Skyline Segmentation using Shape-constrained MRFs

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science Engineering

by

Rashmi Vilas Tonge
200902022
rashmivilas.tonge@students.iiit.ac.in

Center For Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2014

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Skyline Segmentation using Shape-constrained MRFs" by Rashmi Vilas Tonge, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. C. V. Jawahar

To *Parents*

Acknowledgments

I would first like to thank my advisers Prof. C. V. Jawahar and Prof. Subhransu Maji for all the guidance and support. I got to learn the valuable lessons from both of them. For example, I learned "*art of perfection*" and the "*devil is in detail*" from Prof. Jawahar, while I learned the "*never before art of excellent coding*" and passion from Prof. Maji. I would also like to thank Prof. K. S. Rajan for his guidance and support in *Honors Project* and helping me choose my "*calling*". I have been fortunate enough to have worked under and learned from Prof. Jawahar, Prof. Maji and Prof. Rajan.

I would also like to thank my research mentor, Nisarg Raval who gave me first-hand experience of the research in vision and subsequently made my transition easy. It was great fun and an unforgettable experience for me to be part of CVIT. I would like to thank all the CVIT-mates for giving your valuable advice and guidance. A special thanks to Prof. P. J. Narayanan, Prof. Jayanthi Sivaswamy and Prof. Anoop Namboodiri for making CVIT what it is. I would also like to thank all the professors, TAs and seniors for their teachings, lessons and guidance.

Coming to more of a personal territory, I would like to thank from the bottom of my heart to my friends at IIIT — Pranneetha, Avni and Mounica for the fun, emotional support, for always being there for me and for making the college life truly memorable.

Last but not the least, most importantly I would like to thank my parents and brother for unconditionally loving and supporting me, motivating me to stay focused, basically tolerating all the mess, that is me. Finally, I would like to thank the God (if exists) well, for everything.

Abstract

MRF energy minimization has been used for image segmentation in a wide range of applications. Standard MRF energy minimization techniques are computationally expensive. Besides, incorporating higher order priors such as shape and parameters related to it is either very complex or computationally expensive or requires prior information such as shape location. Furthermore, semantic understanding is not achieved using pure MRF formulation, i.e. information about the structure of a skyline such as depth cannot be known through output. Standard semantic segmentation methods using geometric context information is restricted to very few geometric classes or the ones which exploit specific "tiered" structure is computationally exponential in number of labels.

Our aim is to extract the detailed structure of a skyline, i.e. individual buildings and their depth. In this case, there is no restriction on the number of labels. The problem is challenging due to numerous reasons such as complex occlusion patterns, large number of labels and intra-region color and texture variations, etc. We propose an approach for segmenting the individual buildings in typical skylines. Our approach is based on a Markov Random Field (MRF) formulation that exploits the fact that such images contain overlapping objects of similar shapes exhibiting a "tiered" structure. Our contributions are the following:

1. We introduce a dataset **Skyline-12** consisting of 120 skyline images from the 12 cities all over the world. All the images are manually annotated with addition of meta-data like initial boundaries and seeds.

2. We include an analysis and integration of low-level features such as color, texture and shape very useful for the segmentation of skylines.

3. We propose a fast, accurate and robust method to extract individual buildings of a skyline exploiting "tiered" structure of a skylines and incorporating rectangular shape prior in MRF formulation.

For simple shapes such as rectangles, our formulation is significantly faster to optimize than a standard MRF approach, while also being more accurate. We experimentally evaluate various MRF formulations and demonstrate the effectiveness of our approach in segmenting skyline images.

We propose both *Interactive* and *Automatic* methods for segmenting skylines. While interctive setting gives an accurate output and a fast approach to segment skylines given input seeds from user, automatic setting provides about 25% improvement over *state-of-art* low level automatic segmentation

methods. Our approach can be generalized to different shapes as well as detailed structure of a skyline can be used in many applications such as 3D reconstruction of a skyline from single image.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Segmentation is one of the oldest and one of the most common problems in computer vision where each pixel is labeled to be a part of a segment. Semantic Segmentation is relatively new problem in image segmentation, where each pixel is assigned to a meaningful label. Generally, labels in a semantic segmentation setting are limited or belong to some known set of classes, e.g. sky, ground, vertical-horizontal, etc. We take semantic segmentation one step further in a sense that we aim to extract meaningful labels in a setting with unlimited labels.

In computer vision, a lot of problems are considered where 'tiered' structures in an image are exploited to segment it into meaningful tiers or classes. For example, a center-viewing image of a street is classified into tiers such as left facing buildings, right facing buildings and buildings facing to the center. On the contrary, we consider large set of labels i.e. all buildings in a skyline and we do not impose restriction on number of labels (buildings). In our case each building or an object becomes a horizontal tier and a skyline becomes a set of such overlapping tiers.

In day-to-day scenario, we often see such 'tiered' structures, i.e. where an object occludes another object forming tiers, e.g. *Cars* in a parking lot, *pine trees* in a jungle or *buildings* in a skyline. A few such cases are shown in Figure 1.1. In each of these settings, objects of similar shapes occlude one another creating a 'tiered' structure. Not only we have a structural information in these cases but also, repetition of similar shapes gives us a knowledge of the shape of objects into consideration. We propose a fast, efficient and accurate method to extract detailed structure of *skylines* via exploiting the 'tiered' structure and shape information.

1.2 Problem and Challenges

We are interested in extracting the detailed structure of buildings within photographs of skylines as shown in Figure 1.2. The skylines of cities such as Chicago, New York, Hong Kong and Tokyo, among others are a subject of great interest among professional and amateur photographers alike, hence one

Figure 1.1: In the figures, some of the 'tiered' structures are shown where objects of similar shapes occlude each other to create tiers of various shapes.

can find an immense number of these pictures on the web. Some of these cities are known for their exceptionally tall buildings, others for their unique designs, and these photographs provide a gist of their architectural styles. Automatic segmentation of individual buildings from images can be used in a number of applications for designers and artists such as renderings of these from novel viewpoints, information overlays, creation of virtual cities, and other applications such as 'geo-location' by matching individual buildings to a dataset of known buildings.

1.2.1 Challenges

The proposed task is quite challenging due to multiple reasons -

1. **Complex Occlusion Patterns -** Skylines typically contain many tightly packed buildings that partially occlude one another leading to complex occlusion patterns. Few examples where such complex patterns are formed are shown in Figure 1.3.

2. **Intra-building Variations -** Furthermore, different facades of the same building can appear quite different from one another due to sunlight, i.e. different facades of the same building display different colors and textures. Few sample buildings where each facades look different due to lights are shown in Figure 1.4.

3. **Reflections -** In some cases, the buildings are reflected on the facades of other buildings. In such cases, it becomes hard to perceive these buildings as different buildings or as a single building even for human perception. Few such examples are shown in Figure 1.5.

4. **Viewpoint -** Due to various viewpoints from which photograph is taken, same building looks completely different in another photograph of the same skyline. Few such examples are shown in Figure 1.6.

5. **Large number of labels -** All the "state-of-the-art" algorithms are designed for binary or limited number of labels. Each Skyline typically contains $25 - 60$ buildings. All the standard algorithms

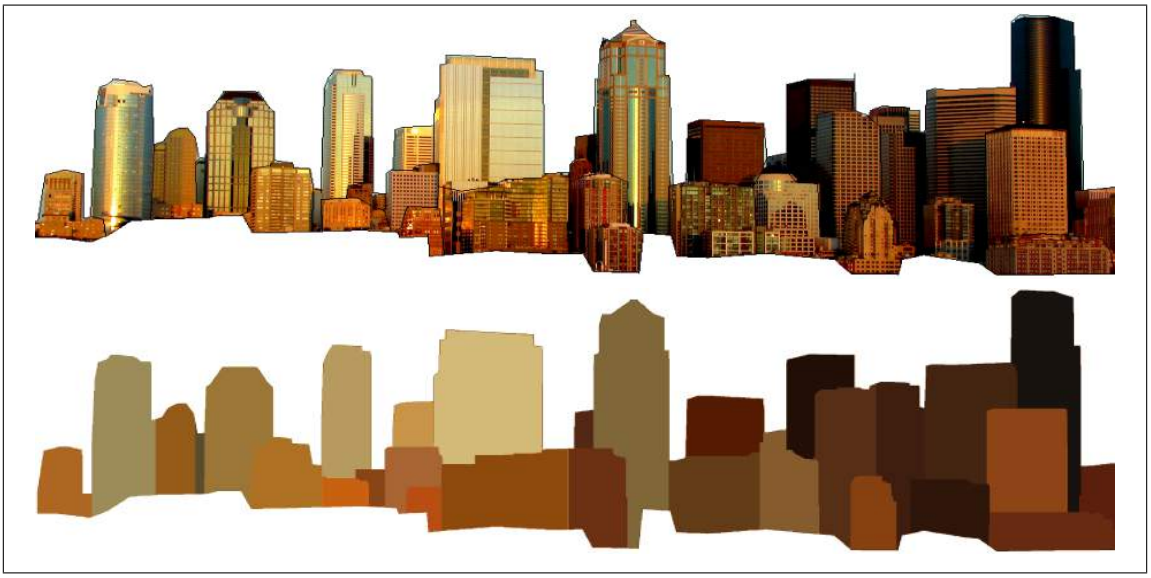

Figure 1.2: Photo of skyline of Miami and its labeling of individual buildings using our method – *refined MRF*.

fail or turn out to be impractical for such large number of labels. One such skyline is shown in Figure 1.7.

However, these images are highly structured – buildings are typically convex objects, roughly rectangular, and all the buildings stand on the ground plane. These constraints can be incorporated as priors for automatic segmentation algorithms.

1.3 Related Work

Pixel labeling is a very general framework that can be used to solve a variety of problems in computer vision. Pixel labeling problems in 2D are usually NP-hard [13], but can be solved efficiently in some special cases [30, 35, 55, 39]. Optimization algorithms for pixel labeling problems is a very active area of research [58].

Semantic segmentation is a relatively new field where pixels are assigned to meaningful labels. Current semantic segmentation algorithms typically do not consider detailed labels. For example, datasets such as PASCAL VOC [22], or MSRC [56] consider labeling of pixels into one of the dozens of labels. In geometric labeling [34], the goal is to roughly label pixels into a number of coarse level orientations such as frontal, left/right-facing, or semantic categories such as ground, sky or porous. Semantic segmentation of tiered scenes [24, 66] consider topological tiers, i.e. vertical and horizontal tiers placed over or beside each other. Our work can be considered in the framework of semantic pixel labeling. There has been significant interest in the recent past to understand the natural outdoor by looking at the buildings, mountains and surroundings [34, 6]. Semantic understanding of the outdoor with additional

Figure 1.3: Figure shows few complex occlusion patterns in the skylines. As can be seen, it is very hard to perceive the buildings individually by humans itself.

Figure 1.4: Figure shows few cases where different facades of the same building display completely different colors and textures.

geometric cues can help in 3D layouts and better visualization. Our work is related to this, except we aim to extract the fine-grained detailed structure of the regions within the image.

Optimization for labeling pixels is a widely studied area of research. Most of the successful methods for semantic segmentation [38, 65] cast it as an energy minimization problem consisting of local and pairwise potentials in Markov Random Fields. Methods like [10, 50] popularized this framework for *binary* interactive segmentation of natural images in an energy minimization framework. Graph cut with α-expansion [13] has emerged as a popular approach to solve multi-label segmentation. The optimization reduces to a sequence of binary labeling problems each of which can be computed using graph-cuts. Although, extremely general, the process can be expensive for large images both in terms of computational complexity and memory. We introduce methods that are an order of magnitude faster and more accurate for labeling skyline images that exploit the spatial structure of the objects. Several leading approaches for semantic segmentation are based on MRFs – a probabilistic model of pixel labels that incorporates local evidence and smoothness of nearby pixels labels. These approaches, though general purpose, do not easily allow the incorporation of higher-order priors such as the overall shape and size of the regions. To this end we propose a *shape-constrained MRF* that allows explicit control

Figure 1.5: Figure shows three cases where other buildings are reflected in the facades of the buildings. In the middle image, whole mini-skyline is reflected in the front facade.

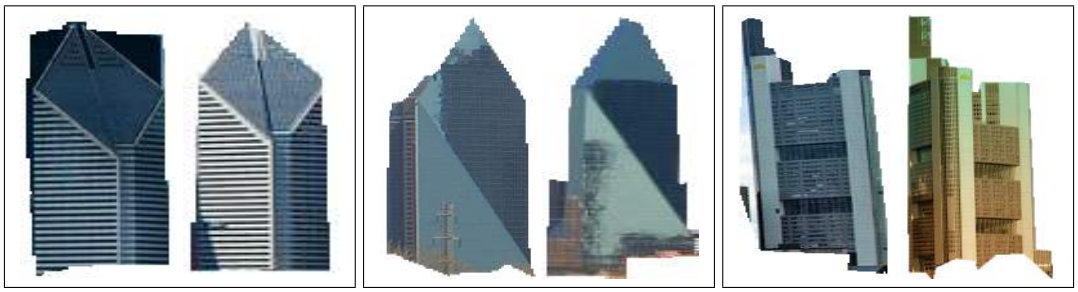

Figure 1.6: Figure shows three cases in which same buildings look completely different in two different skylines in terms of color, structure, texture and viewpoint. For example, in the last image, a single facade is visible in the left building while two facades are visible for the right building.

over the shape, and utilizes the fact the 'tired' structure exhibited by occluding buildings implies that only the upper boundary of an object is 'owned' by each object.

If the domain is 1D or of low treewidth, then optimal labelings can be found using dynamic programming (DP). A simple example involves labeling each scanline of an image independently [54]. [5] uses DP for optimization of a 2D labeling involving binary segmentation where the foreground object is constrained to be a connected region bounded by two horizontal curves. For tiered scenes, Felzenzwalb and Veksler [24] introduced a dynamic programming based solution to obtain a globally optimal solution. However the complexity scales exponentially with the number of labels, hence is impractical for our setting.

Our approach is segmentation with a shape prior, which is a natural approach for getting more robust segmentation results [43, 16]. Segmentation with shape priors usually leads to NP-hard problems. In rare cases [18, 60, 19] when a global minimum can be computed, the shape prior is very restrictive. Zheng *et al.* [66] propose a faster approximation to [24] by decomposing multi-label tiered labeling to a series of binary labeling problems exploiting the topological priors. Our approach takes a similar route, but we incorporate higher order priors such as the overall shape and aspect ratio of each region that cannot be easily expressed as topological priors. Another approach for incorporating topological

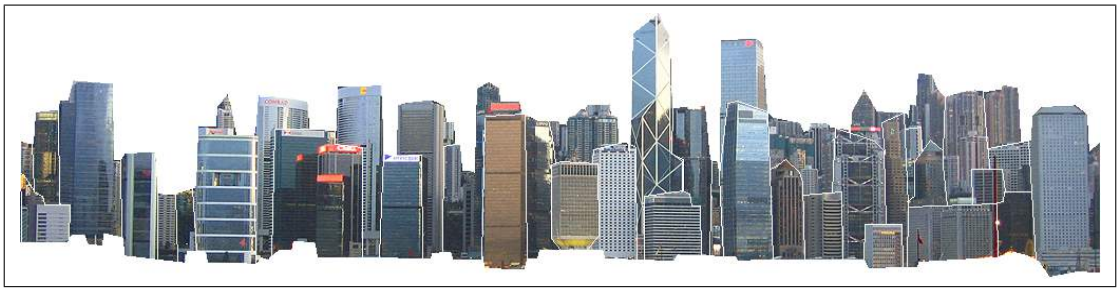

Figure 1.7: Figure shows a skyline with large number of labels.

priors such as inclusion or exclusion is [64], but is also computationally expensive. Freedman and Zhang [28] propose an approach for incorporating shape priors in a MRF formulation, but it assumes that the location of the shape is known making it unsuitable for our case. In our setting both topological and shape priors play a key role, and we show that the combination can improve results without sacrificing speed.

We experimentally evaluate color and texture models for representing the appearance of buildings, and find that texture based Gaussian mixture models can provide significant improvement over color models (Section 3.1). These serve as local evidence (or 'unary' potentials) in a Markov Random Field (MRF) formation of our problem. We study the problem in an automatic as well as interactive setting. In the interactive setting, we assume that we are provided with an image, some 'seed' pixels for each building, and the upper and lower boundaries delineating the region containing all the buildings (as seen in Figure 2.3). In the 'automatic' setting we are only provided with the image and the upper and lower boundaries. On our dataset we found that automatic methods [34] for obtaining such regions work reasonably well, hence we focus on the task of segmenting the individual buildings. Our evaluation metrics and tasks are described in Section 2.2.

We propose several greedy approaches to optimize the proposed MRF formulation (Section 3.3). Similar to approaches like α-expansion [13], we pick one label at a time and update the pixels with respect to that label. However, unlike expansion moves where only background pixels can change to foreground, we allow refinement moves where foreground labels can change to *any* background as well. The 'tiered' structure of the labels allows us to infer the background label *underneath* each foreground pixel. Furthermore, one can order the buildings from front to back based on the y-coordinate of the 'seeds', which serves as a natural order in which we consider region refinement. One such approach called *rectangle MRF* does this via an explicit search over all potential rectangles for each building. This search can be done quickly even on relatively high resolution images using 'integral images'. Another approach called *tiered MRF* does this via a dynamic programming, approximating the upper boundary of a building as a 1D monotonic curve, i.e., the x-coordinates along the curve are monotonic. The former approach allows us control the shape of each region but does a poor job at approximating its upper boundary. Hence we propose a hybrid approach called *refined MRF*, that starts with the solution

of rectangle MRF and refines the upper boundary within the horizontal bounds of the rectangle using dynamic programming. This achieves the best results while being an order of magnitude faster than α-expansion using graph-cuts (Section 2.3).

The automatic setting is suitable for low-level image segmentation methods such as SLIC [1], graph-based segmentation [23], and gPb regions [3]. However, none of these methods explicitly consider shape priors. We show that starting from a set of regions automatically selected from any such segmentation method, one can improve the results using shape priors (Section 4.2). Automatic segmentation methods exploit the local similarity in defining segments and boundaries [23, 1, 3]. While all these methods are quite accurate for generic segmentation, skylines prove to be much harder due to intra-region color and texture variations. We show that our automatic approach can be initialized from any of these unsupervised segmentation techniques and provides a significant boost over them by exploiting shape priors.

1.4 Contributions

Following are the major contributions of the thesis —

1. We introduce a dataset **Skyline-12** consisting of 120 skyline images from the 12 cities all over the world. All the images are manually annotated with addition of meta-data such as initial upper and lower boundaries delineating the buildings region and input seeds provided by user for an interactive setting.

2. We include an analysis and integration of low-level features such as color, texture and shape, very useful for the segmentation of skylines.

3. We propose a fast, accurate and robust method to extract individual buildings of a skyline exploiting 'tiered' structure of the skylines and incorporating rectangular shape prior in MRF formulation. We propose both *interactive* and *automatic* settings of the method.

1.5 Outline of The Thesis

In Chapter 2, we introduce a dataset of skylines form various cities all around the world created specifically for the purpose of skyline segmentation with the necessary metadata. We also define the evaluation measures for both interactive and automatic setting. Futhermore, we define the MRF energy minimization setting and explain some of the baseline "state-of-the-art" methods such as Standard MRF, Grab Cut, etc. that form the baseline for our method. Further, we explain design objectives of our algorithm. Finally, we explain some of the basic techniques such as Hungarian Matching, K-means algorithm for clustering and Integral Images used in the Thesis.

In Chapter 3, we explain the interactive setting in Skyline Segmentation. We start by defining the MRF energy minimization formulation for Skyline Segmentation. We move on to discuss Region Representation using color, texture and spatial distance features subsequently construction of unary potential

for our model. We discuss semantic segmentation for tiered scenes next. Furthermore, we discuss the qualitative results with different parameters of unary potential construction. We then explain our approach for interactive skyline segmentation. Moreover, we explain the shape-constrained MRF methods such as *rectangle MRF*, *tiered MRF*and *refined MRF*. We discuss the results both quantitative and qualitative comparison and Analysis.

In chapter 4, we explain the automatic setting in Skyline Segmentation. We start by explaining baseline low-level automatic segmentation methods such as SLIC, Graph based Segmentation and gPb detector, over which we build our automatic approach. We then describe the Automatic Skyline Segmentation Approach and discuss results. Finally, we explain the automatic method using Geometric Context to obtain initial upper and lower boundaries.

In chapter 5, we explain the extension of *rectangle MRF*, *The Triangle MRF*. Further, we discuss the various directions our work can take in future. Finally, we conclude the thesis by specifying our contributions.

Chapter 2

Dataset and Baselines

2.1 Skyline-12 Dataset

We deal with a very specific problem of segmenting individual buildings in a skyline. As discussed in the Chapter 1, skyline is a typical example of a 'tiered' structure with rectangular shape priors. We are interested in designing an MRF based segmentation algorithm exploiting the 'tiered' structure of the skyline and incorporating rectangular shape priors.

As there are no standard datasets for the purpose, we created a dataset **Skyline-12** consisting 120 images of skylines from 12 cities all around the world - *Chicago, Dallas, Frankfurt, Hong Kong, Miami, New York, Philadelphia, Seattle, Shanghai, Singapore, Tokyo* and *Toronto*. All the images are downloaded from *flickr* having *creative license content*. All the photographs are taken at different times during day. Photographs take at night time were avoided as they are difficult to separate even for human perception due to lack of individual differentiation caused by artificial lightings and overall darkness. All the images consist of dense and complex skylines having numerous buildings placed within multiple horizontal layers. An average skyline in the dataset consists of 35 buildings. All the images in the dataset have high resolution, with an average resolution of 2500×1500. The largest image has resolution of 10476×4092 and the smallest image has resolution of 576×384. Few sample images from each city are shown in Figure 2.1 and Figure 2.2.

In addition to various skyline images, following meta-data is also provided -

1. **Annotations -** All the images are annotated at the pixel level. Annotations are stored as a label matrix of the size of an image with label at each pixel. Upper and lower regions are labeled 1, while all the other regions (buildings) are labeled from 2 to $NumberOfBuildings + 1$.

2. **Upper and lower boundaries -** It is assumed that upper (separating *sky* and *buildings*) and lower (separating *buildings* and *trees/water bodies, etc.*) are provided to the algorithm. Given upper and lower boundary, individual buildings are segmented in middle region. Thus, upper and lower boundaries for each image are provided in the form of $c \times 2$ matrix, where c is number of columns in the image. The first column of the matrix represents the path for the upper boundary while

Figure 2.1: Skyline-12 Dataset. Above shown are sample images from the dataset from the cities — *Chicago, Dallas, Frankfurt, Hong Kong, Miami* and *New York*

the second represents the path for the lower boundary. Thus, element in the 50^{th} the row and 1^{st} column will represent the row index of the upper boundary in 50^{th} column.

