30] to speed up processing. Fig. 5.3 shows some examples.

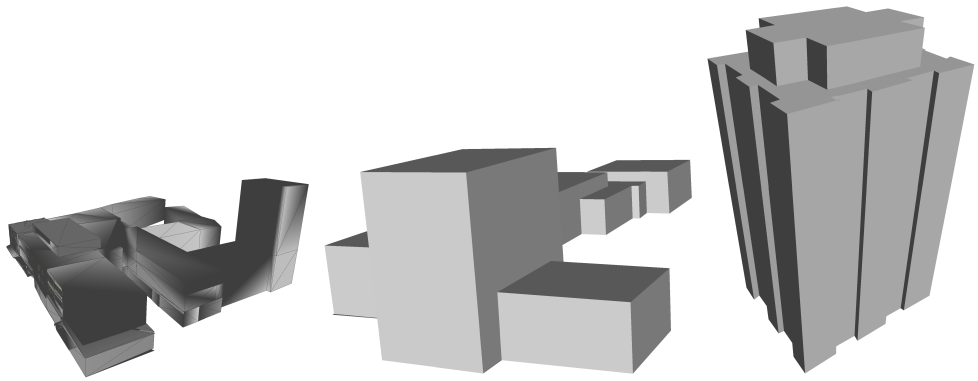


Figure 5.3: Some examples of 3DBM models in our dataset.

The number of images taken of each building depended upon access and the complexity of the architecture - 118 images were taken in total. We used a Sony NEX-6 camera with 4912×3264 pixel resolution. The camera was calibrated using the MATLAB Camera Calibration Toolbox to determine focal length (15.7mm) and principal point. The NEX-6 corrects for barrel distortion - our calibration procedure confirmed that it is negligible.

The camera was held roughly horizontally, but no attempt was made to precisely control height, roll or tilt. We attempted to take generic views of the buildings, but the exact viewing distance and vantage depended upon access and foreground obstructions. This dataset will be made available at `elderlab.yorku.ca/resources`.

To use the 3DBM dataset to evaluate single-view 3D reconstruction algorithms, we need to determine the rotation $\mathbf{\Omega}$ and translation τ of the camera relative to each of the 3DBMs. To this end, we manually identified between 5-20 point correspondences $(\mathbf{w_i}, \mathbf{x_i})$

in the 3DBM model and the 2D image, and then used a standard nonlinear optimization method (MATLAB fmincon) to minimize the projection error.

5.6 Evaluation

Our algorithm generates a 3D CAD model which is hard to compare to other state-of-the-art algorithms. In this case, we rendered the CAD model into a depth map for easier comparison.

We compare LS3D against the CRF-based Make3D algorithm[143] and four state-of-the-art deep learning approaches: the multi-scale deep network of Eigen *et al.* [46] (trained on NYUv2), the fully convolutional residual network (FCRN) of Laina *et al.* [99] (trained on Make3D and NYUv2), the deep ordinal regression network(DORN) of Fu *et al.* [55] (trained on NYUv2 and Kitti) and the very recent PlaneNet algorithm of Liu *et al.*[105] (trained on ScanNet).

The LS3D method estimates range only up to an unknown scaling factor α. Although the FCRN and DORN algorithm are trained to estimate absolute range, the Eigen algorithm is trained to minimize a partially scale-invariant loss function, and therefore should not be expected to deliver accurate absolute range estimates. Moreover, global scaling error has been reported as a significant contributor to overall error for such methods [46]. For these reasons we estimate a global scaling factor α for each algorithm and image independently by fitting the range estimates to the 3DBM ground truth. In particular, we estimate the value of α that minimizes the RMS deviation of estimated range $\hat{d}$ from ground truth range d, over all pixels that project from the 3DBM model.

The LS3D algorithm is not guaranteed to return a range estimate for every pixel that projects from the 3DBM, particularly when foliage and other objects intervene. To account for this, we employ two different methods to compare error between the LS3D and competing methods. In the *intersection* method, we measure the RMS error for all algorithms only for the intersection of the pixel set that projects from the 3DBM and the pixel set for which LS3D returns a range estimate. In the *diffusion* method, we interpolate estimates of range at 3DBM pixels for which LS3D does not return an estimate, by solving Laplace's equation with boundary conditions given by the LS3D range estimates at pixels where estimates exist and there are reflection boundary conditions at the frame of the image.

This allows us to compare RMS error for all algorithms over all pixels projecting from the 3DBMs. The input and output resolution of each algorithm varies - our 4912×3264 pixels images were resized to meet the input requirements of each algorithm.

Qualitative results are shown in Fig. 5.4. Make3D and the deep networks deliver range estimates that are sometimes correlated with ground truth, but these estimates are noisy and highly regularized. They generally fail to capture the dynamic range of depths over the 3DBM surface (deep red to dark blue). Moreover in some cases the estimates seem wildly inaccurate. In Column 1, for example, both versions of FCRN completely fail. In Column 2, all competing algorithms except perhaps DORN estimate the left face of the building as farther away than the right. In Column 3, all networks seem to fail. In Column 4 all networks fail to capture the receding depth of the left wall of the building.

The LS3D results are qualitatively different. The crisp architectural structure of each building is captured, along with the full dynamic range of depths. As expected, where good connectivity is achieved errors are minimal (Columns 2,4). In Column 1 and 3, however, some limitations can be seen, stemming from the failure to extract parts of the building occluded by vegetation.

We find that on average the LS3D method returns a range estimate for 83.3% of pixels projecting from the 3DBM model. For quantitative evaluations, we first average the error over all images of a particular building, and then report the mean and standard error over the 57 buildings in the dataset. Table 5.1 shows quantitative results based on the intersection measure of error. Of the prior algorithms, we find that PlaneNet[105] performs best, achieving a mean error of 9.35m (23.4%). However, LS3D beats this by a substantial margin (24.6%), achieving a mean error of 7.05m (17.7%). Matched-sample t-tests confirm that this improvement is statistically significant (Table 5.1). A comparison of the LS3D performance with and without occlusion constraints shows that these constraints yield a substantial improvement in performance. Mean errors are somewhat higher for all methods when using the diffusion method to evaluate over all pixels projecting from the 3DBM model (Table 5.2). PlaneNet is again the best of the deep networks, achieving a mean error of 10.6m (26.1%). However, LS3D beats this by 22.9%, achieving a mean error of 8.17m (19.4%). Matched-sample t-tests again confirm that this improvement is statistically significant (Table 5.2).

Our method is not intended to reconstruct an entire image or to operate on non-

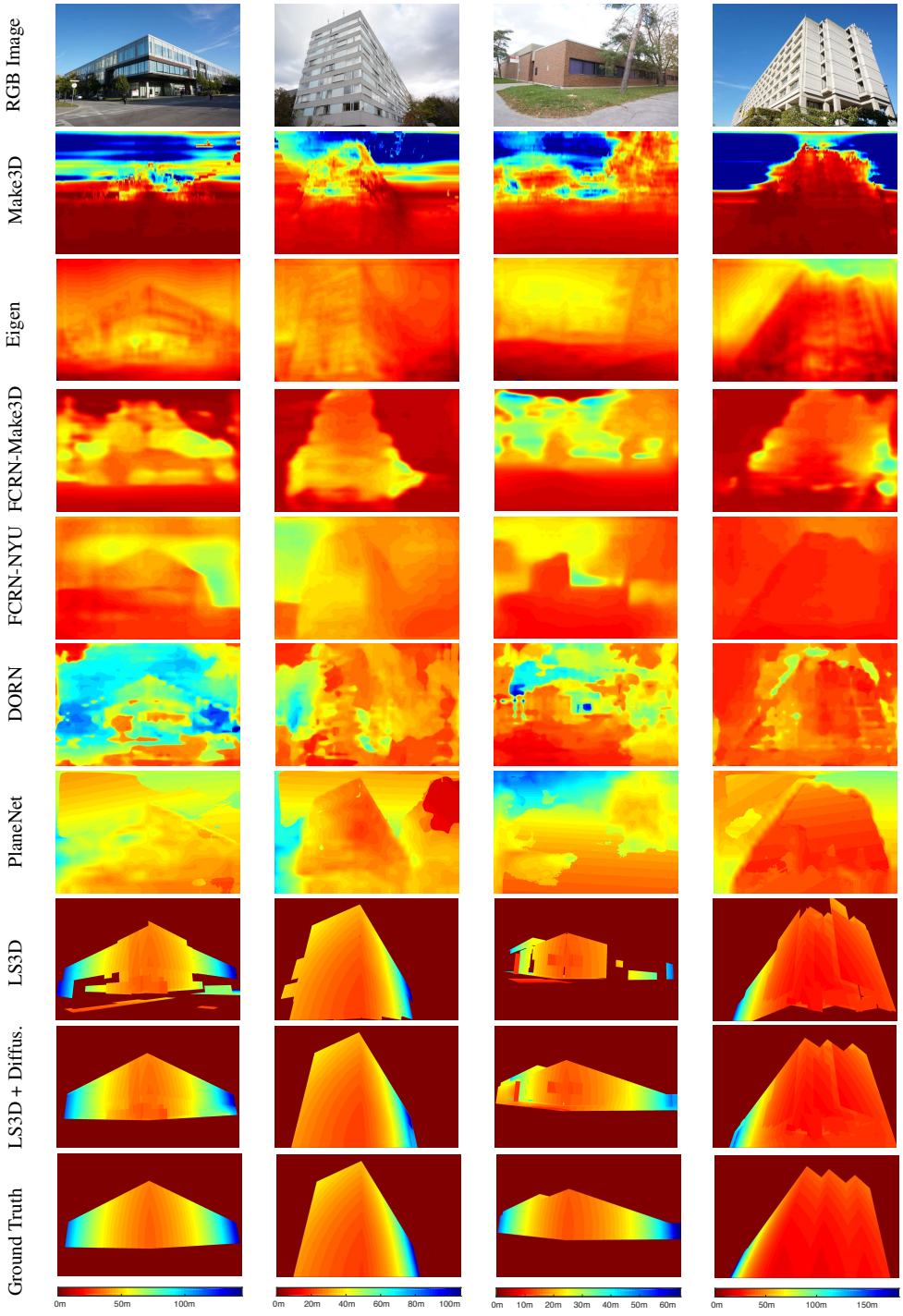


Figure 5.4: Example results for Make3D [143], Eigen [46], FCRN [99], DORN [55], PlaneNet [105], and the proposed LS3D method, with and without diffusion.

	Error Rate		p Value	
Methods	*RMSE(m)*	*RMSPE(%)*	*RMSE*	*RMSPE*
Make3D[143]	25.3	63.3	7.11E-18	4.36E-30
Eigen[46]	11.9	31.4	2.59E-07	2.74E-10
FCRN(Make3D)[99]	14.1	34.9	1.40E-10	1.32E-13
FCRN(NYU)[99]	11.0	28.1	3.91E-08	1.42E-09
DORN[55]	11.5	29.0	1.23E-08	1.05E-10
planeNet[105]	9.33	24.0	8.1E-03	1.2E-03
DenseNet[3]	10.4	26.2	7.93E-23	8.37E-35
DenseNet[3] eval. on test set	13.3	30.7	2.87E-22	4.46E-33
DenseNet[3] fine tuned eval. on test set	12.5	30.9	3.45E-22	7.78E-33
LS3D (no occlusion constraint)	8.02	20.2	5.60E-03	3.21E-02
LS3D (with occlusion constraint)	**7.03**	**18.0**	N/A	N/A

Table 5.1: Quantitative results using the intersection method of evaluation. Errors are computed only for pixels where the LS3D method returns a range estimate. p-values for matched-sample t-tests of the LS3D method (with occlusion constraint) against competing deep network algorithms are reported.

Manhattan structure. Nevertheless, we have evaluated its performance on the indoor $NYU2$ dataset. We achieve a mean error of 1.08m on the subset of pixels for which a range estimate is returned. This is not competitive with deep networks trained on NYU, for which mean error is on the order of 0.5 - 0.64m over the entire image, but is better than Make3D (1.21m). We believe the higher performance of deep networks on NYU2 is due to deviation from Manhattan constraints and the fact that DNNs overfit to the constant camera pose and similarity of environments in the dataset.

