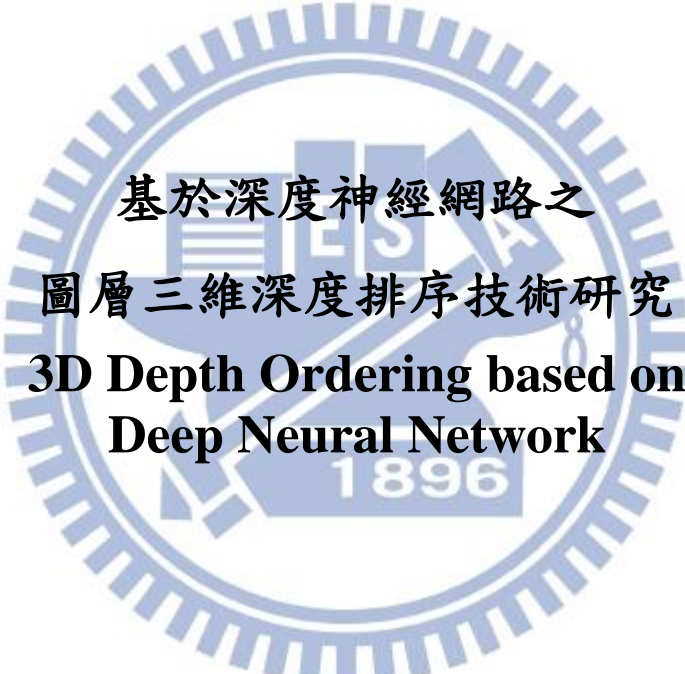


國立交通大學

電子工程學系 電子研究所

碩 士 論 文

The logo of National Central University (NCU) is a circular emblem with a gear-like outer border. Inside the circle, there is a stylized building or monument, and the year '1896' is inscribed at the bottom. The text of the thesis is overlaid on this watermark.

基於深度神經網路之
圖層三維深度排序技術研究
3D Depth Ordering based on
Deep Neural Network

研 究 生：廖姿婷

指 導 教 授：王聖智 教授

簡鳳村 教授

中 華 民 國 一 〇 二 年 八 月

基於深度神經網路之
圖層三維深度排序技術研究
**3D Depth Ordering based on
Deep Neural Network**

研 究 生：廖姿婷

Student : Tzu-Ting Liao

指 導 教 授：王聖智 教授

Advisor : Prof. Sheng-Jyh Wang

簡鳳村 教授

Prof. Feng-Tsun Chien



A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of Master of Science
in
Electronics Engineering

August 2013

Hsinchu, Taiwan, Republic of China

中 華 民 國 一 〇 二 年 八 月

基於深度神經網路之圖層三維深度排序技術研究

研究生：廖姿婷

指導教授：王聖智 教授

簡鳳村 教授

國立交通大學

電子工程學系 電子研究所碩士班

摘要

在這篇論文裡，我們提出一套方法用來找出單張影像的三維深度。不同於利用場景幾何資訊來推論深度的做法，我們不對場景有任何的幾何假設，且只擷取影像裡小區域的深度特徵來幫助我們找到深度。在我們的做法中，首先將影像切割成數個區塊，我們主要專注在小區域裡的深度排序。利用深度神經網路去分類小區域裡深度排序的類別，且自動學習出適合的特徵，然而利用傳統的方式學習深度神經網路會面臨梯度消失的問題。因此，我們的學習過程有兩部分：無監督式預先學習以及有監督式微調模型。學習完深度神經網路後，對於一張影像我們把影像切割成小區域進行測試。然而小區域對於深度排序的結果會不一致。因此，我們結合小區域的相對深度資訊並且把這些資訊轉換成有向圖。再藉由找出最小反饋邊集合以得到整體一致的深度排序結果。

3D Depth Ordering based on Deep Neural Network

Student : Tzu-Ting Liao Advisor : Prof. Sheng-Jyh Wang

Prof. Feng-Tsun Chien

Department of Electronics Engineering, Institute of Electronics
National Chiao Tung University

Abstract

In this thesis, we propose a method to estimate 3-D depth map from a single image. Unlike approaches employing a geometric model behind the scene to infer the depth map, we propose to estimate the 3-D scenes by extracting local depth cues without any structure assumptions in the scene instead. In our approach, first we partition the image into several regions. We focus on inferring the depth ordering in a local patch. Then we apply the model of deep neural network to figure out which depth order class the local patch belong and to automatically learn the appropriate features in our problem. However, training the deep neural network by classical method will encounter the vanishing gradient problem. To tackle the problem, our training algorithm consists two phase: the unsupervised pre-training phase and the supervised fine-tuning phase. After training the deep neural network, we test the image by feeding the local patch into the deep neural network. In practice, the resultant depth orders that are sorted by the deep neural network and from different local patches may be contradictory. Hence, we combine these local depth order reasoning to construct a direct graph. By finding the minimum feedback arc set, we can obtain a depth order with global consistency.

誌謝

在此特別感謝指導教授 王聖智老師以及 簡鳳村老師的指導，讓我在研究所這段時間學習到許多事。在研究上，老師給我許多空間並在遇到困難時引導我，鼓勵我們獨立思考以及自我解決問題；在研究外，老師也分享許多自身的經驗並告訴我們為人處事之道，是研究外的另一大收穫。

感謝這兩年來實驗室所有成員，謝謝博士班碩士班學長姐：慈澄、禎宇、家豪、心憫、耀笙、柏翔、彥廷、秉修、儲培、利容，不管在研究上或是生活上給我們許多建議與幫助；同屆的同學：秉宸、政銘、介暉、佳峻，一起討論課業或是研究內容；以及學弟妹：冠廷、非凡、汝欣、振源、子銘、奕中、浩瑋，幫忙我標大量訓練影像。感謝溫馨的 Vision Lab 在這兩年帶給我許多歡樂以及陪伴，帶給我許多美好的回憶，希望未來大家能健康順利。

感謝在我整個求學過程的老師以及同學，讓我可以有所成長完成學位；感謝室友芊縈，一起走過大學碩士這六年，經歷許多點點滴滴，帶給我許多歡笑的回憶；感謝炫谷的包容與扶持，不管我遇到任何，困難或是決定，都可以支持我並鼓勵我；感謝我的家人，給我許多資源讓我完成學位，且陪伴我並教育我，讓我一步步走下去。

Content

Chapter 1. Introduction	1
Chapter 2. Backgrounds	3
2.1 Depth Estimation using Geometric Models	3
2.2 Depth Estimation using Depth Perception Models	8
Chapter 3. 3D Depth Order Reasoning	11
3.1 Data Set	11
3.2 Image Segmentation	12
3.2.1 Contour Detection and Image Segmentation	12
3.3 Local Depth Ordering	16
3.3.1 Deep Neural Network	17
3.3.2 Training	19
3.3.3 Testing	30
3.4 Global Depth Ordering	31
3.4.1 Combine the Local Depth Ordering	32
3.4.2 Direct Acyclic Graph	32
Chapter 4. Experimental Results	36
4.1 Deep Neural Networks	36
4.2 Local Depth Ordering	37
4.3 Global Depth Ordering	38
4.3.1 Based on Ground Truth Segmentation	38
4.3.2 Based on Contour Detection and Image Segmentation	42
Chapter 5. Conclusion	45

List of Tables

Table 4-1 Accuracy rate. 37

List of Figures

Figure 2-1 Examples of object types : SKY, GROUND, PLANE, and CUBIC. 3

Figure 2-2 Depth assignment for type PLANE and CUBIC. 4

Figure 2-3 Experimental result of Jung et al. in [5]. (a) Input image. (b) Depth map. 4

Figure 2-4 Geometric labels of Hoiem’s system [6]. 5

Figure 2-5 The occlusion cues used in Hoiem’s boundary system. 6

Figure 2-6 lustration of Hoiem’s algorithm [6]. 6

Figure 2-7 Illustration of five valid junctions [6]. 7

Figure 2-8 Result of Hoiem’s occlusion boundary algorithm. (a) Occlusion boundary result. (b) (Top row) Estimated max depth. (Down row) Estimated min depth. 7

Figure 2-9 The convolutional filters used by Saxena [4]. 8

Figure 2-10 Multiple scale structure of features in [1]. (a) An illustration of neighboring location with multi scale features. (b) Actual neighborhood of the superpixel S3C. (c) Collected features for superpixel S3C. 8

Figure 2-11 (a) Original images. (b) Ground truth depth map. (c) Depth map by [4]. 9

Figure 2-12 MRF model by [10] (a) Global depth reasoning. (b) Each junction produces three directed edges in the depth order graph. (c) MRF to encourage the global consistency. (d) The edge potential in our MRF gives high penalties (solid) if the segments’ orders contradict between two nodes. 10

Figure 2-13 (a) Original images. (b) Ground truth depth ordering. (c) Depth ordering by [8]. 10

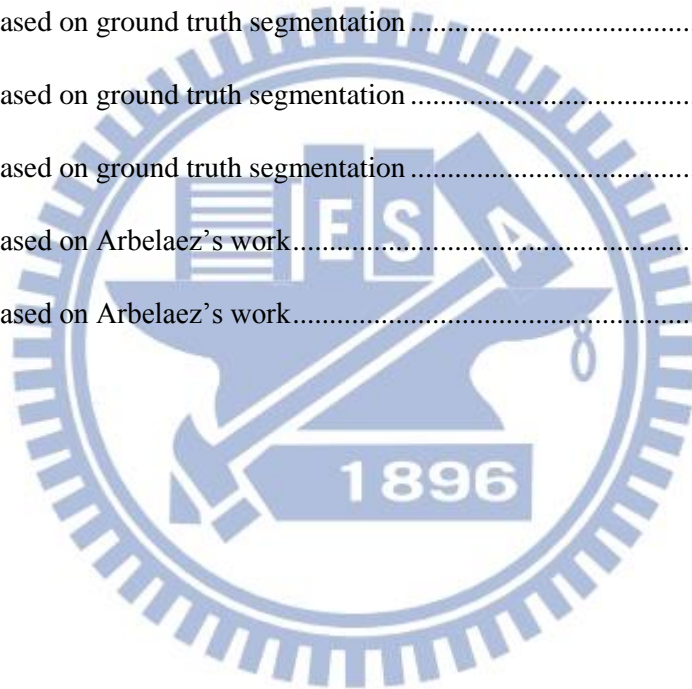
Figure 3-1 Flowchart of our algorithm. 11

Figure 3-2 (a) Image, (b) Segmentation results, (c) Depth Order. 12

Figure 3-3 Filters for creating textons in [14]. 13

Figure 3-4 Multiscale P_b by [14]. Left Column, Top to Bottom: The brightness, color a, color b and texture channels. Rows: The Oriented gradient of histograms for $\theta=0$ and $\theta=\pi/2$, and the maximum response over eight orientations . Beside the original image, the combination of oriented gradients. The lower right panel, the final output of the multiscale contour detector.	14
Figure 3-5 (a) Image. (b)A sparse affinity matrix connecting pixels within a fixed radius. Pixels i and j have a low affinity, whereas i and k have high affinity. (c) The first four generalized eigenvectors resulting. (d) Partitioning the image by clustering on the eigenvectors , (e) Spectral P_b	15
Figure 3-6 Hierarchical segmentation from contours by [14] . (a) Image, (b) Maximal response of contour detector gP_b , (c) Oriented Watershed Transform-Ultrametric Contour Map (OWT-UCM), (d) The initial oversegmentation corresponding to the finest level of the UCM, (e)(f) Contours and corresponding segmentation obtained by thresholding the UCM at level 0.5.	16
Figure 3-7 (a) Image, (b)(c)(d) Segments of the image, (e)(f)(g) RGB color of the image.	17
Figure 3-8 2 possible depth ordering of 2-part model.	18
Figure 3-9 6 possible depth ordering of 3-part model.	18
Figure 3-10 (a) Deep Neural Network for 2-part model, (b) Deep Neural Network for 3-part model.	19
Figure 3-11 Restricted Boltzmann Machine (RBM).....	20
Figure 3-12 Markov chain that uses alternating Gibbs sampling.....	22
Figure 3-13 Contrastive Divergence (CD).....	23
Figure 3-14 Greedy Layer-Wise Pre-training	25
Figure 3-15 The Combination of RBM and Gaussian RBM (3-part Model).....	25
Figure 3-16 Greedy Layer-Wise Pre-training of our model (3-part Model).	26
Figure 3-17 The meaning of features of hidden units \mathbf{h}_1 and \mathbf{h}_2 (3-part Model).....	26
Figure 3-18 Feature of hidden units \mathbf{h}_1 (bottom), \mathbf{h}_2 (middle) and \mathbf{h}_3 (top) in 2-part model.	27
Figure 3-19 Feature of hidden units \mathbf{h}_1 (bottom), \mathbf{h}_2 (middle) and \mathbf{h}_3 (top) in 3-part model.	28
Figure 3-20 Feature of hidden units \mathbf{h}_1 (bottom), \mathbf{h}_2 (middle) and \mathbf{h}_3 (top) in 2-part model after fine-tuning.	29
Figure 3-21 Feature of hidden units \mathbf{h}_1 (bottom), \mathbf{h}_2 (middle) and \mathbf{h}_3 (top) in 3-part model after fine-tuning.	30
Figure 3-22 Nearest posterior (a) 2-part patch, (b) 3-part patch.	31

Figure 3-23 Nodes of direct graph.	33
Figure 3-24 Example of computing the depth relation between node E and F.	34
Figure 3-25 Direct graph.	34
Figure 3-26 (a) Direct graph, (b)Direct acyclic graph with feedback minimum arc set.	35
Figure 3-27 Depth ordering.	35
Figure 4-1 Testing	36
Figure 4-2 Local Depth Ordering, (a) Image, (b) 2-part patch, (c) 3-part patch	37
Figure 4-3 Histogram of accuracy rate.	38
Figure 4-4 Results based on ground truth segmentation	39
Figure 4-5 Results based on ground truth segmentation	40
Figure 4-6 Results based on ground truth segmentation	41
Figure 4-7 Results based on Arbelaez's work.	42
Figure 4-8 Results based on Arbelaez's work.	43



Chapter 1. Introduction

3-D depth estimation is a fundamental issue of computer vision. 3-D depth information is required in many applications, such as segmentation, object recognition, and scene understanding. Up to now, many techniques have been explored [1, 2]. Most of the prior works focus on stereo, structure from motion and multi-view images that are taken at the same scene, but from different view points. Then 3-D depth can be estimated based on binocular or motion cues. However, in some applications, it is usually that we only have a single image and want to recover the 3-D depth [3-9].

It is well known that the image is a projection of the real world. Humans can immediately grasp the 3-D structure of a single scene but it is still a challenging task for computers. We want to make the computer understand the 3-D depth structure from a single image, just like what humans can do. When the 3-D real world is projected onto a 2-D image plane, it often occurs occlusion between two objects. Objects in the image are mostly mutual occlusion. Hence, for a computer to comprehend the 3-D knowledge, it needs to understand the occlusion relationship in an image.

In recent work, the study of occlusion reasoning focuses on how to recover the hidden information behind the 2-D image plane. Some researchers [3, 4, 7] investigated monocular cues, such as color and texture, surface layout, boundary, contour, junction, etc., to extract the 3D information in very specific scenes. Another approach [5, 6, 9] infers the 3-D depth based on the semantic labels of the scene, such as “ground” always supports “vertical surface,” and these are placed before “sky.” Nevertheless, these semantic labels are not always used in most images.

In this thesis, the major goal of our approach is to infer the depth order directly based on monocular cues instead of relying on any contextual information or prior knowledge of the scene structure. Our monocular cues are similar to T-junctions or boundaries in an image [3,

7]. However, in previous work, Dimiccoli[3] and Palou[7] used hand-crafted features to describe T-junctions or boundaries. And the proposed rules in [3, 7] of inferences didn't employ any learning process.