3. **Input Seeds -** Interactive setting of the proposed approach needs the user-given seeds for each building in each image. For fair and coherent evaluation of the methods, these seeds are provided with the dataset for all the images. The same seeds are used for all the methods during evaluation. Seeds are stored in the form of cell-array of size $\mathcal{N}$, where $\mathcal{N}$ if number of buildings in an image. n^{th} cell contains seed pixel for the n^{th} building in form of the matrix of the size $ns \times 2$, where ns is number of seed pixels for the building. In n^{th} cell the seeds are stored for the buildings with label n in ground truth and predicted interactive labeling for convenient evaluation as explained in the next section.

In Figure 2.3, few examples with corresponding input seeds, upper and lower boundaries and annotations are shown. Images within each city in the dataset are randomly split into *Training*, *Validation*

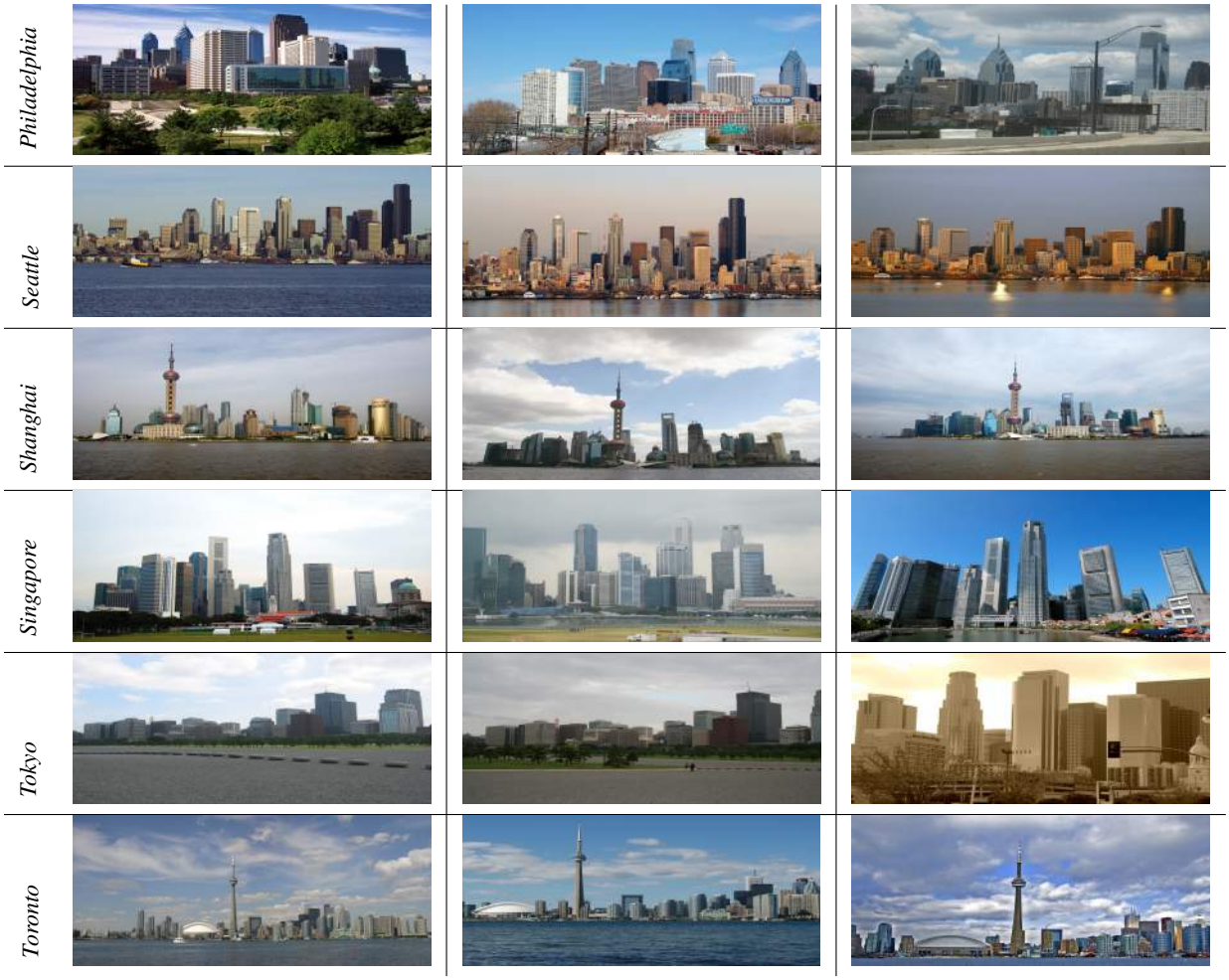

Figure 2.2: Skyline-12 Dataset *Continued*. Above shown are sample images from the dataset from the cities — *Philadelphia, Seattle, Shanghai, Singapore, Tokyo* and *Toronto*

and *Testing* sets of 3, 3 and 4 images respectively. The split results into training and validation sets of 36 images each and testing set of 48 images. One might be concerned about the potential overlap of images from the same city in the training and test set. However most of our modeling is image specific, with the exception of few parameters such as α and β (described in the next chapters) that trade off color and texture weights, the texton dictionary used to estimate texture histograms, as well as the MRF parameters such as λ and τ. These parameters are kept fixed across all images. In an experiment we randomly split the cities into two halves, and using all the images from cities in one half for estimating optimal parameters, while predicting the results on the later half, showed a difference in MAO of about 0.1% compared to using the entire set for training. Hence, we believe that the overlap is not a concern for over-fitting in our approach.

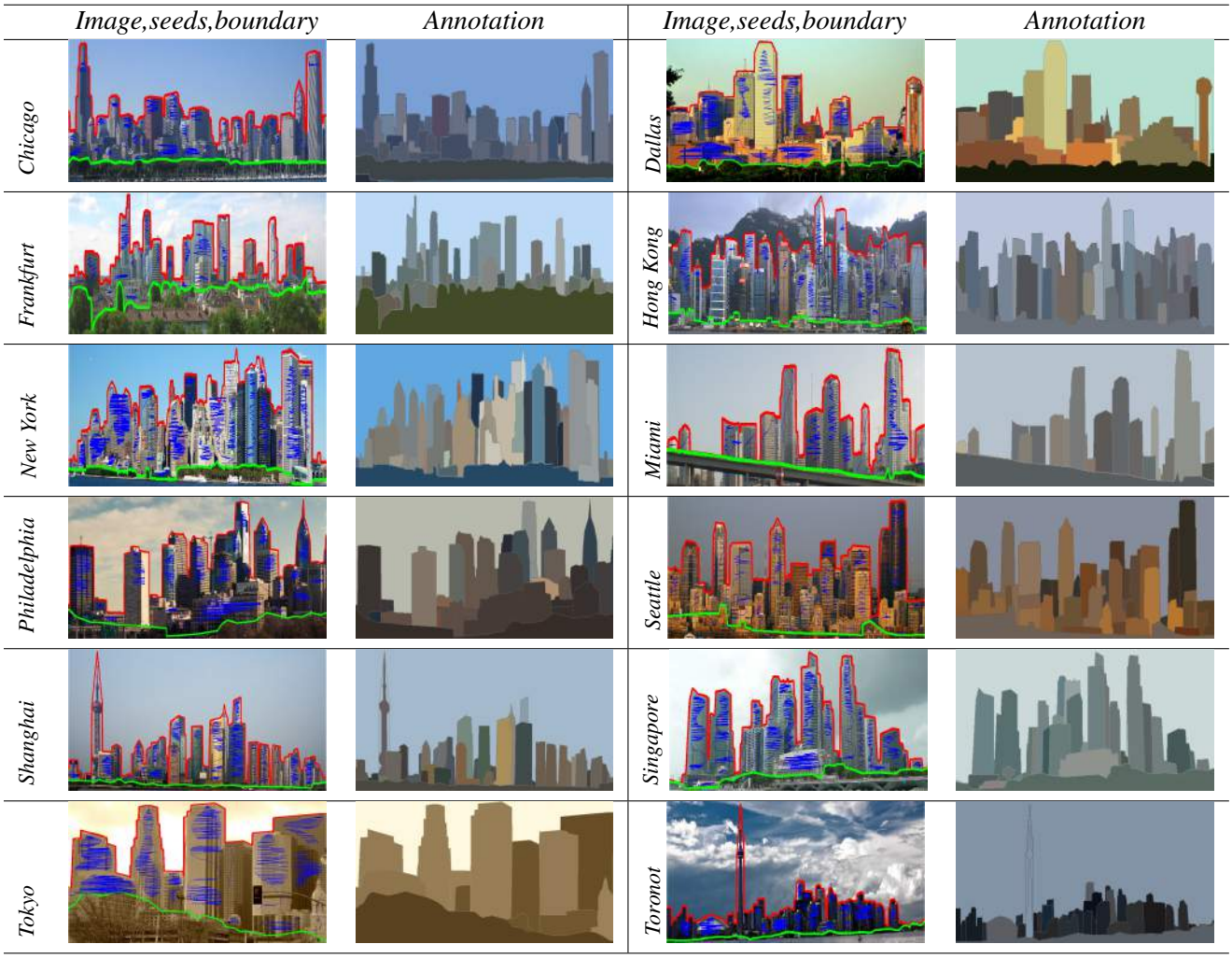

Figure 2.3: Skyline-12 Dataset Metadata. Above shown are images, inputs seeds(*blue*), upper(*red*) and lower(*green*) boundary as well as, annotations (ground truth) for all the 12 cities.

2.2 Evaluation Measures

A common measure to evaluate segmentation methods can be to report the percentage of accurately labeled pixels, which is basically *per-class accuracy*. This measure can be hugely misleading, i.e. labeling all the pixels as one building leads to perfect score for that building (though not for the other buildings), which is clearly incorrect. Biases in different methods (i.e. tendency of over-segmentation or under-segmentation) can lead to misleading high or low accuracies for the individual buildings.

To rectify this problem, evaluation measure based on *intersection of the regions divided by their union* is used. For evaluation of both interactive and automatic settings, evaluation measure is defined as below -

2.2.1 Interactive Setting

In this setting consider an input image I, the upper and lower boundaries delineating the regions containing buildings, thus only the buildings in the middle region are evaluated and, upper and lower regions are not considered while evaluation. The seed pixels S_i are user-provided (or provided with the dataset) for each building b_i, $i \in 1, \ldots, N$. Output of the methods is a labeling of all the pixels in the building region (middle region) into one of the N labels or background. Performance is measured as the average overlap of the segmentations of each buildings b_i as explained below.

Let G_I and P_I denote the ground truth and predicted labeling, and let G_I^i and P_I^i denote the sets of pixels labeled as i. Then, the overlap is computed as the intersection over union of these sets. The $\mathsf{AverageOverlap}(G_\mathcal{I}, P_\mathcal{I})$ is defined as:

$$\mathsf{AverageOverlap}(G_\mathcal{I}, P_\mathcal{I}) = \frac{1}{N} \sum_{i=1}^{N} \frac{G_\mathcal{I}^i \cap P_\mathcal{I}^i}{G_\mathcal{I}^i \cup P_\mathcal{I}^i} \tag{2.1}$$

$\mathsf{AverageOverlap}(G_\mathcal{I}, P_\mathcal{I})$ is averaged over all the images in the test set TS and reported as the single *Mean Average Overlap*(MAO) score for a method P. The similar evaluation measure is used in past as *Best Spatial Support (BSS), Segment Accuracy* and *overlap* in [3], [22] and [47] respectively.

$$MAO_P = \frac{1}{48} \sum_{I \in TS} \mathsf{AverageOverlap}(G_\mathcal{I}, P_\mathcal{I}) \tag{2.2}$$

2.2.2 Automatic Setting

In this setting, we are only given image I and upper and lower boundaries as discussed earlier. As no seeds are provided in this setting, we need to match the regions in the ground truth labeling to the regions in the predicted labeling. The output of a segmentation algorithm is a labeling of each pixel in the image into M regions. We compute similar average overlap scores as before, but first compute a bipartite matching between the ground-truth regions and segmented regions. For all N ground truth regions, we compute the bipartite matching $m : N \to M$ of highest score where the score of matching is given by the intersection over union of the pixels. The average overlap in this setting is defined as:

$$\mathsf{AverageOverlap}(G_\mathcal{I}, P_\mathcal{I}) = \max_{m \in \mathcal{M}} \frac{1}{N} \sum_{i=1}^{N} \frac{G_\mathcal{I}^i \cap P_\mathcal{I}^{m(i)}}{G_\mathcal{I}^i \cup P_\mathcal{I}^{m(i)}} \tag{2.3}$$

Here unassigned ground truth regions get a score of zero. Then, the MAO is computed by averaging $\mathsf{AverageOverlap}(G_\mathcal{I}, P_\mathcal{I})$ as shown in equation 2.2. This measure is similar to the *Best Segment Score* (BSS) criteria used in [47] with the key difference that each segmented region can contribute to only one building, i.e two ground truth segments cannot be matched to the single segment in the predicted labeling and vice versa.

For a given automatic method, we report MAO scores after performing the matching of labels within each image, i.e. the regions in the ground truth are mapped to the regions in the predicted labeling with

maximum overlap. To achieve such matching $m : \mathcal{N} \rightarrow \mathcal{M}$, Hungarian Matching Algorithm described in Section 2.5.1 is used. The cost matrix constructed using region overlaps, i.e. consider a cost matrix C of $\mathcal{N} \times \mathcal{M}$ size, then -

$$C_{ij} = -1 \times G_{\mathcal{I}}^{i} \cap P_{\mathcal{I}}^{j} \tag{2.4}$$

The overlap is multiplied by -1 because, we need to maximize the ovarlap between matched region. As required in Hungarian matching, dummy rows and columns are added to make C a square matrix. Then, the Hungarian Matching finds the $m : \mathcal{N} \rightarrow \mathcal{M}$ such that overlap between matched regions is maximized.

2.3 MRF Formulation of Segmentation

For extracting individual buildings from skyline, we propose an approach based on a Markov Random Field(MRF) formulation, which is based upon the minimization via graph cuts in [13]. In segmentation problem every pixel $p \in P$ must be assigned a label in some finite set L. For the skyline segmentation problem, a set L consists of individual buildings to be segmented. The goal is to find an optimal labeling F where each pixel $p \in P$ is assigned to the label $F_p \in L$ where F is both piecewise smooth and consistent with observed data. This problem can be formulated as shown below -

$$E(F) = E_{data}(F) + E_{smooth}(F) \tag{2.5}$$

Here, term E_{smooth} measures the extent to which F is not piecewise smooth while term E_{data} measures the disagreement between the labeling F and the observed data. The E_{data} is formed as -

$$E_{data} = \sum_{p \in P} D_p(F_p) \tag{2.6}$$

Here, D_p is the value of the *Data term* at pixel p for label F_p measuring how well the label F_p fits pixel p according to observations. The computation of the data term D_p for our setting is described in Section 3.1.

The smoothness term E_{smooth} is formulated as -

$$E_{smooth} = \sum_{p,q \in \mathcal{N}} V_{pq}(F_p, F_q) \tag{2.7}$$

Here, $\mathcal{N}$ is the set of interacting pixels which is set of adjacent pixels in 4-neighborhood in our case. Basically, $V_{pq}(F_p, F_q)$ denotes the cost of smoothness when neighboring pixels p, q are assigned the labels F_p and F_q. Thus, each pair of pixels p, q has its own distinct penalty.

Thus, final energy formulation [44] is as shown -

$$E(F) = \sum_{p \in P} D_p(F_p) + \sum_{p,q \in \mathcal{N}} V_{pq}(F_p, F_q) \tag{2.8}$$

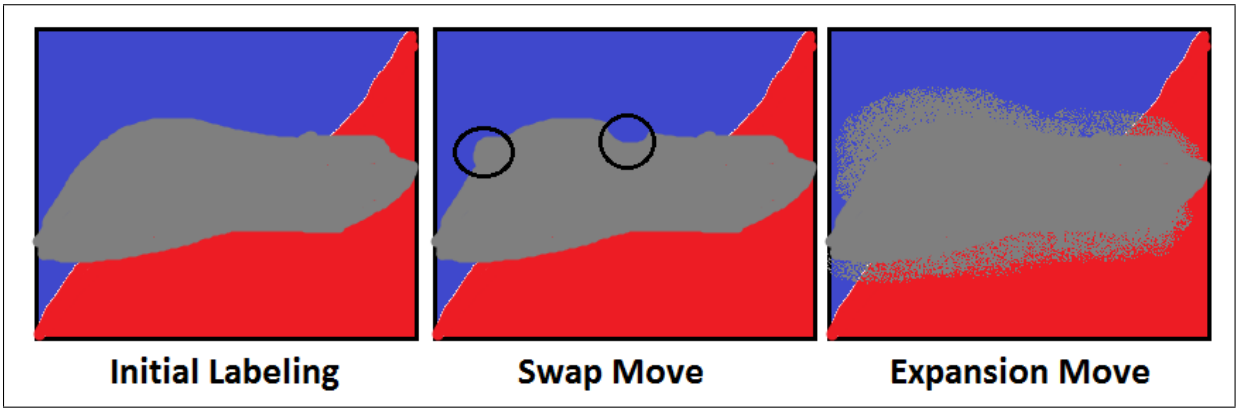

Figure 2.4: $\alpha\beta$-swap and α-expansion moves - Above shown are the large standard moves. In the swap move, few pixels swap their labels between *blue* and *gray* without affecting *red* labels at all. In the expansion move, *gray* label is expanded, changing label of few pixels to *gray* from both *blue* and *red* labels.

Finding global minima of the energy is enormously computationally expensive. Number of possible labelings are exponential in number of pixels $|P|$, which is typically in thousands. Thus, the global energy minimization is NP-hard; therefore it is impossible to find rapid global minima unless $P = NP$. Thus, the standard MRF provides the approximate solution.

2.3.1 Standard MRF

Moves changing the label of a single pixel have been used in Iterated Conditional Modes [9] and Stimulated Annealing [29, 30] but, these methods have very slow rate of convergence. That is why, the larger moves are considered where labels of multiple pixels are changed in a single move. [57, 7] use larger moves in Stimulated Annealing but, instead of selecting the best possible move, they randomly select connected subset of pixels that change their label from α to β. Generally, the algorithms are proposed with respect to two types of large moves to generate the labeling that is local minimum of the energy in Equation 2.8. These moves allow large number of pixels to change their labels simultaneously.
-

1. **$\alpha\beta$-swap :** Given pair of labels α,β, a move from labeling F to labeling F' is called $\alpha\beta - swap$ if $P_l = P'_l$ for any label $l \neq \alpha, \beta$, where P_l and P'_l are set of pixels labeled l in labeling F and F' respectively. That means only difference between labeling F and F' is that some pixels which were labeled α in F are labeled β in F' and some pixels which were labeled β in F are labeled α in F'.

2. **α-expansion :** Given a label α, a move from labeling F to labeling F' is called $\alpha - expansion$, if $P_\alpha \subset P'_\alpha$ and $P_l \subset P'_l$ for label $l \neq \alpha$. That means, this move allows any set of pixels to change their labels to α.

These moves are shown in Figure 2.4. Although, local minimum can be found with respect to both the moves, for skyline segmentation we consider energy minimization only with respect to $\alpha - expansion$.

2.3.1.1 Standard MRF : α-Exapnsion Algorithm

The expansion algorithm finds a local minimum when when expansion moves are allowed. Given a labeling F, there are exponential number of expansion moves possible. [51, 36, 12] used graph cuts to find the exact global minimum of a certain type of energy functions only for 1D labels. The standard MRF uses efficient graph cut (min-cut/max-flow) optimizations to find the optimal $\alpha - expansion$ given a labeling F.

Algorithm 1 α**-expansion Algorithm for Skyline Parsing**

Require: data D, pairwise V

1: Initialize, initial labeling F such that $F_p = min_{l \in L} D_p(l)$
2: **for** iter := 1 **to** K **do**
3: **for** Each building $\alpha \in L$ **do**
4: Find $F' = argmin E(F')$ among F' within one α-expansion of F using Max flow/Min Cut.
5: **if** $E(F') < E(F)$ **then**
6: Set $F = F'$
7: **end if**
8: **end for**
9: **end for**
10: Return F

This approach can be said a variant of "fastest descent" technique. The standard MRF approach for parsing skylines is outlined in Algorithm 1. The efficient graph-based methods to find the optimal α-expansion move (Step 5) given a labeling F is described in Section 2.3.1.2. This is the key step of the algorithm. The Step 3-10 is called an *Iteration*. In each iteration, the algorithm finds an optimal expansion move for every label in random order. These algorithms are guaranteed to terminate in a finite number of iterations. In fact, under the assumptions that D_p and V in Equation 2.8 are constants independent of the image size $P = m \times n$, the convergence can be easily proved in $\mathcal{O}(P)$ iterations [59]. Although for all practical purposes, we use $2 * |L = numBuildings|$ iterations, i.e. $K = 2$.

2.3.1.2 Optimal Expansion Move using Graph Cuts

Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be a weighted graph with two distinguished terminals. A *cut* $\mathcal{C} \in \mathcal{E}$ is a set of edges such that the terminals are separated in the induced graph $\mathcal{G}(\mathcal{C}) =< \mathcal{V}, \mathcal{E} - \mathcal{C} >$ and no proper subset of $\mathcal{C}$ separates the terminals in $\mathcal{G}(\mathcal{C})$. The cost of $\mathcal{C}$ is the sum of its edge weights denoted by $|\mathcal{C}|$. The *Minimum Cut* problem is to find the cheapest cut among all cuts separating the terminals.