Additionally, in order to evaluate the impact of fine-tuning on our small dataset, we split the dataset into 50% training and 50% testing. The training set was used to fine tune DenseDepth model[3], which was pre-trained on the NYU2 dataset. The source code for DenseDepth was obtained from `github.com/ialhashim/DenseDepth`. We found that fine tuning led to a 6% on intersection and 5% on diffusion RMSE improvement.

Figure 5.5(a) shows best, median and worst case performance of our LS3D algorithm on our dataset. The worst case does not look that bad qualitatively, but the algorithm

	Error Rate		p Value	
Methods	*RMSE(m)*	*RMSPE(%)*	*RMSE*	*RMSPE*
Make3D[143]	27.1	65.9%	1.72E-19	7.52E-33
Eigen[46]	13.2	34.2	2.88E-08	4.21E-12
FCRN(Make3D)[99]	15.8	38.1	1.60E-12	5.76E-16
FCRN(NYU)[99]	12.3	30.7	2.01E-09	1.64E-12
DORN[55]	13.0	31.8	5.42E-12	3.60E-13
PlaneNet[105]	10.6	26.5	1.98E-05	1.49E-04
DenseNet[3]	12.0	29.8	7.91E-23	3.63E-32
DenseNet[3] eval. on test set	11.3	26.9	3.45E-22	5.74E-33
DenseNet[3] fine tuned eval. on test set	10.7	19.3	1.38E-22	3.21E-33
LS3D (with occlusion constraint)	**8.11**	**19.7**	N/A	N/A

Table 5.2: Quantitative results using the diffusion method of evaluation. Errors are computed for all pixels projecting from the 3DBM model. p-values for matched-sample t-tests of the LS3D method (with occlusion constraint) against competing deep network algorithms are reported.

incorrectly underestimates the depth of a small part of the building in the lower right corner of the image, and this leads to a large quantitative error.

LS3D has three main free parameters: 1) the minimum length of a line segment, 2) the maximum endpoint separation of connected orthogonal segments, and 3) the maximum separation of connected collinear line segments. Both 1) and 2) are currently set to 100 pixels, and 3) is set to 300 pixels. The dependence of performance on the exact value of these parameters is shown in Fig. 5.5(b). This analysis shows that these threshold values are reasonable, and that variation of up to $\pm 50\%$ in threshold values leads to at most a 10% reduction in coverage and a 7% increase in error.

Our current Matlab implementation of LS3D takes about 21 seconds to produce a 3D model from a 640×480 image. It could be made much faster by optimizing in C++.

The mesh surfaces generated from LS3D can be texture mapped from the input image. Fig. 5.6 shows an example texture mapped model.

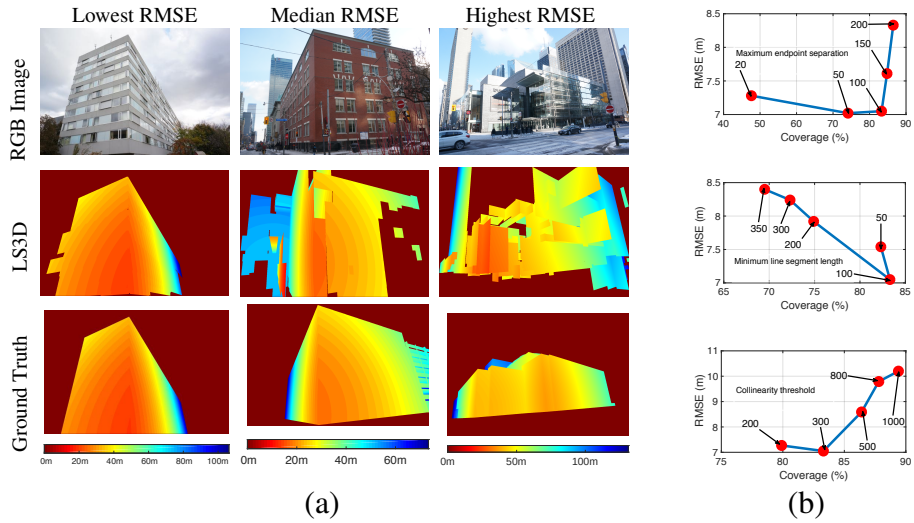


Figure 5.5: (a) Best, median and worst case LS3D performance on the 3DBM dataset. (b) LS3D parameter sensitivity analysis.

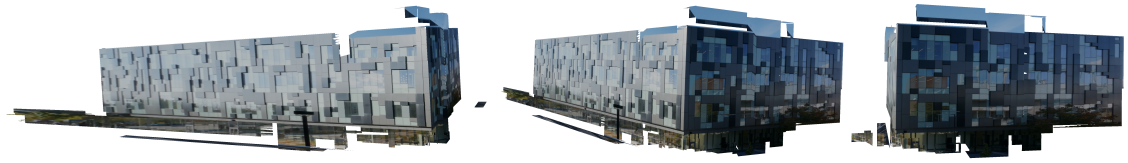


Figure 5.6: An example of texture mapped 3D model generated from LS3D.

5.7 Failure Mode Analysis

Our algorithm is sensitive to camera saturation, shadows, image noise, accidental alignments and occlusions. The three examples with highest error from the 3DBM dataset are shown in Fig 5.7. From these examples we can observe two main source of error in the reconstruction.

1. Accidental alignments.

2. Free-floating scattered structures.

Accidental alignments: line segments from different buildings sometimes align accidentally to form one line segment. This can create large errors during the LS3D recon-

struction. Accidental alignment may be caused by texture on the building, shadows or occluding structures.

Free-floating scatter structures: the second source of error is the scattered structures floating in the scene. Our algorithm assumes a minimum number of planes in the scene. When this assumption fails, free floating structure may appear. For example, a structure in the far background may be incorrectly aligned with a surface in the foreground, creating a huge numerical error during the evaluation.

5.8 Conclusion & Future Work

We have developed a novel algorithm called LS3D for single-view 3D Manhattan reconstruction. This geometry-driven method uses no appearance cues or machine learning yet outperforms state-of-the-art deep learning methods on the problem of 3D Manhattan building reconstruction. While this algorithm is not designed to reconstruct general 3D environments, we believe it will be useful for architectural applications.Future work will explore a mixture-of-experts approach which fuses the LS3D approach to reconstructing Manhattan portions of the environment with deep learning approaches for estimating the 3D layout of non-Manhattan structures.

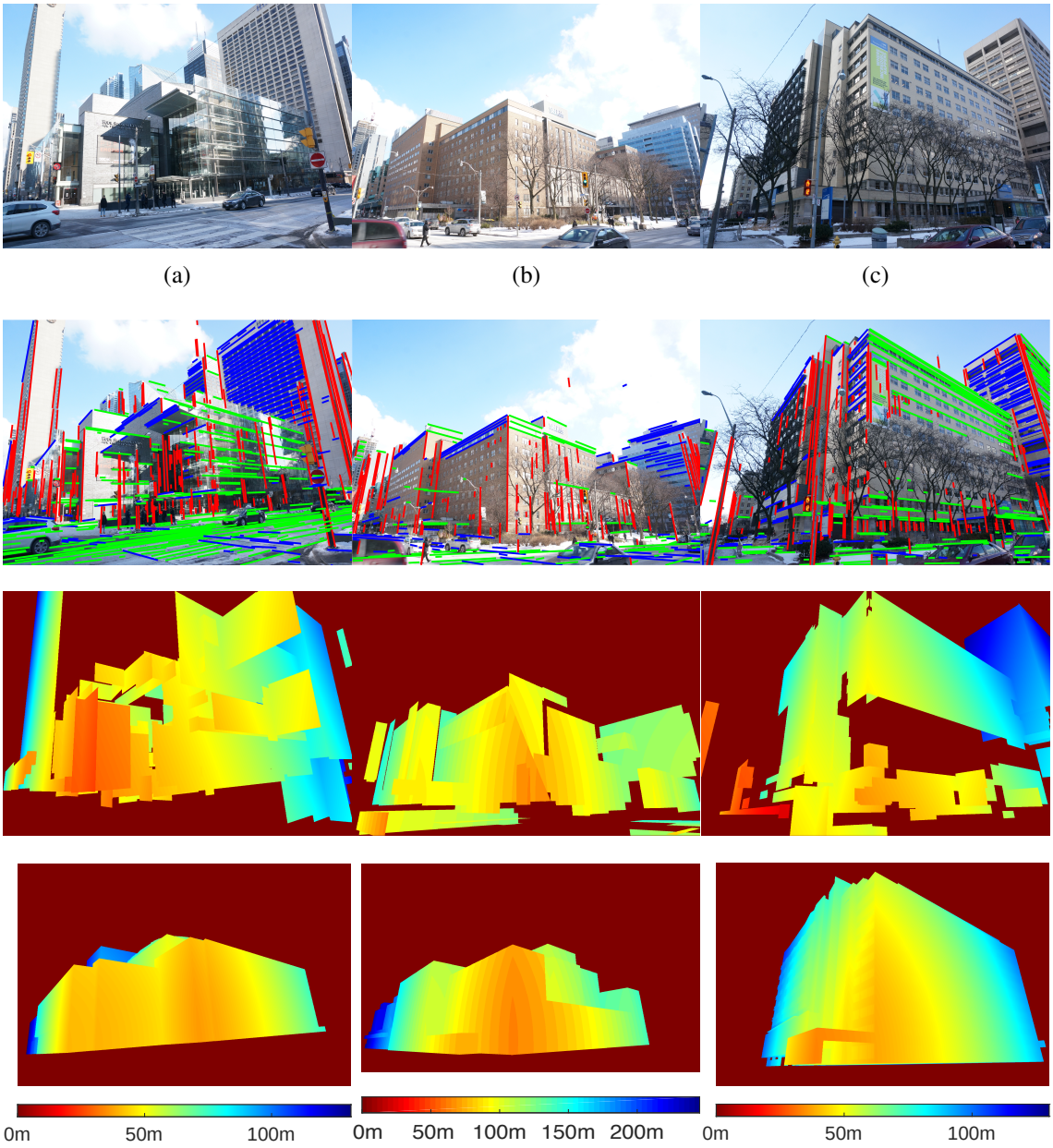


Figure 5.7: Examples with highest error (a) 37.2m (b) 30.8m (c)23.8m. First row is original image, second row is line segments overlaying on image, third row is estimated depth map, fourth row is ground truth.

Chapter 6

Discussion and Future Work

6.1 MCMLSD: Line Segment Detection

The MCMLSD line segment detection algorithm processes images of man-made scenes and identifies line segments that can be used by other algorithms. MCMLSD uses an edge detection algorithm [50] that relies upon an estimate of the standard deviation of the noise in the image. Performance suffers if this estimate is incorrect. Thus an algorithm that automatically estimates the noise parameter could have some benefit the line segment detection algorithm.

6.2 Road Segmentation

Follow-up work [5, 27, 28] on the road segmentation project has shown some improved segmentation performance. Future work will focus on application of this algorithm. One possible application a smartphone system to monitor snowplow vehicle status that could potentially save millions of dollars[53]. This smart phone based system powered by our algorithm can be easily mounted on a dashboard and report the road weather condition to a control centre in real time.

6.3 LS3D: Building Reconstruction From A Single Image

Lines are important features for estimating 3D geometrical information from 2D images. Perspective projection distorts many 3D properties such as angles, distances and ratios of distance, but one property is always preserved - line straightness. We can use lines to find vanishing points, estimate the horizon, and estimate the 3D structure of Manhattan buildings. Unlike deep learning based approaches, it is a computationally inexpensive way to estimate 3D geometry from an image with high generalizability.

However, this method does have limitations. First, the low level linear features alone contains limited semantic information about the scene. By combining it with high level features, this drawback can be reduced. Secondly, our LS3D algorithm (Chapter 5) is limited to Manhattan structures. Buildings with curves or oblique angles cannot be reconstructed. Nor can objects that are not piecewise planar. Part II of this thesis will introduce a novel algorithm to handle reconstruction of more general objects.

There are many opportunities to improve our LS3D algorithm. Ideas include:

1. Combine line cues with other features such as colour and texture.

2. Replace the Elder/Zucker edge detector [50] with a more recent detector such as the holistically nested edge detector [176], or the structured edge detector [43] to improve line segment detection.