The proposed algorithm first uses a segmentation algorithm to partition the image into several regions. The assumption is that these regions are with topological depth ordering. Hence, cyclic occlusion can't be dealt with the proposed method. Second, the local depth order is found by discovering the depth information from the local patches which have been partitioned into more than one part by segmentation. Therefore, the deep neural network is used to explore the depth features and infer the depth relation in the local patches. Deep neural network consists of feature detector units in each layer. The units of lower layers detect simple features. Then, the output of lower-layer feature detector will be fed into a higher layer which detects more complex features. These features are automatically learned by training data instead of by hand-crafting. In the past years, the classical methods are efficient when applied to shallow neural networks but not efficient when adapted to deep neural networks [10]. Until recent year, the Restricted Boltzmann Machine (RBM) proposed by Hinton *et al.* [11, 12] has been proved useful to perform pre-training in the neural network in the training processes. The training strategy consists two phases: the unsupervised pre-training phase and the supervised fine-tuning phase. After training the deep neural network, we can infer the depth order of local patch. However, the depth reasoning of local patches may be inconsistent. We can combine the local depth reasoning to construct a direct graph. By removing the minimum feedback arc set, we can find a globally consistent result across the segments.

This thesis is organized as follows. We will first discuss some related works in Chapter 2. In Chapter 3, we describe the proposed method that infers the 3-D depth order for a single image. Some experimental results are shown in Chapter 4 and we provide our conclusion in Chapter 5.

Chapter 2. Backgrounds

In the past few years, many depth estimation algorithms have been proposed [4, 6]. In this chapter, we will introduce some related works about depth estimation of a single image in vision-based systems. These algorithms can be roughly classified into two groups depending on the type of the information and assumption which have been used: the geometric models and the depth perception models. The former estimates the 3-D depth of a single image by considering the 3D scene geometry [5, 6, 9]. The latter strategy infers the 3D depth from the monocular features in the image without any assumptions about the scene structure [3, 4, 7, 8]. In Section 2.1 and Section 2.2, some relevant works about geometric models and depth perception models will be introduced, respectively.

2.1 Depth Estimation using Geometric Models

In [5], Jung *et al.* used object classification based on Bayesian learning algorithm to extract the depth value. In their approach, they estimate the depth map based on the linear perspective depth cue with the condition that the type of input images need to contain the vanishing point and be outdoor scene. Before object classification, the vanishing point is detected and the image is divided into several segments. Objects in a single-view image are classified into four types: sky, ground, cubic, and plane. (See Figure 2-1)



Figure 2-1 Examples of object types : SKY, GROUND, PLANE, and CUBIC.

According to the inferred type, relative depth values are assigned to each type to generate a 3D model. Ground can be regarded as a horizontal plane. The depth value of GROUND increases as it is getting closer to the vanishing point. Figure 2-2 illustrates the depth assignment for the cubic type and plane type. For plane-type objects, such as cars, they have a constant depth depending on where the bottom position of the object is. For cubic objects, such as buildings, the depth value varies with the distance from the vanishing point. Their results are shown in Figure 2-3.

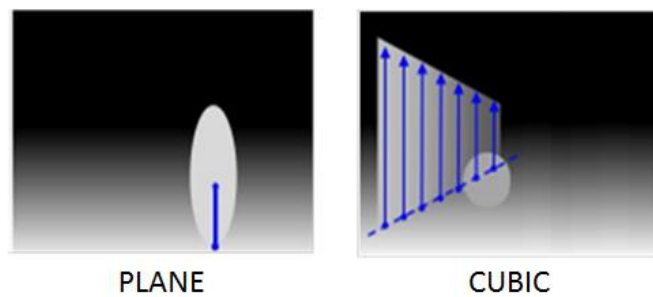


Figure 2-2 Depth assignment for type PLANE and CUBIC.

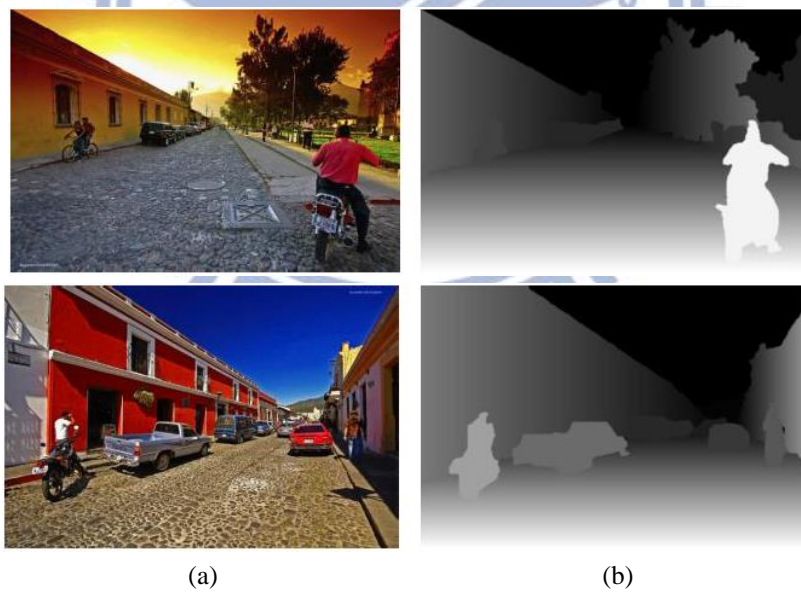


Figure 2-3 Experimental result of Jung et al. in [5]. (a) Input image. (b) Depth map.

In [6], Hoiem *et al.* described an algorithm that recovers the occlusion boundary from an image. They argued that people can perceive the depth of a scene if they get the whole structure of the scene. By recovering the occlusion relationship between objects, relative depth ordering could be determined. Specifically, their work can be divided into two parts. To

understand the geometry of an image, they labeled the image into geometric classes to form the surface layout of a scene [13]. With the geometric labels, they used the classification results to learn the occlusion boundaries in an image. Using the vertical/ground structure, they estimated objects depth by detecting the attachment of ground and vertical objects. For some regions which are occluded by other regions, the occlusion relationship is used to estimate the max/min depth of these regions.

In their approach, they used the results of surface layout estimation [13]. Each pixel belongs to ground plane, vertical surface or sky. Figure 2-4 shows a classification result. Different colors mean different main classes (ground, vertical, sky). The marks represent different subclass labels of vertical regions. These surface layout cues are part of the occlusion cues and are helpful to get the information boundaries. They also used other cues to recognize the boundaries, such as boundary cues, region cues, and depth-based cues. The detail cues are listed in Figure 2-5. Surface layout cues use the result of the surface layout algorithm and are very useful for detecting the occlusion boundaries since most edges between different surface labels are occlusion boundaries. Geometric labels of surface layout can also reveal figure/ground information.

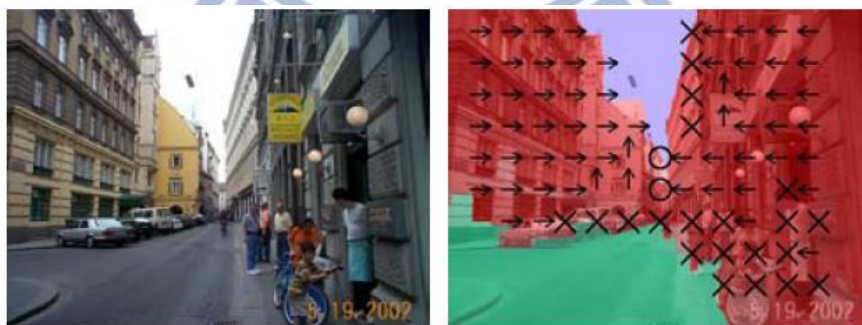


Figure 2-4 Geometric labels of Hoiem's system [6].

Occlusion Cue Descriptions	Num
Boundary	7
B1. Strength: average Pb value	1
B2. Length: length / (perimeter of smaller side)	1
B3. Smoothness: length / (L1 endpoint distance)	1
B4. Orientation: directed orientation	1
B5. Continuity: minimum diff angle at each junction	2
B6. Long-Range: number of chained boundaries	1
Region	18
R1. Color: distance in $L^*a^*b^*$ space	1
R2. Color: difference of $L^*a^*b^*$ histogram entropy	1
R3. Area: area of region on each side	2
R4. Position: differences of bounding box coordinates	10
R5. Alignment: extent overlap (x,y,overall,at boundary)	4
3D Cues	34
S1. GeomContext: average confidence, each side	10
S2. GeomContext: signed difference of S1 between sides	5
S3. GeomContext: sum absolute S2	1
S4. GeomContext: most likely main class, both sides	1
S5. GeomTJuncts: two kinds for each junction	4
S6. GeomTJuncts: if both junctions are GeomTJuncts	2
S7. Depth: three estimates (log scale), each side	6
S8. Depth: diffs of S7, abs diff of first estimate	4
S9. Depth: diff of min overestimate, max underestimate	1

Figure 2-5 The occlusion cues used in Hoiem's boundary system.

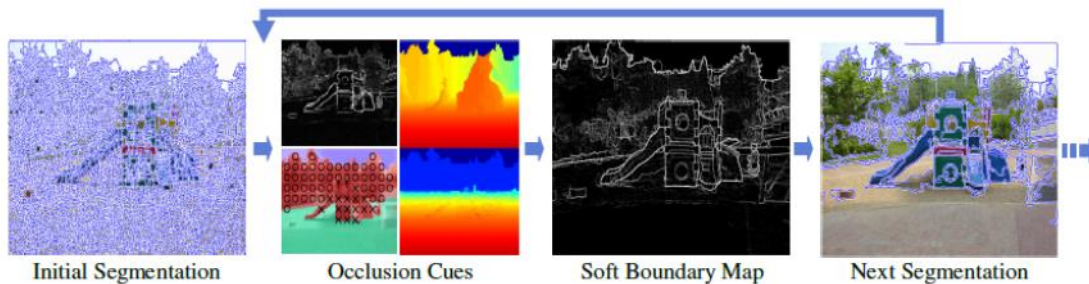


Figure 2-6 Illustration of Hoiem's algorithm [6].

To identify occlusion boundary, they learned a classifier which classifies boundaries to three different types: non-occlusion, occluded and occlusion. Their proposed algorithm starts with an over-segmentation algorithm, which assumes most boundaries are preserved in the edges between these segmented regions. Usually there are thousands of regions at the beginning and then the algorithm gradually removes these unlikely edges to get the final boundaries. They used the cues to predict the likelihood of being a boundary for each edge. At each time, the boundary likelihood of each edge is re-calculated and the most unlikely edges are removed until the boundary likelihood of all the remaining edges are above a given threshold. As regions grow larger and edges become larger, they refine the boundary

prediction and remove unlikely boundaries again. This process continues iteratively until no new region forms. The flow chart of their system is shown in Figure 2-6.

In order to estimate the likelihood of boundary labels, they used CRF (Conditional Random Field) model to enforce boundary continuity and consistency. More precisely, the boundary likelihoods of connected edges are related. Hoiem *et al.* considered all possible labels of the image, instead of estimating each boundary confidence alone. For example, in a junction where three edges are connected, there are 27 combinations of junctions but only 5 of them are possible. The valid types are shown in Figure 2-7. Figure 2-8 (b) shows one of their results, the region to the left of an arrow is in front of the region to the right of the arrow.

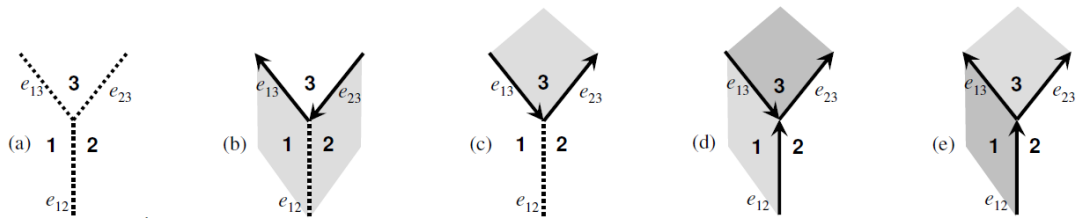


Figure 2-7 Illustration of five valid junctions [6].

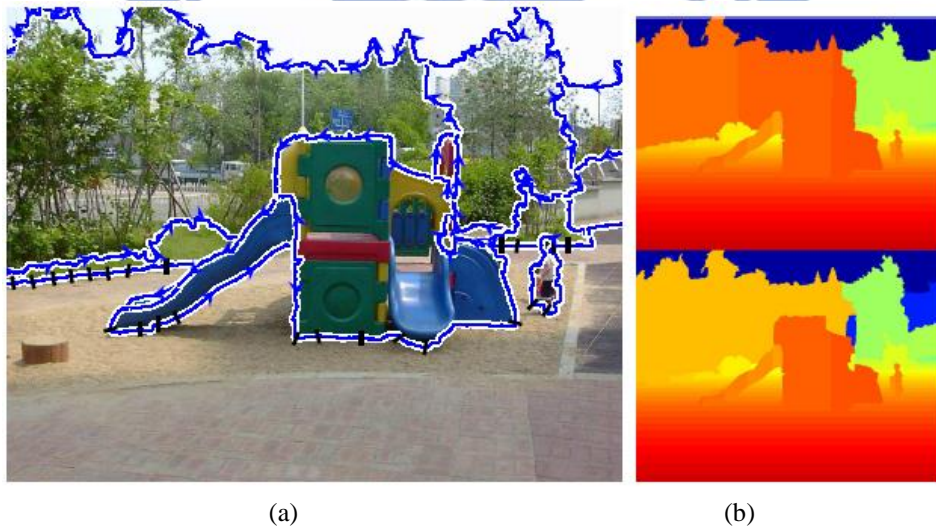


Figure 2-8 Result of Hoiem's occlusion boundary algorithm. (a) Occlusion boundary result. (b) (Top row) Estimated max depth. (Down row) Estimated min depth.

In these aforementioned methods [5, 6], depth can be roughly estimated using scene geometric. However, these algorithms can't estimate the depth, when there are no semantic labels in the image.

2.2 Depth Estimation using Depth Perception Models

In contrast to algorithms that attempt to get the absolute depth value by using depth perception models, some authors have developed methods that infer relative depth information with local depth perception only.

In [4], Saxena *et al.* presented an algorithm that used the Markov Random Field (MRF) model to infer the 3-D structure based on super-pixel features. They used the superpixel segmentation algorithm to divide the image into small uniform regions. And they assumed that each region is a planar surface. They used two kinds of 3-D structure perceptions. First, for each superpixel, they computed some features to capture the monocular cues. Figure 2-9 shows the filters used for texture energies and gradients. Since local image features are insufficient to estimate the depth, they appended local features to multi-scale features and combined features from neighboring superpixels to capture more “contextual” information (See Figure 2-10). Second, they also computed features for boundaries in the image. When two neighboring superpixels of an image have different appearance, they are often grasped to different objects.



Figure 2-9 The convolutional filters used by Saxena [4].

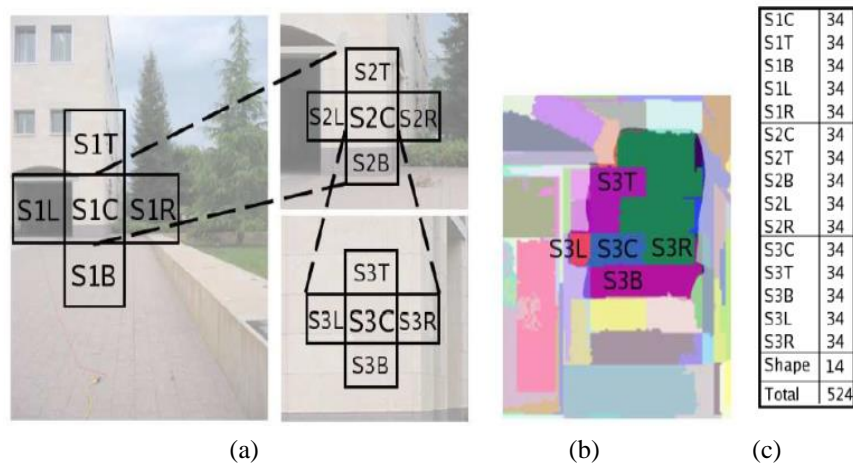


Figure 2-10 Multiple scale structure of features in [1]. (a) An illustration of neighboring location with multi scale features. (b) Actual neighborhood of the superpixel S3C. (c) Collected features for superpixel S3C.

With depth perceptions, they inferred the “plan parameters” that capture both the 3-D location and orientation of the 3-D surface for each superpixel in the image using an MRF model. By supervised learning, the MRF models the relationship between superpixels and infer both the 3-D location and orientation of the 3-D surface for each superpixel. Moreover, the MRF model also considers the relationship between adjacent regions, such as connected structure, co-planar structure, and co-linearity. For connected structure, except in the case of occlusion, neighboring planes are more likely to be connected to each other. For co-planarity structure, neighboring regions are more likely to belong to the same plane if they have similar features. For co-linearity, long straight lines in the image represent straight lines in 3-D, such as edges of buildings. Their results are shown in Figure 2-11.