Given an input labeling F and a label α, the goal is to find the labeling $\hat{F}$ that minimizes E in Equation 2.8 over all the labelings within one α-expansion of F. The technique is based on computing a

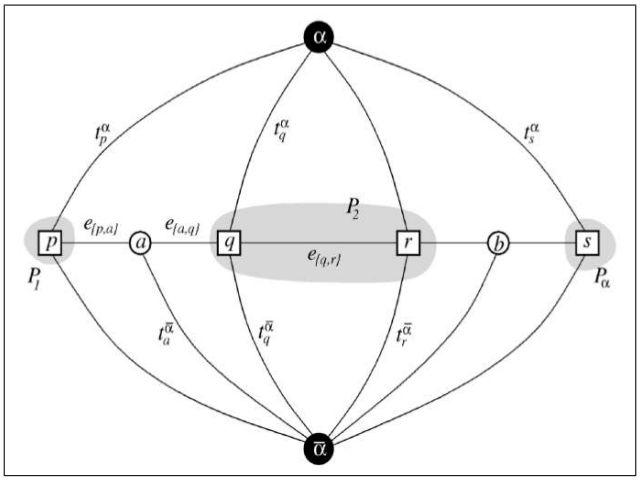

Figure 2.5: Graph construction for α-expansion - Above shown is a graph for hypothetical 1D image with terminals α and $\bar{\alpha}$ and corresponding nodes and edges.

labeling corresponding to minimum cut of graph $\mathcal{G}_\alpha =< \mathcal{V}_\alpha, \mathcal{E}_\alpha >$. The structure of the graph is determined by the current labeling f and label α. The structure of the graph $\mathcal{G}_\alpha$ is shown in the Figure 2.5 for 1-D image. Set of vertices include label α and $\bar{\alpha}$, all the pixels $p \in P$ as well as, additionally for each pair of neighboring pixels $\{p, q\} \in \mathcal{N}$ such that $f_p \neq f_q$, an auxiliary node $a_{\{p,q\}}$ is introduced at the boundaries between sets P_l for $l \in \mathcal{L}$. Thus, the set of vertices is -

$$\mathcal{V}_\alpha = \{\alpha, \bar{\alpha}, \bigcup_{\{p,q\}\in\mathcal{N}, F_p \neq F_q} a_{pq}\}$$

Each pixel $p \in P$ is connected to terminals α and $\bar{\alpha}$ by *t*-links t_p^α and $t_p^{\bar{\alpha}}$ respectively. Each pair of neighboring pixels $\{p, q\} \in \mathcal{N}$ such that $f_p \neq f_q$ is connected by an *n*-link e_{pq}, additionally for each such pair a triplet $\mathcal{E}_{pq} = \{e_{pa}, e_{aq}, t_a^{\bar{\alpha}}\}$ is created. Then, the set of edges are -

$$\mathcal{E}_\alpha = \{\bigcup_{p\in P} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\{p,q\}\in\mathcal{N}, F_p \neq F_q} \mathcal{E}_{pq}, \bigcup_{\{p,q\}\in\mathcal{N}, F_p \neq F_q} e_{pq}\}$$

The weights assigned to the edges are as shown in Table 2.1.

Any cut $\mathcal{C}$ on $\mathcal{G}_\alpha$ must include exactly one t-link for any pixel $p \in P$: If neither t-link were in $\mathcal{C}$, there would be a path between terminals α and $\bar{\alpha}$; while if both the t-links were cut, then a proper subset of $\mathcal{C}$ would be a cut. This defines a natural labeling $f^{\mathcal{C}}$ corresponding to cut $\mathcal{C}$ on graph $\mathcal{G}_\alpha$. Formally -

$$f_p{}^{\mathcal{C}} = \begin{cases} \alpha & \text{if } t_p{}^\alpha \in \mathcal{C} \\ f_p & \text{if } t_p{}^{\bar{\alpha}} \in \mathcal{C} \end{cases}$$

In other words, a pixel p is assigned to label α if cut $\mathcal{C}$ separates p from terminal α, while it retains its old label if a cut $\mathcal{C}$ separates p from $\bar{\alpha}$. For $p \notin P_\alpha$, $\bar{\alpha}$ represents labels in the initial labeling.

Then, the following statements can be proven easily-

edge	weight	for
$t_p^{\bar{\alpha}}$	∞	$p \in P_\alpha$
$t_p^{\bar{\alpha}}$	$D_p(F_p)$	$p \notin P_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in P$
e_{pa}	$V(F_p, \alpha)$	$\{p, q\} \in \mathcal{N}, F_p \neq F_q$
e_{aq}	$V(\alpha, F_q)$	
$t_a^{\bar{\alpha}}$	$V(F_p, F_q)$	
e_{pq}	$V(F_p, \alpha)$	$\{p, q\} \in \mathcal{N}, F_p = F_q$

Table 2.1: In above table, weights assigned for each type of edges and pixels are shown in the Graph for α-expansion.

1. A labeling $F^{\mathcal{C}}$ corresponding to a cut $\mathcal{C}$ on $\mathcal{G}_\alpha$ is one α-expansion away from the initial labeling F.

2. If $\{p, q\} \in \mathcal{N}$ and $F_p \neq F_q$ then minimum cut $\mathcal{C}$ on $\mathcal{G}_\alpha$ satisfies -

   (a) If ${t_p}^{\alpha}, {t_q}^{\alpha} \in \mathcal{C}$ then $\mathcal{C} \bigcup \mathcal{E}_{p,q} = \phi$

   (b) If ${t_p}^{\bar{\alpha}}, {t_q}^{\bar{\alpha}} \in \mathcal{C}$ then $\mathcal{C} \bigcup \mathcal{E}_{p,q} = {t_a}^{\bar{\alpha}}$

   (c) If ${t_p}^{\bar{\alpha}}, {t_q}^{\alpha} \in \mathcal{C}$ then $\mathcal{C} \bigcup \mathcal{E}_{p,q} = e_{pa}$

   (d) If ${t_q}^{\bar{\alpha}}, {t_p}^{\alpha} \in \mathcal{C}$ then $\mathcal{C} \bigcup \mathcal{E}_{p,q} = e_{pq}$

   These properties are shown in Figure 2.6.

3. Let $\mathcal{G}_\alpha$ be constructed as above given F and α. Then, there is a one to one correspondence between elementary cuts on $\mathcal{G}_\alpha$ and labelings within one α-expansion of F. Moreover, for any elementary cut $\mathcal{C}$, we have $|\mathcal{C}| = E(F^{\mathcal{C}})$.

Thus, the optimal expansion move can be found using standard Min Cut/Max flow algorithm. Minimum cuts can be efficiently found by standard combinatorial algorithms with different low-order polynomial complexities [2]. For example, a minimum cut can be found by computing the maximum flow between the terminals, according to a theorem due to Ford and Fulkerson [25]. In our experiments, we use of a new max-flow algorithm that has the best speed on the graphs over many modern algorithms [11].

In the figure 2.7 the standard MRF output for a sample skyline is shown. The qualitative comparison between standard MRF output and our method is done in Section 3.4.

2.3.2 GrabCut

GrabCut [50] is a method to accurately solve the problem of foreground/background segmentation. GrabCut extends the Graph Cut method explained in the previous section and develops an iterative version of the optimization. The method proceeds as below -

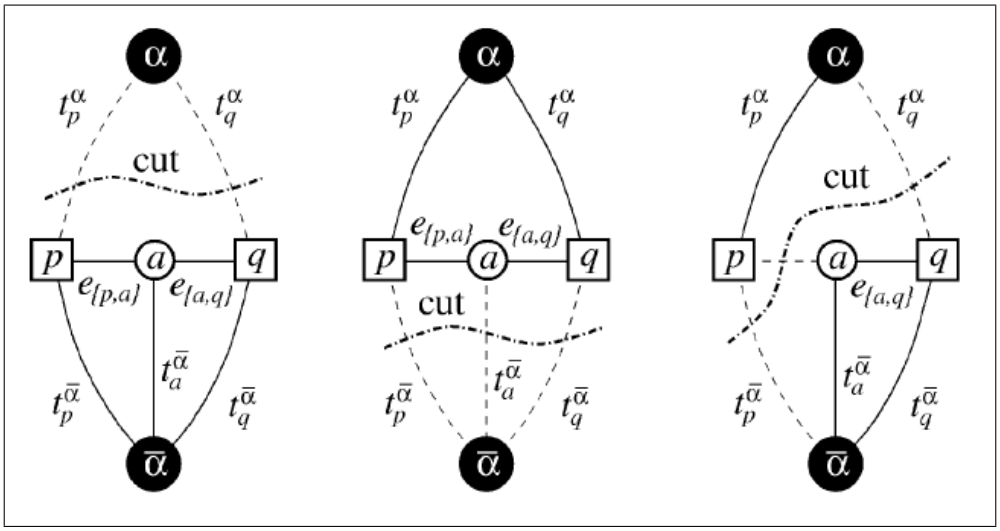

Figure 2.6: Graph cut properties - Above shown are the properties of minimum cut $\mathcal{C}$ on graph $\mathcal{G}_\alpha$. Property (a) is shown in leftmost graph, (b) is shown in the middle graph while Properties (c) and (d) are shown in the rightmost graph.

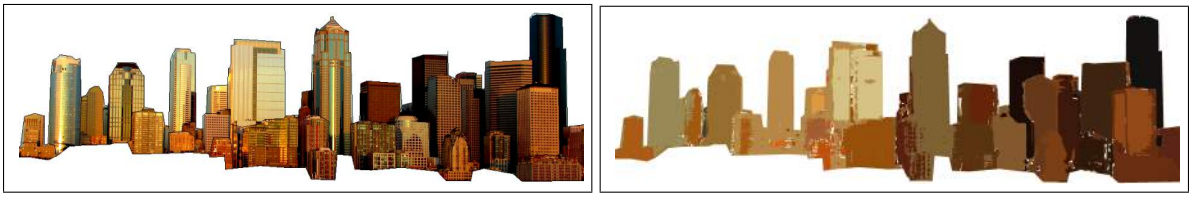

Figure 2.7: Standard MRF - In the *left*, shown is the example skyline while in the *right* a standard MRF output using α-expansion via Graph Cuts is shown.

1. In this method, GMM components are calculated from initial user input as explained in Section 3.1.1 and each pixel is assigned to a GMM component corresponding to minimum unary potential.
2. The corresponding GMM parameters such as mean($\mu(k_n)$), covariance($\Sigma(k_n)$) and the weights($\pi(k_n)$) are learned from the image data for each component.
3. The segmentation is estimated using min cut (refer Section 2.3.1.2).
4. Step 1-3 are repeated until convergence.

The method guarantees proper convergence due to iterative energy minimization. The iterative process also allows user editing and refining. The method is impractical for our purpose as we need to segment about 30-50 labels for image and this method is suitable for foreground/background segmentation. Iterative procedure until convergence for each label involving a min cut in each iteration is highly inefficient

for our purpose. We present a much faster, efficient and accurate method which takes into consideration multiple features such as color, texture and shape.

2.4 Design Objectives

MRF segmentation discussed in previous Section 2.3 segments the image into individual buildings (labels) such that the energy of the labeling (Equation 2.8) is approximately minimized. The algorithm becomes inefficient for skyline segmentation due to multiple reasons -

1. For an image of size $m \times n$, standard MRF takes $\mathcal{O}(m^3 n^3)$ computations in the worst case scenario for each label using α-expansion.

2. Incorporating shape priors and higher order priors into MRF formulation becomes very complex. Methods that have incorporated shape priors and higher order priors into MRF formulation in the past ([18, 60, 19, 66, 64, 28]) are either too restrictive or computationally expensive or require prior information (Section 1.3). Thus, it is impractical to incorporate rectangular shape priors as well as, higher order priors such as height, width and aspect ratio into segmentation of skylines using the methods available.

3. The pixel-wise classification, i.e. segmentation of the skyline is often not enough for the applications where additional semantic information such as order/depth of each building is required.

The main reason behind these failures of Standard MRF for segmenting the skylines is because the algorithm does not consider the overall structure of the skylines and shapes of individual buildings. We propose an effective and efficient algorithm to segment skylines into individual buildings via exploiting the unique 'tiered' structure of the skyline as well as, incorporating the general rectangular shapes of the building into MRF formulation.

A lot of scenarios exhibit 'tiered' structure, e.g. cars in parking lot, pine trees in forest, oranges arranged in a heap and buildings in a skyline. The skylines are 'tiered' in a sense that objects of the similar shape (buildings) occlude one another forming the unique and complex 'tiered' structure. The Dynamic Programming methods [66, 24] propose segmentation methods optimizing MRF energy of the tiered labeling exploiting the 'tiered' structure of the setting. Such methods fail for the complex occlusion patterns in the skyline as well as, large number of labels. The 'tiered' structure of the skylines not only provides the global structural information but also helps deducing the local ordering of the buildings in the z-dimension, i.e. the building is farther to camera than the one occluding it, the building closer to the bottom boundary is closer to camera than others, etc. This information about the tiered structure and the ordering of the buildings helps to split the overall labeling problem into bunch of binary segmentation problems, i.e. each building can be segmented iteratively in the order from bottom to top. We also propose a simplified tiered labeling problem using Dynamic Programming based on the structure and ordering of the buildings in Section 3.3.2.

While 'tiered' structure of the skyline helps us simplifying the overall labeling problem into ordered binary segmentation problems, incorporating shape priors and higher order priors still remains a computationally and conceptually complex problem without considering the general rectangular shape of the buildings in a skyline. We assume the buildings of the skylines are rectangular, the hard assumption being the buildings are strictly rectangular (Section 3.3.1) and the soft assumption being the left and right boundaries of the buildings are rectangular while upper boundary is an x-monotonic curve (Section 3.3.3), e.g. towers. Instead of searching for the optimal labeling in the labeling space like in Standard MRF, we search over parametric rectangular shape family. We can easily incorporate the higher order priors such as height, width and aspect ratio into MRF formulation via restricting the shape parameters.

2.5 Other Related Concepts

The various tools and techniques subsequently used in the method are described below -

2.5.1 Hungarian Matching

Hungarian matching [40] is used to solve an assignment problem where n jobs are need to be assigned to n nodes such that cumulative cost is minimum. The Hungarian Matching is based on following theorem -

If a number is added to or subtracted from all of the entries of any one row or column of a cost matrix, then on optimal assignment for the resulting cost matrix is also an optimal assignment for the original cost matrix.

The following algorithm applies above theorem to a given $n \times n$ cost matrix to find an optimal assignment -

1. Subtract the smallest entry in each row from all the entries of its row.
2. Subtract the smallest entry in each column from all the entries of its column.
3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.
4. *Test for Optimality* -
   (a) If the minimum number of covering lines is n, an optimal assignment of zeros is possible and optimal assignment is done.
   (b) If the minimum number of covering lines is less than n, an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5.
5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

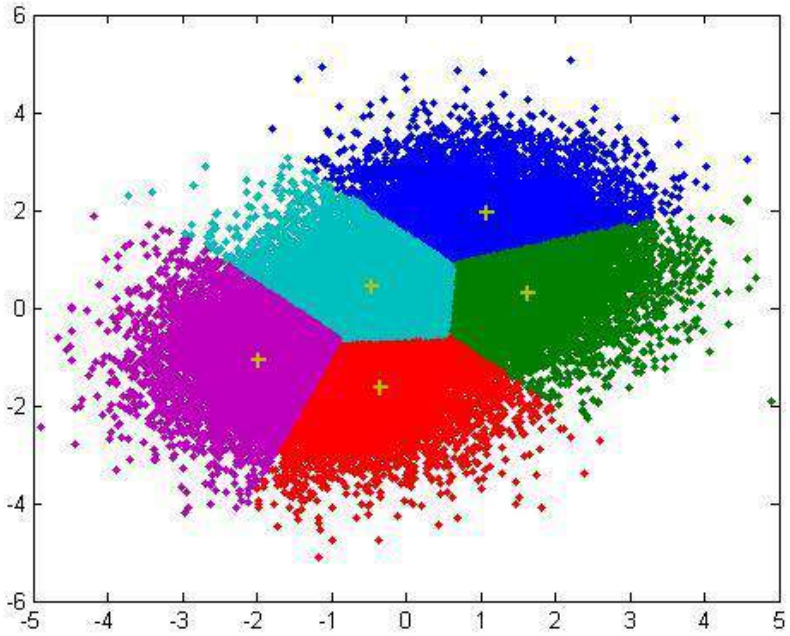

Figure 2.8: k-means Algorithm - Above shown in the image are different clusters represented by different colors, data points assigned to the nearest cluster center shown by symbol **+**.

2.5.2 K-means Clustering

k-means [45] is an unsupervised clustering algorithm, used in color and texture modeling in Section 3.1.1 and 3.1.2 as well as automatic superpixels generation algorithm such as SLIC in Section 4.1.1. Given N data points of d dimensions each and k number of clusters, k-means clusters the data into k clusters so as to minimize *within-cluster* sum of squares.

Let $\{x_1, \ldots, x_N\}$ be the set of d dimensional real vectors, k-means clustering aims to partition the N observations into k sets ($k \leq N$) $S = (S_1, S_2, \ldots, S_k)$ so as to minimize the *within-cluster* sum of squares -

$$S = argmin_S \sum_{i=1}^{k} ||x_j - \mu_i||^2 \tag{2.9}$$

where, $\mu_i = \frac{\sum_{x_j \in S_i} x_j}{|S_i|}$ is mean of all the data points in cluster S_i and $|S_i|$ is the number of data points in the cluster S_i.

The k-means algorithm uses an iterative refinement technique to solve the optimization problem. The iterative procedure is also referred to as Lloyds algorithm. It proceeds in three steps -

1. **Initialization Step -** The algorithm starts with randomly initializing the means to $\mu_1, \mu_2, \ldots, \mu_k$.

2. **Assignment Step -** In the assignment step, each data point is assigned to its nearest mean according to Euclidean distance.

$$S_i = \{x_p : ||x_p - \mu_i|| \leq ||x_o - \mu_j|| \forall 1 \leq j \leq k\} \tag{2.10}$$

3. **Update Step -** During the update step, the mean of each cluster is recomputed after the new assignments from the previous step.

$$\mu_i = \frac{\sum_{x_j \in S_i} x_j}{|S_i|} \tag{2.11}$$

The algorithm converges when in the assignment step, the clusters do not change. There is no guarantee that the algorithm will converge to the global optimum, and the result depends on the initialization of the cluster means. One common practice is to randomly choose k points from the data points as the initial cluster means, and run it multiple times with different initializations. There are other variants of initializations in the literature, for example k-means++ [4], which avoids the poor clusterings found by the standard k-means algorithm..

2.5.3 Integral Images

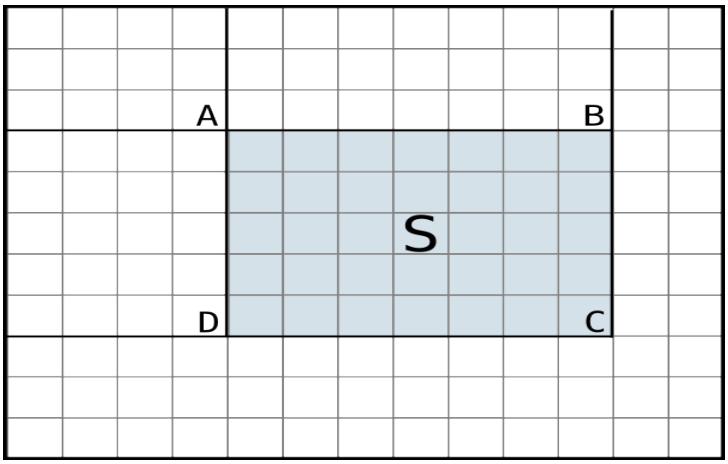

Figure 2.9: To find a sum of the rectangle $ABCD$ as shown in the figure, compute $I(C) + I(A) - I(B) - I(D)$.

Integral Image (or Summed area table or Cumulative Sums) [17, 62] is a data structure and algorithm for quickly and efficiently generating sum of values in rectangular subset of a grid or image or matrix.

The value at any pixel (x, y) in an integral image is the sum of the pixel values above and to the left of (x, y), inclusive.

$$I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{2.12}$$

where, $I(x, y)$ is value of integral image at (x, y) and $i(x, y)$ is pixel value at (x, y) in an original image.

Moreover, an integral image can be calculated very efficiently with a single pass over the image as described in below equation -

$$I(x, y) = i(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1) \tag{2.13}$$

Once an integral image is computed, the task of evaluating the sum within any rectangle is matter of constant computation with just four array references. Consider the Figure 2.9, having $A = (x_0, y_1)$, $B = (x_1, y_1)$, $C = (x_1, x_0)$ and $D = (x_0, y_0)$. Then, the sum of $i(x, y)$ over the rectangle spanned by A, B, C and D is -

$$\sum_{x_0 < x \leq x_1, y_0 < y \leq y_1} i(x, y) = I(C) + I(A) - I(B) - I(D) \tag{2.14}$$

The small variant of the integral images are *columnwise* or *rowwise* cumulative sums. Instead of computing and storing the the sums of the entire rectangle, the sums of the elements in the row or column upto the point are stored. For example, in *Columnwise* cumulative sums -

$$I(x, y) = i(x, y) + I(x, y - 1) \tag{2.15}$$

and in *rowwise* cumulative sums -

$$I(x, y) = i(x, y) + I(x - 1, y) \tag{2.16}$$

The further variant of the Rectangular integral images are Triangular integral images which will be explained in Section 5.1.1.

Chapter 3

Interactive Segmentation

We formulate the overall labeling problem as an energy minimization problem as described in 2.3. For set of pixels P and set of possible labels L, the energy of a labeling $F : P \rightarrow L$, is defined as

$$E(F) = \sum_{p \in P} D_p(F_p) + \sum_{p,q \in N} V_{pq}(F_p, F_q) \tag{3.1}$$

Where, the smoothness term (or pairwise potential) is defined as below.

$$V_{pq}(a, b) = \lambda \exp\left(-\gamma \left(I_p - I_q\right)^2\right) \cdot \mathcal{1}(a \neq b) \tag{3.2}$$

Here, I_p denotes the image intensity at pixel p. The smoothness term encourages coherence in regions of similar colors-levels. The power of the exponent $(I_p - I_q)^2$ is often called as contrast term and is computed using Euclidean distance in color space. The optimal labeling can be obtained by $F^* = arg\min_f E(f)$.

The unary term D_p measures the color and texture similarity of the pixel compared to the color and texture models estimated from a set of seed pixels (Section 3.1). In the interactive setting these seeds are provided as input, as described earlier in Section 2.1. In the automatic setting, seeds are initialized from unsupervised low-level segmentation algorithms as explained later in Section 4.2. In the next section, construction of unary potential term D_p is described in detail.

3.1 Region Representation

Unary potential (or Data term) represents the disagreement between labeling F and observed data. In other words, $D_p(b)$ measures the cost of labeling pixel p with building (label) b with respect to observed data. In our modeling, various features such as color, texture and spatial distance are used to compute robust region representation.

We start with initial SLIC superpixel segmentation [1] with $regionSize = 30$ and $regularizer = 0.001$. Before obtaining the SLIC superpixels, the image is converted to CIE Lab [63] color space from RGB color space. The reason behind converting to Lab color space is that it is designed to uniform, i.e. perceptual "closeness" that humans observe corresponds to the Euclidean distance in the Lab

color space. As shown in Figure 3.1, “L” component corresponds to lightness from white to black, “a” component corresponds to red-greenness and “b” component corresponds to yellow-blueness.