3. Generalize our LS3D algorithm to other settings such as the Atlanta World assumption to allow it to process more complex scenes.

4. Generalize our LS3D algorithm to process curves.

5. Combine LS3D with deep learning approaches

6. Extend the algorithm to scenes with more than 3 vanishing points.

Combining LS3D with deep learning approaches could be a great direction to take as it would benefit from both traditional methods and the power of deep learning. Deep learning methods provides a low-resolution depth map for the scene. There are two potential ways to fuse Manhattan MTs with a depth map computed by a deep learning algorithm:

1. Use the depth map as a base, then use the MTs to refine the depth map.
2. Use the depth map as an additional constraint in the LS3D linear programming optimization process.

The first approach provides a way to smooth the noisy depth map surface. The second approach aims to counter the problem of false plane alignment.

Part II

Single-View 3D Estimation of General Objects

Chapter 7

Introduction

Multi-view methods for 3D object shape estimation such as stereopsis and motion parallax provide direct depth cues via triangulation, however these methods have limitations. Their accuracy is inversely proportional to the square of the distance to the object, making them less effective for distant objects, and they can fail when surface texture is too faint to generate features that can be tracked reliably across frames.

These limitations highlight the value of single-view cues such as shape from shading and texture. However, these surface cues also have limitations for 3D surface reconstruction. While providing useful local information about its qualitative surface shape, global shape estimation is subject to depth sign (convex/concave) and more general bas-relief ambiguities [160, 14, 89, 159].

Information carried by the shape of the object boundary has the potential to constrain these ambiguities. Here we consider in particular the problem of estimating the shape and pose of a smooth solid object from its bounding contour. To be clear on terminology, we follow [88] and refer to the set of surface points grazed by the view vector as the 3D *rim* of the object, and the perspective projection of these points onto the image as the 2D *bounding contour*.

It is well known that the bounding contour provides strong local constraints on shapes at the rim: convex points on the bounding contour must project from convex points on the 3D surface, while concave points on the bounding contour must project from saddle points. Moreover, the magnitude of curvature at a point on the 2D bounding contour is

proportional to the transverse curvature of the surface at the corresponding point on the 3D rim [88].

Here we consider whether the 2D bounding contour can also provide useful information about the *global* shape and pose of an object. We are motivated by evidence that human judgement of surface shapes is strongly influenced by the shape of the bounding contour [160, 159], and often even the bounding contour alone is enough to provide a compelling sense of volumetric shape [162, 47] (Fig. 7.1). Moreover, most natural and artifactual objects possess symmetries [127, 138] that will be carried to some degree by the 3D rim, and the distortions of those symmetries induced by perspective projection should provide cues to depth.

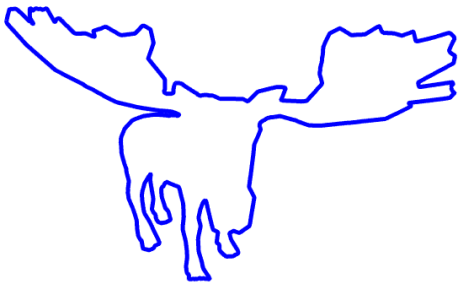


Figure 7.1: Volumetric shape from the bounding contour.

The 3D rim may be useful in its own right for certain applications, including free space estimation in autonomous navigation and surface contact point selection for robotic grasp planning. We also note that human stereoscopic 3D perception of an object is strongly driven by disparity cues at the object boundary [59]. This suggests that accurate estimation of the 3D rim will be crucial for 2D to 3D film conversion.

Part II of the thesis is organized as follows: Chapter 8 is a literature review, Chapter 9 will introduce my work of reconstructing 3D shapes given only the 2D contour, and Chapter 10 concludes with a discussion and proposal of future work.

Chapter 8

Literature Review

8.1 Introduction

Early computer vision algorithms for single-view 3D shape estimation exploited the bounding contour in conjunction with surface cues within an optimization framework. Typically, user interaction and/or some inflation term in the objective function was required to avoid a trivial (flat) solution. Users were required to specify the depth of some surface points [128], a fixed volume that the shape must fill [161], or an inflation term that indirectly determines the volume [123]. More recent approaches have managed to avoid user interaction or arbitrary inflation terms. Shape Collage [32] employs a non-parametric example-based approach for local surface patch estimation within an MRF framework and a thin-plate model to integrate the local patches into a global shape. A distinct branch of research explores the problem of class-conditional single-view 3D object reconstruction, which allows strong within-category regularities to be exploited [84]. The more recent state of the art algorithms are deep auto-encoders[174, 186, 115, 170, 60].

This chapter is organized as follows: In section 8.2 I will review the puffball reconstruction algorithm, in section 8.3 I will review the probabilistic learning based approaches, in section 8.4 I will review the deep learning based approaches and section 8.5 is the conclusion.

8.1.1 Puffball

A strategy for single-view estimation of smooth solid shapes was introduced in the interactive sketching interface dubbed *Teddy* [77] and later studied by Twarog [163] under the name *Puffball*. In this approach, the solid shape is defined as the envelope of spheres centred on the interior skeleton [18] of the shape in the image, and bi-tangent to the occluding contour (Fig. 8.1).

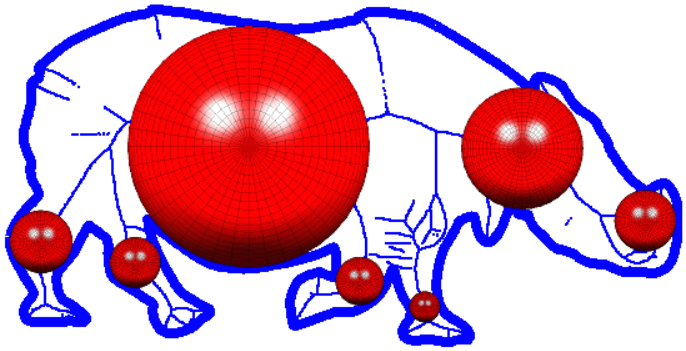


Figure 8.1: Puffball reconstruction (taken from [163]).

The puffball solution I for a silhouette S can be written as:

$$I(S) = \bigcup \left\{ B^3(p,r) \,|\, B^2(p,r) \subset S \right\} \tag{8.1}$$

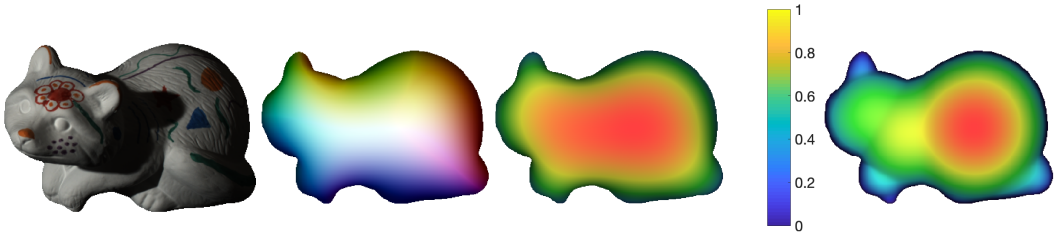
where $B^3(p,r)$ is the spherical ball centred on p with radius r and $B^2(p,r)$ is the maximal circular region centred on p with radius r contained within S.

The method is simple and can produce surprisingly reasonable results in some cases. However, a major limitation of this approach is that the 3D rim of the object is assumed to be planar and fronto-parallel, which in general is not true. Note also that surface normals at the rim are orthogonal to the optic axis, consistent with orthographic, but not perspective projection.

8.1.2 Probabilistic Learning Based Approach

In their Shape, Illumination and Reflectance from Shading (SIRFS) approach [12], Barron and Malik adopted a probabilistic framework, employing priors over shape, reflectance

and illumination together with surface constraints imposed by the boundary of the object. An example of the SIRFS algorithm output is shown in Fig. 8.2. Shapes are rendered orthographically.



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

Figure 8.2: An example of SIRFS and puffball algorithm output. The color bar represents the Z coordinate (increasing toward the eye). (a) original input image, (b) surface normal from SIRFS, (c) depth map from SIRFS. (d) depth map from puffball.

The SIRFS algorithm can be formulated as an unconstrained optimization problem:

$$\min_{Z,L} g(I - S(Z, L)) + f(Z) + h(L) \tag{8.2}$$

where $I = R + S(Z, L)$ is the log intensity of the image and R is a log-reflectance image. Z is a depth-map, L is a spherical-harmonic model of illumination, and S is a function of Z and L to produce a log-shading image. $g(R)$, $f(Z)$, and $h(L)$ are the cost function for reflectance, shape, and illumination respectively.

Reflectance cost function: The reflectance cost function $g(R)$ is formulated as

$$g(R) = \lambda_s g_s(R) + \lambda_e g_e(R) + \lambda_a(R) \tag{8.3}$$

where $g_s(R)$ is smoothness prior, $g_e(R)$ is parsimony prior, and $g_a(R)$ is absolute prior. The λ multipliers are learned through cross-validation on the training set.

Shape cost function: The shape prior consists of four components: 1) An assumption of smoothness encoded by the variation of mean curvature. 2) An assumption of isotropy of orientation of surface normals. 3) Given the orthographic projection, surface normals

must be orthogonal to the optic axis at the rim. 4) Shapes should resemble some noisy or incomplete external observation. These four components are formulated as the cost function:

$$f(Z) = \lambda_k f_k(Z) + \lambda_i f_i(Z) + \lambda_c f_c(Z) + \lambda_o f_o(Z, \hat{Z}) \tag{8.4}$$

Where $f_k(Z)$ is the smoothness model, $f_i(Z)$ is the isotropy model, f_c is the bounding contour model, and $f_o(Z, \hat{Z})$ encourages Z to be similar to some observation $\hat{Z}$. The λ multipliers are learned through cross-validation on the training set.

Illumination cost function: The illumination is modelled as a multivariate Gaussian model:

$$h(L) = \lambda_L (L - \mu_L)^T \Sigma_L^{-1} (L - \mu_L) \tag{8.5}$$

Where μ_L and Σ_L are the parameters of a multivariate Gaussian model of the coefficients of a low-order spherical harmonic model. Their dimensionality is 9 for grayscale and 27 for RGB images. λ_L is the multiplier learned from the training set. L is a feature vector from the spherical-harmonic model of illumination[132]. This feature vector is the derivation for the irradiance in terms of spherical harmonic coefficients of the lighting. The first 9 spherical harmonics were used as feature vector.

Karsch *et al* [85] extended this probabilistic framework to include surface normal constraints at internal geometric contours (self-occlusions and folds). They found that reconstructions from the silhouette improved more with the addition of these internal contours than with smooth shading. In fact, once internal contours were incorporated, adding shading cues was found to *lower* performance. This highlights the importance of contour cues for single-view 3D reconstruction.

Self-occlusion cost function: The boundary of a self-occlusion implies a discontinuity in depth, and thus the normal of the surface at the boundary should be aligned with the normal to the occluding contour. An appropriate cost function can be formulated as:

$$f_{selfocc}(Z) = \sum_{i \in C_{selfocc}} \sqrt{(N_i^x(Z) - n_i^x)^2 + (N_i^y(Z) - n_i^y)^2} \tag{8.6}$$

where $N = (N^x, N^y, N^z)$ is the surface normal and (n^x, n^y) is the normal of the 2D boundary.

Internal fold cost function: An internal fold in the surface denotes a discontinuity in surface normals across a contour along the object. It could be convex or concave. The

idea of the fold cost function is to constrain normals at pixels that lie across a fold to have convex or concave orientation and to be oriented consistently in the direction of the fold.

$$f_{fold}(Z) = \sum_{i \in C} \max(0, \epsilon - (N_i^l \times N_i^r) \cdot u) \tag{8.7}$$

where $u = (u_x, u_y, 0)$ is a fold's tangent vector in the image plane, and N_i^l, N_i^r is two corresponding normals across pixel i in the fold contour C. ϵ is a constant and it was set to $\frac{1}{\sqrt{2}}$.