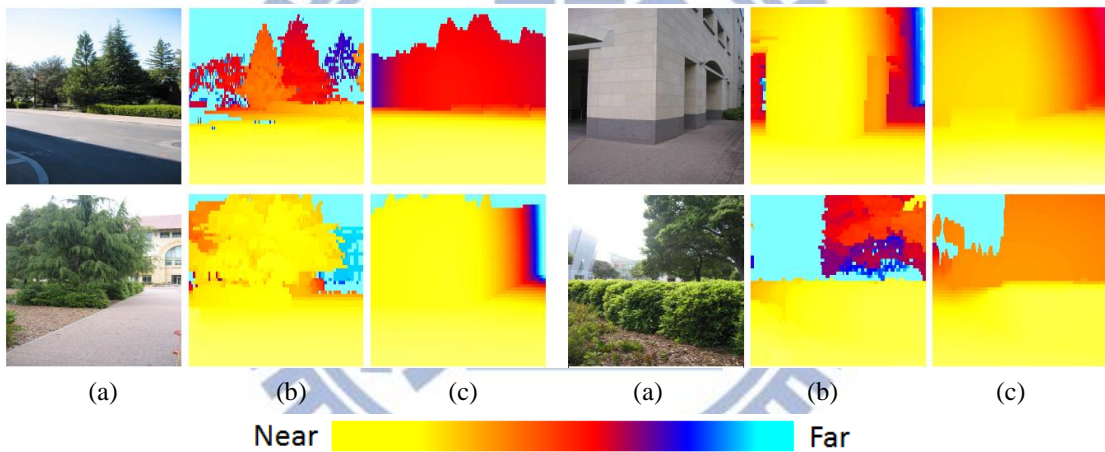


Figure 2-11 (a) Original images. (b) Ground truth depth map. (c) Depth map by [4].

In [8], Jia *et al.* argued that boundary and junction characteristics are important cues for depth ordering. They designed new features on boundaries and junctions, and used these as basic elements to learn the local depth ordering. However, the local depth order reasoning may be inconsistent. Therefore, they used an MRF model to get a globally consistent ordering. In addition, in order to produce a better object segmentation for depth ordering, they proposed to explicitly enforce closed loops and long edges for the occlusion boundary detection.

Jia *et al.* detected the occlusion boundaries in an image and used the result of detecting to partition the image into segments. There are two sets of feature for depth ordering: the

junction feature and the boundary feature. The boundary convexity is calculated for the possibility of the top layer.

After the local inference for depth ordering, they use MRF graph to infer a globally consistent depth ordering. The node potential is defined by the local depth reasoning. The edge potential is defined as the consistency between the segment's orders. Figure 2-12 shows the detail of the MRF model. Their results are shown in Figure 2-13.

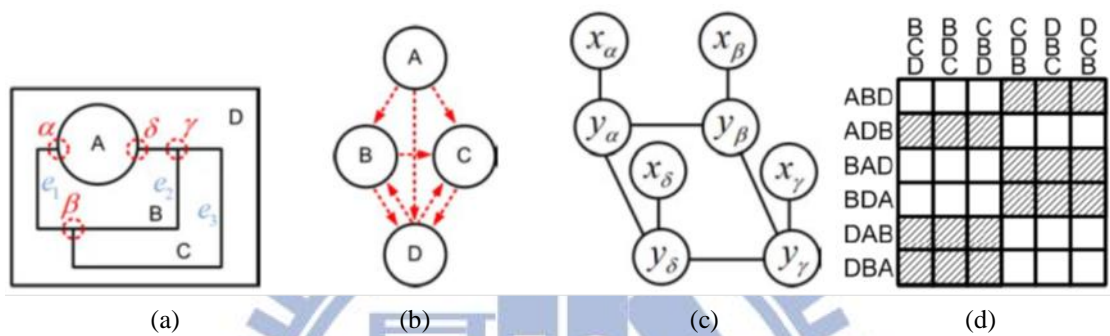


Figure 2-12 MRF model by [10] (a) Global depth reasoning. (b) Each junction produces three directed edges in the depth order graph. (c) MRF to encourage the global consistency. (d) The edge potential in our MRF gives high penalties (solid) if the segments' orders contradict between two nodes.

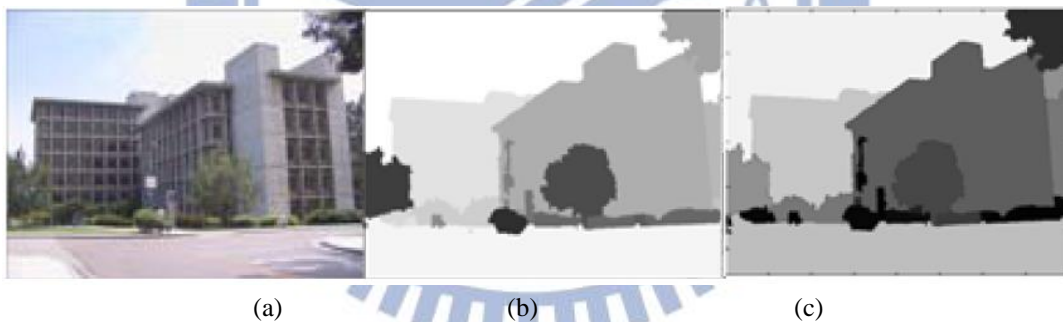


Figure 2-13 (a) Original images. (b) Ground truth depth ordering. (c) Depth ordering by [8].

Chapter 3. 3D Depth Order Reasoning

The goal of our work is to estimate the 3-D depth ordering from a single image. We aim to use local perceptions as the main cues for depth estimation. We directly use the local patch information: the shape information and the appearance information. In our approach, the assumption is that the depth order of regions which are partitioned by segmentation exists, so the image is partitioned by a perfect segmentation and doesn't have cyclic occlusion.

Our work has the following aspects: a) the rules of inferences are designed with learning process which is data-driven, b) we consider both the T-junction features and the boundary features, c) the global ordering is produced by aggregating local decisions and contradictory decisions can be dealt with by a graphical approach.

Our system starts from dividing an image into several regions. Because we assume each region in distinct depth order, we can use the local depth cues to find the depth order of neighbor segments. Due to possible inconsistent depth reasoning resulted from the previous step, we combine the information to produce a better consistent depth order. In Figure 3-1, we illustrate the flowchart of the proposed algorithm. In this chapter, first we introduce which data set we used and the details of our model will be introduced latter.



Figure 3-1 Flowchart of our algorithm

3.1 Data Set

The proposed algorithm is evaluated based on the Berkeley Segmentation Data Set (BSD500) [14]. Some similar works on this topic also use this data set [3, 15]. The ground-truth segmentation boundaries are hand-drawn by different human subjects. Based on these segmentation results, we produce human-labeled depth order for each segment. Figure

3-2 shows an example of this data set.

The Berkeley Segmentation Dataset contains 500 images. Here, we use 400 images as the training data and 100 images as the testing data.

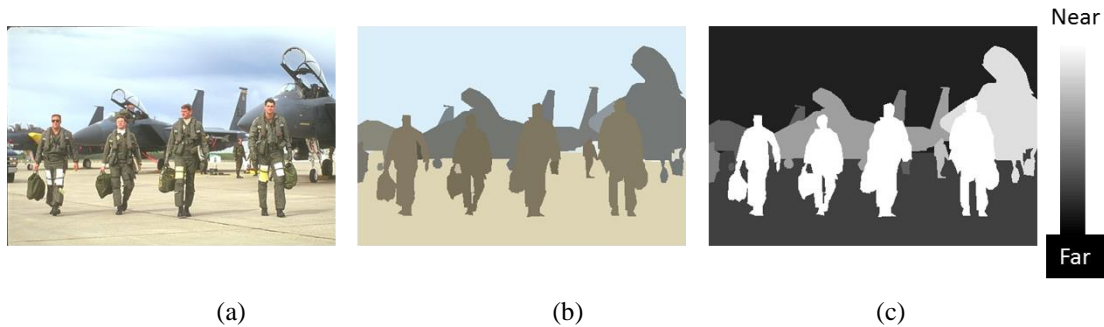


Figure 3-2 (a) Image, (b) Segmentation results, (c) Depth Order.

3.2 Image Segmentation

In the aforementioned work [4, 6], they used over-segmentation techniques and perform grouping segments in regions. The reason they used small regions was to prevent incorrect segmentation results and to collect useful cues, such as colors, textures, to reduce the probability of incorrect segmentation. In contrast, since our local depth inference is region-based, we need to do image segmentation work at first.

There are numerous research works dealing with image segmentation. In our approach, our algorithm is based on the results of ground truth segmentation or Arbelaez's work [14]. Figure 3-2 shows an example of the ground truth segmentation. In 3.2.1, we will introduce Arbelaez's work.

3.2.1 Contour Detection and Image Segmentation

Arbelaez [11] proposed an algorithm that dealt with two computer vision tasks: contour detection and image segmentation. The contour detector combines multiple local cues into a globalization framework based on spectral clustering. Then, the major part of the segmentation algorithm is to transform the result of contour detector into a hierarchical region tree.

In Section 3.2.1.1 and Section 3.2.1.2, we will briefly introduce the contour detection and the segmentation algorithm.

3.2.1.1 Contour Detection

For contour detection, Arbelaez used a function $Pb(x, y, \theta)$ that predicts the posterior probability of the a boundary with orientation θ at each pixel (x, y) . This posterior probability measures in local image brightness, color, and texture channels.

The Pb contour detector is the combination of oriented gradient signals $G(x, y, \theta)$ on four channels. The oriented gradient signal $G(x, y, \theta)$ processes each channel independently. The brightness and color channels correspond to the CIE Lab color space. The texture channel computed by a set of 17 Gaussian derivative and center surround filter. (See Figure 3-3)

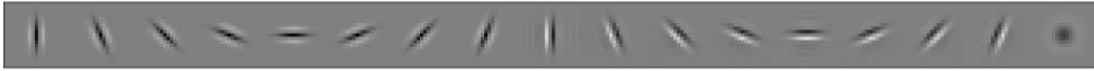


Figure 3-3 Filters for creating textons in [14].

In order to detect fine as well as coarse structure, they extended the Pb detector to three scales. Then they linearly combined these local cues into a single multi scale oriented signal:

$$mPb(x, y, \theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma(i,s)}(x, y, \theta), \quad (3-1)$$

where s indexes scales, i indexes feature channels (brightness, color a, color b, and texture), and $G_{i,\sigma(i,s)}(x, y, \theta)$ measures the oriented gradient of histogram in channel i with radius $\sigma(i, s)$ centered at (x, y) and angle θ . The parameters $\alpha_{i,s}$ weight the relative contribution of each gradient signal. Figure 3 4 shows the multiscale Pb .

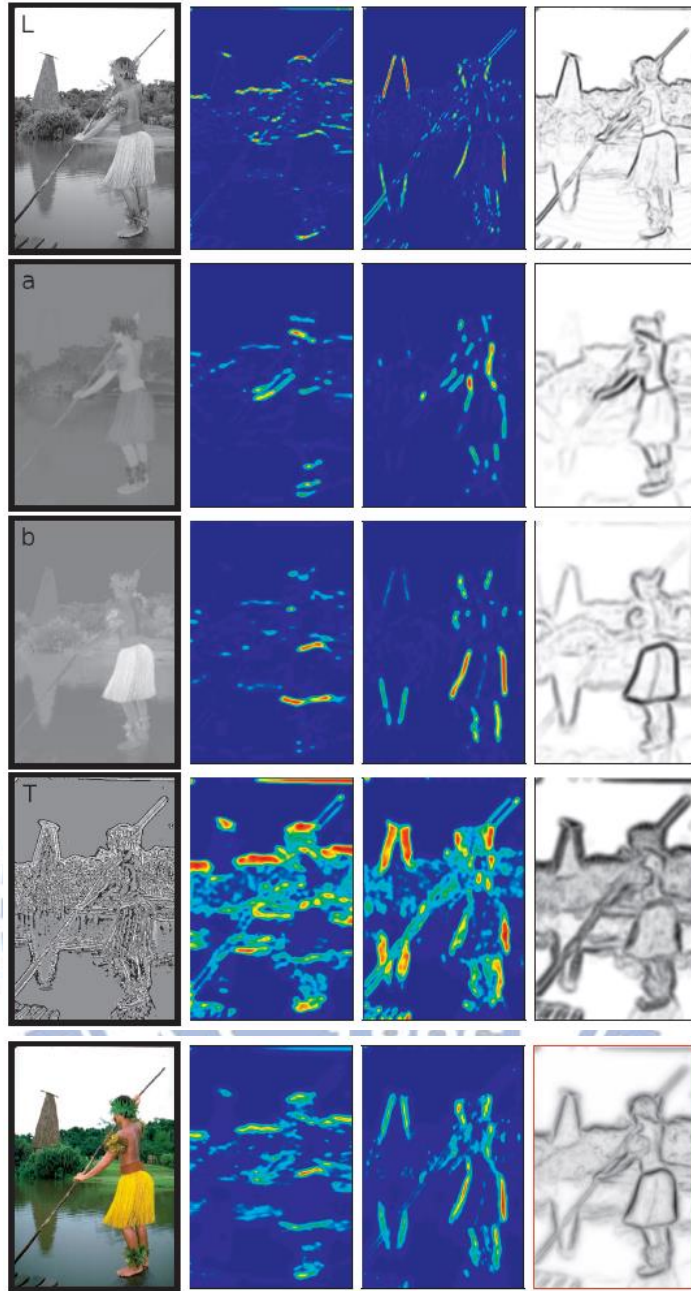


Figure 3-4 Multiscale Pb by [14]. Left Column, Top to Bottom: The brightness, color a, color b and texture channels. Rows: The Oriented gradient of histograms for $\theta=0$ and $\theta=\pi/2$, and the maximum response over eight orientations . Beside the original image, the combination of oriented gradients. The lower right panel, the final output of the multiscale contour detector.

To generate a globalization results, the multiscale local cues are considered as information of spectral clustering. The affinity value W_{ij} is defined by the maximal value of mPb along the line connecting pixels i and pixels j . The equation is as fallow:

$$W_{ij} = \exp\left(-\frac{\max_{p \in I_j}\{mPb(p)\}}{\rho}\right). \quad (3-2)$$

To circumvent this difficulty, they observed that the eigenvectors themselves carry contour information. Treating each eigenvector \mathbf{v}_k as an image, they convolved with Gaussian directional derivative filters at multiple orientations obtaining oriented signals $\{\nabla_{\theta}\mathbf{v}_k(x, y)\}$. The information from different eigenvectors is then combined to provide the “spectral” component of the boundary detector:

$$sPb(x, y, \theta) = \sum_{k=1}^n \frac{1}{\sqrt{\lambda_k}} \cdot \nabla_{\theta}\mathbf{v}_k(x, y), \quad (3-3)$$

where the parameters λ_k weight the directional derivatives. Figure 3-5 presents an example of the eigenvectors, their directional derivatives, and the resulting sPb signal.

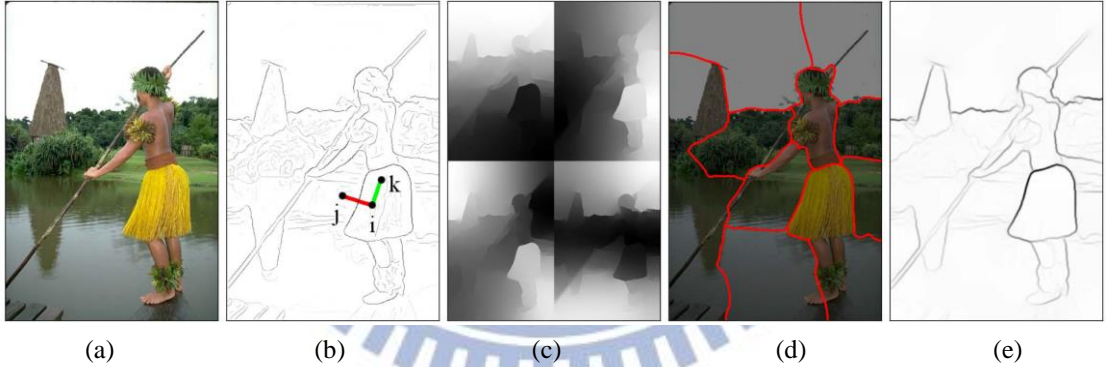


Figure 3-5 (a) Image. (b)A sparse affinity matrix connecting pixels within a fixed radius. Pixels i and j have a low affinity, hereas i and k have high affinity. (c) The first four generalized eigenvectors resulting. (d) Partitioning the image by clustering on the eigenvectors , (e) Spectral Pb.