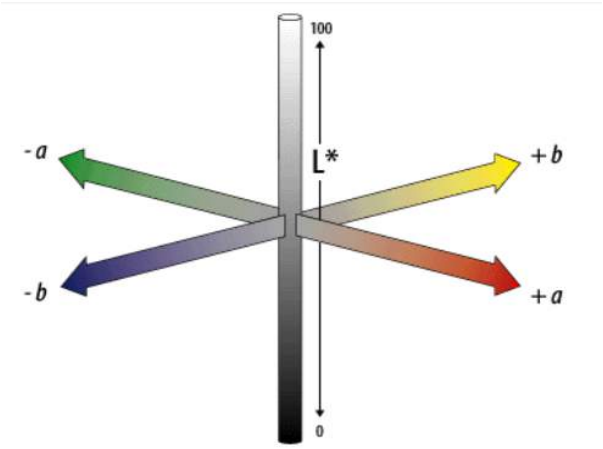

Figure 3.1: Lab color space - “L” component corresponds to lightness from white to black, “a” component corresponds to red-greenness and “b” component corresponds to yellow-blueness

The superpixels that contain the seed pixels are assigned to the majority label, i.e. if a superpixel contains a seeds from two different buildings, then it is assigned a label which has a majority seeds in the superpixel. To assign the affinity of a pixel to a region (i.e. value of the unary potential at the pixel), we use color and texture features as explained below.

3.1.1 Color Modeling

Color is an obvious and the first observation distinguishing one building from another. Thus, while constructing unary potential, we make use of the color information provided by the image.

Consider I_p as color intensity at pixel $p \in P$ in CIE Lab color space [63]. Unlike texture modeling, it is highly impractical to construct adequate color space histograms, as there are as many as 255^3 possible beans. Thus, color data is modeled using GMMs using, a model already used for soft segmentation in [50, 52, 15]. While in these methods 2 GMMs are generated one for each of the background and foreground, for skyline parsing as many as B GMMs are used, one for each building (label). Each GMM is taken to be a full-covariance Gaussian Mixture with K components (In this case, $K = 5$). In order to generate robustness and tractability, in the optimization framework, an additional vector $k = \{k_1, \ldots k_N\}$ is introduced, with $k_n \in \{1, \ldots, K\}$. Thus, to each pixel p a unique GMM component $C_p(b)$ is assigned, a component from b^{th} building (label). $C_p(b, k)$ represents the contribution of color model towards the unary potential at a pixel p in the k^{th} cluster for building (label) b.

For each label, set of foreground (here, foreground means the label into consideration) color vectors P_b are acquired using seed pixels, i.e. all the pixels belonging to superpixels labeled as current foreground are foreground colors. This set of foreground colors vectors is partitioned into K clusters using k-means algorithm. For each cluster k and the foreground label b, mean color $\mu(b, k)$, the covariance

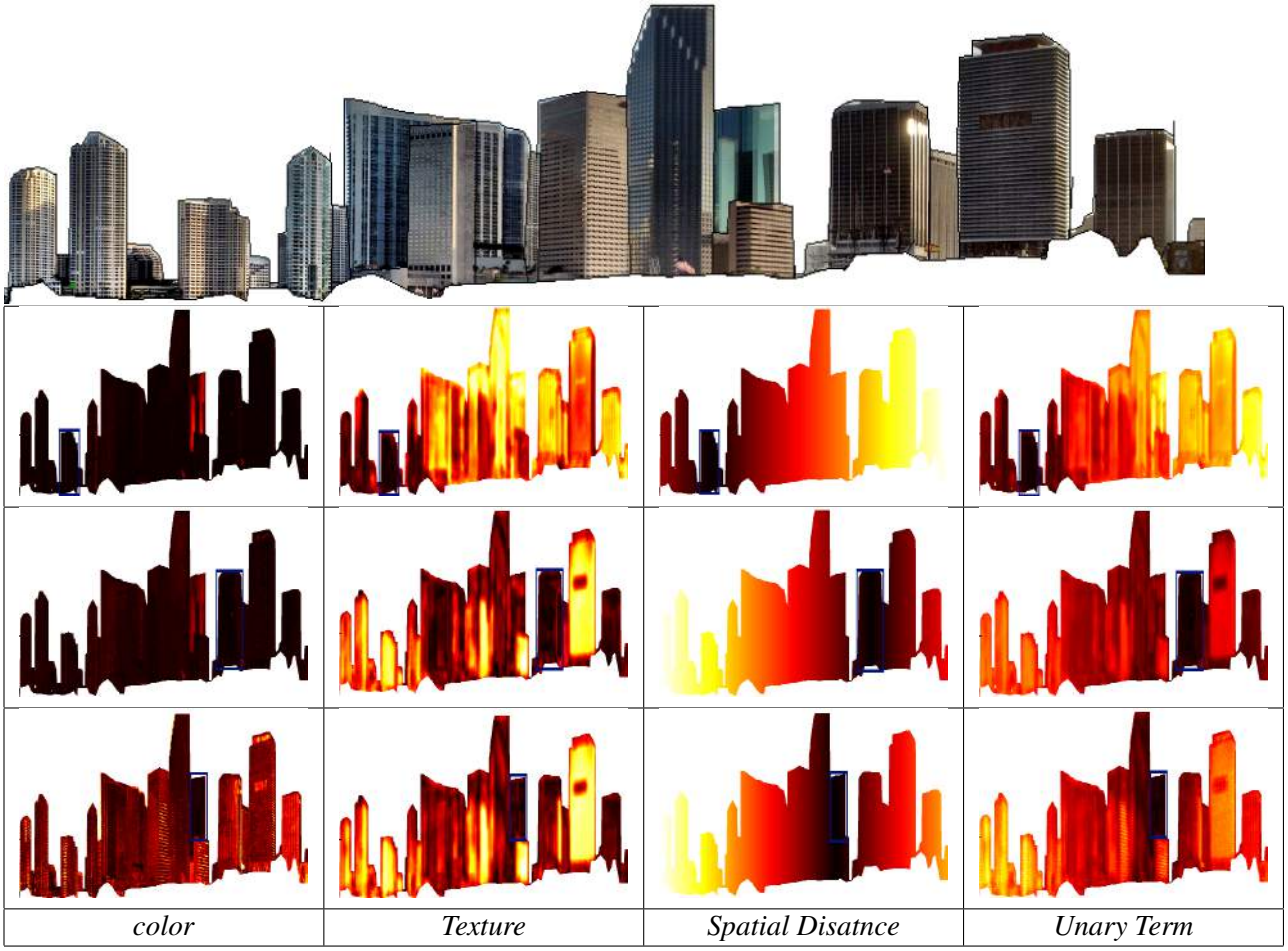

Figure 3.2: Unary Potential. Above shown are unary potentials with corresponding color, texture and spatial distance component for 3 buildings from the image shown in the first row. The corresponding buildings are outlined by *blue* rectangle. The darker region in the images correspond to lower cost (unary potential). Thus, all the corresponding buildings are darker than the rest of the skyline.

matrix $\Sigma(b,k)$ and the mixture weighting coefficient for the cluster $\pi(b,k)$ is calculated as shown below:

$$\mu(b,k) = \frac{\sum_{p \in P_b} I_p}{|P_b|} \tag{3.3}$$

$$\Sigma(b,k) = \sum_{p \in P_b} (I_p - \mu(b,k))(I_p - \mu(b,k))^T \tag{3.4}$$

$$\pi(b,k) = \frac{|P_b|}{\sum_{i \in B} |P_i|} \tag{3.5}$$

Then, the log likelihoods of the foreground can be modeled as being derived from an oriented elliptical Gaussian distribution using covariance matrix $\Sigma(b,k)$. Thus -

$$C_p(b,k) = -\log f(p|b,k) - \log \pi(b,k) \tag{3.6}$$

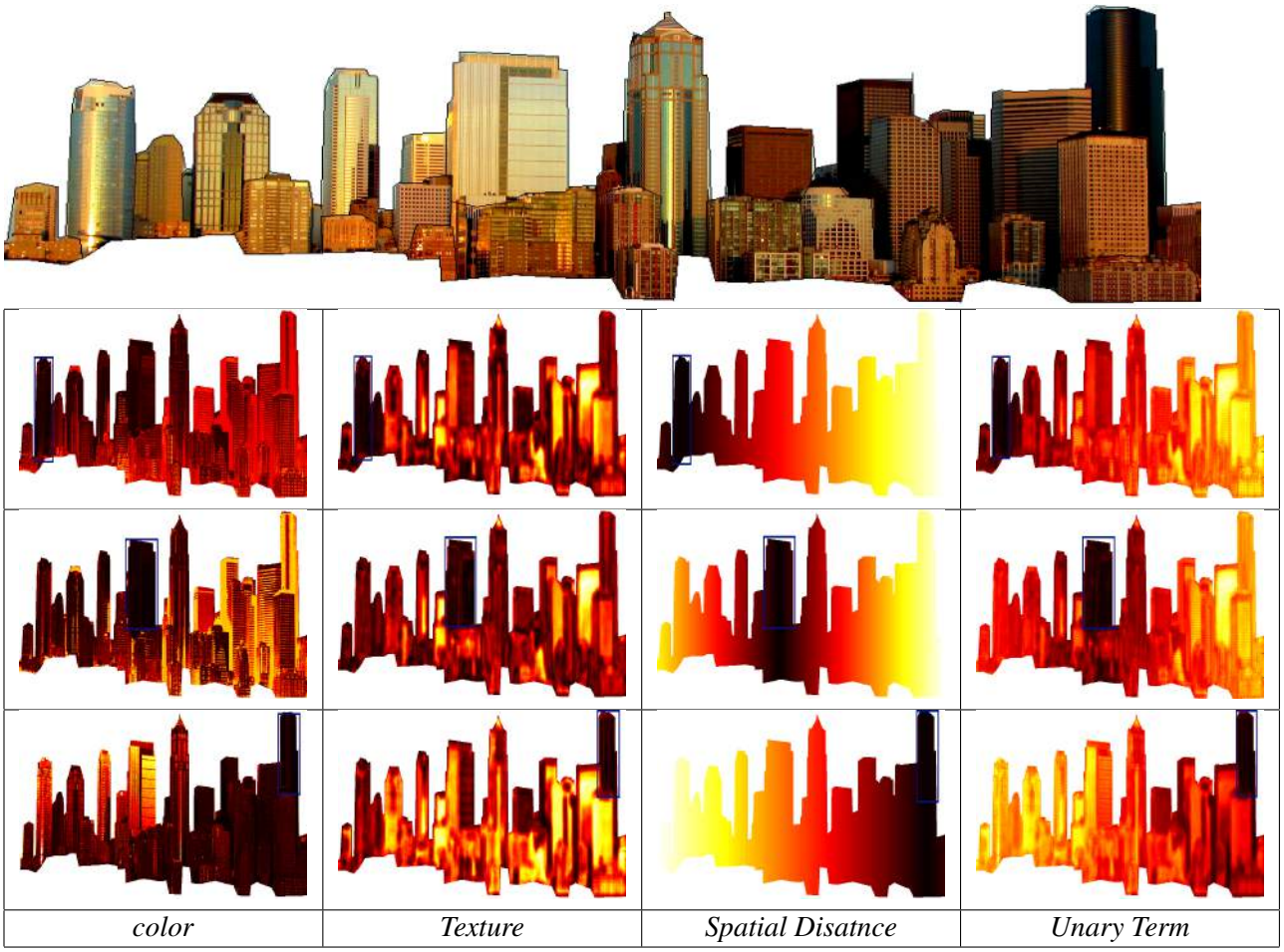

Figure 3.3: Unary Potential Continued. Above shown are unary potentials with corresponding color, texture and spatial distance component for 3 buildings from the image shown in the first row. The corresponding buildings are outlined by *blue* rectangle. The darker region in the images correspond to lower cost (unary potential). Thus, all the corresponding buildings are darker than the rest of the skyline.

Here, $f(.)$ is Gaussian probability distribution and $\pi(.)$ are mixture weighing coefficients. Then -

$$C_p(b,k) = -\log \pi(b,k) + \frac{1}{2}\log det\Sigma(b,k) + \frac{1}{2}[I_p - \mu(b,k)]\Sigma(b,k)^{-1}[I_p - \mu(b,k)]^T \tag{3.7}$$

Therefore, the parameters of the model are -

$$\theta = \{\pi(b,k), \mu(b,k), \Sigma(b,k), b = 1,\ldots,B, k = 1,\ldots,K\} \tag{3.8}$$

i.e. the weights π, means μ and covariances Σ of the $B \times K$ gaussian components for the distributions for all the buildings (labels) and $C_p(b,k)$ represents the color contribution towards the unary potential.

The color potentials for few buildings in two skylines are shown in figures 3.2 and 3.3.

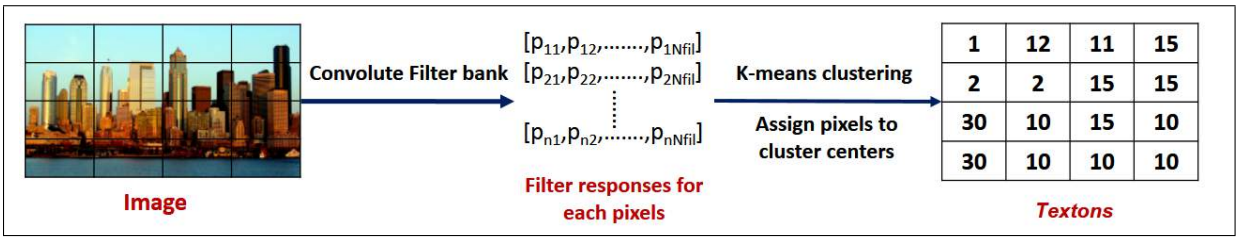

Figure 3.4: Training *Textons* - Above image shows texton training. Each pixel in the Image is converted into $\mathcal{N}_{fil}$ dimensional vector and these vectors are clustered. Finally, each pixel is assigned to nearest texton (cluster center).

3.1.2 Texture Modeling

Texture is very important feature distinguishing buildings apart. Due to sunlight, angle of photography, etc. external factors, different facades of the same building display different colors but, the facades generally display similar textures. Thus, in skyline images *texture* plays a very important part in unary potential construction. Texture model is build over pre-trained *Textons* as in [46, 48]. To generate a *Texton* for each pixel, texture descriptors as in [46, 48] are used as described below.

Generally, texture is represented using filter responses of various orientations and spatial frequencies, i.e. image is convolved with the bank of filters to construct texture descriptor. Thus, each pixel value is represented by $\mathcal{N}_{fil}$ real-valued filter responses, which is quite redundant considering textures are the entities which are repetitive by definition. Therefore, over the texture filter responses should be quite the same. Thus, there should be several distinct texture filter response vectors representing different textures, named *textons*.

Textons are trained over the training set of **Skyline-12** dataset using the below process -

1. Each image in the training set is convolved with a filter bank containing both odd and even filters at multiple orientations as well as, a radially symmetric center-surround filter to get a vector of filter responses to every pixel. Thus, each pixel is transformed to a $\mathcal{N}_{fil}$ dimensional vector of filter responses.

2. Now, all these vectors are clustered using k-means clustering algorithm as described in Section 2.5.2 into $\mathcal{N}_T$ clusters. These cluster centers are called as *Textons*. For texture modeling in our case, $\mathcal{N}_T = 32$ is used.

This process is shown in the form of flow diagram in Figure 3.4. Thus, by the end of training we have $\mathcal{N}_T$ cluster centers or textons.

Now, while parsing any skyline image, each pixel is assigned to one of these *textons* only once in the beginning, i.e. each pixel is assigned a *texton* number based on the minimum distance of the filter response vector at that pixel from the *texton* centers.

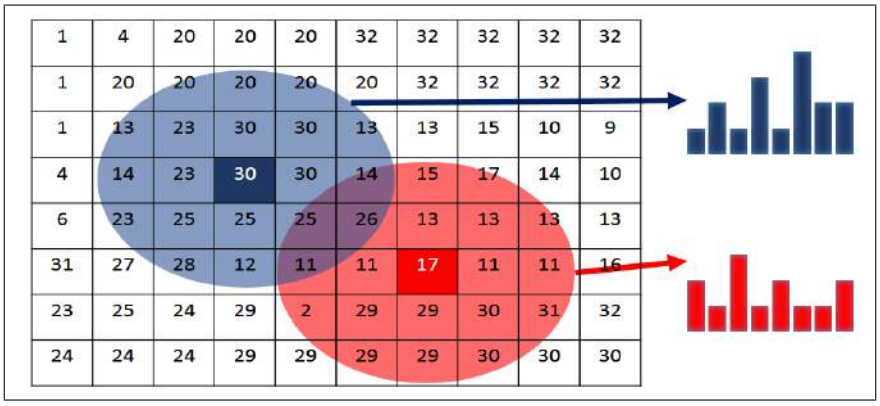

Figure 3.5: In the image, a hypothetical image is shown with each pixel assigned to a texton. For the two sample pixels, a *red* and *blue* circle is a circular window within which texton histograms are computed.

Considering the notion of discrete texture elements *'texels'*, we include the notion of texel neighborhood into computing texutre similarities. To compute pairwise texture similarities, windowed texton histograms are compared. The window $W(p)$ is defined for a pixel p, a disk of radius tr centered at pixel p. For our case, we use $tr = 10$. Each histogram has NT bins ($NT = 32$ for our case), one for each texton channel. The value of the i^{th} histogram bin for pixel p is found by counting the number of pixels inside the window $W(p)$ assigned to i^{th} texton. Thus, the histogram represents texton frequencies in a local neighborhood. This can be written as -

$$h_p(i) = \sum_{j \in W(p)} I[T(j) = i] \tag{3.9}$$

where $I[.]$ is an indicator function, i.e.

$$I(x) = \begin{cases} 1 & \text{if } x = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

and $T(j)$ returns the texton assigned to pixel p. The process is shown in Figure 3.5.

For each building (label) b in the image, texture contribution towards unary potential is computed as described below -

1. As described in section 3.1.1 and 3.1, starting with initial SLIC [1] superpixels, the foreground pixels are found and texton histograms related to those pixels are clustered using k-means clustering into K clusters (We use $K = 3$ for our purpose) to get cluster centers μ_{ibk} for i^{th} texton and k^{th} cluster.

2. Then the contribution of the texture towards unary potential of the k^{th} cluster at the p^{th} pixel for the b^{th} building (label) $T_p(b, k)$ is computed as χ^2-distance of the local texton histogram, $h_p(i)$

computed at pixel p for i^{th} texton using Equation 3.9 from the k^{th} cluster center, i.e.

$$T_p(b,k) = \sum_{i=1}^{T=32} \frac{(h_{ibk} - h_p(i))^2}{(h_{ibk} + h_p(i))} \tag{3.11}$$

3.1.3 Construction of Unary Potential

Besides color and texture component, horizontal spatial distance also contributes towards unary potential. Due to general vertical left and right edges of the building, horizontal distance from the mean seed emphasizes the horizontal spatial affinity of the pixels assigned with the same label as that of the seed, i.e. the pixels of the same building are likely to be horizontally closer to the seed pixels equally from both left and right sides due to its rectangular shape. Thus, the horizontal spatial distance $S_p(b)$, of the p^{th} pixel from the mean seed of the b^{th} building is computed as:

$$S_p(b) = ||\mu_b - x_p||_1 \tag{3.12}$$

where μ_b is the horizontal co-ordinate of the mean seed of the b^{th} building (label) and x_p is the horizontal co-ordinate of the p^{th} pixel.

Finally, unary potential $D_p(b)$ for pixel p and the building (label) b is computed as,

$$D_p(b) = \alpha(\beta \min_k C_p(b,k) + (1-\beta) \min_k T_p(b,k)) + (1-\alpha) S_p(b) \tag{3.13}$$

where, color contribution $C_p(b,k)$, texture contribution $T_p(b,k)$ and spatial distance $S_p(b)$ are computed according to Equations 3.7, 3.11 and 3.12 respectively.

3.1.4 Qualitative Analysis

Figure3.2 and Figure 3.3 show the color, texture, spatial component as well as, the resulting unary potential for few buildings from two different skylines. Table 3.1 presents the quality of the unary potentials. Labels F are obtained by taking the pixel-wise minimum of the costs of each label, i.e. -

$$F_p = \arg\min_{l \in B} D_p(l) \tag{3.14}$$

where, F_p is label obtained at pixel p, B is the set of all possible labels (buildings) and $D_p(l)$ is the unary cost of label l at pixel p. The label obtained as such is evaluated to obtain MAO on validation set of 36 images as explained in Section 2.2.1. Table 3.1 shows MAO for 4 cases - Color+Texture+Spatial Distance($\alpha = 0.35$ and $\beta = 0.2$), without color($\beta = 0$), without Texture ($\beta = 1$) and without spatial distance ($\alpha = 1$).

The table 3.1 shows that all the three components (color, shape and texture) contribute to the final success. Color alone is not sufficient, possibly due to wide appearance variations of facades of a building caused by sunlight and other external factors. Adding texture significantly improves the performance. The performance of the combination is not sensitive over a wide range of α and β. For instance, MAO

for validation set does not vary significantly over values of α between 0.2-0.4 and between 0.2-0.6. Optimum values of α and β obtained on the validation set are 0.35 and 0.20. While calculating unary potentials for various experiments, all three models are normalized to unit variance.

Description	MAO
Color + Texture + Spatial	53.4%
w/o Color	50.3%
w/o Texture	37.2%
w/o Spatial Distance	33.1%

Table 3.1: Quality of unary potentials - MAO scores on the validation set using unary potentials only.

3.2 Semantic Segmentation of Tiered Scenes

Semantic segmentation aims at pixel-wise classification of the image such that each label represents a meaningful entity. For example, in our case we aim to extract the individual buildings of a skyline using the depth order of each building which can be used later to re-build the meaningful structure of a skyline. For this purpose we use the 'tiered' structure of a skyline where each building is occluded by others forming complex tiers. In the Section 4.3 is described another example which assigns each pixel to a meaningful geometric class. In this section, a Dynamic Programming Approach is explained to efficiently minimize MRF energy formulation for a specific tiered structure.

[24] uses Dynamic Programming for labeling problem to minimize energy as given by Equation 3.1. The dynamic programming solution proposed in [24] finds a globally optimal solution to the labeling problem of the 'tiered' structure shown in Figure 3.6. In this case, the image is partitioned in top, bottom and middle region, while the middle regions is further partitioned into multiple vertical regions.

While standard expansion algorithm in Section 2.3 often gets trapped in a local minima quite far from the globally optimal solution and fails to find an acceptable solution to the labeling problem in Figure 3.6, the dynamic programming solution manages to find a global optimum in $\mathcal{O}(m^2 n \mathcal{K}^2)$ for an image with m rows, n columns and $\mathcal{K}$ labels in the middle region. Also, unlike other minimization techniques, this method does not require the pairwise potential term (or smoothness term) $V_{pq}$to be of certain form (metric or semi-metric). Instead of imposing restrictions on V_{pq}, the labeling structure is restricted.