8.1.3 Deep Learning Approaches

A distinct branch of research explores the problem of class-conditional single-view 3D object reconstruction, which allows strong within-category regularities to be exploited [84]. The more recent state of the art algorithms are deep auto-encoders. While typically trained and evaluated on a small number of object classes (e.g., chairs, cars, planes in MarrNet [174]), newer versions (GenRe) have been shown to generalize well to new object classes [186].

Fig. 8.3 is a schema of a basic auto-encoder structure. It consists of two components, encoder and decoder. In Wu's work[174], the encoder is ResNet-18[69] that encodes a 256×256 RGB image into a $8 \times 8 \times 512$ feature map. The decoder contains four sets of 5×5 fully convolutional and ReLU layers, followed by four sets of 1×1 convolutional and ReLU layers. Its outputs depth and the surface normal at a resolution of 256×256.

Taking the depth map and the surface normal estimation as input, Wu[174, 175] used another auto-encoder structure to generate a 3D voxel model. This voxel model can be further refined[186] by an additional auto-encoder network (GenRe model).

The Chamfer distance (CD)[13] was used to evaluate the reconstruction quality. Both GenRe and MarrNet were trained on chairs, cars, and airplanes.The algorithms were evaluated on instances from the training (seen) classes and from 10 other (unseen) categories: bench, boat, cabin, display, lamp, phone, rifle, sofa, speaker, and table. Results are shown in Table 8.1. The additional refinement network improves the 3D reconstruction performance.

Drawbacks of the deep learning approach include the large quantity of training data

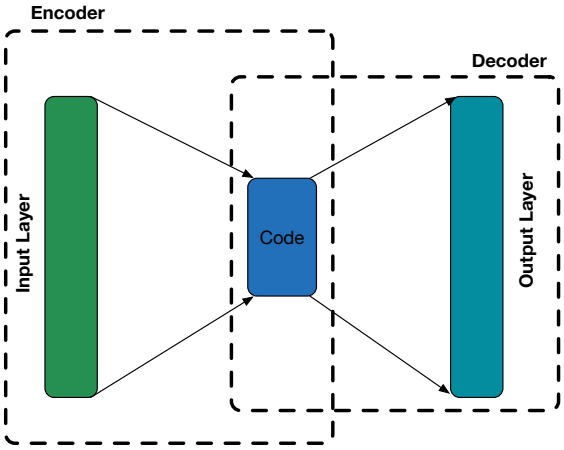


Figure 8.3: Schema of a basic Autoencoder

Table 8.1: Comparison of GenRe and MarrNet mean error on ShapeNet.

Models	Mean Error on Seen Classes	Mean Error on Unseen Classes
MarrNet	0.070	0.120
GenRe	0.064	0.106

required and the millions of free parameters that must be learned, which limits the explainability of the approach.

8.1.4 Conclusion

The puffball, probabilistic learning and deep learning approaches all have their unique advantages. Puffball is simple. The probabilistic learning approach allows for combination with reflectance, illumination, self-occlusion and fold cues. Instead of using handcrafted features, the deep learning approach learns features from training data, but suffers from a lack of explainability.

Chapter 9

3D Object Rim Reconstruction from 2D Occluding Contour

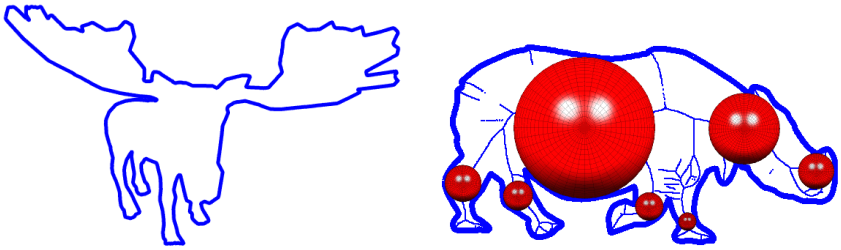
9.1 Introduction

In 1984, Koenderink wrote the seminal paper "What does the occluding contour tell us about solid shape?", in which he pointed out important qualitative relationships between the local shape of the occluding contour in the image and the local shape of the object surface [88]. However, this paper does not speak to whether the occluding contour can tell us anything *quantitative* about solid shapes. While strict quantitative constraints relating the occluding contour to solid shapes are unlikely, we posit here that typical regularities of common objects and rules of projection induce dependencies that can be used to derive statistical estimates of quantitative solid shapes from the occluding contour.

We follow Koenderink [88] and refer to the set of surface points grazed by the view vector as the 3D *rim* of the object, and the perspective projection of these points onto the image as the 2D *occluding contour*. Koenderink noted that the occluding contour provides strong local constraints on the qualitative surface shape at the rim: convex points on the occluding contour must project from convex points on the 3D surface, while concave points on the occluding contour must project from saddle points [88].

Here we consider whether the 2D occluding contour can also provide useful *quantitative* information about the global shape and pose of an object. We are motivated by

evidence that human judgement of surface shape is strongly influenced by the shape of the occluding contour [160, 159] - often the occluding contour alone is enough to provide a compelling sense of volumetric shape [162, 47] (Fig. 9.1(a)). Moreover, most natural and artifactual objects possess symmetries [127, 138] that will be inherited to some degree by the 3D rim, and the distortions of those symmetries induced by perspective projection should provide cues for depth.



(a) (b)

Figure 9.1: (a) The occluding contour can evoke a strong sense of solid shape. (b) Puffball reconstruction [77, 163].

The topologies of both the occluding contour and 3D rim can in general be quite complex; here we make two simplifications. First, we ignore self-occlusions, where the view vector both grazes and pierces the object, restricting our attention to the boundary of the object. One of the motivations for doing this is that self-occlusions can be trickier to detect in real images, since the figure and ground often have similar illuminations, colours and textures. Second, we ignore holes in the object projections, further focusing our attention on the outer boundary of the object. We leave analysis of self-occlusions and more complex topologies for future work.

With these simplifications, we partition the problem of estimating 3D shape from the occluding contour into two parts: 1) Estimation of the 3D rim from the 2D occluding contour, and 2) Estimation of the visible surface shape from the estimated 3D rim.

9.1.1 Estimating the 3D Rim from the 2D Occluding Contour

In what follows we will first demonstrate a statistical link between the shape of the occluding contour and depth variation in the 3D rim through a simple intuitive model that links

the depth of a point on the rim to the distance of the corresponding point on the occluding contour from the object's centre of mass. We then explore multivariate normal and auto-encoder models that more completely capture this statistical relationship. We show that a statistical model that links both the position and the tangent of the occluding contour to the depth of the rim generally yields superior results.

We note that the 3D rim may be useful in its own right for certain applications, including free space estimation in autonomous navigation and surface contact point selection for robotic grasp planning. We also note that human stereoscopic 3D perception of an object is strongly driven by disparity cues at the object boundary [59]. This suggests that accurate estimation of the 3D rim will be crucial for 2D to 3D film conversion.

9.1.2 Estimating the Surface Shape from the 3D Rim

Can the estimated 3D shape of the rim be used on its own, i.e., without direct surface cues, to deliver information about the 3D shape of the visible object surface? To explore this question, we develop and evaluate a generalization of the puffball approach [163]: a 3D object is completed by the union of maximal osculating spheres tangent to the estimated 3D rim and contained within its view cone.

Here we are not trying to compete with auto-encoder methods, which make use of many training instances of a small number of objects categories, and use all of the cues (e.g., shading, texture, self-occlusions, part structure) afforded by the colour imagery. Rather, we focus on the scientific question of whether the occluding contour carries quantitative information about 3D object shape, and if so, how that information can be harnessed. It is our hope that in the long run, a better understanding of the information afforded by the occluding contour will ultimately lead to better (and more explainable) multi-cue algorithms for single-view 3D object reconstruction.

Generally, prior work has focused on orthographic projection, which is unrealistic and also ignores important 3D information available in the distortions induced by perspective projection. A specific contribution of the present paper is to examine the 3D information afforded by the occluding contour when viewed in perspective.

9.1.3 Summary of Contributions

In summary, we make five specific contributions:

1. We introduce two novel datasets consisting of 3D object rims and their 2D projections. We will make these datasets public to encourage continuing research on 3D shapes from contours.

2. We demonstrate a statistical connection between the 3D shape of the object rim and the observable 2D shape of its occluding contour, and capture this relationship with a series of novel statistical models.

3. We show that these models can be used to make predictions of the depth variation in the 3D object rim from the 2D occluding contour alone.

4. We introduce a novel spherical completion approach for reconstructing the visible surface based solely on the estimated 3D rim.

5. We show that our approach yields more accurate estimates of the 3D object shape and pose than competing approaches [163, 12].

9.2 Datasets

We employ two datasets of solid 3D objects. The first dataset comprises 122 scanned objects originally employed by Mehrani *et al* [114] and obtained directly from the authors. (These 3D scans were originally sourced from a variety of online datasets including BigBIRD [150] and YCB [21]). We randomly split the Mehrani dataset into training and test sets of 61 objects each. For each FOV we generated 1,000 random image projections for each training object and 20 random projections for each test object.

The second dataset is the ShapeNet Core [24] dataset of 52,472 synthetic objects. We split the ShapeNet Core dataset objects into random training (60%), validation (20%) and 20% test (20%) partitions. For each FOV we generated 20 random image projections for each training object and 1 random projection for each test object.

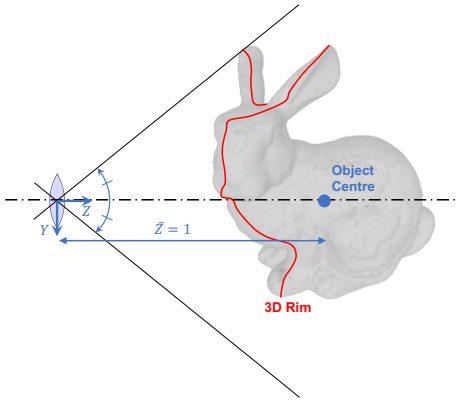
(a) (b)

Figure 9.2: Example 3D objects from (a) the Mehrani dataset [114] and (b) the ShapeNet Core dataset [24].

Samples from both datasets are shown in Fig. 9.2. We selected these two dataset because, at the time of writing, they provide the highest quality 3D mesh models for the goal of 3D object reconstruction.

Perspective projections were formed using a virtual pinhole camera with unit focal length and field of view (FOV) $\in (2, 4, 8, 16, 32, 64)$ deg (Fig. 9.3). We employ a camera-centred coordinate frame with Z representing distance from the lens plane along the optical axis. Objects were centred on the optic axis with centroids at a depth of $\bar{Z} = 1$. (Note that under perspective projection, points on the rim will have an average depth $Z < 1$.) The size of each object was adjusted so that the object was just contained within the field of view, grazing the frustrum at least one point. The rim was sampled at 32 points with equal arc-length separation, and these 32 points were projected to the image to form the occluding contour. Each 3D rim $\gamma_3(s) = (X(s), Y(s), Z(s))$ is thus a 32×3 matrix, and the corresponding 2D occluding contour $\gamma_2(s) = (x(s), y(s))$ is a 32×2 matrix.

To facilitate learning, we rotated all contours in the image plane to align the first principal component with the x-axis of the image, with the taller side on the left (i.e., points with $x < 0$ have greater y-variance then points with $x > 0$.)



(a)

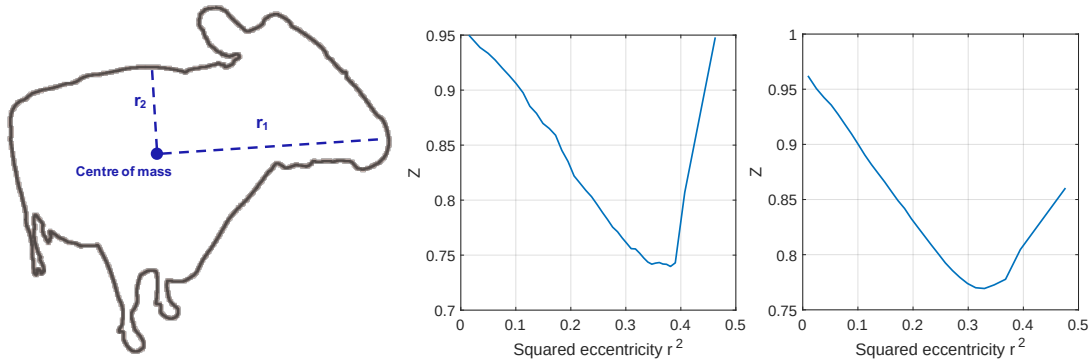
Figure 9.3: Viewing geometry. Objects are centred on the optical axis, with centroids at a distance of $\bar{Z} = 1$, and scaled so that the maximum angular eccentricity is half the prescribed field of view.