The signals mPb and sPb convey different information, as the former fires at all the edge, while the latter extracts only the most salient curves in the image. They used linear combination to get both behaviors. So the final globalized probability of boundary is the written as follow:

$$gPb(x, y, \theta) = \sum_s \sum_i \beta_{i,s} G_{i,\sigma(i,s)}(x, y, \theta) + \gamma \cdot sPb(x, y, \theta), \quad (3-4)$$

where the parameter $\beta_{i,s}$, γ weight the contribution of mPb and sPb .

3.2.1.2 Segmentation

The suppressed gPb contours produced in the previous section are often not closed. To produce high-quality image segmentations, they linked this contour detector with a generic grouping algorithm consisting of two steps.

First, they introduced a new image transformation called the Oriented Watershed Transform for constructing a set of initial regions from an oriented contour signal. Second, using an agglomerative clustering procedure, they formed these regions into a hierarchy which can be represented by an Ultrametric Contour Map, the real-valued image obtained by weighting each boundary by its scale of disappearance.

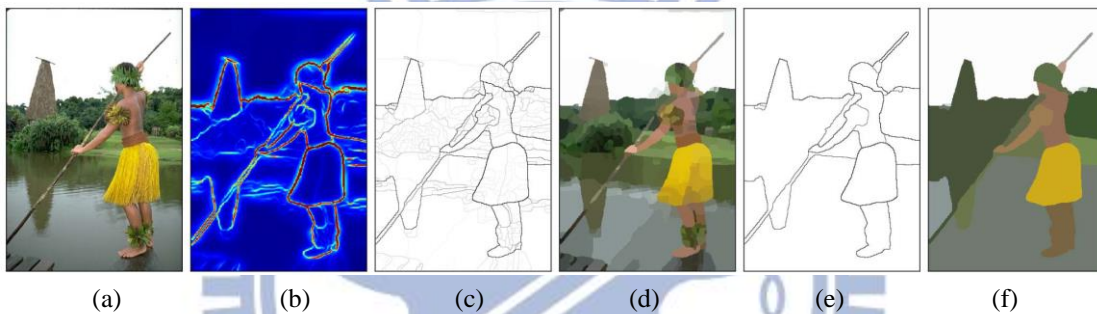


Figure 3-6 Hierarchical segmentation from contours by [14] . (a) Image, (b) Maximal response of contour detector gPb , (c) Oriented Watershed Transform-Ultrametric Contour Map (OWT-UCM), (d) The initial oversegmentation corresponding to the finest level of the UCM, (e)(f) Contours and corresponding segmentation obtained by thresholding the UCM at level 0.5.

3.3 Local Depth Ordering

With the segmentation results of the image, we want to find the depth relation of each region. However, in [3, 7], these authors design the rules of inferences without any learning process. They use hand-crafting local cues to infer the depth. On the contrary, our approach is a learning-based framework and is data-driven. We use a deep architecture to automatically learn the depth perceptions to solve this task.

In Section 3.3.1, we will introduce our deep neural network model. In Section 3.3.2, we will introduce how to train deep neural network. In Section 3.3.3, we will show some testing results.

3.3.1 Deep Neural Network

We want to use deep neural network to find the depth relation in a local patch. In our work, the local patches we used are 16×16 pixels. We can use sliding window searching for the segmentation results of the image to get lots of training data to our models. In our model, we only deal with the local region partitioned into 2 or 3 parts. Because there are few local patches which is partitioned into more than 3 parts, there are 2 deep neural network model that one deal with 2 parts in a local patch, another one deal with 3 parts in a local patch.

The input of the deep neural network contains shape and appearance information of the local patch. The shape information is the group of pixels that the part contains and shows in binary value. The appearance information is the RGB color of the local patch with real value between 0 and 1. Figure 3-7 shows an example of input data. Top row means the input of 2-part model. Bottom row means the input of 3-part model.

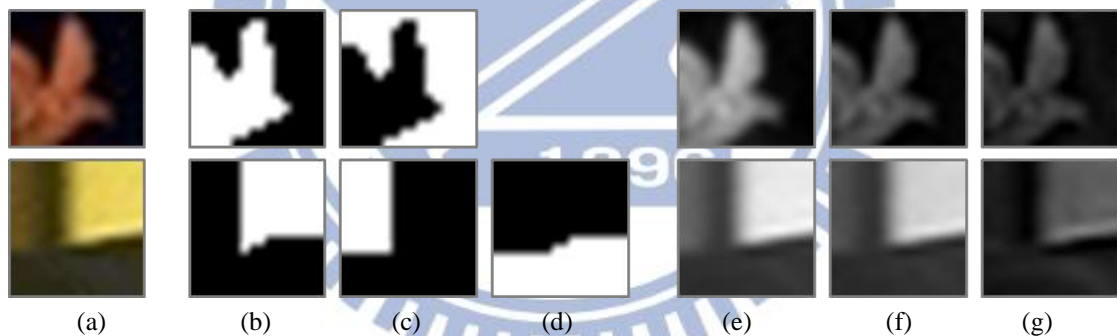


Figure 3-7 (a) Image, (b)(c)(d) Segments of the image, (e)(f)(g) RGB color of the image.

The target of the deep neural network is the depth ordering of segment in local patch. We consider this as a classification problem. For example, if the local region contains part A, B, and C, there are 6 possible depth ordering ABC, ACB, BAC, BCA, CAB, and CBA. Each depth order belongs to one of class. Figure 3-8 and Figure 3-9 show an example of possible depth ordering.

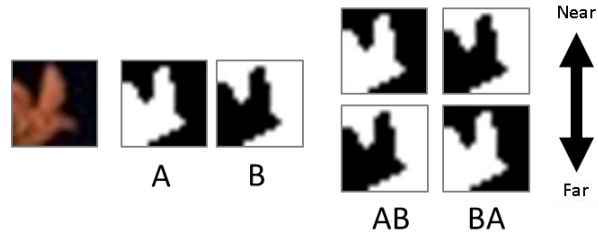


Figure 3-8 2 possible depth ordering of 2-part model.

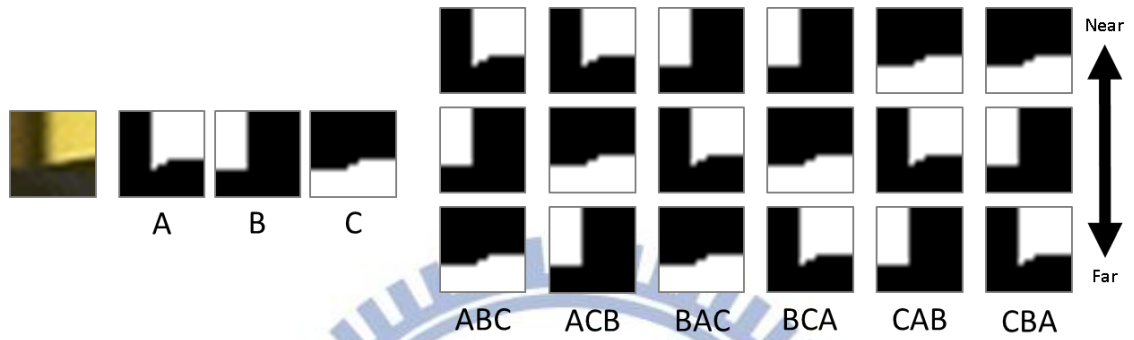


Figure 3-9 6 possible depth ordering of 3-part model.

In the neural network, the hidden units and the target units apply a sigmoid function and softmax function as active function respectively. The model deals with 2 segments in local patch with layers of size $(2*16*16+3*16*16)-700-800-800-2$. The other model deals with 3 segments in local patch with layers of size $(2*16*16+3*16*16)-800-900-900-6$. Figure 3-9 shows our deep neural network model.

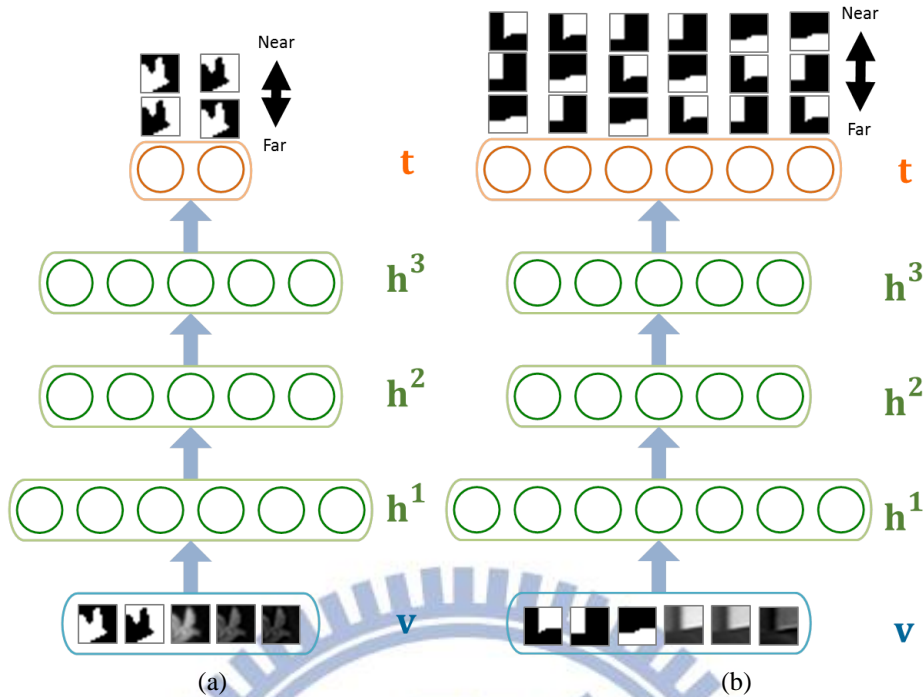


Figure 3-10 (a) Deep Neural Network for 2-part model, (b) Deep Neural Network for 3-part model.

3.3.2 Training

There are three major problems when training a deep neural network. First, the classical methods are efficient when applied to a shallow neural network but not efficient when adapted to a deep neural network. For example, using backpropagation to train a deep neural network will get stuck in local optima. The gradients in the early layers are tiny, making it infeasible to train neural network with many hidden layers. That is why the neural networks are limited to few hidden layers. Second, the classical methods require labeling all the training data. But in the real world almost all data are unlabeled. Third, a discriminative training procedure like backpropagation ignores the structure in the input and only tries to model the relationship between inputs and outputs. This idea is not sufficient when the input contains a lot of structure that can be modeled by features and the output is a class label that is more related to these latent variables than it is to the raw input.

Using unsupervised pre-training can solve these problems. Unsupervised pre-training initializes parameters, which are close to good solution. It also can discover features, which

model the structure of the training images. Once a good set of features has been discovered by unsupervised learning, discriminative learning models the relationship between the features and the class labels. So fine-tuning the model works better than discrimination with random initialization. These features are found by the input data which contain a lot of information, which is enough for labeling the target which typically contain much less information.

Hinton introduced a greedy learning algorithm for unsupervised pre-training in [16]. They used the pre-training result as the initial parameters of model and fine-tuning the model in a supervised fashion. The greedy layer-by-layer learning can find model parameters quickly even when the model contains many layers. It also can make efficient use of very large sets of unlabeled data. The greedy layer-by-layer learning algorithm is that treating each layer as a Restricted Boltzmann Machine (RBM) which is trained in an unsupervised way. Once the previous layers have been trained, the next layer is trained from the transformation of input by the previous layer. In this section, we will introduce the details of the training algorithm.

3.3.2.1 Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) is a two layer, bipartite, undirected graphical model with a set of binary or real-valued visible units \mathbf{v} and a set of binary hidden units \mathbf{h} (See Figure 3-11). The connections of RBM are only between visible units $\mathbf{v} \in \{0,1\}$ and hidden units $\mathbf{h} \in \{0,1\}$. In this section, we describe RBM with binary visible units. The real-valued case having similar properties with binary case will be introduced latter.

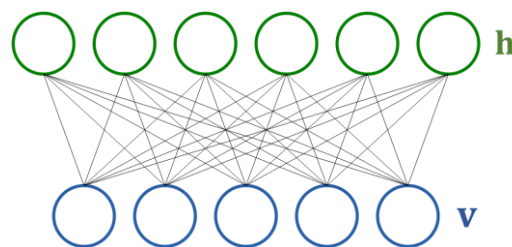


Figure 3-11 Restricted Boltzmann Machine (RBM)

RBM is an energy-based model with the energy function being defined as follows:

$$\text{Energy}(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = - \sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} v_i W_{ij} h_j, \quad (3-5)$$

where v_i, h_j are the binary states of visible units i and hidden units j , b_i, c_j are their biases, W_{ij} is the weight between them, and $\boldsymbol{\theta} = \{\mathbf{b}, \mathbf{c}, \mathbf{W}\}$ are the model parameters. The probability distribution to every pair of a visible and a hidden vector defined by in terms of the energy function:

$$P(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \frac{e^{-\text{Energy}(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})}}{\mathcal{Z}}, \quad (3-6)$$

The constant \mathcal{Z} is a normalizing factor which is called the partition function, is given by summing over all possible pairs of visible and hidden vector:

$$\mathcal{Z} = \sum_{\mathbf{v}, \mathbf{h}} e^{-\text{Energy}(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})}. \quad (3-7)$$

The probability distribution over visible vector \mathbf{v} is defined as follows:

$$P(\mathbf{v}; \boldsymbol{\theta}) = \frac{1}{\mathcal{Z}} \sum_{\mathbf{h}} e^{-\text{Energy}(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})}. \quad (3-8)$$

Because there are no connects between visible units and between hidden units in RBM, it is very easy to get the conditional distributions:

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j = 1|\mathbf{v}) \quad \text{and} \quad P(h_j = 1|\mathbf{v}) = \sigma \left(c_j + \sum_i v_i W_{ij} \right) \quad (3-9)$$

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i = 1|\mathbf{h}) \quad \text{and} \quad P(v_i = 1|\mathbf{h}) = \sigma \left(b_i + \sum_j h_j W_{ij} \right) \quad (3-10)$$

where $\sigma(x)$ is the logistic sigmoid function $\frac{1}{(1+\exp(-x))}$. According to the conditional independent property, it performs block Gibbs sampling in training stage.

In order to train RBM as a probabilistic model, the criterion to maximize is log-likelihood. The gradient of the log-likelihood of a training data \mathbf{v} can be written as follows:

$$\frac{\partial \log P(\mathbf{v})}{\partial \theta} = -E_{P(\mathbf{h}|\mathbf{v})} \left[\frac{\partial \text{Energy}(\mathbf{v}, \mathbf{h})}{\partial \theta} \right] + E_{P(\mathbf{v}, \mathbf{h})} \left[\frac{\partial \text{Energy}(\mathbf{v}, \mathbf{h})}{\partial \theta} \right]. \quad (3-11)$$

This leads to very simple learning rule for performing stochastic steepest ascent in the log-likelihood of the training data \mathbf{v} :

$$\Delta W_{ij} = \epsilon (E_{data}[v_i h_j] - E_{model}[v_i h_j]), \quad (3-12)$$

$$\Delta b_i = \epsilon (E_{data}[v_i] - E_{model}[v_i]), \quad (3-13)$$

$$\Delta c_j = \epsilon (E_{data}[h_j] - E_{model}[h_j]), \quad (3-14)$$

where ϵ is a learning rate, $E_{data}[\cdot]$ is referred as the data-dependent expectation, and $E_{model}[\cdot]$ is referred as the model expectation. The data-dependent expectation term is an expectation under the training set empirical distribution $P(\mathbf{h}|\mathbf{v})$. Because of the conditional independent property, this term is easy to calculate in parallel. The model expectation term is an expectation under the model distribution $P(\mathbf{v}, \mathbf{h})$. This term is intractable. However, it can be approximated by the Markov chain Monte Carlo (MCMC) algorithm such as Gibbs sampling. The idea is simple: starting from any configuration \mathbf{v}^0 , and alternating Gibbs sampling \mathbf{v}^t and \mathbf{h}^t by conditional distribution $P(\mathbf{v}^t|\mathbf{h}^{t-1})$ and $P(\mathbf{h}^t|\mathbf{v}^t)$. Figure 3-12 shows the MCMC algorithm.