Let $\mathbb{L}$ be the set of all possible labels. In a *tiered labeling* of an image I with m rows and n columns, each column is partitioned into 3 regions corresponding to top, bottom and middle regions. Pixels in the top are labeled as T, those in the bottom as B and pixels in the middle are labeled as l where $l \in \mathcal{M}$ and $\mathcal{M} = \mathbb{L} \setminus T, B$. Thus, the tiered labeling is defined by a sequence of n triplets (i_k, j_k, l_k) for each column k. For each column, we have a triple (i, j, l) with $0 \leq i \leq j \leq m$ and $l \in \mathcal{M}$. This triple defines a label in column k. Pixels in rows $0, 1, \ldots, i-1$ are labeled T, pixels in rows $j, \ldots, m-1$ are labeled

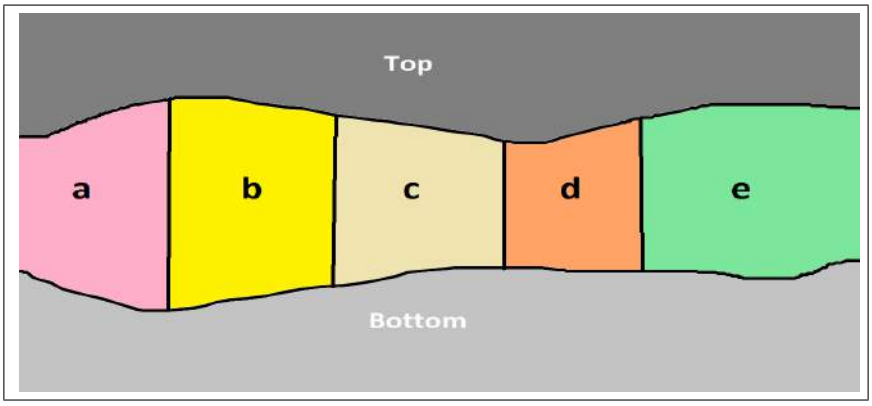

Figure 3.6: Tiered Structure - The labeling is defined by two horizontal curves that partition the image into a top, bottom and middle region. The middle region is subpartitioned by vertical boundaries. The top region is labeled T , the bottom region is labeled B and the middle regions take labels from M .

B and the rows in the middle are labeled l. The labeling is shown in Figure 3.7. Thus, the sequence of indices $\alpha = (i_0, \ldots, i_{n-1})$ defines the boundary between top an middle regions and $\beta = (j_0, \ldots, j_{n-1})$ defines the boundary between middle and bottom regions. Both, curves α and β are x-monotonic, as they cut each column exactly once. Two curves cannot cut each other as $i_k \leq j_k$.

3.2.1 Dynamic Programming

Let $Z = \{(i, j) | 0 \leq i \leq j \leq m\}$ and $S = Z \times M$. The tiered labeling is defined by sequence of n triples -

$$s_k = (i_k, j_k, l_k) \in S, 0 \leq k \leq n$$

Then, the labeling is a path in $S \times 0, 1, \ldots, n-1$ going through points (s_k, k), simultaneously tracing α and β. For a tiered labeling F, the energy in Equation 3.1 can also be expressed as -

$$E(F) = \sum_{k=0}^{n-1} C_k(s_k) + \sum_{k=0}^{n-2} H_k(s_k, s_{k+1}) \tag{3.15}$$

This form is obtained by putting data term and vertical smoothness term in C_k and horizontal smoothness term between column k and $k+1$ into the term H_k. C_k and H_k can be computed in $\mathcal{O}(1)$ using cumulative sums (integral images) as described in Section 2.5.3. The standard Dynamic Programming (DP) for solving the energy equation in 3.15 is described below. The method builds n tables, E_k indexed by states $s \in S$, the value in $E_k[s]$ is the cost of an optimal sequence of $k+1$ states ending in state s. If $(s_0, \ldots, s_k)$ is an optimal sequence ending in s_k then $(s_0, \ldots, s_{k-1})$ is an optimal sequence ending in

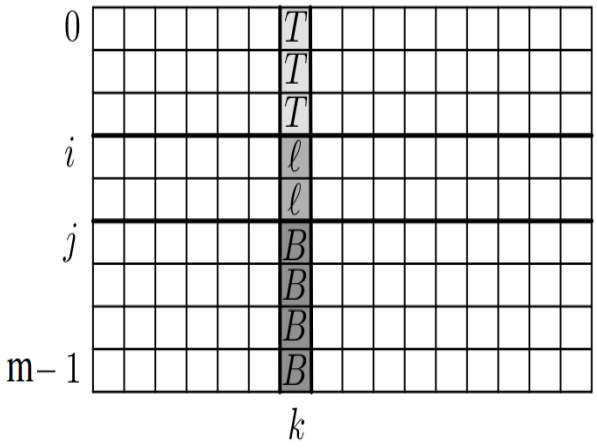

Figure 3.7: The labeling in each column is defined by 2 indices $0 \leq i \leq j \leq n$ and a label l for the middle region.

s_{k-1}. This gives a recurrence relation -

$$E_0[s] = C_0[s]$$
$$E_k[s] = C_k(s) + \min_{\bar{s}}(E_{k-1}[\bar{s}] + A_{k-1}(\bar{s}, s)) \tag{3.16}$$

While computing E_k the optimum previous states in table $P_k[s]$ are also recorded. After computation of all the tables, the optimal solution is obtained by selecting -

$$s_{n-1} = argmin_s E_{n-1}[s]$$

and then tracing the optimal path using P_k. While computing E_k, for each state $s \in S$ all the possible previous states $\bar{s} \in S$ are searched. Since, such n tables are needed to be computed, the complexity of the algorithm is $\mathcal{O}(n|S|^2) = \mathcal{O}(m^4 n K^2)$.

3.2.2 Fast Algorithm

Let g be an array of size m. The running-min of g is an array h of size m defined as -

$$h[i] = \min_{i' \leq i} g[i']$$

The simple way to compute h in $\mathcal{O}(m)$ is to first set $h[0] = g[0]$ and then sequentially set $h[i] = \min(g[i], h[i-1])$.

Consider the standard DP algorithm in Section 3.2.1. In each stage the basic computation is of the form -

$$G[s] = \min_{\bar{s} \in S}(E[\bar{s}] + A(\bar{s}, s))$$

where, E specifies the cost of solutions up to the previous column and G the solutions up to the current column. Both E and G have $|S| = \mathcal{O}(m^2 K)$ entries.

For each $s \in S$ the search for $\bar{s}$ takes $\mathcal{O}(m^4 K)$ computations. To speed the process up, the search for $\bar{s}$ is broken into different sub-cases. For each state $s = (i, j, l)$ the energy is minimized separately over l'. The algorithm works by breaking minimization for fixed l, l' over 6 cases — $Z_1(i,j), \ldots, Z_6(i,j)$ such that $Z = Z_1(i,j) \cup \ldots \cup Z_6(i,j)$. Then, $G_{l,l'}$ is computed as follows -

$$F_t[i,j] = \min_{(\bar{i},\bar{j}) \in Z_t(i,j)} (E_{l,l'}[\bar{i},\bar{j}] + H_{l,l'}((\bar{i},\bar{j}),(i,j)))$$

$$F_{l,l'}[i,j] = \min_{1 \le t \le 6} F_t[i,j]$$

The sets $Z_t(i,j)$ are defined in terms of the positions of $\bar{i}, \bar{j}$ relative to i, j. The subsets are $i \le j \le \bar{i} \le \bar{j}$, $i \le \bar{i} \le \bar{j} \le j$, $\bar{i} \le i \le \bar{j} \le j$, $i \le \bar{i} \le j \le \bar{j}$, $\bar{i} \le \bar{j} \le i \le j$ and $\bar{i} \le i \le j \le \bar{j}$.

For each t computing F_t via Brutes force takes $\mathcal{O}(m^4)$. Below described is the method for computing F_t in $\mathcal{O}(m^2)$ for the case -

$$Z_t(i,j) = \{(\bar{i},\bar{j}) | \bar{i} \le i \le \bar{j} \le j\}$$

The other cases are analogous.

There are two key ideas -

1. The search for $\bar{i}$ can be separate from search for $\bar{j}$.

2. Resulting searches can be done quickly with running-min computations.

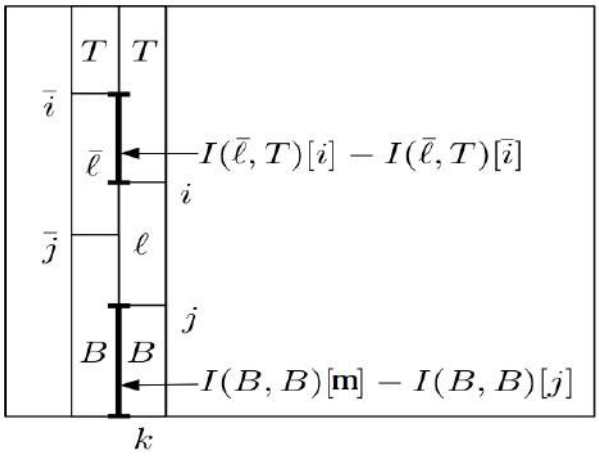

Figure 3.8: Computing $H((\bar{i},\bar{j}),(i,j))$ when $\bar{i} \le i \le \bar{j} \le j$. The horizontal V_{pq} can be defined using integral images.

Let $I(a,b)[x]$ be the sum of horizontal smoothness terms for labels a, b between previous and current columns from rows 0 to $x-1$, i.e. sum of $V_{pq}(a,b)$ for p in previous column, q in current column and

both p, q in rows 0 to $x-1$. Then the smoothness term in $A((\bar{i},\bar{j}),(i,j))$ can be expressed using these cumulative sums as follows (see Figure 3.8) -

$$A((\bar{i},\bar{j}),(i,j)) = d1 + d2 + d3 + d4 + d5$$

where -

$$\begin{aligned}
d1 &= I(T,T)[i] \\
d2 &= I(l',T)[i] - I(l',T)[\bar{i}] \\
d3 &= I(l',l)[\bar{J}] - I(l',l)[i] \\
d4 &= I(B,l)[j] - I(B,l)[\bar{j}] \\
d5 &= I(B,B)[m] - I(B,B)[j]
\end{aligned}$$

After re-grouping the terms -

$$A((\bar{i},\bar{j}),(i,j)) = \mathcal{I}(i) + \mathcal{J}(j) + \bar{\mathcal{I}}(\bar{i}) + \bar{\mathcal{J}}(\bar{j}) + \mathcal{C}$$

Thus -

$$\begin{aligned}
F_t[i,j] &= \min_{\bar{i}\le i\le\bar{j}\le j} (E[\bar{i},\bar{j}] + \mathcal{I}(i) + \mathcal{J}(j) + \bar{\mathcal{I}}(\bar{i}) + \bar{\mathcal{J}}(\bar{j}) + \mathcal{C}) \\
&= \mathcal{I}(i) + \mathcal{J}(j) + \mathcal{C} + \min_{i\le\bar{j}\le j} (\bar{\mathcal{J}}(\bar{j}) + \min_{\bar{i}\le i} (E[\bar{i},\bar{j}] + \bar{\mathcal{I}}(\bar{i}))) \qquad (3.17)
\end{aligned}$$

Since, neither $E[\bar{i},\bar{j}]$ nor $\bar{\mathcal{I}}(\bar{i})$ depend on j the optimum $\bar{i}$ is solved as a function of i and $\bar{j}$,

$$E'[i,\bar{j}] = \min_{\bar{i}\le i} (E[\bar{i},\bar{j}] + \bar{\mathcal{I}}(\bar{i}))$$

E' is computed using m running-min computations, one for each choice of $\bar{j}$. For each $\bar{j}$, let g be an array of size $\bar{j}+1$ with -

$$g[i] = E[i,\bar{j}] + \bar{\mathcal{I}}(i)$$

Then $E[i,\bar{j}] = h[i]$ where h is a running-min of g. Since each running-min computation takes $\mathcal{O}(m)$, E' is computed in $\mathcal{O}(m^2)$. Once E' is computed -

$$F_t[i,j] = \mathcal{I}(i) + \mathcal{J}(j) + \mathcal{C} + \min_{i\le\bar{j}\le j} (\bar{\mathcal{J}}(\bar{j}) + E[i,\bar{j}])$$

Once again using running-min, for each i let g be an array of size $n-i+1$ with -

$$g[j-i] = \bar{\mathcal{J}} + E'[i,j]$$

Then $F_t[i,j] = \mathcal{I}(i) + \mathcal{J}(j) + \mathcal{C} + h[j-i]$ where h is the running-min of g. Thus, for each t, $\mathcal{O}(m^2)$ time is needed. To compute $F_{l,l'}$, $\mathcal{O}(m^2)$ and by searching over l, l', $\mathcal{O}(m^2K^2)$ time for computing F, i.e. E_k in each stage of standard DP algorithm. This leads to an $\mathcal{O}(m^2nK^2)$ labeling algorithm.

The Fast tiered labeling algorithm proposed above by Felzenzwalb and Veksler [24] has certain restrictions on the labels as it cannot handle multiple layers of buildings in a skyline since, the path cuts each column only twice. Also, complexity scales exponentially with the number of labels, impractical for our setting. Zheng *et al.* [66] propose a faster approximation to [24] by decomposing multi-label tiered labeling to set of binary labeling problems exploiting topological priors. Taking similar route, we incorporate higher order priors such as overall shape and aspect ratio of the region which cannot be as easily expressed as topological priors.

In the section 3.3 we propose the effective and efficient method for interactive segmentation of the skyline into individual buildings.

3.3 Interactive Skyline Segmentation

We formulate the overall labeling problem as energy minimization as described in Equation 3.1. A standard approach to solve multi-label MRF as described above is α-expansion or $\alpha\beta$-swap algorithm [13] as mentioned in Section 2.3. In $\alpha\beta$-swap algorithm, in each iteration, a pair of labels is picked and only these labels can be interchanged between themselves. Thus, it takes $|B|^2$ iterations in each cycle instead of $|B|$, where $|B|$ is the number of buildings. On an average, skyline images in our dataset have 35 buildings, i.e. for each image it will take at least $\binom{35}{2} = 1188$ iterations as opposed to 35 during α-expansion.

In α-expansion, in each iteration label α is picked and a binary segmentation problem is formulated by replacing all the other labels to a single background label as follows:

$$E(F) = \sum_{p \in P} D'_p(F_p) + \sum_{p,q \in \mathcal{N}} V_{pq}(F_p, F_q) \tag{3.18}$$

where, F is a binary labeling such that $F : P \rightarrow \{0, 1\}^P$, i.e. each pixel is assigned a label 1 or 0 depending upon if it is foreground (α) or background respectively. Unary potential (observed data) for the foreground label α, $D'_p(1) = D_p(\alpha)$ as computed in Section 3.1 and unary potential (observed data) for the background, $D'_p(0) = D_p(L_p^{bg})$ where L_p^{bg} is the current background label at pixel p. From the current segmentation (or initial segmentation) background labels at each pixel are known except at the foreground pixels, i.e. pixels labeled α. Thus, only expansion moves are considered by setting background costs of such pixels high, i.e. higher background unary potential at the foreground pixels.

However, due to unique tiered structure of buildings (labels) in skylines, we can induce the background labels for pixels labeled α for copying the background labels from top to bottom as illustrated in Figure 3.10, i.e. for each column in the foreground labeled area, a background label of the building directly above that column is assigned. This allows us to simultaneously expand and contract the regions with label α. This is extremely beneficial in the sense that it allows us to only adjust the upper boundary of each building at a time leading to faster algorithms.

As explained in [13] and Section 2.3, an optimal solution to a binary segmentation problem can be obtained using graph cuts. Although this is an effective and general purpose approach, running graph

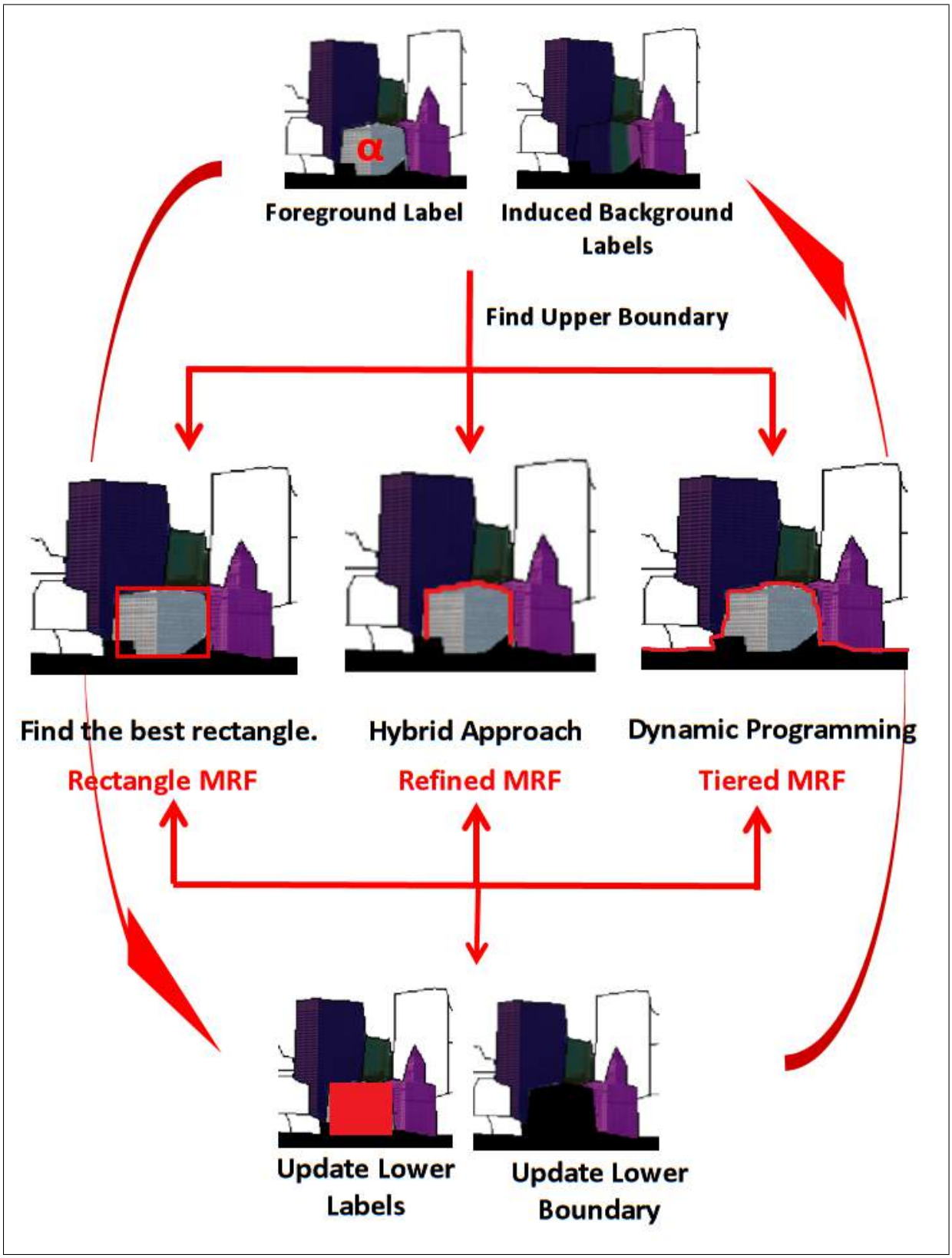

Figure 3.9: Approach - In the image a single iteration of our method is shown for the building labeled α. Approach starts with inducing background as shown in Figure 3.10 followed by computation of upper boundary using either *rectangle MRF*, *tiered MRF* or *refined MRF*. Finally, lower boundary and labels are updated. The process continues in the same way for each building.

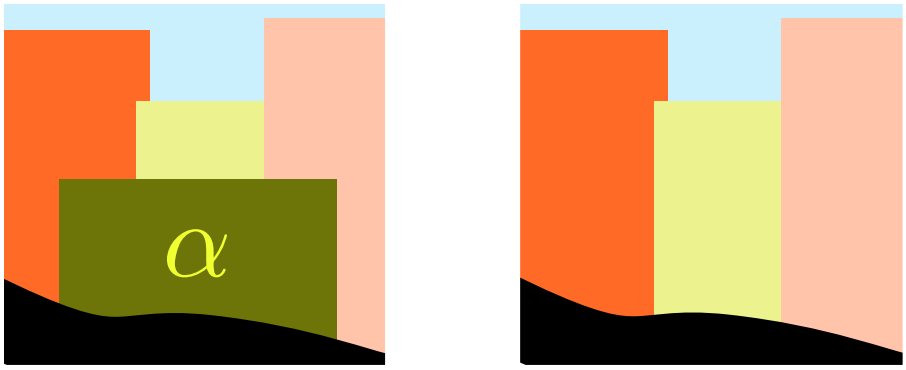

Figure 3.10: Given a label α (left) one can infer the background labels underneath α by copying the labels from the top to bottom because of the tiered structure (right).

cuts can be quite expensive on large images such as ours, requiring several minutes to find the optimal labeling. Worst case complexity for binary segmentation using Graph cut as explained in [13] with graph of V nodes and E edges is $\mathcal{O}(EV^2)$. In our case, number of nodes $V = P$ where P is number of pixels. For an image with m rows and n columns number of pixels, $P = m \times n$. Each node has 2 t-links and 8 n-links (considering 8-neighborhood), i.e there are totally $10 \times V = 10mn$ edges in the graph. Then, the complexity of optimization using Graph cuts for each binary segmentation iteration in terms of number of rows(m) and number of columns(n) is $\mathcal{O}(m^3n^3)$.

We propose much faster algorithm. Our key idea is to replace the search over binary segmentations by a search over a parametric shape family. For buildings we can explicitly search over the space of feasible rectangles much faster than possible segmentations. Furthermore, the 'tiered' structure of the buildings provides a natural ordering of the buildings according to their depth order, i.e. buildings that are closer to the camera or buildings in the lowermost layer are nearer to the lower boundary (separating buildings from water body or trees, etc.) than the buildings behind them. In practice the buildings are ordered according to the lowest seed pixel, i.e., the building with the lowest seed is considered first and so on.

Our approach is outlined in Algorithm 2.

Each step in Algorithm 2 is discussed in detail below -

1. Initial Segmentation F, Lower and upper boundaries (l, u) as well as, data term D and pairwise potential V are required as an input to the method. Initial segmentation is obtained by finding best rectangle as described in Section 3.3.1 for each label from bottom to top. The initial segmentation for a building is shown in Figure 3.11.

2. **Step 2-9** is called an iteration. In each iteration, all the labels ordered from bottom to top are visited once.

Algorithm 2 Greedy skyline segmentation

Require: data D, pairwise V, boundary (l, u)

1: Initialize, initial labeling F from unary labels
2: **for** iter := 1 **to** K **do**
3: Initialize, frontier $f \leftarrow l$
4: **for** α := 1 **to** N **do**
5: $\Omega_\alpha \leftarrow \mathsf{upperBoundary}(\alpha, F, D, V, f, u)$
6: $f \leftarrow \max(f, \Omega_\alpha)$
7: F $\leftarrow \mathsf{updateLabels}(F, \Omega_\alpha)$
8: **end for**
9: **end for**

Figure 3.11: Initial Segmentation - In the figure shown, is an initial segmentation for a sample image, obtained via finding best rectangle for each building from bottom to top.