9.3 Estimating the 3D Rim from the 2D Occluding Contour

9.3.1 Eccentricity Model

Why do we expect the occluding contour to carry information about depth variation in the 3D rim? In explaining perspective projection to a child, one might start with the fact that as objects get closer to the eye they get bigger in the image. Applying the inverse of this logic to the bounding contour, we might predict that more eccentric points on the occluding contour (i.e., points that are further from the centre of mass) project from points on the rim that are closer to the eye (Fig. 9.4(a)).

To explore this idea, we examine the statistical relationship between eccentricity $r(s)$ of points on the occluding contour and the depth ($Z(s)$) values of the corresponding points on the rim. Empirically, we find that the relationship between $Z(s)$ and $r(s)^2$ is nearly linear, and so to form a model we bin the ground truth Z values from our training datasets as a function of $r(s)^2$. The bin width is selected to minimize leave-one-out cross-validation error over objects for the Mehrani dataset, and error on the validation partition for the ShapeNet Core dataset.



(a) (b) (c)

Figure 9.4: The eccentricity model for estimating distance $Z(s)$ from the image plane. (a) The cue is the squared distance $r(s)^2$ of the occluding contour from the centre of mass of the contour in the image. (b-c) Histogram models for the Mehrani and ShapeNet datasets.

The systematic relationship between eccentricity and depth can be seen clearly in the resulting histograms - Figs. 9.4(b-c) show the results for 64 deg FOV. This clearly demonstrates that the occluding contour carries information about depth variation in the 3D rim. We explore the performance of this simple model in our Evaluation section below, but intuitively it seems unlikely that the eccentricity model captures all of the statistical information that the occluding contour can provide. We therefore turn now to more general statistical models that we hope can more fully capture this relationship.

9.3.2 Normal Models

When exploring a statistical relationship for the first time it is natural to consider a normal model, and here we consider two. In our base model, we assume that the occluding contour $\alpha(s) = (x(s), y(s))$ is jointly normal with the unknown depth coordinate $Z(s)$: $(\alpha(s), Z(s)) \sim \mathcal{N}(\alpha(s), Z(s); \mu_1, \Sigma_1)$. Importantly, we model the covariance across all pairs of points (s_1, s_2) on the occluding contour and rim. As a result the model consists of a 96-dimensional mean vector μ_1 and a 96×96 covariance matrix Σ_1.

We also explore a second, augmented normal model. The prevalence of orientation regularities such as parallelism and rectilinearity, and the importance of a linear perspective in human perception, suggests that the local orientation of the occluding contour may

also be an important cue to depth. While the tangent vector $t(s)$ of the occluding contour is implicitly defined by the contour $\alpha(s) = (x(s), y(s))$ itself, the linear nature of the normal model may limit its capacity to capture the influence of the tangent on the depth of the rim. In our second model, we therefore augment the occluding contour coordinates with the tangent vector $t(s) = (t_x(s), t_y(s))$: $(\alpha(s), t(s), Z(s)) \sim \mathcal{N}(\alpha(s), t(s), Z(s); \mu_2, \Sigma_2)$. This model consists of a 160-dimensional mean vector μ_2 and a 160×160 covariance matrix Σ_2.

Maximum likelihood estimates of these parameters are estimated from the training data. The parameterized models can then be used for inference: given a partially observed test vector $\alpha(s)$ (base model) or $(\alpha(s), t(s))$ (augmented model), the expectation of the unobserved depth $Z(z)$ can be estimated using the standard conditional expectation formula ([16], Eqn. 2.81):

$$\mu_{Z|u} = \mu_Z + \Sigma_{Zu}\Sigma_{uu}^{-1}(u - \mu_u) \tag{9.1}$$

where u represents $\alpha(s) = (x(s), y(s))$ for the base model and $(\alpha(s), t(s))$ for the augmented model.

9.3.3 Auto-Encoder Model

Given the success of auto-encoders for pixel-wise single-view 3D object reconstruction, it is natural to consider an auto-encoder for estimating depth variation in the 3D rim from the occluding contour. We trained the auto-encoder to minimize squared error in depth Z. For both the Mehrani and the ShapeNet datasets we found empirically that a simple architecture with just one encoder layer and one hidden layer performs best (Fig. 9.5): larger architectures tend to reduce performance on unseen data.

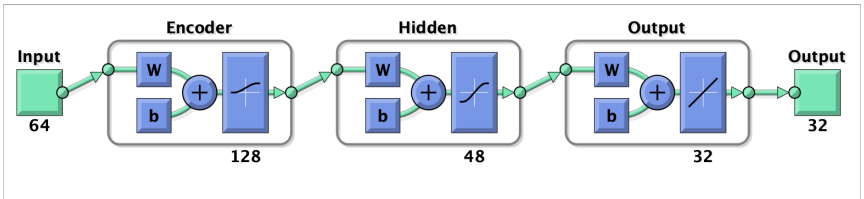


Figure 9.5: Optimized auto-encoder models for the Mehrani dataset. The model for ShapeNet is identical but with 16 hidden units.

Given their difference in size, we optimize the number of units in each layer separately for the two datasets. We perform this optimization by maximizing the Pearson correlation between the ground truth depth and the estimated depth for a field of view of 64 deg, using leave-one-out cross-validation on the Mehrani training dataset and the ShapeNet validation dataset. For both datasets, we found a 128-unit encoder layer to be optimal. For the Mehrani dataset, we found an 48-unit hidden layer to be optimal, while for the ShapeNet dataset we found 16 hidden units to be optimal.

9.4 Estimating the Surface Shape from the 3D Rim

Can we use an estimate of the 3D rim to generate an estimate of the visible object surface? In the puffball approach [77, 163], the solid shape is defined as the union of spheres centred on the interior skeleton [18] of the shape in the image, and bi-tangent to the occluding contour (Fig. 9.1(b)). A major limitation of this approach is that the 3D rim of the object is assumed to be planar and fronto-parallel, which in general will not be the case. Here we propose a generalization of the puffball method that can be applied to oblique non-planar 3D rims. The method produces a 3D volumetric estimate of the object (Fig. 9.6).

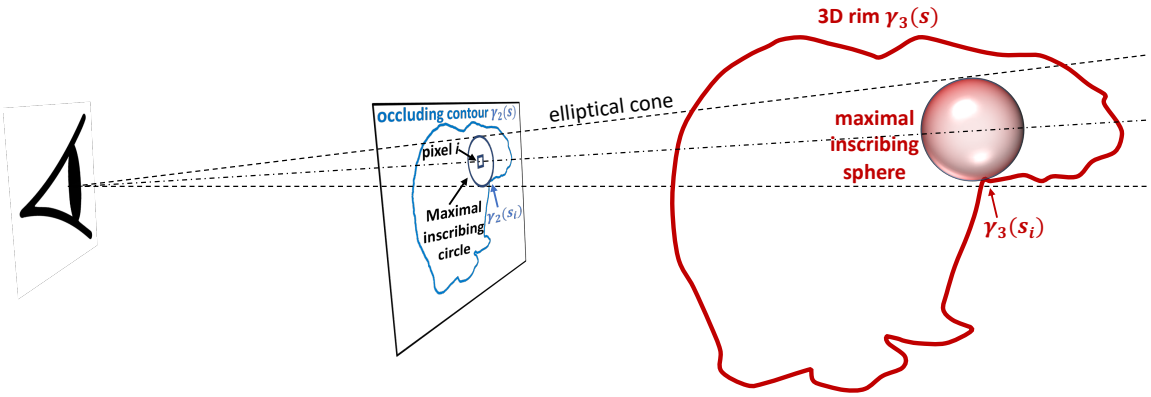


Figure 9.6: Spherical surface completion.

For each pixel i of the image interior to the occluding contour we identify the maximal inscribing circle centred on the pixel. This circle lies entirely within the shape, is tangent to at least one point $\gamma_2(s_i)$ on the occluding contour, and defines an elliptical cone with

an axis passing through the optical centre and the pixel i. This cone is tangent to the rim at the corresponding rim point $\gamma_3(s_i)$. We identify the unique maximal sphere that lies within this cone and is tangent to the cone at $\gamma_3(s_i)$; note that the projection of this sphere is the inscribing circle. The union of these spheres over all pixels i in the interior of the 2D occluding contour defines our estimate of the solid 3D shape. Note that the surface normal of the estimated shape will be orthogonal to the view vector at each rim point, as required for a smooth solid object.

In practice, the computation involves identifying the orthogonal projection of each of the k 3D rim points onto each of the n interior pixel rays to define kn tangent spheres. Then we must check the distance of all other $k-1$ rim points from the centre of each sphere, eliminating any spheres that subsume other rim points, and therefore identifying the unique maximal inscribing sphere for each pixel. The computation thus has complexity k^2n.

Since the 3D locations and radii of the inscribing spheres tend to vary smoothly in a local pixel neighbourhood, the spherical completion method generates smooth solid completions. This will work well for smooth objects, but less well for objects with sharp folds at the rim.

9.5 Evaluation

9.5.1 Estimating the 3D Rim from the 2D Occluding Contour

We first evaluate our models for estimating the 3D rim from the occluding contour. As perspective projection cues to depth generally increase with FOV, we first train and evaluate the accuracy of these models as a function of FOV, on the Mehrani training and test sets, respectively. We vary the FOV over $[2, 4, 8, 16, 32, 64]$ deg.[1] As a measure of performance we use the Pearson correlation between the ground truth depth values Z and the estimated

[1]Some useful points of reference: Human monocular field of view is roughly 135 deg horizontally and 180 deg vertically. The diameters of the human fovea, parafovea and perifovea are roughly 5, 8 and 18 deg, respectively. A typical super-telephoto 400m lens will have a FOV of roughly 6 deg. A standard 50mm lens will typically have a FOV of roughly 40 deg. A recent iPhone has a standard FOV of roughly 57.5 deg and a telephoto FOV of roughly 31.8 deg.

depth values $\hat{Z}$ over all points on the rim.

Fig. 9.7 shows that correlation increases monotonically for all methods as a function of FOV. We find that the normal and auto-encoder models outperform the simple eccentricity model by a large margin. The advantage of the augmented normal model over the base normal model confirms our intuitions that the local orientation of the occluding contour is informative about the depth of the rim. Interestingly, the auto-encoder model was found to underperform the normal models.

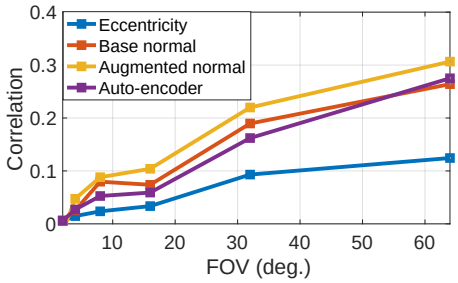


Figure 9.7: Pearson correlation between the ground truth depth values Z and the estimated depth values $\hat{Z}$ over all points on the rim, averaged over objects in the Mehrani test dataset. Models were trained on the Mehrani training dataset.

To provide a qualitative feel the 3D rim estimates, Fig. 9.8(second column) shows best, median and worst case estimates of the rim depth Z (as measured by correlation) for test shapes in the Mehrani dataset. In the best case the estimate is excellent and the correlation almost perfect, but in the worst case the algorithm flips the sign of the depth. The median cases are most representative: the low frequency trend of the rim depth is captured, but finer details are lost and the amplitude of depth variation is attenuated. Analogous results for the ShapeNet dataset are shown in the supplementary material.