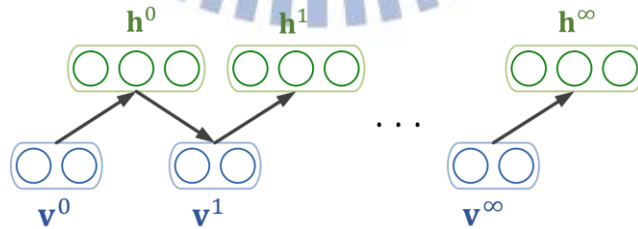


Figure 3-12 Markov chain that uses alternating Gibbs sampling.

However, running a long MCMC chain is still expensive. A much faster algorithm proposed by Hinton [16] is called Contrastive Divergence (CD). Contrastive Divergence reduces the Markov chain with a sample from training dataset. When the Markov chain is reduced to alternate one time, the \mathbf{v}^1 can be considered as the reconstruction data. It still gives good

results. According to the Contrastive Divergence (CD) learning algorithm, the gradient of log-likelihood can be approximated by follows:

$$\Delta W_{ij} = \epsilon(E_{data}[v_i h_j] - E_{recon}[v_i h_j]), \quad (3-15)$$

$$\Delta b_i = \epsilon(E_{data}[v_i] - E_{recon}[v_i]), \quad (3-16)$$

$$\Delta c_j = \epsilon(E_{data}[h_j] - E_{recon}[h_j]), \quad (3-17)$$

where $E_{recon}[\cdot]$ is referred as reconstruction data expectation. Figure 3-13 shows the Contrastive Divergence (CD) learning algorithm.

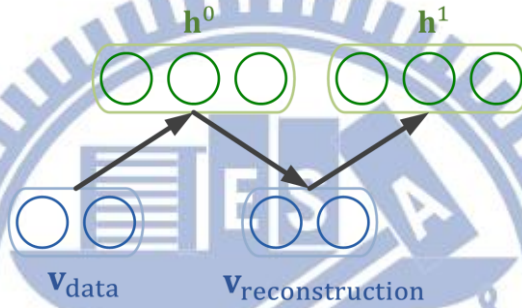


Figure 3-13 Contrastive Divergence (CD).

Although the maximized criterion is not log-likelihood anymore, the experimental results show that the parameter update almost always decreases the log-likelihood.

3.3.2.2 Gaussian Restricted Boltzmann Machine

For real-valued visible units, we use Gaussian RBM. The Gaussian RBM has a similar property with the RBM. Their energy function is defined as follows:

$$\text{Energy}(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = \sum_i \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_j c_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} W_{ij} h_j, \quad (3-18)$$

and the conditional distribution becomes:

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j = 1|\mathbf{v}) \quad \text{and} \quad P(h_j = 1|\mathbf{v}) = \sigma\left(c_j + \sum_i \frac{v_i}{\sigma_i} W_{ij}\right), \quad (3-19)$$

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i = 1|\mathbf{h}) \quad \text{and} \quad P(v_i = 1|\mathbf{h}) = \mathcal{N}\left(v_i | b_i + \sigma_i \sum_j h_j W_{ij}\right), \quad (3-20)$$

The conditional distribution is also easy to compute in parallel. The Gaussian RBM still uses the Contrastive Divergence (CD) learning algorithm to approximate the gradient of log-likelihood.

RBM and Gaussian RBM can be considered as a set of feature detectors to the higher-order correlations between visible units. However, a single layer of binary features is not sufficient to model the structure in a set of images. We can treat the activities of hidden units \mathbf{h} given the training data as the visible units for learning a second layer of features.

3.3.2.3 Greedy Layer-wise Pre-training

In order to learn more structured features, we need more layers to model the structure in a set of images. The greedy layer-wise pre-training treats the activities of hidden units \mathbf{h} in the previous RBM as the visible units in the next RBM. It can be thought that the previous layer of feature detectors is considered as the visible units for learning the next RBM. Units of lower layers detect simple features and feed into high layers, which detect more complex features. This layer-by-layer learning can be repeated as many times as desired.

The detail of greedy layer-wise pre-training is as follows: (1) learning the first RBM in which the visible units are the training data, (2) freezing the weights of first RBM \mathbf{W}^1 and treating the real-valued of probabilities of the conditional distribution $E[P(\mathbf{h}^1|\mathbf{v}; \mathbf{W}^1)]$ as the visible units in second RBM, and (3) Freezing the weights of first and second RBM \mathbf{W}^1 and \mathbf{W}^2 and treating the real-valued of probabilities of the conditional distribution $E[P(\mathbf{h}^2|\mathbf{h}^1; \mathbf{W}^2)] \times E[P(\mathbf{h}^1|\mathbf{v}; \mathbf{W}^1)]$ as the visible units in third RBM. Figure 3-14 shows the flow of greedy layer-wise pre-training.

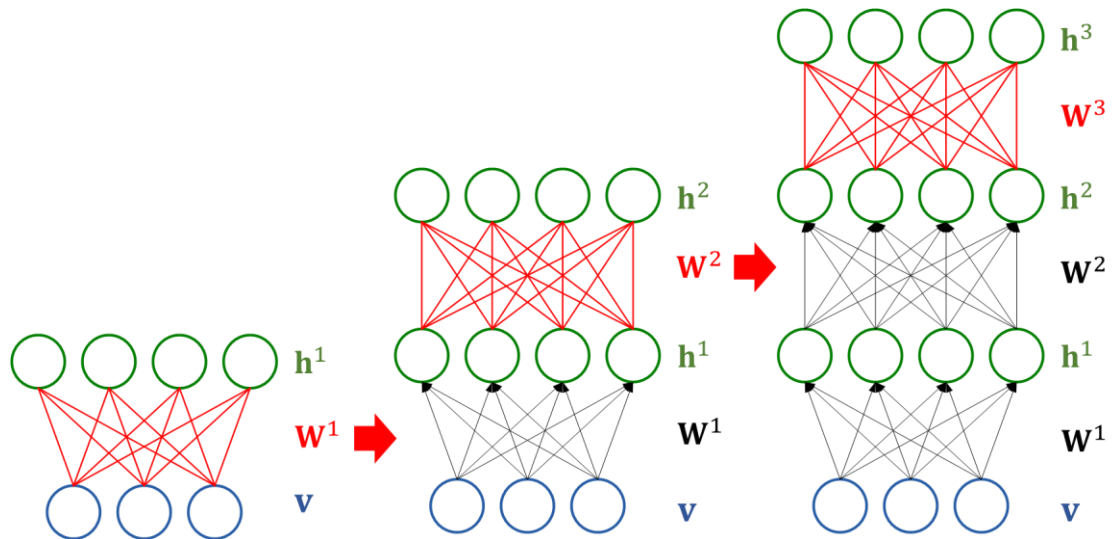


Figure 3-14 Greedy Layer-Wise Pre-training

In our approach, our models have 3 hidden layers, so we need to use 3 stacked RBM to pre-train our models. Otherwise, the inputs of our models are shape information with binary-value and appearance information with real-valued, so the first layer will be the combination of RBM and Gaussian RBM. (See Figure 3-15) The detailed greedy layer-wise pre-training is the same as above. (See Figure 3-16)

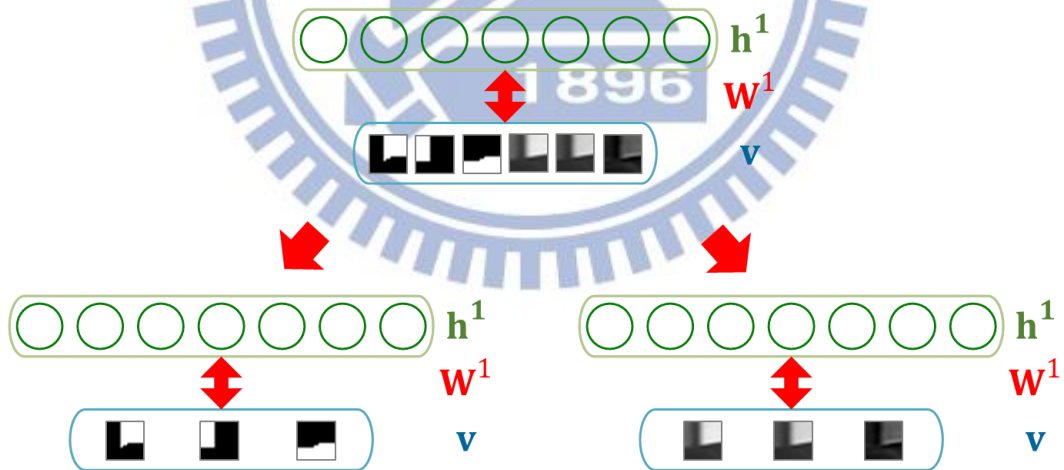


Figure 3-15 The Combination of RBM and Gaussian RBM (3-part Model).

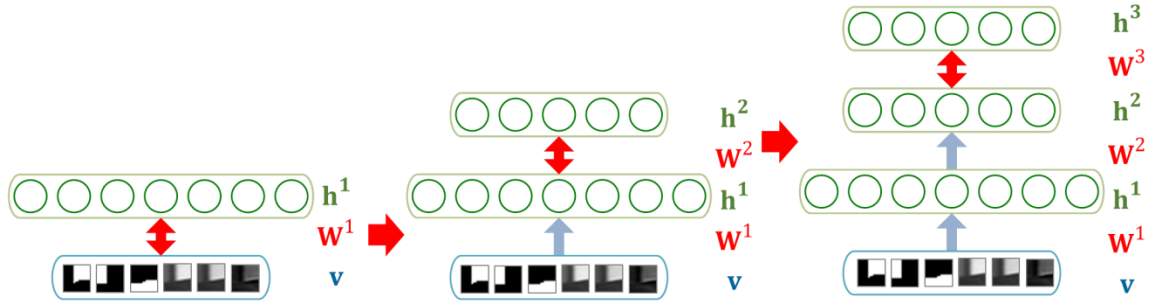


Figure 3-16 Greedy Layer-Wise Pre-training of our model (3-part Model).

With greedy layer-wise pre-training, our proposed model is capable of discovering a lot of hierarchical features. We show the features of hidden units \mathbf{h}^1 by directly displaying the weights. The weights connecting to the segment visible units show in a gray image. The weights connecting to the appearance visible units are shown in a RGB image. The features of hidden units \mathbf{h}^2 are the linear combination of features of hidden units \mathbf{h}^1 with the contribution weight between hidden units \mathbf{h}^1 and hidden units \mathbf{h}^2 . (See Figure 3-17)

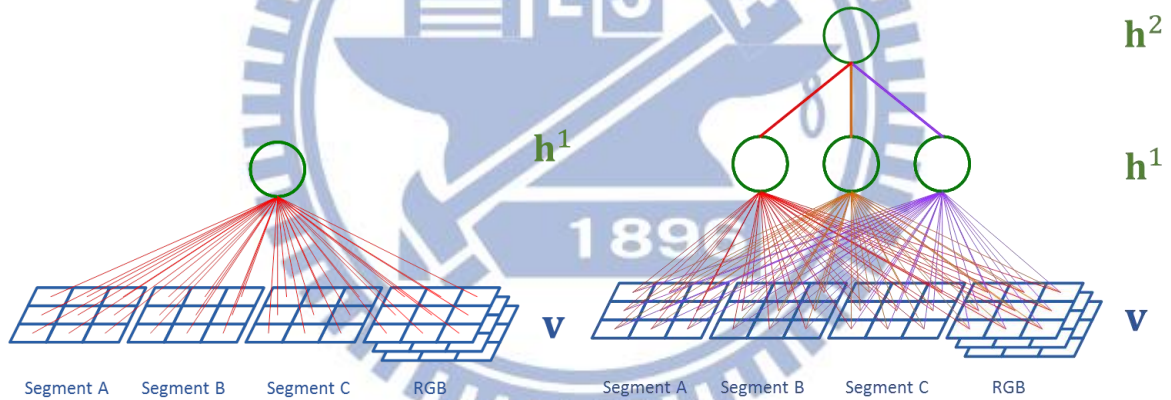


Figure 3-17 The meaning of features of hidden units \mathbf{h}^1 and \mathbf{h}^2 (3-part Model).

Figure 3-18 and Figure 3-19 show the feature learning by the proposed models. As expected, high level features are the combination of low level features.

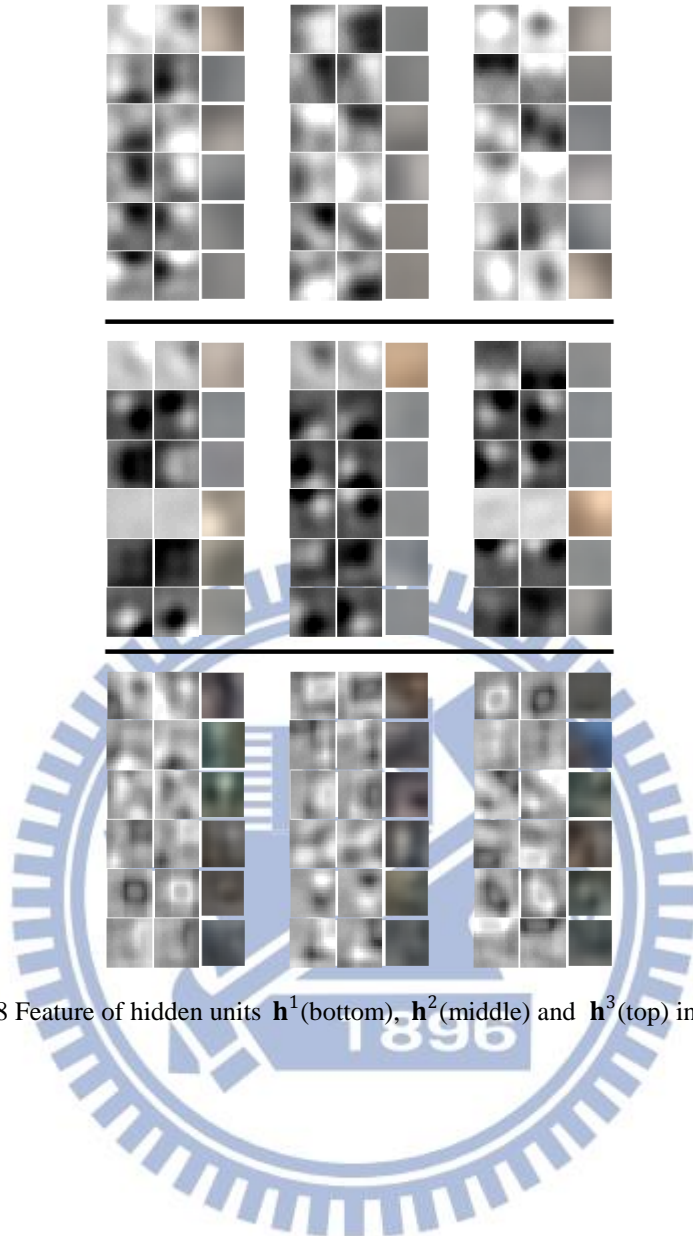


Figure 3-18 Feature of hidden units h^1 (bottom), h^2 (middle) and h^3 (top) in 2-part model.