3. In **Step 3**, 'Frontier' f is initialized to the lower boundary l of the building region. Through updating frontier, we keep updating of the lowermost building in each iteration.

4. At each iteration we pick the next building (label) α in the ordered list. List is ordered from the buildings with lowermost seeds (lowest y-co-ordinate) to the buildings with the uppermost seeds (highest y-co-ordinate), i.e. lowermost buildings are picked first followed by upper buildings in that order.

5. Once the building (label) α is picked, background labels are induced by copying labels from top to bottom as shown in Figure 3.10 and binary segmentation problem is formulated as described in Equation 3.18. In **Step 5**, upper boundary for each building, Ω_α is estimated by finding the optimal solution the Equation 3.18 in each iteration.

$$\Omega_\alpha \leftarrow \mathsf{upperBoundary}(\alpha, F, D, V, f, u)$$

6. Then, in **Step 6**, frontier is updated by taking columnwise *max* of the current frontier F and the upper boundary, Ω_α computed in previous step.

$$f \leftarrow \max(f, \Omega_\alpha)$$

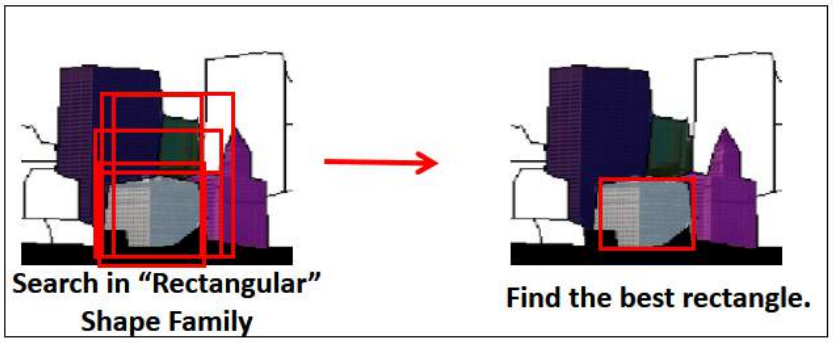

Figure 3.12: Rectangle MRF - In the figure shown, is the process of finding best rectangle for a sample building. We search in the rectangular parametric shape family to find the best rectangle.

Figure 3.13: Rectangle MRF - In the figure shown, is the *rectangle MRF* output for the sample image. For each label (building), the best rectangle is searched.

7. In **Step 7**, corresponding labeling F is updated as well by labeling all the pixels between previous frontier and upper boundary Ω_α. Updating the labeling is important as in consequent iteration **Step 2-9**, background labels at the foreground pixels will change accordingly.

$$F \leftarrow \mathsf{updateLabels}(F, \Omega_\alpha)$$

The approach described above is outlined in the Figure 3.9 for a single iteration.

Below we describe three efficient ways of searching over upper boundaries in **Step 5** - Rectangle MRF, Tiered MRF and Refined MRF.

3.3.1 Rectangle MRF

Most of the buildings in the skyline can be described as exactly rectangular or can be fairly approximated as a rectangle. Thus, in this formulation we constrain the buildings to be exactly rectangular i.e., for each building we need to estimate only three values for its segmentation (L, R, T) - left, right and

top co-ordinates of the rectangle describing the segmentation respectively, where L and R are column indices and, T is row index of the rectangle in an image. We do not need to estimate bottom row index of the rectangle as it is fixed by the current frontier.

Values (L, R, T) have to be within the feasible set, i.e.,

1. The possible rectangles have to be within current upper and lower boundaries i.e., the rectangle should be within buildings (middle) region.

$$L \leq T \leq$$

2. The rectangle should enclose the seed pixels of the building, i.e. maximum value of L can be the leftmost seed pixels and minimum value of the R can be the rightmost seed pixel.

3. We can also constrain the aspect ratios to a desired range as well as, enforce width and height constraints learned on training data. By learning on training data we can have a range of aspect ratio for rectangle to form a valid building, i.e. most of the buildings have much more height than width. We can enforce these constraints by limiting the rectangle search space to the rectangles within desired aspect ratios, width and height.

4. We can also limit the number of rectangles to be searched by setting a variable *stepSize* specifying sampling interval in width and height of the rectangles in the search space.

The sample output is shown in the Figure 3.13.

Algorithm 3 Rectangle MRF

Require: label α, data D, pairwise V, labeling F, Seeds $({S_\alpha}^x, {S_\alpha}^y)$, Lower and upper bounds (l, u)
Ensure: Upper boundary Ω_α

```
Induce Background labels from α and F
for L := 1 to min(S_α^x) do
    for R := max(L + minWidth, max(S_α^x)) to n do
        for T := min(l) to max(u) do
            score_{L,R,T} ← E^α_{LRT}(F)
            if score_{L,R,T} < minScore then
                minScore = score_{L,R,T}
                Ω_α ← (L, R, T)
            end if
        end for
    end for
end for
```

The energy of the rectangle can be computed in $\mathcal{O}(1)$ using integral images of unary and pairwise potential as explained in Section 2.5.3. For a region of size $m \times n$, i.e. m rows and n columns, there are $\mathcal{O}(mn^2)$ rectangles to consider, i.e there are m possibilities for Top boundary (T) and for each possible

value of T, there are n possibilities for left boundary (L) and right boundary (R) each, thus resulting $\mathcal{O}(mn^2)$ rectangles to search.

Consider, ID_{fg} be the integral image corresponding to foreground data term $D(\alpha)$, ID_{bg} be the integral image corresponding to background fata term $D(l_p^{bg})$, IV_y be the row-wise integral image corresponding to horizontal component of pairwise potential Vy and IV_x be the column-wise integral image corresponding to vertical component of pairwise potential, Vx. Consider Bf to be the minimum of the lower boundary (frontier) between L and R, i.e. $Bf = \min_{L \le y \le R} f$. Then, for the rectangle (L, R, T), sum of energy in the rectangle, E^{α}_{LRT} is computed as below -

1. Sum of the unary potential corresponding to Rectangle (L, R, T) is computed subtracting sum of background unary potential from the foreground unary potential, i.e. $\mathcal{O}^{fg}_{LRT} = ID_{fg}(R, Bf) + ID_{fg}(L, T) - ID_{fg}(R, T) - I_{fg}(L, Bf)$. Similarly, $\mathcal{O}^{bg}_{LRT}$ is computed.

2. Sum of Horizontal pairwise potential is calculated along the top edge of the rectangle using row-wise integral image, i.e. $Vy_{LRT} = IV_y(R, T) - IV_y(L, T)$.

3. Sum of Vertical pairwise potential is calculated along the two vertical edges - *left* and *right* edges using column-wise integral images, i.e. $Vx_{LRT} = IV_x(L, Bf) - IV_x(L, T) + IV_x(R, Bf) - IV_x(R, T)$

4. Finally, sum of energy corresponding to building α is calculated as -

$$E^{\alpha}_{LRT} = \mathcal{O}^{fg}_{LRT} - \mathcal{O}^{bg}_{LRT} + Vy_{LRT} + Vx_{LRT} \tag{3.19}$$

3.3.2 Tiered MRF

Figure 3.14: Tiered MRF - In the *left* shown is a setting for Dynamic Programming used in *tiered MRF*. In the image shown are column j and $j-1$. u_{j-1} and u_j represent the row indices for upper boundary in columns $j-1$ and j respectively, while l_{j-1} and l_j represent the row indices for lower boundary in columns $j-1$ and j respectively. k and i are the row indices in DP as explained in Equation 3.20. In the *right*, the output of *tiered MRF* is shown for a sample image.

Constraining the upper boundaries to be rectangles can be a poor approximation for many buildings. We propose **Tiered MRF** where the shape of upper boundary is refined to estimate accurately as in

ground truth. However, instead of a $2D$ curve we restrict the upper boundary Ω to be 'x-monotonic', i.e. it intersects each column exactly once. This is a good approximation to the buildings generally seen in skylines which are convex. Almost all the buildings (which are convex), can be represented by an upper boundary which intersects each column only once.

The key advantage of the assumption of the x-monotonicity of the buildings is that the optimal solution (i.e. upper boundary) can be found using a simple extension of the dynamic programming algorithm proposed in [24, 66] explained in Section 3.2.

In our case, we need to compute the single horizontal path Ω_α (upper boundary) in each iteration crossing each column only once. Thus, the path is defined by sequence of row indices $i_1, i_2, \ldots, i_{n-1}$ such that $l_j \leq i_j \leq u_j$ for each column j where l_j and u_j denote upper and lower bounds at column j. At each column j we maintain the optimal cost of a path ending in each row i as well as, we also maintain back-pointers (optimum paths in previous columns) $P_j[i]$. Like 3.2.1, if $(i_0, \ldots, i_j)$ is an optimal sequence ending at column j then $(i_0, \ldots, i_{j-1})$ is an optimal sequence ending at column j. This gives the recurrence relation for $E_{i,j}$ for $i \in [l_j, u_j]$-

$$\begin{aligned} E_{i,-1} &= 0, \forall i \\ l_{-1} &= u_{-1} = u_0 \\ E_{i,j} &= \min_{k \in [l_{j-1}, u_{j-1}]} E_{k,j-1} + \mathcal{C}_{i,j} + |\mathcal{H}_{k,j} - \mathcal{H}_{i,j}| + \mathcal{V}_{i,j} + \tau|k-i| \end{aligned} \tag{3.20}$$

Where, $\mathcal{C}_{i,j} = \sum_{t=l_j}^{i} D'_{(t,j)}(1) - D'_{(t,j)}(0)$ is the sum of the data term for binary segmentation in column j from lower bound l_j to i. $\mathcal{H}_{k,j} = \sum_{t=l_j}^{i} V_{(t,j-1),(t,j)}$ is the sum of horizontal smoothness term for the neighboring pixels in column $j-1$ and j from lower bound l_j to row index i. $\mathcal{V}_{i,k} = V_{(i+1,j),(i,j)}$ is the vertical smoothness term between the boundary pixels i and $i+1$ in column j. Here, $V_{p,q}$ is the cost of an edge between pixels p and q (Equation 3.18, 3.1), i.e. horizontal smoothness term opposes vertical edge in the boundary between similar color pixels and vertical smoothness term opposes horizontal edge in the boundary between similar color pixels. The last term τ forces the path to be smoother, i.e. nearly rectangular. The setting is shown in Figure 3.14.

The term $\mathcal{C}$ and $\mathcal{H}$ can be pre-computed as described in Section 2.5.3 using cumulative sums, allowing evaluation of $C_{i,j}$ in $\mathcal{O}(m)$ time, $\mathcal{O}(1)$ for each row $j \in [l_{j-1}, u_{j-1}]$. $C_{i,j}$ needs to be evaluated for each row i and each column j, thus mn times. Thus, the complexity of computing the optimal path is $\mathcal{O}(m^2 n)$ for each iteration (label).

Tiered MRF provides an efficient way to find an upper boundary Ω_α in each iteration to solve the overall labeling problem. But, it does not take the rectangular shape of the building into account while finding the upper boundary.

3.3.3 Refined MRF

The *Rectangle MRF* strictly outputs rectangular segments, often producing incorrect upper boundary since quite a lot of buildings have non-rectangular upper boundary shapes. Even though the *Tiered MRF*

Figure 3.15: Refined MRF - Above shown is a segmented output for the sample image using *refined MRF*. In *refined MRF*, left and right boundaries are considered to be rectangular while the upper boundary is considered an x-monotonic curve.

does not constrain the boundaries to be rectangular, neither does it respect the rectangular shape of the building. Hence, we propose a hybrid approach — *Refined MRF*.

In Refined MRF, we find the *Left* and *Right* $(\mathcal{L}, \mathcal{R})$ boundary of the building from the optimal rectangle found using *Rectangle MRF* as described in Section 3.3.1 i.e. in Refined MRF, we constrain the left and right edge of the upper boundary to be rectangular (vertical). Then, the upper boundaries of the buildings between $\mathcal{L}$ and $\mathcal{R}$ are refined using dynamic programming as described in Section 3.3.2. The recurrence relation in Equation 3.20 changes as follows -

$$\begin{aligned} E_{i,L-1} &= 0, \forall i \\ l_{L-1} &= u_{L-1} = u_L \\ E_{i,k} &= \min_{j \in [l_{k-1}, u_{k-1}]} E_{j,k-1} + \mathcal{O}_{i,k} + |\mathcal{A}_{j,k} - \mathcal{A}_{i,k}| + \mathcal{V}_{i,k} + \tau|j-i| \end{aligned} \tag{3.21}$$

The optimal solution is obtained by selecting -

$$i_R = argmin_i(E_{i,R})$$

This algorithm maintains overall shape while allowing better and more accurate fits to the upper boundary.

For a building of width $d = |\mathcal{L} - \mathcal{R}|$, the upper boundary can be computed using Refined MRF in $\mathcal{O}(mn^2)$ computations to search optimal rectangle using Rectangle MRF and $\mathcal{O}(m^2 d)$ computations for dynamic programming, thus $\mathcal{O}(mn^2 + m^2 d)$.

3.4 Results

Outputs of the *tiered MRF*, *rectangle MRF* and *refined MRF* for the sample image from each city are shown in Figure 3.16,3.17,3.18 and 3.19.

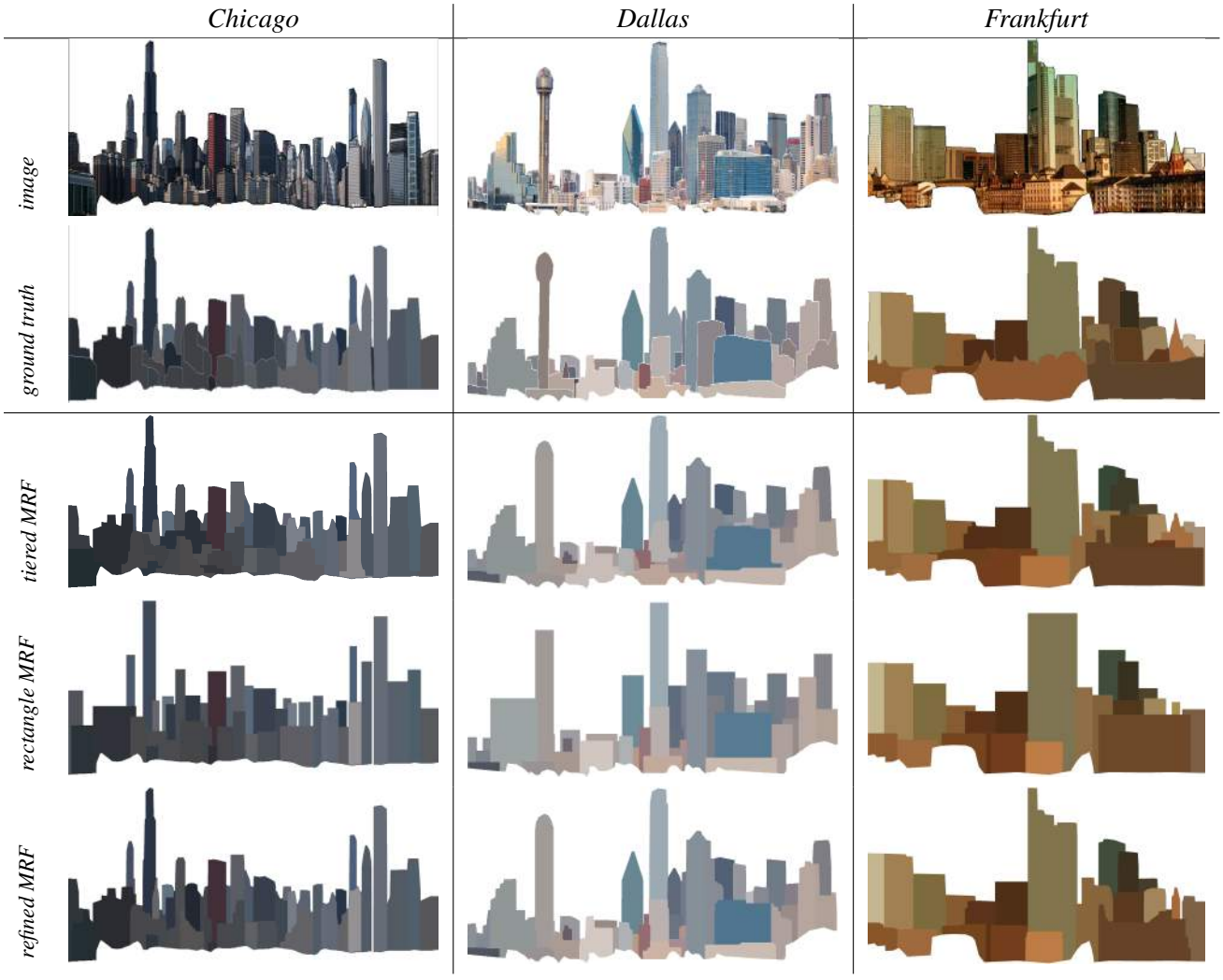

Figure 3.16: Interactive Skyline segmentation results. In first and second row, original skylines within the upper and lower boundary and the corresponding ground truth segmentation are shown. In third, fourth, and fifth row, outputs of the interactive *tiered MRF*, *rectangle MRF*, *refined MRF* are shown respectively.

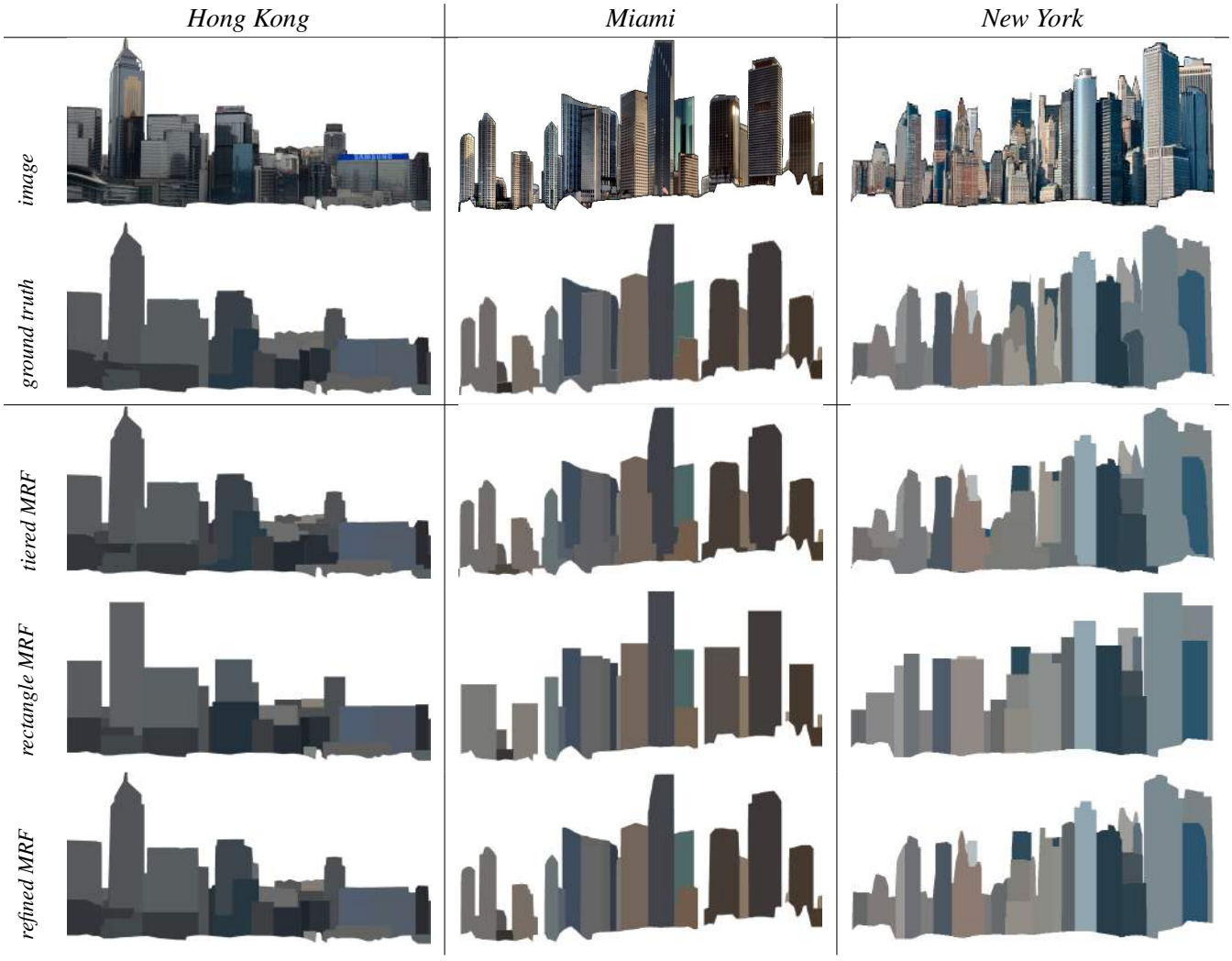

Figure 3.17: Interactive Skyline segmentation results Continued. In first and second row, original skylines within the upper and lower boundary and the corresponding ground truth segmentation are shown. In third, fourth, and fifth row, outputs of the interactive *tiered MRF*, *rectangle MRF*, *refined MRF* are shown respectively.