To explore the generality of these approaches, we also train and evaluate on the ShapeNet dataset, focusing on a FOV of 64 deg, close to the FOV for the standard lens of a typical smart phone camera. Table 9.1 shows performance of all models on both Mehrni and ShapeNet test sets, trained on their respective training sets. Interestingly, we see that the auto-encoder model performs better than the normal models on the Shapenet dataset. This may be due to the fact that the ShapeNet objects tend to be less smooth, resulting in highly non-linear statistical dependencies that are more easily captured by the multi-layer auto-

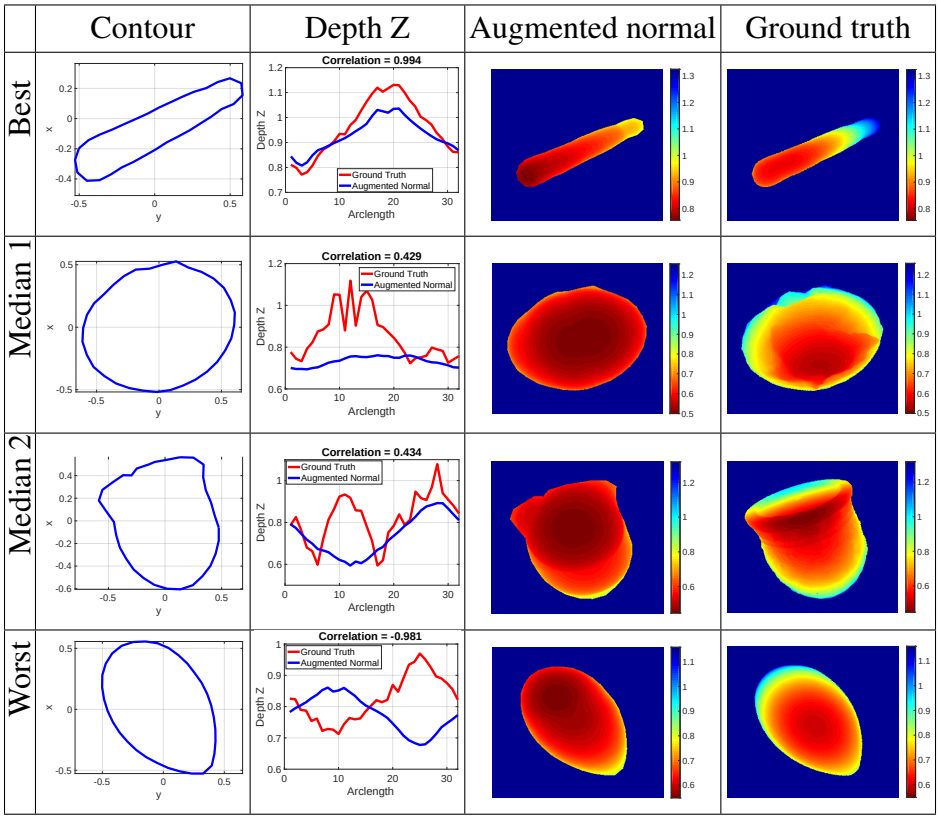


Figure 9.8: Best, median and worst case estimates (as measured by correlation of the 3D rim depth estimates $\hat{Z}$ with ground truth Z) for the augmented normal model on the Mehrani dataset. The last two columns show the estimated depth of the visible surface inferred by spherical completion from the estimated rim, and ground truth depth, respectively.

encoder.

We also examine generalization across the datasets, training on one and testing on the other (Table 9.2. While we see a drop in performance in both cases, the drop is less profound when generalizing from the ShapeNet to the Mehrani dataset. This is unsurprising, given that the ShapeNet training dataset is much larger than the Mehrani training set, one would expect better generalization. But this difference may also be due to the greater diversity of the ShapeNet dataset. Since it contains both smooth and less smooth objects it supports inference for both, whereas the Mehrani dataset, containing primarily smooth ob-

Table 9.1: Within-dataset Pearson correlation between the ground truth depth values Z and the estimated depth values $\hat{Z}$ over all points on the rim, mean±std. err. over test objects, for 64 deg FOV.

Model	Mehrani	ShapeNet
Eccentricity	0.12 ± 0.008	0.16 ± 0.003
Base normal	0.26 ± 0.016	0.31 ± 0.004
Augmented normal	0.31 ± 0.015	0.33 ± 0.004
Autoencoder	0.27 ± 0.015	0.36 ± 0.004

jects, could be expected to fail when presented with less smooth objects from the ShapeNet dataset.

Table 9.2: Between-dataset Pearson correlation between the ground truth depth values Z and the estimated depth values $\hat{Z}$ over all points on the rim, mean±std. err. over test objects, for 64 deg FOV. ShapeNet→Mehrani: Train on ShapeNet training set, test on Mehrani test set. Mehrani→ShapeNet: Train on Mehrani training set, test on ShapeNet test set.

Model	ShapeNet→Mehrani	Mehrani→ShapeNet
Eccentricity	0.097 ± 0.009	0.15 ± 0.003
Base normal	0.25 ± 0.014	0.23 ± 0.005
Augmented normal	0.25 ± 0.014	0.22 ± 0.005
Autoencoder	0.25 ± 0.014	0.23 ± 0.004

Finally, we make use of the ShapeNet category labels to examine how 3D rim estimation accuracy may depend upon the type of object being viewed, and whether there is generalization across categories. We focus here on the augmented normal model, and consider three levels of generalization: (1) Train on all training data, (2) Train individually on the training set for each category, (3) Train on all categories but the test category (leave one out).

Fig. 9.9 shows the results ranked by performance when trained on all training data. We see substantial variation in performance across categories. Performance is generally better for smooth, single-part objects (e.g., clock, bottle, basket) that are more volumetric,

and worst for objects that contain multiple parts or are less volumetric or contain parts that are almost 2D (e.g., bag, earphone, knife seen in Fig. 9.10). On the other hand, we see almost no drop in performance when the test category is not included in training, indicating that the model is learning more general geometric principles rather than memorizing categories.

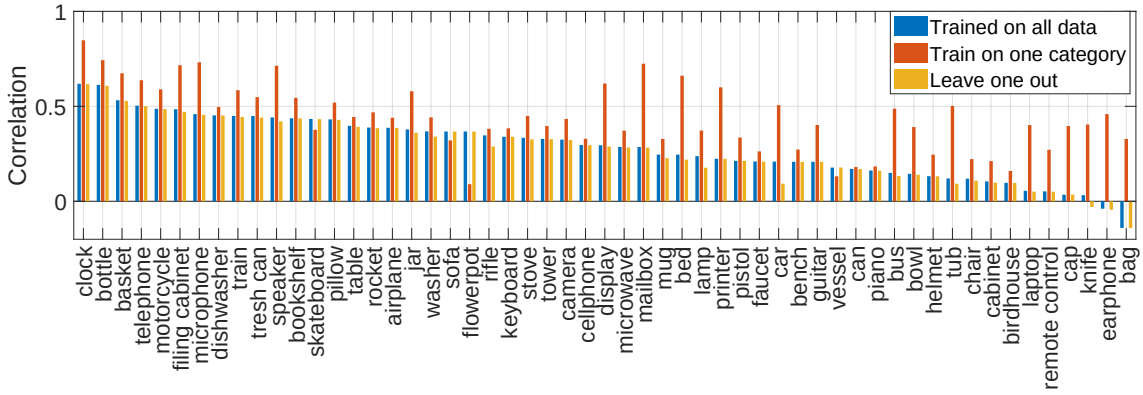


Figure 9.9: Categorical evaluation on Augmented normal. (a)Trained on all data. (b) Trained and evaluated on each category independently. (c)Leave one out evaluation on each category.

Figure 9.10: Examples of the three easiest (left) and three hardest (right) ShapeNet categories, in terms of 3D rim estimation from the occluding contour.

9.5.2 Estimating the Surface Shape from the 3D Rim

To evaluate the potential of using the estimated 3D rim for surface reconstruction, we apply our spherical completion method to 3D rims estimated using the eccentricity, base normal, augmented normal and auto-encoder models. We compare the results against the shape-from-contour version of SIRFS SIRFS [12] and the Puffball method [77, 163].

We also evaluate an idealized model that applies the spherical completion method to the ground truth 3D rim: this provides an indication of how improvements to 3D rim estimation could influence the accuracy of surface completion. We evaluate all methods in terms of the mean Pearson correlation and RMS error between ground truth and estimated surface depth over the pixels on the object.

Table 9.3 shows the results. We find the eccentricity model to be weak. It does produce a lower RMS error than SIRFS and Puffball, but only because it generates more conservative (flatter) shape estimates: the correlations with ground truth are substantially lower than both SIRFS and Puffball.

Table 9.3: Pearson correlation and RMS error between the ground truth surface depth values and the estimated depth values over all pixels of the shape, mean±std. err. over test objects, for 64 deg FOV.

	Mehrani		ShapeNet	
Model	Correlation	RMS Error	Correlation	RMS Error
SIRFS	0.60 ± 0.008	0.262 ± 0.003	0.17 ± 0.003	0.238 ± 0.001
Puffball	0.58 ± 0.01	0.172 ± 0.003	0.27 ± 0.003	0.207 ± 0.001
Eccentricity	0.43 ± 0.01	0.154 ± 0.003	0.07 ± 0.003	0.222 ± 0.001
Base normal	0.58 ± 0.01	0.166 ± 0.002	0.26 ± 0.004	0.223 ± 0.001
Augmented normal	$\mathbf{0.60 \pm 0.01}$	$\mathbf{0.168 \pm 0.002}$	0.29 ± 0.01	0.225 ± 0.001
Auto-encoder	0.58 ± 0.01	0.165 ± 0.003	$\mathbf{0.32 \pm 0.004}$	$\mathbf{0.224 \pm 0.001}$
Ground truth rim	0.78 ± 0.01	0.129 ± 0.002	0.61 ± 0.003	0.187 ± 0.001

However, we find that the normal and auto-encoder models perform substantially better than both SIRFS and Puffball on both Mehrani and ShapeNet datasets. The augmented normal model performs best on the Mehrani dataset, increasing correlation with ground truth by 8% over SIRFS and Puffball, and reducing RMS error by 52% and 29% over SIRFS and Puffball, respectively.

For the ShapeNet dataset, we find that the normal models and auto-encoder perform comparably, increasing correlation with ground truth by about 19% over SIRFS and Puffball, and reducing RMS error by 28% and 3% over SIRFS and Puffball, respectively.

At the same time, we see that much higher accuracy is achieved by the spherical completion model if based upon the ground truth 3D rim. This underlines the value in continuing research on the inference of 3D shape from the bounding contour.

The spherical completion method is appropriate for smooth objects, but not all objects are smooth. Is it possible given only the occluding contour to predict whether spherical completion should be applied? To explore this question, we analyze the accuracy of surface completion as a function of the maximum turning angle of the occluding contour (Fig. 9.11). We see that for both datasets, correlation for all methods is relatively high for small maximum turning angles but drops substantially as the maximum turning angle increases. This suggests that the turning angle statistics can be used to help guide the selection of surface completion methods.

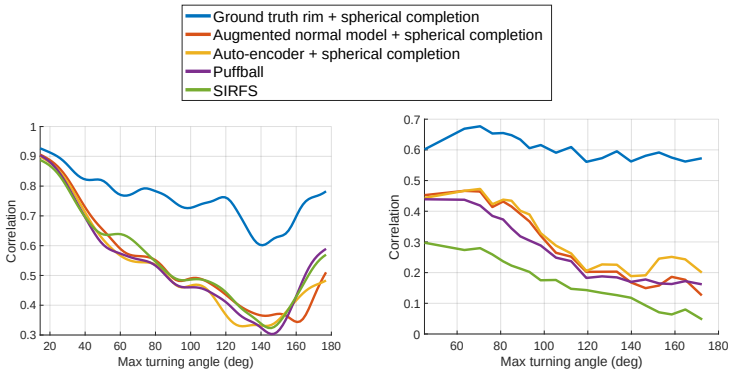


Figure 9.11: Correlation of estimated and ground-truth surface depth for (a) Mehrani and (b) ShapeNet datasets, as a function of the maximum turning angle of the occluding contour.

Fig. 9.8 (last two columns) shows surfaces reconstructed by spherical completion from the augmented normal 3D rim estimate for best, median and worst case estimates of the rim depth Z (as measured by correlation) for test shapes in the Mehrani dataset.