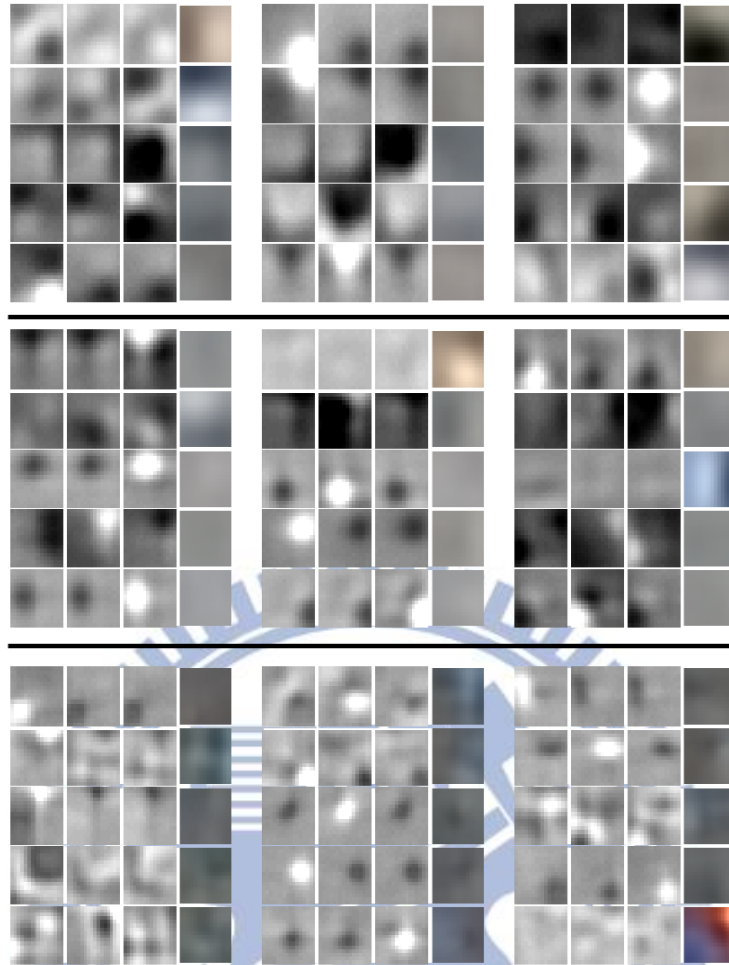


Figure 3-19 Feature of hidden units \mathbf{h}^1 (bottom), \mathbf{h}^2 (middle) and \mathbf{h}^3 (top) in 3-part model.

3.3.2.4 Fine-tuning

With greedy layer-wise pre-training, the proposed model can discover a lot of hierarchical features which contain a lot of information about inputs. The global fine-tuning then learns the model between features and targets for discrimination. We replace stochastic activities by deterministic, real-valued probabilities and use backpropagation through the whole models to fine-tune the weights for optimal outputs. Initializing by unsupervised pre-training produces a much better results than random initialization. Figure 3-20 and Figure 3-21 show the feature learning by our models.

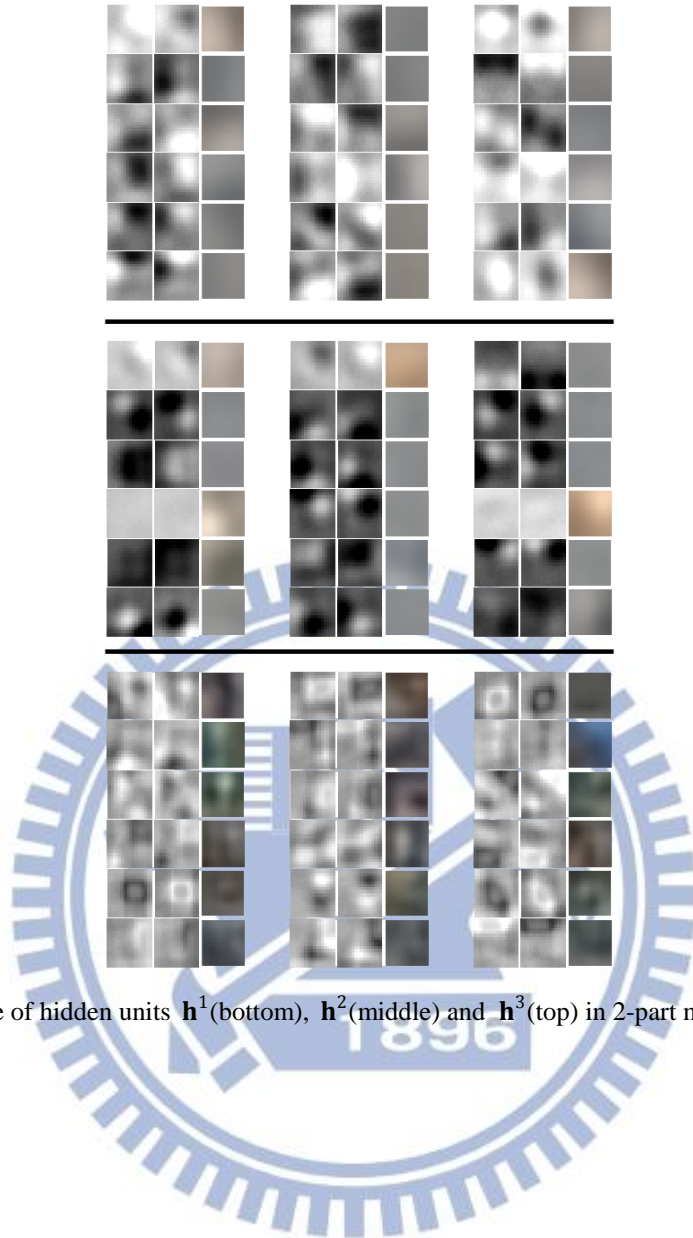


Figure 3-20 Feature of hidden units \mathbf{h}^1 (bottom), \mathbf{h}^2 (middle) and \mathbf{h}^3 (top) in 2-part model after fine-tuning.

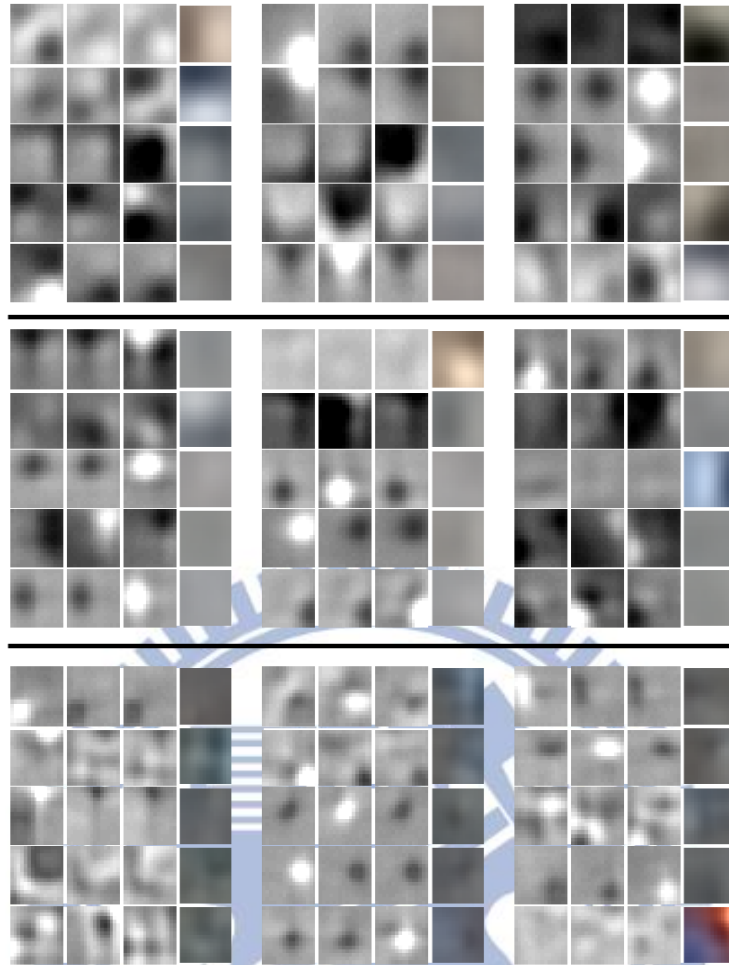


Figure 3-21 Feature of hidden units h^1 (bottom), h^2 (middle) and h^3 (top) in 3-part model after fine-tuning.

3.3.3 Testing

After our models have been trained, we want to infer the depth ordering given the original image and the segmentation results. We use sliding window searching for the segmentation results to find the local patches which are partitioned into 2 or 3 parts. Then, we use the deep neural networks as a classifier to compute the posterior of all possible depth ordering. If a patch is partitioned into part A and part B, the posterior values are considered as the possibility of depth ordering AB and BA. If a patch is partitioned into part A, part B and part C, the posterior values are considered as the possibility of depth ordering ABC, ACB, BAC, BCA, CAB and CBA. Figure 3-22 shows the results of nearest posterior value of each segment in local patch. The nearest posterior value $nPost$ of each segment is defined as

follows:

$$\text{nPost}_{A,B}(A) = \text{Post}_{A,B}(\text{depth order } AB), \quad (3-21)$$

$$\text{nPost}_{A,B,C}(A) = \text{Post}_{A,B,C}(\text{depth order } ABC) + \text{Post}_{A,B,C}(\text{depth order } ACB), \quad (3-22)$$

where A, B, C index segment, $\text{Post}_{A,B,C}(\text{depth order } ABC)$ is the posterior values of depth order ABC calculated by deep neural network in the local patch partitioned into segment A, B and C , and $\text{nPost}_{A,B,C}(A)$ measures the nearest posterior value of segment A in the local patch partitioned into segment A, B and C .

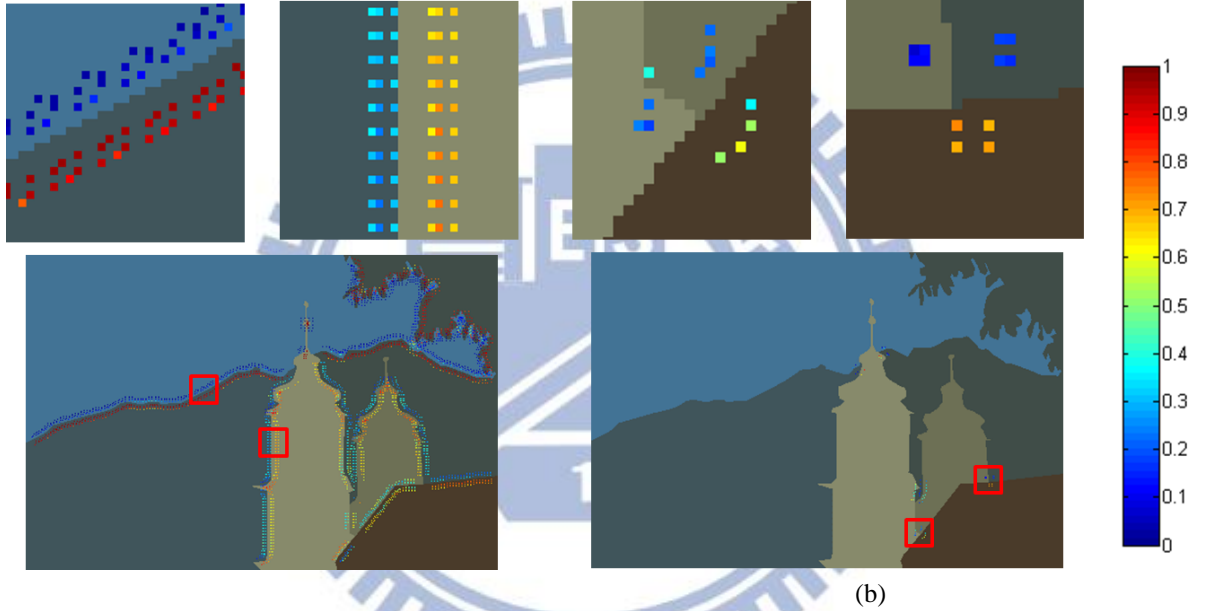


Figure 3-22 Nearest posterior (a) 2-part patch, (b) 3-part patch.

3.4 Global Depth Ordering

Because the depth ordering of segments in local patches may be inconsistent, we combine the local depth reasoning and find a global consistency results across the segments. First, we combine the local depth order reasoning that infers the same neighboring regions. Second, we construct a direct graph with the weights of adjacent matrix being defined by the combination of local depth order reasoning. Because we assume there is no cyclic occlusion existing in our image, we transform the direct graph to the direct acyclic graph with optimal

solution. As we find out the direct acyclic graph, the global ordering can be accomplished.

In Section 3.4.1, we will introduce how to combine the local depth ordering. In Section 3.4.2, we will introduce how to construct a direct graph and transform it to a direct acyclic graph.

3.4.1 Combine the Local Depth Ordering

Since the local depth ordering is based on sliding window the find the local patch. There are a lot of different depth order posterior values inferring the same neighboring regions. We average the posterior values of neighboring regions to get a combination posterior value:

$$cPost_{A,B}(\text{depth order } d) = \frac{1}{N} \sum_i Post_{A,B}(\text{depth order } d, \text{patch } i) \quad (3-23)$$

$$cPost_{A,B,C}(\text{depth order } d) = \frac{1}{N} \sum_i Post_{A,B,C}(\text{depth order } d, \text{patch } i) \quad (3-24)$$

where i indexes the depth ordering results, $cPost_{A,B}(\text{depth order } d)$ measure the combination posterior values of depth order d in the local patch partitioned into part A and B, $cPost_{A,B,C}$ measure the combination posterior values of depth order d in the local patch partitioned into part A, B and C, $Post_{A,B}(\text{depth order } d, \text{patch } i)$ is the posterior value of patch i partitioned into segment A and B with depth order d which is calculated by the deep neural network, $Post_{A,B,C}(\text{depth order } d, i)$ is the posterior value of patch i partitioned into segment A, B and C with depth order d which is calculated by the deep neural network and N is the number of local patches with the same segments.

3.4.2 Direct Acyclic Graph

As we have seen from the discussion in Sec. 3.4.1, the combined posterior values of neighboring regions can't directly give a globally consistent result of depth ordering. We in this section propose a heuristic approach to a consistent result. We treat each segment as a node of a direct graph (See Figure 3-23). The direct edge from node A to node B with weight

w is interpreted as that segment A is in front of segment B with probability w. We construct two depth relation matrixes, one for relation of 2 segments and the other one for relation of 3 segments and then combine them into an adjacency matrix for the direct graph.

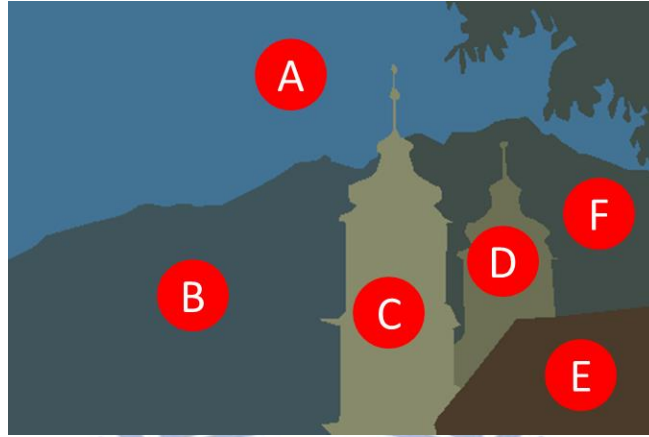


Figure 3-23 Nodes of direct graph.

For relation matrix of 2 segments, it is directly constructed by combining the posterior values. It is defined as follows:

$$\text{RelationMatrix2}_{ij} = \text{cPost}_{i,j}(\text{depth order } ij), \quad (3-25)$$

where i,j index the segment. For relation matrix of 3 segments, it is more complex. It is defined as follows:

$$\begin{aligned} \text{RelationMatrix3}_{ij} &= \sum_k \text{cPost}_{i,j,k}(\text{depth order } ijk) + \text{cPost}_{i,j,k}(\text{depth order } ikj) \\ &\quad + \text{cPost}_{i,j,k}(\text{depth order } kij). \end{aligned} \quad (3-26)$$

Then, we use a simple linear combination to construct adjacency matrix.

$$\text{Adjacencymatrix} = \alpha \cdot \text{RelationMatrix2} + (1 - \alpha) \cdot \text{RelationMatrix3}, \quad (3-27)$$

where the parameters α and $(1 - \alpha)$ respectively weight the relation contribution of RelationMatrix2 and RelationMatrix3. Here we use $\alpha=0.2$, because RelationMatrix3 contain more information about depth ordering. Figure 3-25 shows an example of the direct graph.

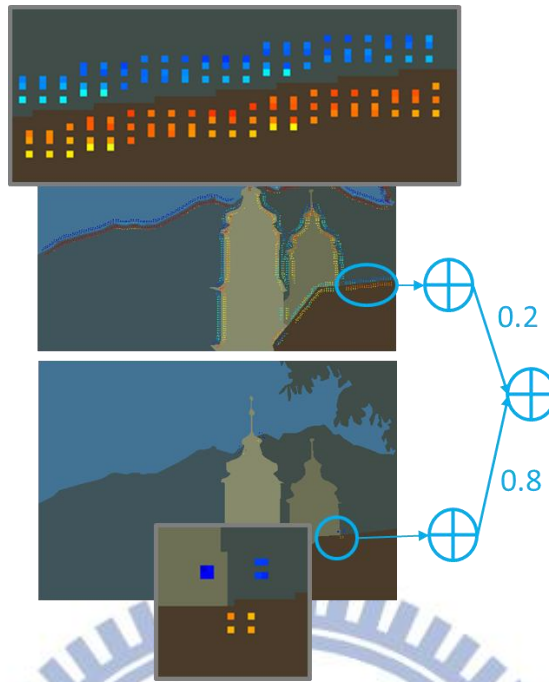


Figure 3-24 Example of computing the depth relation between node E and F.