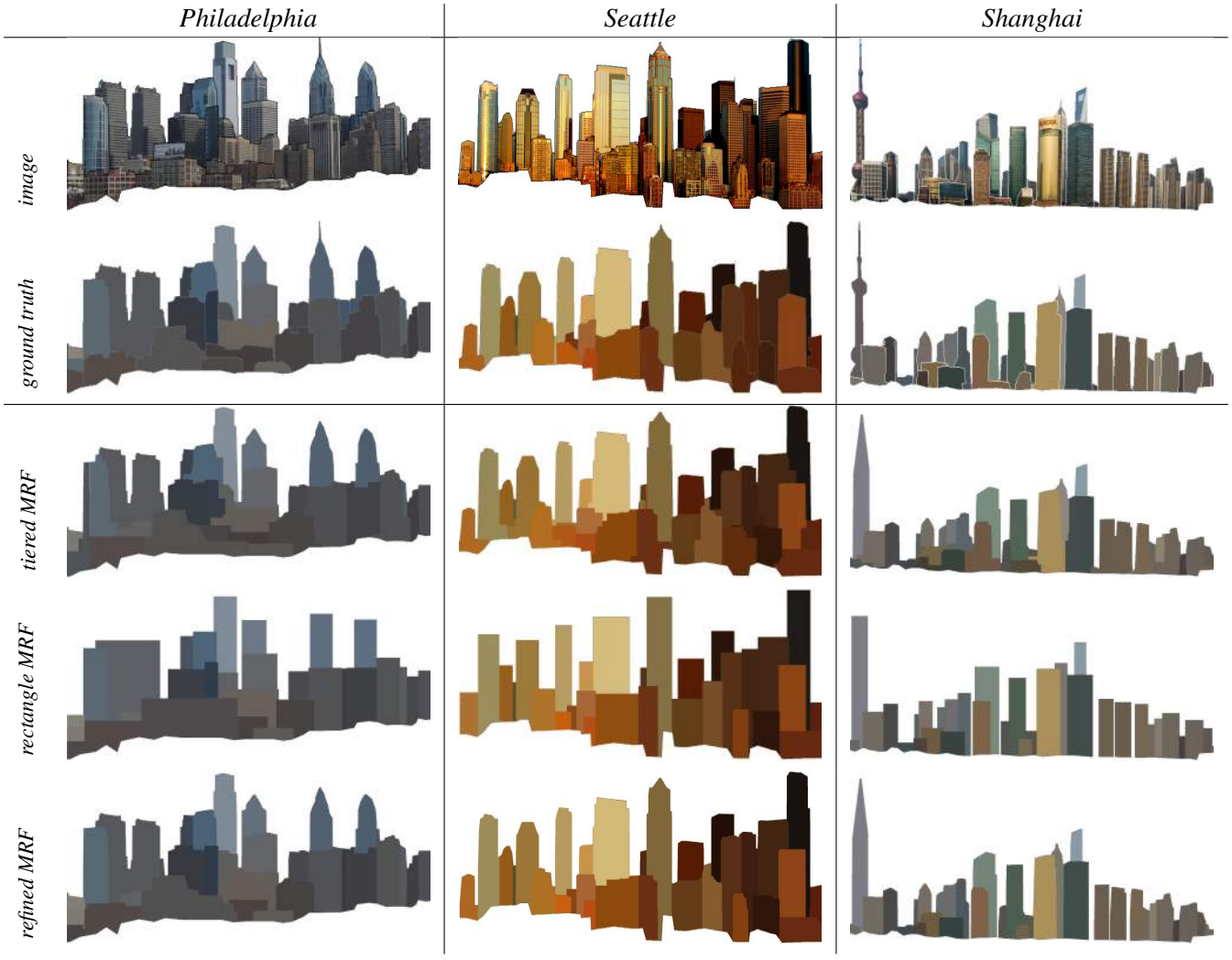

Figure 3.18: Interactive Skyline segmentation results. In first and second row, original skylines within the upper and lower boundary and the corresponding ground truth segmentation are shown. In third, fourth, and fifth row, outputs of the interactive *tiered MRF*, *rectangle MRF*, *refined MRF* are shown respectively.

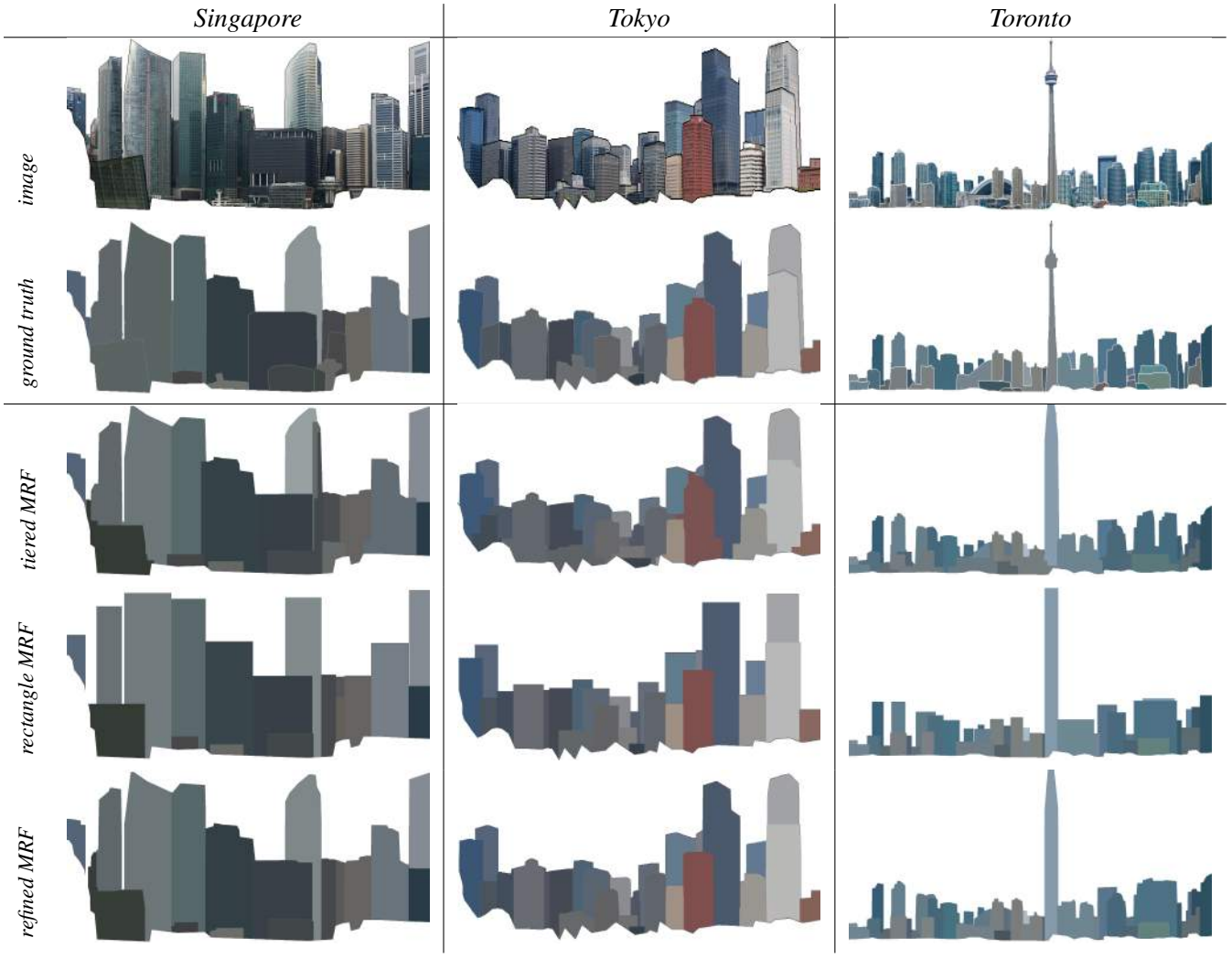

Figure 3.19: Interactive Skyline segmentation results continued. In first and second row, original skylines within the upper and lower boundary and the corresponding ground truth segmentation are shown. In third, fourth, and fifth row, outputs of the interactive *tiered MRF*, *rectangle MRF*, *refined MRF* are shown respectively.

In the interactive setting we compare *tiered MRF*, *rectangle MRF* and *refined MRF* with a *standard MRF* formulation where α-expansion is used to solve the binary labeling problem. We use publicly available code for max-flow/min-cut for optimizing the problem [11]. For a fair comparison we run all the algorithms for $K = 2$ outer iterations (Algo. 2). In our experiments we found that no significant change in labeling after 2 iterations. For speed we also resize all the images to a maximum dimension of 2000 pixels, and the results rescaled to the original size for evaluation.

Table 3.2 presents results in the interactive setting. All the MRF formulations significantly improve over the unary potentials. *Our proposed approaches are about an order of magnitude faster than the α-expansion*. The *rectangle MRF* achieves results almost as good as the *standard MRF* while taking only 5.5s on average per image on commodity desktop with an Intel CPU @ 3.20GHz. Refinement on top improves performance for a small additional time of 3.7s (for a total of 9.2s). Tiered labeling is fast but not competitive showing the value of enforcing shape priors.

Method	MAO	Complexity/building	Speed/image
Unary only	54.5%	n/a	n/a
Standard MRF	62.3%	$\mathcal{O}(m^3n^3)$	69.5s
Tiered MRF	59.4%	$\mathcal{O}(m^2n)$	7.5 s
Rectangle MRF	62.0%	$\mathcal{O}(mn^2)$	5.5 s
Refined MRF	**63.4%**	$\mathcal{O}(mn^2 + m^2d)$	9.2 s

Table 3.2: Speed and accuracy tradeoff in the interactive setting. For various methods MAO scores, *worst case* computational complexities per building, and speed per image (in seconds) averaged over the *test* set are shown. All the methods are run for $K = 2$ outer iterations (Algorithm 2). Images are resized to a maximum dimension of 2000 pixels for speed. The typical image is of size $m \times n = 1255 \times 2000$ pixels and has 34 buildings.

In Figures 3.20, 3.21 and 3.25 the qualitative comparison of *refined MRF* and *standard MRF* is discussed with the help of few examples. Similarly in Figures 3.22 3.23 and 3.24, the qualitative comparison between *refined MRF* and *tiered MRF* is discussed with the help of specific examples. In a typical skyline image, many buildings have two visible facades, each with different color and texture due to sunlight, because of which the unary potentials are unreliable. Here shape priors can provide additional cues to guide segmentation. Figure 3.20 ,3.21 ,3.22, and 3.23 show the significance of shape priors in segmenting buildings. The *refined MRF* outperforms both *standard MRF* and *tiered MRF*, while preserving contiguity and shape of the segments. While it correctly segments buildings in most of the cases, there are images where rectangular shape prior is grossly incorrect. Two such examples are shown in Fig. 3.24 and 3.25. In the first case, *refined MRF* fails due to irregular shapes of crowded and similar buildings. In the later case, the rectangular shape prior is incorrect due to concave shape of the buildings.

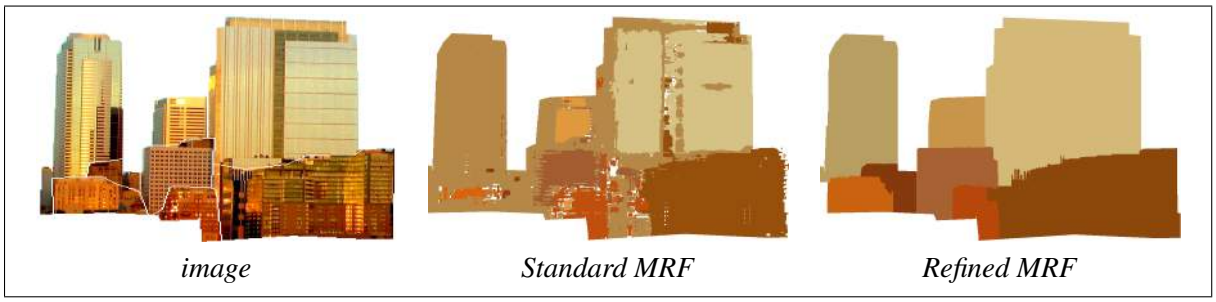

Figure 3.20: An example where *refined MRF* improves over the *standard MRF*. As we can see, the *standard MRF* output is not continuous, spread out with lot of noise, while *refined MRF* gives continuous segments respecting the rectangular shape of the buildings all the while.

Figure 3.21: Another example where *refined MRF* improves over the *standard MRF*. As we can see, in the *standard MRF* output merges two different buildings into one of approximately similar color and textures. *refined MRF* segments the buildings correctly, due to enforced shape priors and *left* and *right* boundaries imposed by rectangle.

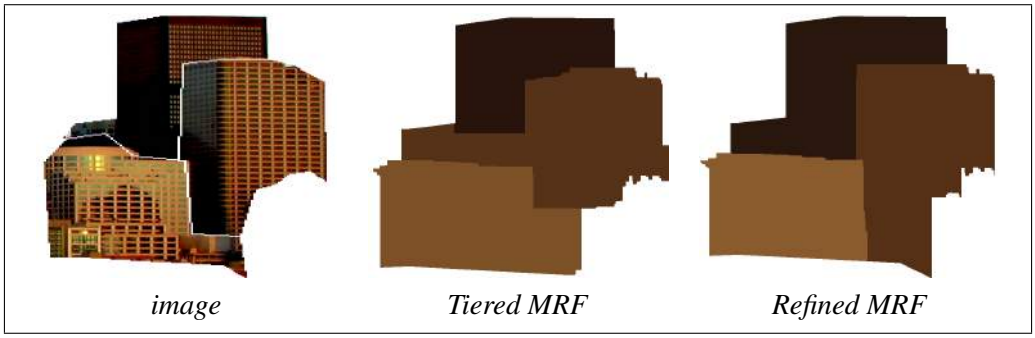

Figure 3.22: An example where *refined MRF* improves over *tiered MRF*. Here, *tiered MRF* output *oversegments* the building, while *refined MRF* output correctly segments it. *tiered MRF* fails here due to lack of shape information to the algorithm, while *refined MRF* correctly segments due to shape enforcement which restricts the algorithm from spreading out boundaries beyond the optimal rectangle.

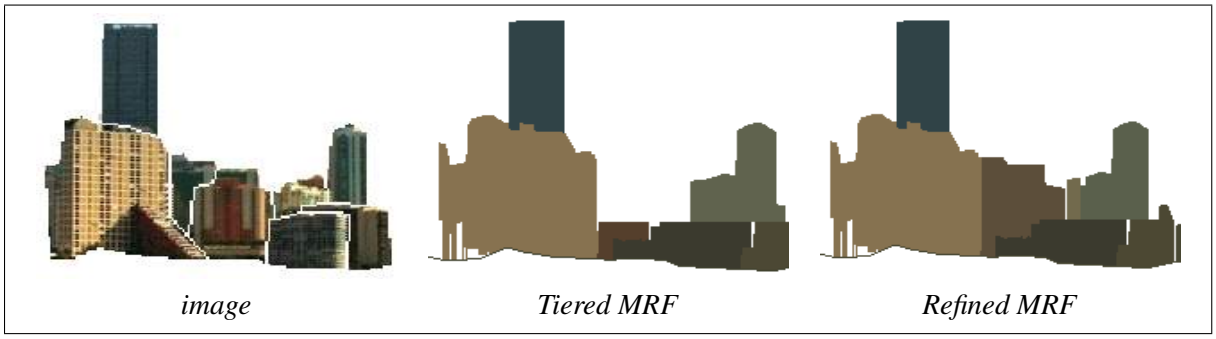

Figure 3.23: An example where *refined MRF* improves over *tiered MRF*. Here, *tiered MRF* output *undersegments* the building, while *refined MRF* output correctly segments it. *tiered MRF* fails here again due to lack of shape information to the algorithm, while *refined MRF* correctly segments due to prior shape information available to it.

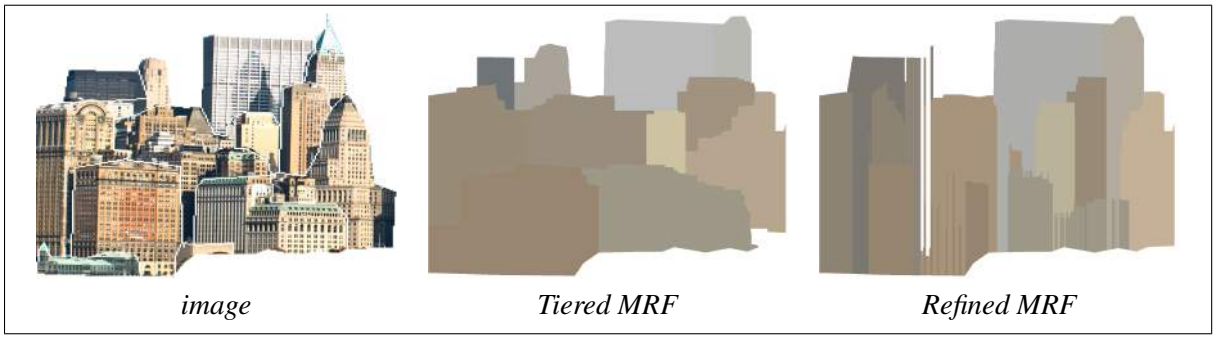

Figure 3.24: An example where *refined MRF* fails over *tiered MRF*. In the section of the image shown, the buildings are generally not rectangular and similarity of color and texture among them makes it difficult to perceive the individual boundaries, while intricate tiered structure loses the rectangular shape. Thus, in this case, shape information is futile.

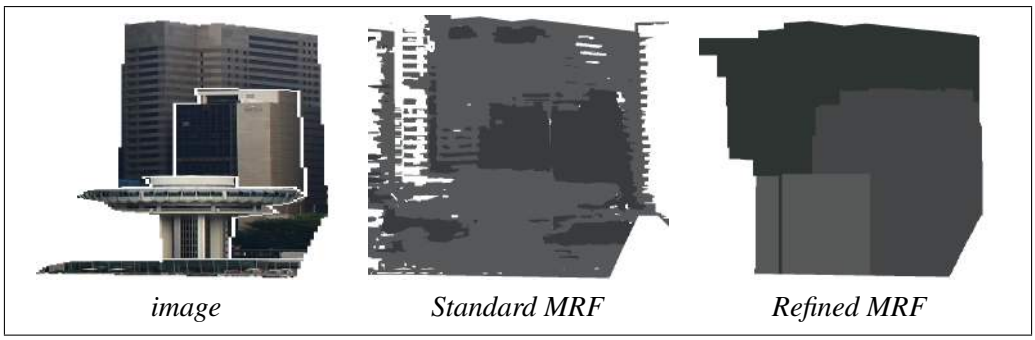

Figure 3.25: An example where *refined MRF* fails over *standard MRF*. In *refined MRF*, we consider the shape of upper boundary to be *x-monotonic*, i.e. boundary intersects each column only once. Thus, *refined MRF* fails to segments concave buildings and is applicable strictly in case of convex upper boundaries.

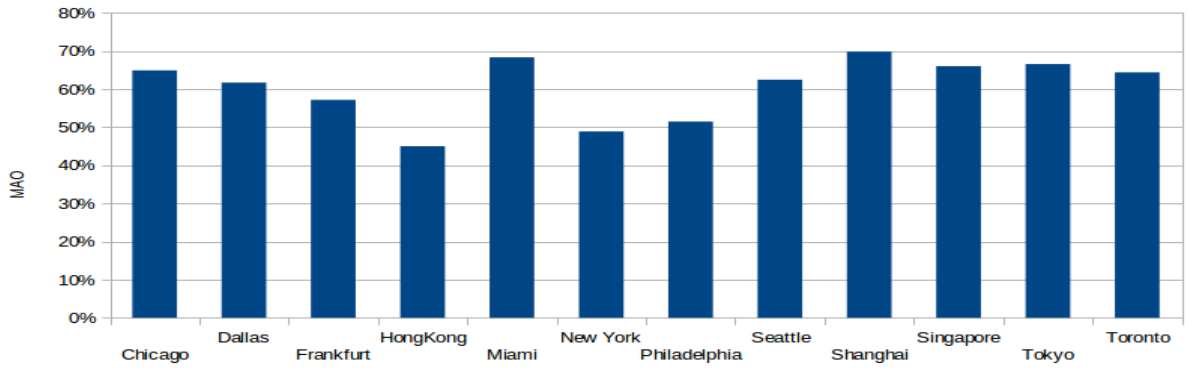

Figure 3.26: Citywise Results Above shown is citywise MAO for the interactive setting.*Shanghai* and *Miami* skylines prove to be the ones with the highest MAO, while *Hong Kong* and *New York* skylines prove to be most difficult to segment.

Figure 3.26 shows *citywise* MAO for the interactive setting. *Shanghai* and *Miami* skylines prove to be the ones with the highest MAO, while *Hong Kong* and *New York* skylines prove to be most difficult to segment. The reason *Hong Kong* and *New York* cities are relatively harder to segment is that *Hong Kong* has a dense skyline with lot of similar buildings (same color and texture) and lot of small buildings, which makes it harder to perceive and segment them individually. *New York* has also very dense skylines and additionally, *New York* skylines have lot pre-war buildings crammed together creating complex occlusion patterns separating which individually is very subjective and depends on different individuals (Figure 3.24).

Chapter 4

Automatic Segmentation

4.1 Automatic Unsupervised Segmentation Methods

Many low-level automatic methods such as SLIC [1], Graph based Segmentation [23], gPb boundary detector [3] perform skyline segmentation that is coarse, not contiguous and does not have semantic understanding. None of these methods explicitly consider shape priors. In the automatic setting, we start with the output of one of these methods and improve upon these outputs using shape priors.

4.1.1 SLIC

Simple Linear Iterative Clustering (**SLIC**) [1] is an adaptation for k-means(2.5.2) for superpixel generation with following distinctions -

1. The number of distance calculations in the optimization is dramatically reduced by limiting search space to a region proportional to a superpixel size, resulting the complexity linear in number of pixels N and independent of the number of superpixels k.

2. A weighted distance measure combines color and spatial proximity while simultaneously providing control over the size and compactness of the superpixels.

SLIC is similar to the approach used as a preprocessing step for depth estimation described in [67], which was not fully explored in the context of superpixel generation.

Initialization Step - For the images converted in CIE Lab color space, the clustering procedure being with an initialization step where k initial centers $C_i = [l_i a_i b_i x_i y_i]^T$ are sampled on a regular grid spaced S pixels apart. To produce roughly equal sized superpixels, the grid interval is $S = \sqrt{N/k}$. The centers are moved to seed locations corresponding to the lowest gradient position in a $a \times 3$ neighborhood to avoid centering a superpixel on an edge and to reduce the chance of seeding a superpixel with a noisy pixel.

Assignment Step - Next, in the assignment step, each pixel i is associated with the nearest cluster center whose search region overlaps its location. This is the key to speeding up the algorithm because

limiting the size of the search region significantly reduces the number of distance calculations, and results in a significant speed advantage over conventional k-means clustering where each pixel must be compared with all cluster centers. The distance measure D determines the nearest cluster center for each pixel. To combine both Lab color distance and spatial distance into a single distance measure, the color and spatial proximity is normalized by their respective maximum distances within cluster $N_c = m$ where m is a constant and $N_s = S = \sqrt{N/K}$.

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \tag{4.1}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{4.2}$$

$$D' = \sqrt{(\frac{d_c}{m})^2 + (\frac{d_s}{S})^2} \tag{4.3}$$

Simplifying -

$$D = \sqrt{d_c^2 + (\frac{d_s}{S})^2 m^2} \tag{4.4}$$

Since the expected spatial extent of a superpixel is a region of approximate size $S \times S$, the search for similar pixels is done in a region $2S \times 2S$ around the superpixel center.

Update Step - In an update step, cluster centers are adjusted to mean $[labxy]^T$ vector of the pixels in the cluster. The L_2 norm is used to compute the residual error E between the new cluster center locations and previous cluster center locations. The assignment and update steps can be repeated iteratively until the error converges.

Postprocessing - Connectivity is enforced in the postprocessing step by reassigning the disjoint pixels to nearby superpixels.

The algorithm is summarized in Algorithm 4.