9.6 Conclusion

In this work, we have introduced the problem of estimating the quantitative 3D shape of the object rim from its 2D occluding contour. Our results show that the shape of the occluding contour and depth variation in the rim are statistically related, and that Gaussian

and auto-encoder models can capture this relationship. We have also introduced a novel spherical completion algorithm that allows an estimate of the visible surface to be reconstructed from the estimated 3D rim, and shown that this produces more accurate surface estimates than prior state-of-the-art approaches. Finally, we have shown that the maximum turning angle of the occluding contour can be used to predict the accuracy of the spherical completion method, which we hope can be used in future work to guide model selection for surface completion.

Chapter 10

Discussion and Future Work

In this work, we showed that shape of the occluding contour and depth variation in the rim are statistically related, and that Gaussian and auto-encoder models can capture this relationship. A spherical completion algorithm was introduced to reconstruct the visible surface from the estimated 3D rim. It was shown that this produces more accurate surface estimates than prior state-of-the-art approaches. To improve the algorithm there are a few approaches we can take. In the shapeNet dataset, the objects mostly contain planar surfaces which cannot be reconstructed by spherical completion. One possible solution is to try to detect such objects and then use minimal surface competition techniques to recover its surfaces.

Part III

Conclusion

Single-view 3D reconstruction is an ill-posed problem and thus solvable only for scenes that satisfy strong regularity conditions. The first part of the thesis introduced a method that focuses on reconstructing Manhattan structures from a single image. We showed that line information is sufficient to recover the visible surfaces of Manhattan structures.

The second part of the thesis introduced an algorithm to reconstruct approximate 3D models of a general object. The goal is to reconstruct 3D models given only the occluding contour. We showed that given the occluding contour in perspective projection, an approximation of the 3D rim can be recovered.

Bibliography

[1] W. J. Adams, J. H. Elder, E. W. Graf, J. Leyland, A. J. Lugtigheid, and A. Mury. The Southampton-York natural scenes (SYNS) dataset: Statistics of Surface Attitude. *Scientific Reports*, 6:35805, 2016.

[2] C. Akinlar and C. Topal. EDLines: A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011.

[3] I. Alhashim and P. Wonka. High quality monocular depth estimation via transfer learning. *arXiv e-prints*, abs/1812.11941, 2018.

[4] E. J. Almazàn, R. Tal, **Y. Qian**, and J. H. Elder. MCMLSD: A dynamic programming approach to line segment detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5854–5862, July 2017.

[5] E. J. Almazan, **Qian, Y.**, and J. H. Elder. Road segmentation for classification of road weather conditions. In G. Hua and H. Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 96–108, Cham, 2016. Springer International Publishing.

[6] J. M. Alvarez, T. Gevers, and A. M. Lopez. 3d scene priors for road detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 57–64. IEEE, 2010.

[7] J. M. Álvarez and A. M. Ĺopez. Road detection based on illuminant invariance. *Intelligent Transportation Systems, IEEE Transactions on*, 12(1):184–193, 2011.

[8] J. M. Álvarez, A. M. López, and R. Baldrich. Shadow resistant road segmentation from a mobile monocular system. In *Pattern Recognition and Image Analysis*, pages 9–16. Springer, 2007.

[9] M. Amthor, B. Hartmann, and J. Denzler. Road condition estimation based on spatio-temporal reflection models. In *German Conference on Pattern Recognition (GCPR), 2015 37th German Conference on*, pages 3–15. Springer, 2015.

[10] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, and A. Konushin. Fast automatic single-view 3-D reconstruction of urban scenes. In *European Conference on Computer Vision*, pages 100–113. Springer, 2008.

[11] O. Barinova, V. Lempitsky, and P. Kholi. On detection of multiple object instances using Hough transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1773–1784, 2012.

[12] J. T. Barron and J. Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687, 2015.

[13] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'77, page 659–663, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

[14] P. N. Belhumeur, D. J. Kriegman, and A. L. Yuille. The bas-relief ambiguity. *International Journal of Computer Vision*, 35(1):33–44, 1999.

[15] Binnette. Origami crane. *Wikimedia*. https://commons.wikimedia.org/wiki/File-Origami.Crane.svg.

[16] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[17] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[18] H. Blum. Biological shape and visual science (Part I). *J. Theor. Biol.*, 38:205–287, 1973.

[19] M. Boldt, R. Weiss, and E. Riseman. Token-based extraction of straight lines. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1581–1594, 1989.

[20] M. Brown, D. Windridge, and J. Guillemaut. A generalisable framework for saliency-based line segment detection. *Pattern Recognition*, 48:3993–4011, 2015.

[21] B. Çalli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *ArXiv e-prints*, Feb 2015.

[22] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[23] J. Casselgren. *Road surface classification using near infrared spectroscopy*. PhD thesis.

[24] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[25] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *European Conference on Computer Vision*, pages 833–851, 2018.

[26] W. Chen, Z. Fu, D. Yang, and J. Deng. Single-image depth perception in the wild. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 730–738. Curran Associates, Inc., 2016.

[27] G. Cheng, **Qian, Y.**, and J. H. Elder. Fusing geometry and appearance for road segmentation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 166–173, 2017.

[28] G. Cheng, Y. Wang, **Y. Qian**, and J. H. Elder. Geometry-guided adaptation for road segmentation. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pages 46–53, 2020.

[29] N. Cho, A. Yuille, and S. Lee. A novel linelet-based representation for line segment detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(5):1195–1208, May 2018.

[30] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, 2008.

[31] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[32] F. Cole, P. Isola, W. T. Freeman, F. Durand, and E. H. Adelson. Shapecollage: Occlusion-aware, example-based shape interpretation. In *European Conference on Computer Vision*, pages 665–678. Springer, 2012.

[33] J. M. Coughlan and A. L. Yuille. Manhattan world: compass direction from a single image by Bayesian inference. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 941–947 vol.2, 1999.

[34] J. M. Coughlan and A. L. Yuille. Manhattan World: Orientation and outlier detection by Bayesian inference. *Neural Computation*, 15(5):1063–1088, 2003.

[35] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski. Self-supervised monocular road detection in desert terrain. In *Robotics: science and systems*. Philadelphia, 2006.

[36] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3D reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[37] S. Dasgupta, K. Fang, K. Chen, and S. Savarese. Delay: Robust spatial layout estimation for cluttered indoor scenes. In *CVPR*, pages 616–624, June 2016.

[38] P. De Cristóforis, M. A. Nitsche, T. Krajník, and M. Mejail. Real-time monocular image-based path detection. *Journal of Real-Time Image Processing*, pages 1–14, 2013.

[39] E. Delage, H. Lee, and A. Y. Ng. Automatic single-image 3D reconstructions of indoor Manhattan world scenes. In *Robotics Research*, pages 305–321. Springer, 2007.

[40] P. Denis, J. H. Elder, and F. J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European Conference on Computer Vision*, pages 197–210. Springer, 2008.

[41] A. Desolneux, L. Moisan, and J.-M. Morel. Meaningful alignments. *International Journal of Computer Vision*, 40(1):7–23, 2000.

[42] A. Desolneux, L. Moisan, and J.-M. Morel. *From Gestalt theory to image analysis: a probabilistic approach*, volume 34. Springer Science & Business Media, 2007.

[43] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1558–1570, 2014.

[44] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[45] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[46] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, pages 2366–2374, 2014.

[47] J. H. Elder. Bridging the dimensional gap: Perceptual organization of contour into two-dimensional shape. In J. Wagemans, editor, *Oxford Handbook of Perceptual Organization*, Oxford, UK, 2014. Oxford University Press.

[48] J. H. Elder, E. J. Almazàn, **Y. Qian**, and R. Tal. MCMLSD: A probabilistic algorithm and evaluation framework for line segment detection, 2020 (In preparation).

[49] J. H. Elder and R. M. Goldberg. Ecological statistics of Gestalt laws for the perceptual organization of contours. *Journal of Vision*, 2(4):324–353, 2002.

[50] J. H. Elder and S. W. Zucker. Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):699–716, Jul 1998.

[51] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[52] P. F. Felzenszwalb and O. Veksler. Tiered scene labeling with dynamic programming. In *CVPR*, pages 3097–3104, June 2010.

[53] J. Freeman. Toronto wasted 31m on poorly managed snow clearing contracts since 2015: Auditor general, Oct 2020.

[54] J. Fritsch, T. Kuehnl, and A. Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[55] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[56] Y. Furukawa and Y. Shinagawa. Accurate and robust line segment extraction by analyzing distribution around peaks in Hough space. *Computer Vision and Image Understanding*, 92:1–25, 2003.

[57] R. Garg, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[58] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 2013.

[59] B. Gillam and T. Flagg. Evidence for disparity change as the primary stimulus for stereoscopic processing. *Perception & Psychophysics*, 36(6):559–564, 1984.

[60] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[61] N. Guil, J. Villalba, and E. L. Zapata. A fast Hough transform for segment detection. *IEEE Transactions on Image Processing*, 4(11):1541–1548, 1995.

[62] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision*, pages 482–496. Springer, 2010.

[63] D. Guru, B. Shekar, and P. Nagabhushan. A simple and robust line detection algorithm based on small eigenvalue analysis. *Pattern Recognition Letters*, 25(1):1–13, 2004.

[64] A. Guzman. *Computer recognition of three-dimensional objects in a visual scene*. PhD thesis, MIT, 1968.

[65] A. Guzman. Decomposition of a visual scene into three-dimensional bodies. In *Proc. of the Fall Joint Computer Conference*, pages 291–304, 1968.

[66] O. Haines and A. Calway. Recognising planes in a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1849–1861, 2015.

[67] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[68] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[69] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[70] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1849–1856, Sept 2009.

[71] M. Hofer, M. Maurer, and H. Bischof. Efficient 3D scene abstraction using line segments. *Computer Vision and Image Understanding*, 157:167–178, 2017.

[72] D. Hoiem, A. A. Efros, and M. Hebert. Automatic Photo Pop-up. *ACM Transactions On Graphics (TOG)*, 24(3):577–584, 2005.

[73] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007.

[74] P. V. Hough. Method and means for recognizing complex patterns, Dec. 18 1962. US Patent 3,069,654.

[75] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma. Learning to parse wireframes in images of man-made environments. In *CVPR*, June 2018.

[76] D. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence 6*, pages 295–324, 1971.

[77] T. Igarashi, T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. In *SIGGRAPH*, pages 409–416, 1999.

[78] H. Izadinia, Q. Shan, and S. M. Seitz. IM2CAD. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2422–2431. IEEE, 2017.

[79] M. Jokela, M. Kutila, and L. Le. Road condition monitoring system based on a stereo camera. In *Intelligent Computer Communication and Processing, 2009. ICCP 2009. IEEE 5th International Conference on*, pages 423–428. IEEE, 2009.

[80] P. Jonsson. Remote sensor for winter road surface status detection. In *Sensors, 2011 IEEE*, pages 1285–1288. IEEE, 2011.

[81] P. Jonsson, J. Casselgren, and B. Thornberg. Road surface status classification using spectral analysis of nir camera images. *Sensors Journal, IEEE*, 15(3):1641–1656, 2015.

[82] V. Kamat-Sadekar and S. Ganesan. Complete description of multiple line segments using the Hough transform. *Image and Vision Computing*, 16(9):597–613, 1998.

[83] T. Kanade. A theory of Origami world. *Artificial Intelligence*, 13(3):279–311, 1980.

[84] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1966–1974, 2015.

[85] K. Karsch, Z. Liao, J. Rock, J. T. Barron, and D. Hoiem. Boundary Cues for 3D Object Shape Recovery. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2163–2170, June 2013.

[86] S. Kawai, K. Takeuchi, K. Shibata, and Y. Horita. A method to distinguish road surface conditions for car-mounted camera images at night-time. In *ITS Telecommunications (ITST), 2012 12th International Conference on*, pages 668–672. IEEE, 2012.

[87] J. Kim and S. Lee. Extracting major lines by recruiting zero-threshold Canny edge links along Sobel highlights. *IEEE Signal Processing Letters*, 22(10):1689–1692, 2015.

[88] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13(3):321–330, 1984.