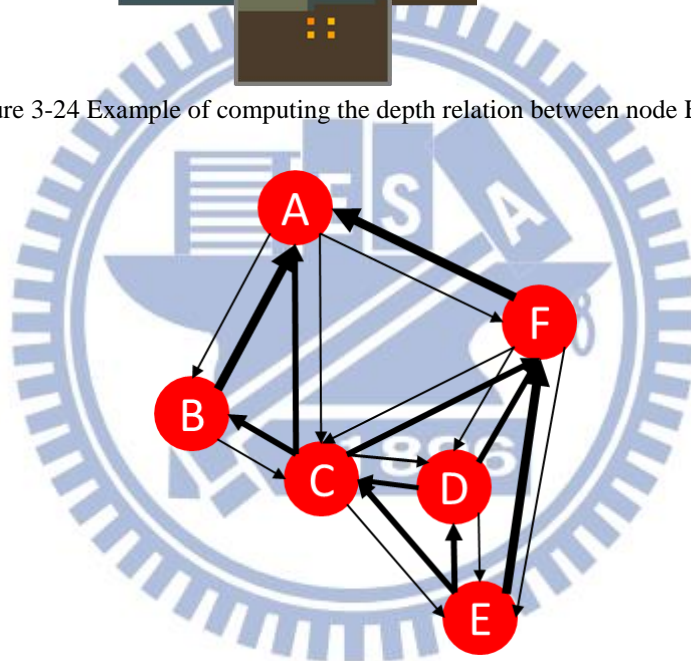


Figure 3-25 Direct graph.

After constructing the direct graph, it is instinctive that we can remove the edges that form a direct acyclic graph. A direct acyclic graph is a result of depth ordering of each segment. The best result is that the summation of weights which have been removed is minimum. This is called the minimum feedback arc set problem. However, it is an NP-hard problem. In our approach, we limit the number of segments to be lower than 15 and use exhaustive search to find the minimum cost. Figure 3-26 (b) shows an example of direct acyclic graph. Figure 3-27 shows the results of an example.

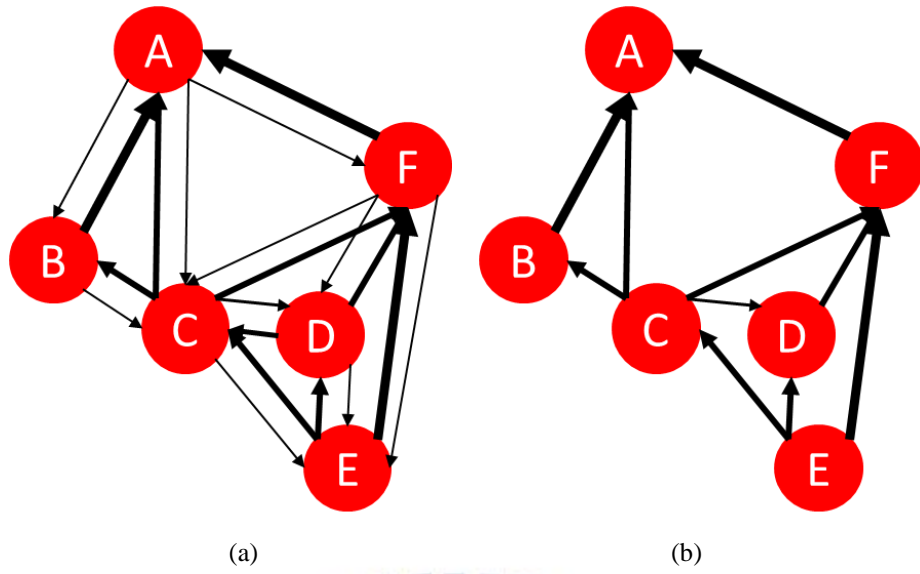


Figure 3-26 (a) Direct graph, (b) Direct acyclic graph with feedback minimum arc set.

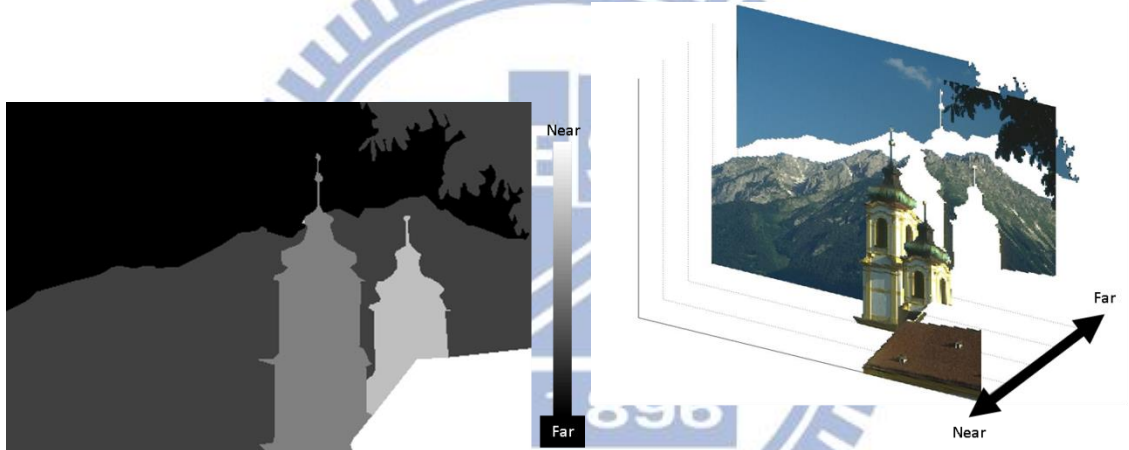


Figure 3-27 Depth ordering.

Chapter 4. Experimental Results

In this chapter, we will show several simulation results of the proposed algorithm. First, we will show the accuracy of the deep neural network and local depth reasoning. Second, we will show the global depth ordering based on the ground truth segmentation or Arbelaez's work.

4.1 Deep Neural Networks

We evaluate deep neural networks on validation data. Figure 4-1 shows how the evaluations work. Given an image, a sliding window moves in the image to find 2-part or 3-part patches. We can compute the accuracy rate of validation data. The accuracy is defined as whether the prediction is equal to the ground truth or not. For example, if a local patch is partitioned into part A, B and C with depth ordering ABC, the accuracy is 1 whenever the prediction of depth ordering is ABC. If the prediction of depth ordering is ACB, the accuracy is still 0. Table 4-1 shows the accuracy rate. We can see that the accuracy rate is higher than 50% (2-part randomly guesses) or 17% (3-part randomly guesses).

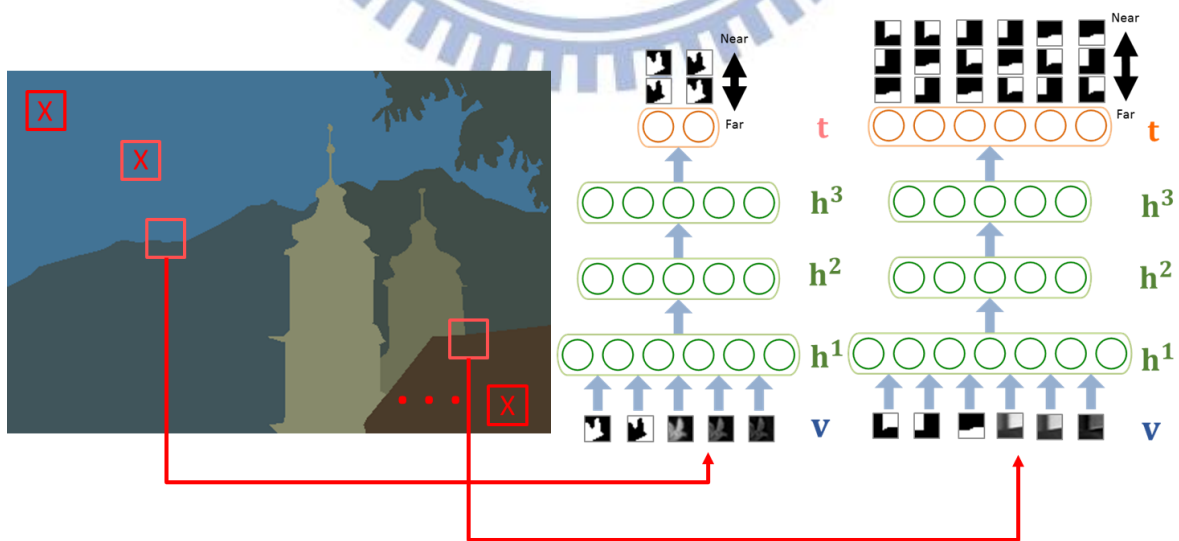


Figure 4-1 Testing

Table 4-1 Accuracy rate.

	Data	Accuracy Rate(Validation Data)
2-part model	Training Data: 382,600 Validation Data: 164,100	71.5%
3-part model	Training Data: 366,200 Validation Data: 157,000	42.4%

4.2 Local Depth Ordering

Figure 4-2 shows some results of local depth ordering. As we expect, the deep neural networks almost can predict correct depth ordering. The posterior means the confidence of depth ordering.

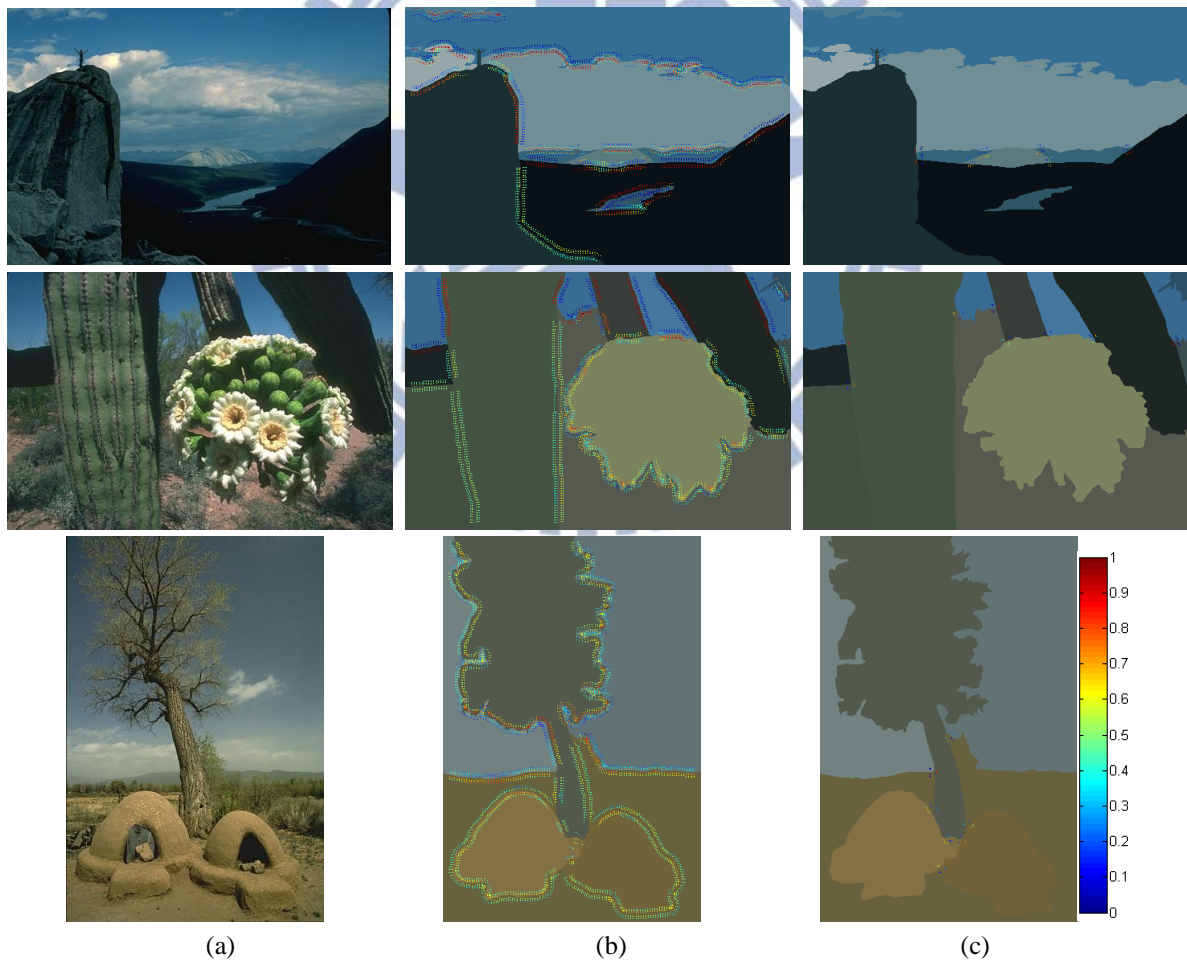


Figure 4-2 Local Depth Ordering, (a) Image, (b) 2-part patch, (c) 3-part patch

4.3 Global Depth Ordering

In this section, we will show simulation results of the proposed single image 3-D depth order algorithm. We use 100 test images from Berkeley Segmentation Dataset (BSD500). Our algorithm is based on the results of ground truth segmentation or Arbelaez's work. In Section 4.3.1 and Section 4.3.2, the results of ground truth segmentation and Arbelaez's work will be shown, respectively.

4.3.1 Based on Ground Truth Segmentation

Since the ground truth segmentation has human-labeled ground truth depth order, we can compute the accuracy rate of global depth order:

$$\text{Accuracy} = \frac{\# \text{ of correct edge}}{\# \text{ of edge}} \times 100\%. \quad (4-1)$$

Given a direct graph, which is produced by the proposed algorithm, we compute the ratio of number of correct edge to number of edge. It checks every edge of the direct graph with the ground truth depth order. Our accuracy rate on 100 testing images is 74%. Figure 4-3 shows the histogram of accuracy rate. Most of image exceed 65% accuracy rate.

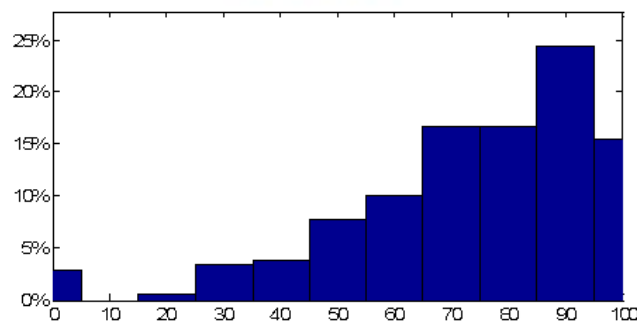


Figure 4-3 Histogram of accuracy rate.