[89] J. J. Koenderink and A. J. Van Doorn. The generic bilinear calibration-estimation problem. *International Journal of Computer Vision*, 23(3):217–234, Jun 1997.

[90] H. Kong, J.-Y. Audibert, and J. Ponce. Vanishing point detection for road detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 96–103. IEEE, 2009.

[91] H. Kong, J.-Y. Audibert, and J. Ponce. General road detection from a single image. *Image Processing, IEEE Transactions on*, 19(8):2211–2220, 2010.

[92] J. Košecká and W. Zhang. Video compass. In *European Conference on Computer Vision*, pages 476–490. Springer, 2002.

[93] T. Kovar. Popup book starter kit. 2017. https://videohive.net/item/popup-book-starter-kit/6808435.

[94] M. Kubovy, A. O. Holcombe, and J. Wagemans. On the lawfulness of grouping by proximity. *Cognitive Psychology*, 35:71–98, 1998.

[95] M. Kubovy and J. Wagemans. Grouping by proximity and multistability in dot lattices: A quantitative Gestalt theory. *Psychological Science*, 6(4):225–234, July 1995.

[96] H. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[97] A. Kushal and S. M. Seitz. Single view reconstruction of piecewise swept surfaces. In *3DV*, pages 239–246, June 2013.

[98] M. Kytö, M. Nuutinen, and P. Oittinen. Method for measuring stereo camera depth accuracy based on stereoscopic vision. In *Three-Dimensional Imaging, Interaction, and Measurement*, volume 7864, page 78640I. International Society for Optics and Photonics, 2011.

[99] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248, Oct 2016.

[100] D. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, pages 2136–2143. IEEE, 2009.

[101] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1288–1296. Curran Associates, Inc., 2010.

[102] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1119–1127, June 2015.

[103] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, June 2018.

[104] S.-H. Lim, S.-K. Ryu, and Y.-H. Yoon. Image recognition of road surface conditions using polarization and wavelet transform. *Journal of The Korean Society of Civil Engineers*, 27(4D):471–477, 2007.

[105] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa. PlaneNet: Piece-wise planar reconstruction from a single RGB image. In *CVPR*, pages 2579–2588, 2018.

[106] D. Liu, Y. Wang, Z. Tang, and X. Lu. A robust and fast line segment detector based on top-down smaller eigenvalue analysis. In *Fifth International Conference on Graphics and Image Processing*, pages 906916–906916. International Society for Optics and Photonics, 2014.

[107] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170, 2015.

[108] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE TPAMI*, 38(10):2024–2039, Oct 2016.

[109] X. Liu, Z. Cao, N. Gu, S. Nahavandi, C. Zhou, and M. Tan. Intelligent line segment perception with cortex-like mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(12):1522–1534, 2015.

[110] X. Lu, J. Yao, K. Li, and L. Li. Cannylines: A parameter-free line segment detector. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 507–511. IEEE, 2015.

[111] A. Mallya and S. Lazebnik. Learning informative edge maps for indoor scene layout prediction. In *ICCV*, pages 936–944, Dec 2015.

[112] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(5):530–549, May 2004.

[113] J. Matas, C. Galambos, and J. Kittler. Robust detection of lines using the progressive probabilistic Hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

[114] P. Mehrani and J. H. Elder. Estimating coarse 3D shape and pose from the bounding contour. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP, (VISIGRAPP 2017)*, pages 603–610. INSTICC, SciTePress, 2017.

[115] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[116] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[117] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *European Conference on Computer Vision*, pages 57–70. Springer, 2010.

[118] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499, 2016.

[119] M. Nieto, C. Cuevas, L. Salgado, and N. García. Line segment detection using weighted mean shift procedures on a 2D slice sampling strategy. *Pattern Analysis and Applications*, 14(2):149–163, 2011.

[120] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[121] R. Omer and L. Fu. An automatic image recognition system for winter road surface condition classification. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1375–1379. IEEE, 2010.

[122] L. E. Ortiz, E. V. Cabrera, and L. M. Gonçalves. Depth data error modeling of the ZED 3D vision sensor from stereolabs. *ELCVIA: Electronic Letters on Computer Vision and Image Analysis*, 17(1):0001–15, 2018.

[123] M. R. Oswald, E. Töppe, K. Kolev, and D. Cremers. Non-parametric Single View Reconstruction of Curved Objects Using Convex Optimization. In *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 171–180. Springer Berlin Heidelberg, 2009.

[124] J. Pan, M. Hebert, and T. Kanade. Inferring 3D layout of building facades from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2918–2926, 2015.

[125] P. Parodi and G. Piccioli. 3D shape reconstruction by using vanishing points. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(2):211–217, 1996.

[126] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *CVPR*, pages 2719–2726, June 2012.

[127] Z. Pizlo, T. Sawada, Y. Li, W. G. Kropatsch, and R. M. Steinman. New approach to the perception of 3D shape based on veridicality, complexity, symmetry and volume. *Vision Research*, 50(1):1–11, 2010.

[128] M. Prasad, A. Zisserman, and A. W. Fitzgibbon. Single view reconstruction of curved surfaces. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1345–1354, 2006.

[129] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *CVPR*, June 2018.

[130] S. Ramalingam and M. Brand. Lifting 3D manhattan lines from a single image. In *2013 IEEE International Conference on Computer Vision*, pages 497–504, Dec 2013.

[131] S. Ramalingam, J. K. Pillai, A. Jain, and Y. Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3065–3072, June 2013.

[132] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, 2001.

[133] C. Rasmussen. Grouping dominant orientations for ill-structured road following. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–470. IEEE, 2004.

[134] C. Rasmussen. Texture-based vanishing point voting for road shape estimation. In *BMVC*, pages 1–10. Citeseer, 2004.

[135] Y. Ren, S. Li, C. Chen, and C.-C. J. Kuo. A coarse-to-fine indoor layout estimation (cfile) method. In *Asian Conference on Computer Vision*, pages 36–51. Springer, 2016.

[136] L. G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[137] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[138] R. Sawada, Y. Li, and Z. Pizlo. Detecting 3D mirror symmetry in a 2D camera image for 3D shape recovery. *Proceedings of the IEEE*, 102:1588–1606, 2014.

[139] T. Sawada. Visual detection of symmetry of 3D shapes. *Journal of Vision*, 10(6):4, 2010.

[140] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems*, pages 1161–1168, 2005.

[141] A. Saxena, S. H. Chung, and A. Y. Ng. 3-D depth reconstruction from a single still image. *International Journal of Computer Vision*, 76(1):53–69, 2008.

[142] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Depth perception from a single still image. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1571–1576. AAAI Press, 2008.

[143] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3D scene structure from a single still image. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):824–840, 2009.

[144] G. Schindler and F. Dellaert. Atlanta World: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–203–I–209 Vol.1, June 2004.

[145] C. Schmid and A. Zisserman. Automatic line matching across views. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 666–671, 1997.

[146] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[147] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2815–2822. IEEE, 2012.

[148] A. G. Schwing and R. Urtasun. Efficient exact inference for 3D indoor scene understanding. In *ECCV 2012*, pages 299–313, 2012.

[149] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. *Computer Vision–ECCV 2012*, pages 746–760, 2012.

[150] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3D database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516, May 2014.

[151] S. Song, S. P. Lichtenberg, and J. Xiao. Sun RGB-D: A RGB-D scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, June 2015.

[152] K. Sugihara. *Machine interpretation of line drawings*, volume 1. MIT press Cambridge, 1986.

[153] T. Suttorp and T. Bucher. Robust vanishing point estimation for driver assistance. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 1550–1555. IEEE, 2006.

[154] R. Tal and J. H. Elder. An accurate method for line detection and Manhattan frame estimation. In *Asian Conference on Computer Vision*, pages 580–593. Springer, 2012.

[155] C. Tan, T. Hong, T. Chang, and M. Shneier. Color model-based real-time learning for road following. In *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, pages 939–944. IEEE, 2006.

[156] J.-P. Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1250–1257. IEEE, 2009.

[157] **Qian, Y.**, E. J. Almazan, and J. H. Elder. Evaluating features and classifiers for road weather condition analysis. In *International Conference on Image Processing (ICIP), 2016 IEEE*. IEEE, 2016.

[158] **Y. Qian**, S. Ramalingham, and J. Elder. LS3D: Single-view Gestalt 3D surface reconstruction from manhattan line segments. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 399–416, 2018.

[159] J. Todd. The visual perception of 3D shape. *Trends in Cognitive Sciences*, 8(3):115–121, 2004.

[160] J. T. Todd and F. D. Reichel. Ordinal structure in the visual perception and cognition of smoothly curved surfaces. *Psychological Review*, 96(4):643–657, 1989.

[161] E. Toppe, M. R. Oswald, D. Cremers, and C. Rother. Image-based 3D Modeling via Cheeger Sets. In *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I*, ACCV'10, pages 53–64, Berlin, Heidelberg, 2011. Springer-Verlag.

[162] P. Tse. A contour propagation approach to surface filling-in and volume formation. *Psychological Review*, 109(1):91–115, 2002.

[163] N. R. Twarog, M. F. Tappen, and E. H. Adelson. Playing with puffball: Simple scale-invariant inflation for use in vision and graphics. In *Proceedings of the ACM Symposium on Applied Perception*, pages 47–54. ACM, 2012.

[164] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity invariant CNNs. In *International Conference on 3D Vision (3DV)*, 2017.

[165] M. Varma and A. Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, volume 3, pages 255–271. Springer-Verlag, May 2002.

[166] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010.

[167] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt. A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure–ground organization. *Psychological Bulletin*, 138(6):1172, 2012.

[168] D. L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. 1972.

[169] H. Wang, S. Gould, and D. Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. In *Proceedings of the 11th European Conference on Computer Vision: Part II*, pages 435–449. Springer-Verlag, 2010.

[170] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018.

[171] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille. Towards unified depth and semantic prediction from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 2800–2809. IEEE, 2015.

[172] Y. Wang, E. K. Teoh, and D. Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004.

[173] S. Waterer. Exercise: Linear perspective. 2015. https://learningmojo.wordpress.com/2015/01/08/exercise-linear-perspective.

[174] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum. Marrnet: 3D shape reconstruction via 2.5D sketches. In *Advances in Neural Information Processing Systems*, pages 540–550, 2017.

[175] J. Wu, C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum. Learning shape priors for single-view 3d completion and reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 646–662, 2018.

[176] S. Xie and Z. Tu. Holistically-Nested edge detection. In *Proceedings of the IEEE international Conference on Computer Vision*, pages 1395–1403, 2015.

[177] D. Xu, W. Ouyang, X. Wang, and N. Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *CVPR*, June 2018.

[178] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. Multi-scale continuous CRFs as sequential deep networks for monocular depth estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[179] Z. Xu, B. Shin, and R. Klette. A statistical method for line segment detection. *Computer Vision and Image Understanding*, 138:61–73, 2015.

[180] Z. Xu, B.-S. Shin, and R. Klette. Accurate and robust line segment extraction using minimum entropy with Hough transform. *IEEE Transactions on Image Processing*, 24(3):813–822, 2015.

[181] Z. Xu, B.-S. Shin, and R. Klette. Closed form line-segment extraction using the Hough transform. *Pattern Recognition*, 48(12):4012–4023, 2015.

[182] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang. Learning attraction field representation for robust line segment detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[183] H. Yang and H. Zhang. Efficient 3d room shape recovery from a single panorama. In *CVPR*, pages 5422–5430, June 2016.

[184] H.-J. Yang, H. Jang, J.-W. Kang, and D.-S. Jeong. Classification algorithm for road surface condition. *IJCSNS*, 14(1):1, 2014.

[185] L. Zhang and R. Koch. Structure and motion from line correspondences: representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation*, 25(5):904–915, 2014.

[186] X. Zhang, Z. Zhang, C. Zhang, J. Tenenbaum, B. Freeman, and J. Wu. Learning to reconstruct shapes from unseen classes. In *Advances in Neural Information Processing Systems*, pages 2263–2274, 2018.

[187] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[188] W. Zhuo, M. Salzmann, X. He, and M. Liu. 3d box proposals from a single monocular image of an indoor scene. In *AAAI*, 2018.