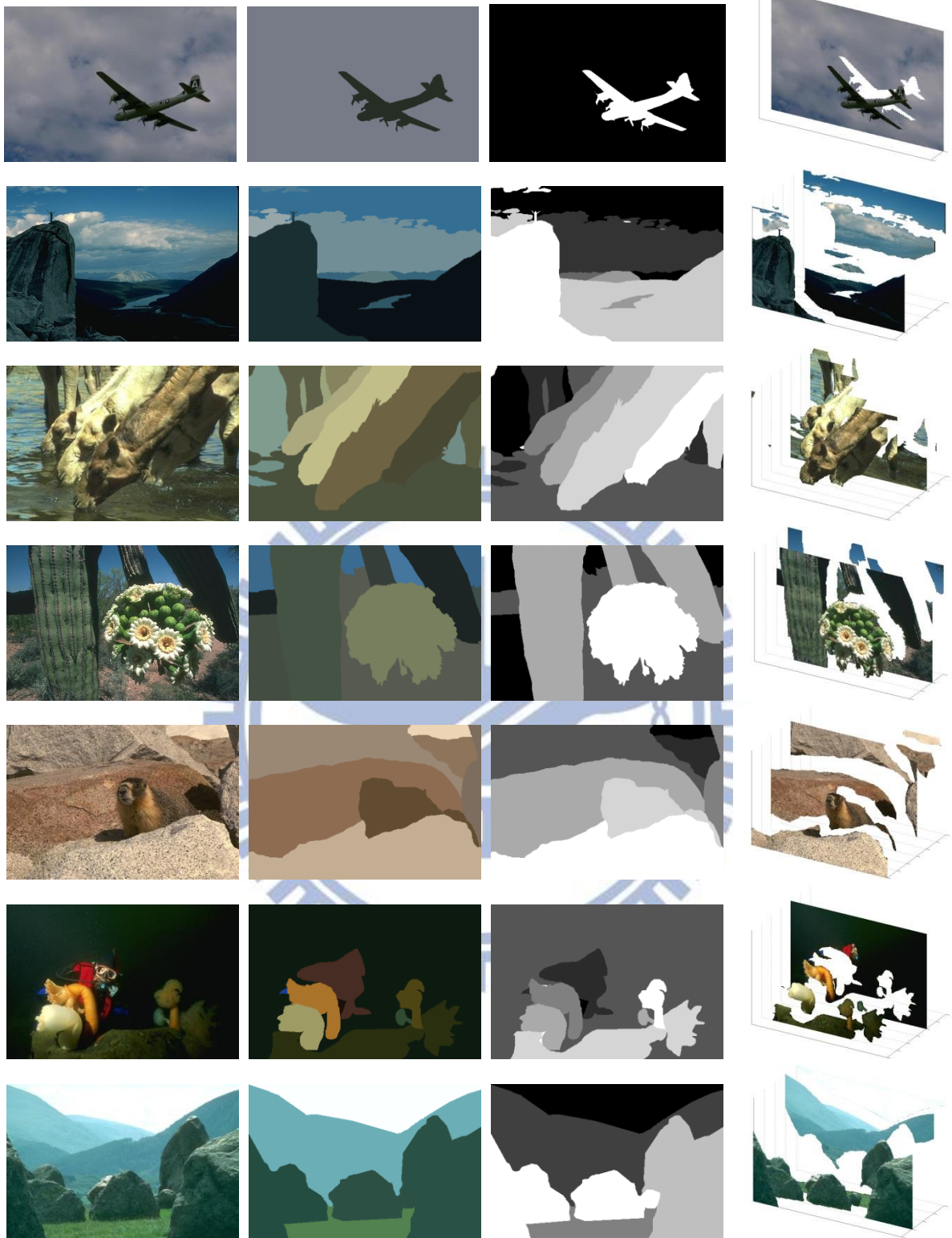


Figure 4-4 Results based on ground truth segmentation



Figure 4-5 Results based on ground truth segmentation

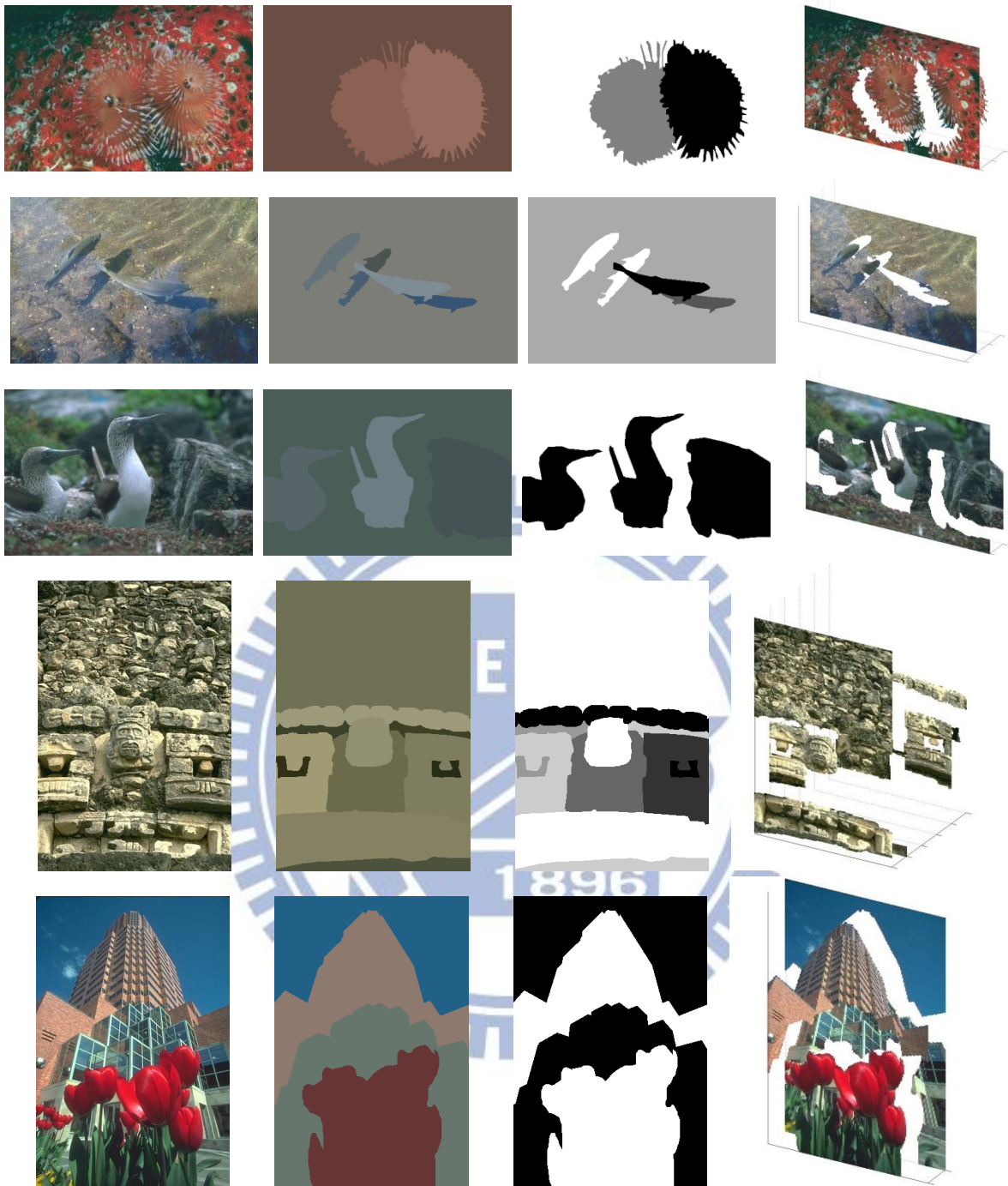


Figure 4-6 Results based on ground truth segmentation

Figure 4-4 and Figure 4-5 show the results that the accuracy rate is higher than 70%. Most of depth relation between two objects is correct. Figure 4-6 shows the results that the accuracy rate is lower than 50%. It often occurs that the number of segments is small, so the accuracy rate is small when only rare edges are correct.

4.3.2 Based on Contour Detection and Image Segmentation

Since our approach relies on the segmentation result, the accuracy of depth order will be low with poor segmentation result. Figure 4-7 and Figure 4-8 show some results based on Arbelaez's work.



Figure 4-7 Results based on Arbelaez's work

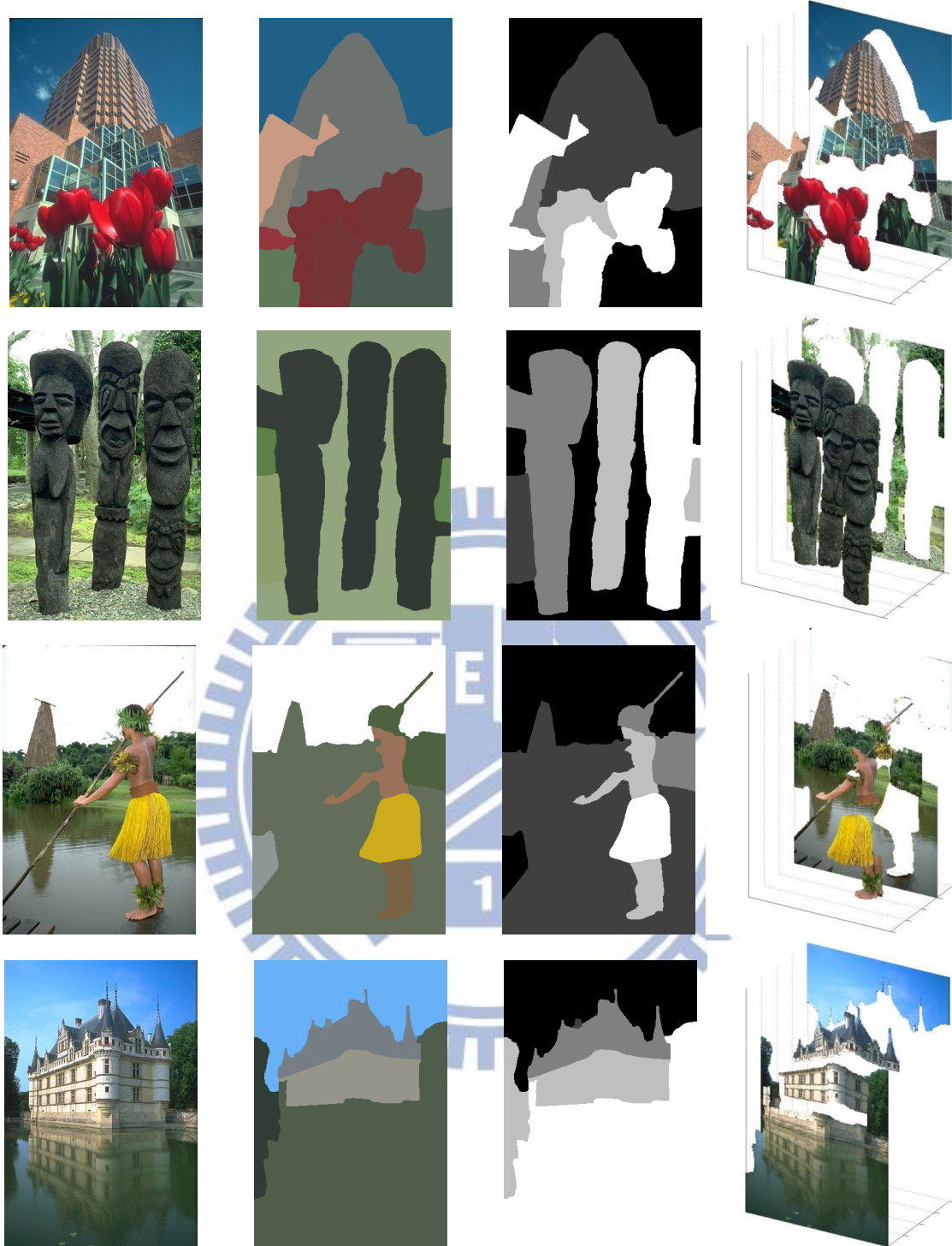
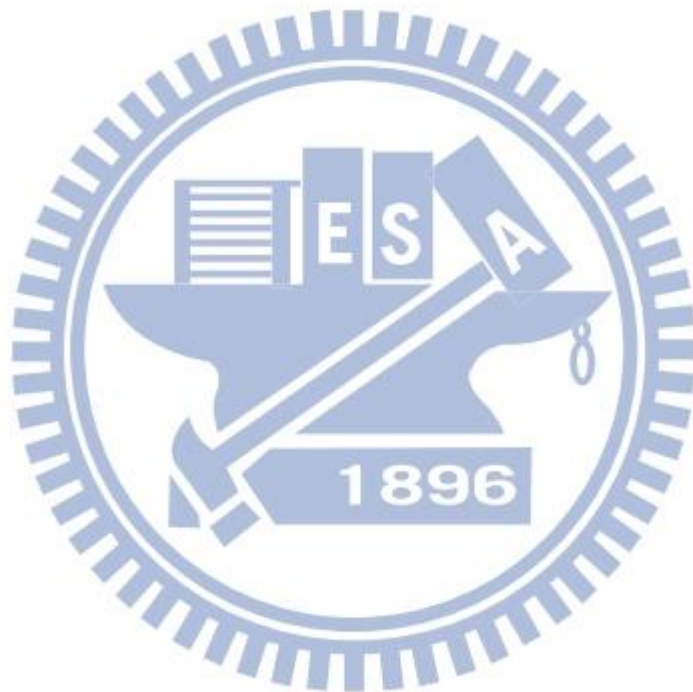


Figure 4-8 Results based on Arbelaez's work

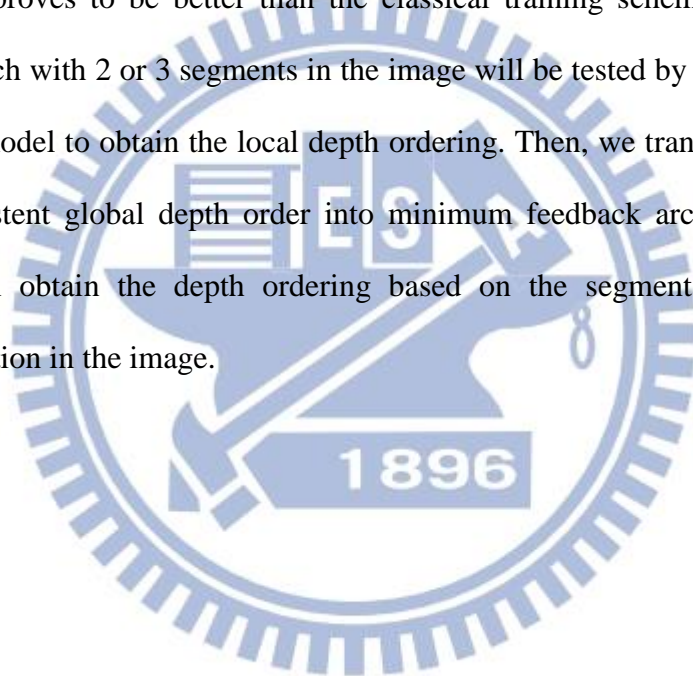
There are some limitations in our algorithm. First, our approach assumes that we have a perfect segmentation results. However, the segmentation results are not correct when adjacent objects are similar in color or texture. This incorrect segmentation result will affect the depth order. Second, since we use exhaustive search to solve the feedback arc problem, we can't

deal with the image which is partitioned into too many regions.



Chapter 5. Conclusion

In this thesis, we have proposed a method to infer the depth ordering of object from a single image based on local shape and appearance information. Our approach has assumed the segmentation result is perfect. With the segmentation result, we know the number of segment in local patch. Then, we can individually train the deep neural networks for 2 segments and 3 segments. The deep neural network can automatically learn the features from the training data. We have used the unsupervised pre-training and the supervised fine-tuning to train our networks, which proves to be better than the classical training scheme. After training the network, local patch with 2 or 3 segments in the image will be tested by feeding into our deep neural networks model to obtain the local depth ordering. Then, we transform the problem of obtaining a consistent global depth order into minimum feedback arc set problem. In our approach, we can obtain the depth ordering based on the segmentation result with no geometric assumption in the image.



Reference

- [1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, pp. 7-42, 2002.
- [2] J. Ponce, D. Forsyth, E.-p. Willow, S. Antipolis-Méditerranée, R. d'activité-RAweb, L. Inria, *et al.*, "Computer vision: a modern approach," *Computer*, vol. 16, p. 11, 2011.
- [3] M. Dimiccoli and P. Salembier, "Exploiting t-junctions for depth segregation in single images," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1229-1232.
- [4] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 824-840, 2009.
- [5] J.-I. Jung and Y.-S. Ho, "Depth map estimation from single-view image using object classification based on Bayesian learning," in *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video*, 2010, pp. 1-4.
- [6] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering occlusion boundaries from an image," *International Journal of Computer Vision* vol. 91, pp. 328-346, 2011.
- [7] G. Palou and P. Salembier, "Occlusion-based depth ordering on monocular images with binary partition tree," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2011, pp. 1093-1096.
- [8] Z. Jia, A. Gallagher, Y.-J. Chang, and T. Chen, "A learning-based framework for depth ordering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 294-301.
- [9] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *IEEE Conference on Computer Vision and Pattern Recognition* 2010, pp. 1253-1260.
- [10] Y. Bengio, *Learning Deep Architectures for AI* vol. 2, 2009.
- [11] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks," presented at the Neural Information Processing Systems, 2006.
- [12] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, pp. 504-507, 2006.
- [13] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *International Journal of Computer Vision*, vol. 75, pp. 151-172, 2007.
- [14] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 898-916, 2011.
- [15] M. Dimiccoli, J.-M. Morel, and P. Salembier, "Monocular depth by nonlinear diffusion," in *Sixth Indian Conference on Computer Vision, Graphics & Image*

Processing, 2008, pp. 95-102.

- [16] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, pp. 1527-1554, 2006.