In practice, a pixel falls in the neighborhood of less than eight cluster centers, meaning that SLIC is $\mathcal{O}(N)$ complex. In contrast, the trivial upper bound for the classical k-means algorithm is $\mathcal{O}(k^N)$ [45], and the practical time complexity is $\mathcal{O}(NkI)$ [21], where I is the number of iterations required for convergence. While schemes to reduce the complexity of k-means have been proposed using prime number length sampling [61], random sampling [41], local cluster swapping [37], and by setting lower and upper bounds [21], these methods are very general in nature. SLIC is specifically tailored to the problem of superpixel clustering.

4.1.2 Graph Based Segmentation

This method described in [23] takes a graph based approach to segmentation. Let $G = (V, E)$ be an undirected graph with vertices $v_i \in V$, the set of elements to be segmented, and edges $(v_i, v_j) \in E$ corresponding pair of neighboring vertices. Each edge $(v_i, v_j) \in E$ has the corresponding weight w_{ij} denoting a non-negative measure of dissimilarity between neighboring elements v_i and v_j. In case of image segmentation, the elements in V are pixels and the weight of an edge is difference in color intensity between two pixels connected by that edge.

Algorithm 4 SLIC Superpixels Generation [1]

Initialize, cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move Cluster centers to the lowest gradient position in 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.
while $E > threshold$ **do**
  for each cluster center C_k **do**
    for each pixel i in $2S \times 2S$ region around C_k **do**
      Compute distance D between C_k and i
      if $D < d(i)$ **then**
        $d(i) = D$
        $l(i) = k$
      end if
    end for
  end for
  Compute new cluster centers
  Compute Residual Error E
end while

In this approach, each segmentation S is a partition of V into connected components where each connected component $C \in S$ corresponds to $G' = (V, E')$ such that $E' \subseteq E$. Any segmentation is induced by a subset of edges in E. In general, the elements in the same component are similar, i..e edge weights between the elements in the same components are low. Similarly, the elements in the different components are dissimilar, i.e. edge weights between the pixels from different components are high.

Predicate D is defined to evaluate the evidence of boundary between two segments (components). The predicate is based on measuring the dissimilarity between pixels along the boundary of components relative to dissimilarity between neighboring pixels within the component, i.e. predicate compares inter-component dissimilarities with within-component dissimilarities. The *Internal Difference* of a component $C \subseteq V$, $Int(C)$ is defined as the largest weight in the minimum spanning tree of the component, $MST(C, E)$, i.e. -

$$Int(C) = \max_{e \in MST(C,E)} w_e \tag{4.5}$$

This means, the internal difference is defined as the minimum edge weight needed to hold the component together or keep the component connected.

Difference between two components $C_1, C_2 \subseteq V$, $Dif(C_1, C_2)$ is defined as the minimum edge weight connecting the two components, i.e. -

$$Dif(c_1, c_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w_{ij} \tag{4.6}$$

If there is no edge connecting C_1 and C_2, then $Dif(C_1, C_2) = \infty$.

The region comparison predicate evaluates the evidence of the boundary between the pair of components by comparing how large $Dif(C_1, C_2)$ is relative to either of the $Int(C_1)$ or $Int(C_2)$. A threshold

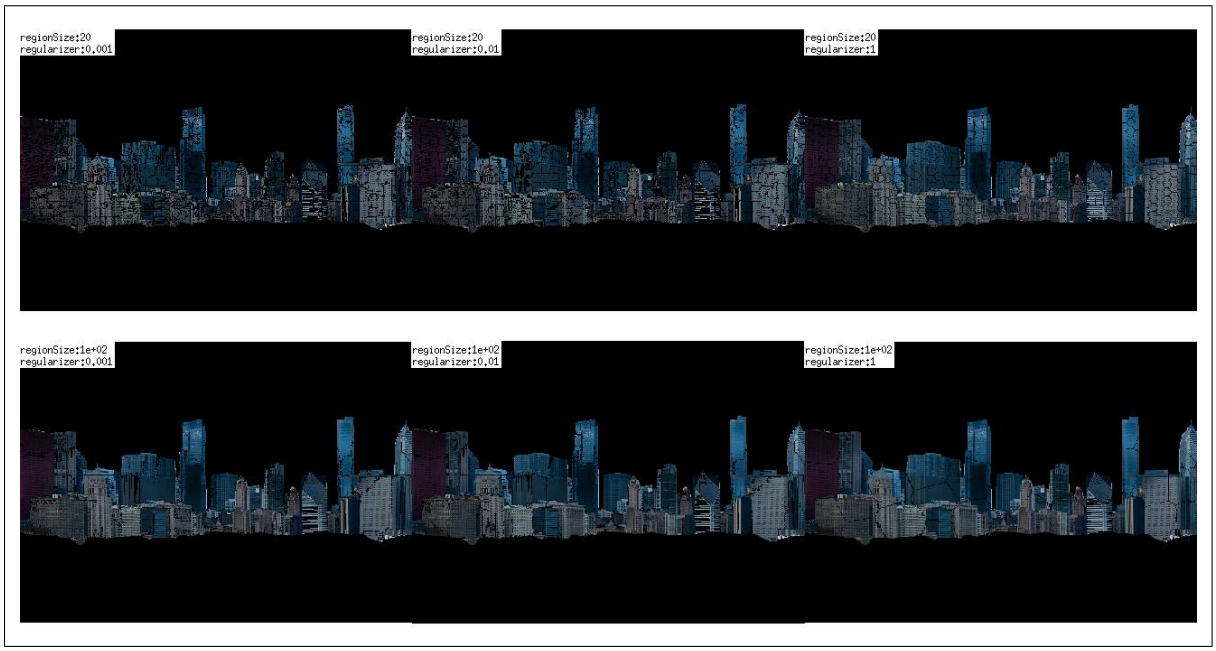

Figure 4.1: SLIC output - In the figure, SLIC outputs are shown for different values of paramters. We use *regionSize* = 100 and *regularizer* = 0.001.

function is used to control the degree to which the difference between the components $Dif(C_1, C_2)$ must be larger than the minimum of the two internal differences, $Int(C_1)$ and $Int(C_2)$ -

$$D(C_1, C_2) = \begin{cases} \text{True} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{False} & \text{Otherwise} \end{cases} \tag{4.7}$$

$MInt(C_1, C_2)$ is defined as -

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{4.8}$$

Here, τ controls the degree to which the $Diff(C_1, C_2)$ must be greater than their internal difference -

$$\tau(C) = \frac{k}{|C|} \tag{4.9}$$

Where $|C|$ denotes the size of C and k is some constant parameter, i.e. for small components larger evidence of boundary is needed. In practice, larger k causes preference for larger components. In our case, we use $k = 500$.

The algorithm to achieve segmentation of V into components $S = (C_1, \ldots, C_r)$ from graph $G = (V, E)$ of n vertices and m edges if as follows -

1. At first, the edge weights are sorted in non-decreasing order $\pi = (o_1, \ldots, o_m)$.

2. Initially, algorithm starts with segmentation S^0 where each vertex v_i ($1 \leq i \leq n$) belongs to its own component, i.e. initially for each component C_i, $|C_i| = 1$.

3. Now, for all the edges in the non-decreasing order by their weights, following procedure is followed to merge the regions (components). S^q is constructed from S^{q-1} as follows -

   Let v_i and v_j be the vertices connected by q^{th} edge in the non-decreasing ordering, i.e. $o_q = (v_i, v_j)$. If v_i and v_j are disjoint components of the S^{q-1} and edge weight w_{o_q} is lesser compared to the internal difference of both these components, then merge these components otherwise do nothing, i.e. -
   Let ${C_i}^{q-1}$ be the component of S^{q-1} containing v_i and let ${C_j}^{q-1}$ be the component containing v_j. Then, if ${C_i}^{q-1} \neq {C_j}^{q-1}$ and $w_{o_q} \leq MInt({C_i}^{q-1}, {C_j}^{q-1})$, then S^q is obtained from S^{q-1} by merging ${C_i}^{q-1}$ and ${C_j}^{q-1}$.

4. Return the final segmentation $S = S^m$.

For an image of N pixels, the running time of this algorithm is $\mathcal{O}(N \log N)$.

Algorithm 5 Graph Based Segmentation [23]

Require: Graph $G = (V, E)$ with n pixels and m edges
Ensure: Segmentation of V into components, $S = (C_1, \ldots, C_r)$
1: Sort E into $\pi = (o_1, \ldots, o_m)$ by non-decreasing weights
2: Start with segmentation S_0 with each v_i is in its own component
3: **for** $q := 1$ **to** m **do**
4: Let ${C_i}^{q-1} \in S^{q-1}$ and $v_i \in {C_i}^{q-1}$ and $v_j \in {C_j}^{q-1}$
5: **if** ${C_i}^{q-1} \neq {C_j}^{q-1}$ and $w_{o_q} \leq MInt({C_i}^{q-1}, {C_j}^{q-1})$ **then**
6: $S^q \leftarrow merge({C_i}^{q-1}, {C_j}^{q-1})$
7: **else**
8: $S^q = S^{q-1}$
9: **end if**
10: **end for**
11: $S \leftarrow S^m$

4.1.3 gPb Detector

gPb [3] contour detector considers a function $Pb(x, y, \theta)$ which predicts the posterior probability of a boundary with with orientation θ at pixel (x, y) [48] by measuring difference in local brightness, color and texture channels. The basic building block of the Pb detector is the computation of an oriented gradient signal $G(x, y, \theta)$ from an intensity image I. At each pixel (x, y), a circular disk of radius σ centered at it, is split into two halves along the diameter angled at θ. Then, the histogram of the intensity values covered by each half is constructed. The gradient magnitude G at location (x, y) is defined by

Figure 4.2: Graph based segmentation output - In the figure, Graph based segmentation outputs are shown for different values of threshold function parameter k. We use $k = 500$.

χ^2 distance between the two half-disk histograms g and h as in Equation 3.11 -

$$\chi^2(g,h) = \frac{1}{2}\sum_i \frac{(g(i)-h(i))^2}{g(i)+h(i)} \tag{4.10}$$

The, the second order Savitzky-Golay filtering [53], i.e. a cylindrical parabola, its axis orientation along θ is fit to a local $2D$ window surrounding each pixel and replacing the response at the pixel by the fit. A strong gradient response means a pixel is most likely to lie on the boundary of two regions. The Pb detector combines the gradient response obtained by transforming the image into four separate channels and processing each one independently. The first three correspond to the channels of the CIE Lab color space [63], which are referred as brightness, color a and color b channels. The fourth channel is a texture channel, which assigns each pixel a *texton* as explained in Section 3.1.2. Then, the gradient response is computed from the image where each pixel is assigned a texton value.

In order to detect both fine and coarse structures, the gradients are considered at three different scales — $[\frac{\sigma}{2}, \sigma, 2\sigma]$ for each of the four channels, brightness, color a, color b and texture. These local cues are then combined into a single multiscale oriented signal -

$$mPb(x,y,\theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma_{(i,s)}}(x,y,\theta) \tag{4.11}$$

where s indexes scale, i indexes feature channels (brightness, color a, color b or texture) and $G_{i,\sigma_{(i,s)}}(x,y,\theta)$ measures the histogram difference in channel i between two halves of a disk of radius $\sigma_{(i,s)}$ centered at (x,y) and divided by a diameter at angle θ. The parameter $\alpha_{(i,s)}$ weighs the relative contribution of each feature signal. Practically, θ is sampled at eight equally spaced orientations in the interval $[0,\pi)$. Taking the maximum of response at each of these orientation gives the measure of boundary strength at each pixel -

$$mPb(x,y) = \max_\theta \{mPb(x,y,\theta)\} \tag{4.12}$$

An optional non-maximum suppression step [14] produces thinned, real-valued contours.

Final step is globalization using spectral clustering. As an input to spectral clustering, a sparse symmetric affinity matrix W is constructed using the *intervening contour cue* [26, 27, 42], i.e. the maximum value of mPb along the line joining two pixels. All the pixels i and j are connected within a fixed radius r with affinity -

$$W_{ij} = \exp(-\max_{p \in \bar{ij}}\{mPb(p)\}/\rho) \tag{4.13}$$

where $\bar{ij}$ is the line segment connecting i and j and, ρ is a constant. We set $r = 5$ and $\rho = 0.1$.

In order to introduce global information, $D_{ii} = \sum_j W_{ij}$ is introduced and then the system $(D - W)v = \lambda Dv$ is solved for generalized eigen vectors $(v_0, v_1, \ldots, v_n)$, corresponding to the $n+1$ smallest eigen values $0 = \lambda_0 \leq \ldots \leq \lambda_n$. We use $n = 16$.

Eigen vectors themselves carry contour information. Thus, treating each eigen vector v_k as an image, the Gaussian directional derivative filters are convolved at multiple orientations θ, obtaining oriented signals $\{\bigtriangledown_\theta v_k(x, y)\}$. Taking derivatives in this manner ignores the smooth variations in the image only considering salient features in the image. The information thus provided from different eigen vectors is then combined to provide the "spectral" component of the boundary detector -

$$sPb(x, y, \theta) = \sum_{k=1}^{n} \frac{1}{\sqrt{\lambda_k}} . \bigtriangledown_\theta v_k(x, y) \tag{4.14}$$

where the weighing by $\frac{1}{\sqrt{\lambda_k}}$ is motivated by physical interpretation of the generalized eigen value problem as a mass-spring system [8].

The signals mPb and sPb convey different information, as the mPb fires at all edges while sPb extracts the most prominent curves in the image. The simple linear combination extracts the benefits from both behaviors. The final *globalized probability of boundary* is then written as a weighted sum of local and spectral signals -

$$gPb(x, y, \theta) = \sum_s \sum_i \beta_{i,s} G_{i,\sigma_{(i,s)}}(x, y, \theta) + \gamma . sPb(x, y, \theta) \tag{4.15}$$

Subsequently gPb is rescaled using a sigmoid to match a probabilistic interpretation.

4.2 Automatic Approach for Skyline Segmentation

In the automatic setting, we start with a baseline segmentation, and refine it using our method. The initial segmentation method is used to estimate the seeds which are then used as input for the interactive segmentation methods described in the earlier section.

For the initial segmentation we use either SLIC [1], graph-based segmentation [23], or gPb regions [3]. The way we estimate seed regions is as follows -

1. A skyline is partitioned into N vertical uniform divisions of the same size.

2. The largest K segments are selected from each such division of the baseline output.

Method	SLIC [1]	Graph based [23]	gPb [3]
Initial	24.56%	20.17%	26.35%
Tiered MRF	27.22%	25.86%	31.51%
Rectangle MRF	**27.33%**	**27.87%**	32.79%
Refined MRF	27.30%	27.42%	**33.13%**

Table 4.1: Performance in the automatic setting. Starting from various baseline segmentation algorithms such as SLIC, graph-based segmentation, and gPb regions, we perform an automatic labeling. The table shows the MAO scores for various the methods. Seeds obtained from gPb regions offer the best performance.

Buildings in a skyline are layered due to varying depth of buildings from camera. The uniform selection of segments is effective in selecting buildings in all layers. Generally, a skyline has $2-3$ such layers. In all experiments we set $N = 20$ and $K = 2$. Thus, we select $N \times K = 40$ uniformly distributed largest segments from the output of a segmentation algorithm and label these as different buildings. This serves as a *baseline*. A number of pixels within the segments are used as seeds for the interactive methods.

Table 4.1 compares various methods in the automatic setting. The *refined MRF* and *rectangle MRF* give significant performance boost over all these methods, an average 40% improvement over graph-based segmentation and 25% improvement over gPb, in few images showing as much as 60% improvement over the baseline. The running time of these methods are similar to those described in Table 3.2.

Among various low-level methods for segmentation, SLIC and graph-based use only color, while gPb uses both texture and color, hence the improved baseline. Nonetheless, our method improves over all of these methods mainly due to the utilization of shape priors. In an interactive setting a user may use this as an input to guide effort in correction. The results for some images using the automatic approaches are shown in the last row of Figure 4.3. Our method may be made fully automatic using methods such as [33, 34] that can estimate the upper and lower boundaries. In our experiments we found that although these methods are fairly good, they still make mistakes. Hence to avoid confounding factors for mistakes in our analysis, we choose to include the boundary as part of the input for the automatic segmentation methods.

4.2.1 Challenges

The automatic setting presents multiple challenges that make it very difficult to accurately segments the skyline -

1. In automatic setting, the exact number of labels (buildings) is not known, thus we only approximate the buildings to be $N \times K$. Thus, over-segmentation (more number of buildings than in the ground truth) or under-segmentations (less number of buildings than in ground truth) cannot be avoided.

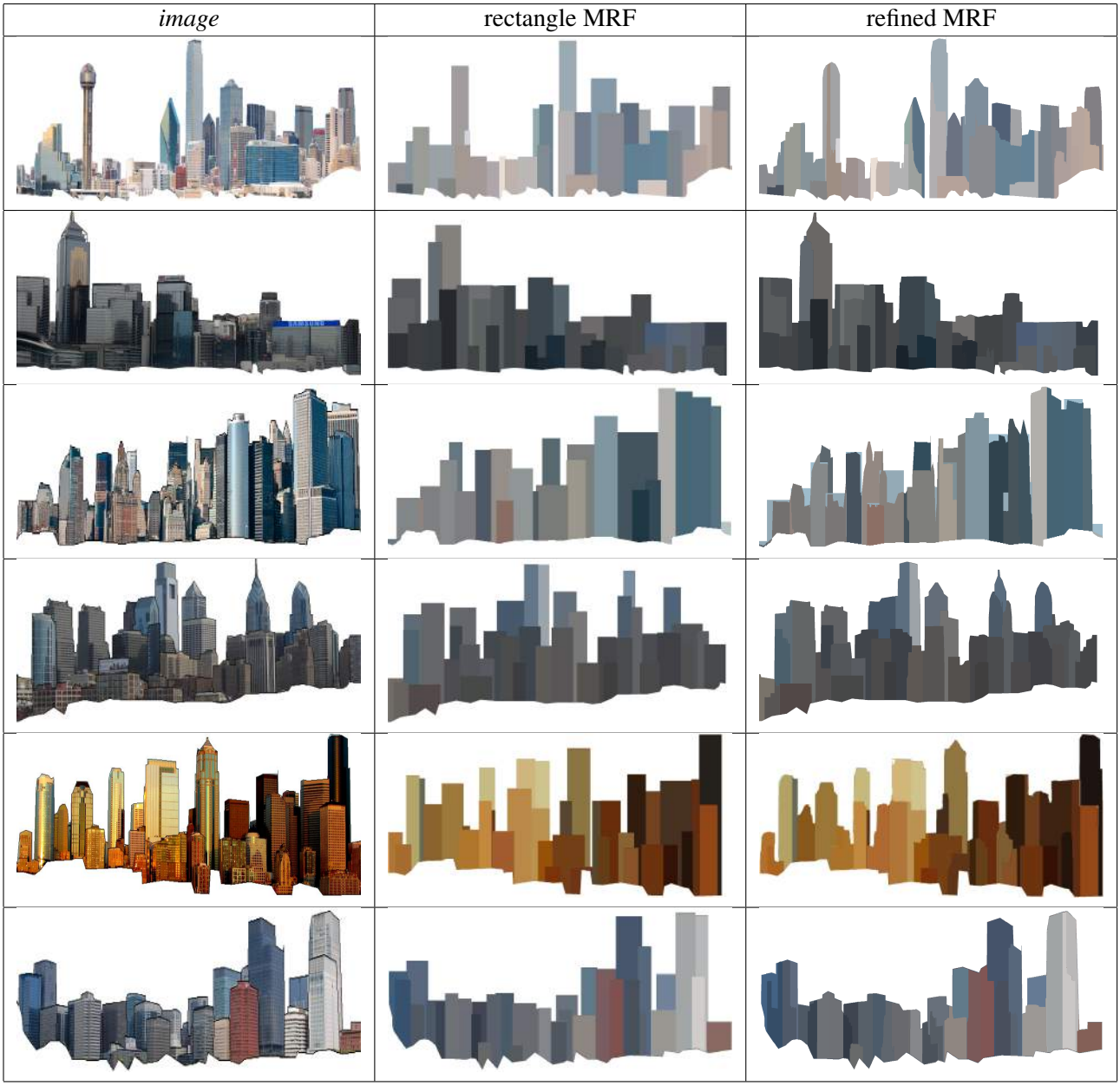

Figure 4.3: Automatic Skyline segmentation results. In first column the original image is shown, while in the second and third column *rectangle MRF* and *refined MRF* outputs are shown respectively.

2. Outputs of the baseline methods — Graph based Segmentation [23] and gPb [3] often give salt-and-pepper noise, i.e. very small segments only adding noise. This noise leads to inaccurate ordering between segments according to their size while selecting largest K segments in each of the vertical partition. We solve this issue by running a *Connected Component Analysis* [20] on the segmented output and removing very small segments.

3. Biggest challenge an automatic setting faces is that of intra-building variation and subsequent choice of incorrect seeds. As explained Section 1.2.1, different facades of the same buildings look different due to sunlight. Thus, the baseline low-level automatic segmentation methods segment these buildings into different segments. Additionally, color and texture model constructed from the segments selected as seeds fails to connect the different segments belonging to the same building as one. That results into over-segmentation because each building is generally split into various segments.

Even though automatic setting faces all these challenges, we still get a hefty 25% average improvement over baseline segmentation methods due to incorporation of shape priors and exploiting overall 'tiered' structure of the skyline.

4.3 Geometric Context for Initial Boundaries

Even in the Automatic setting of our approach, we assume that the initial upper and lower boundaries delineating buildings region from the image is provided by the user. We make this assumptions because, automatic methods [33, 34] for obtaining such regions work reasonably well and, while evaluating our approach, the MAO will also include the failure of obtaining such regions using these methods creating unnecessary noise in the statistics. In this section, we have a look at the automatic method [33, 34] for obtaining such geometric regions and its performance on our dataset.

Hoiem et al.[34] propose a method to label an image of an outdoor scene into coarse geometric classes (labels). The main classes and subclasses considered are —

1. **"support"** - Support surfaces are roughly parallel to the ground and could potentially support a solid object, e.g. road surfaces, lawns, lakes, etc.

2. **"vertical"** - Vertical surfaces are solid surfaces that are too steep to support an object, e.g. walls, trees, people, cliffs, etc. "vertical" class is further divided into three subclasses — "left", "right" and "center" depending on the angle to which viewer faces the vertical surface. Furthermore, the non-planar surfaces are divided into two classes - "porous" and "solid". Porous surfaces include tree leaves, shrubs, etc. while the Solid surfaces are non-planar vertical surfaces having solid continuous surface like automobiles, people, tree trunks, etc.

3. **"sky"** - The sky is regions corresponding to open air and clouds.

4.3.1 Cues for Labeling Surfaces

To label surfaces, multiple different kind of cues are used like material, location, texture gradient, shading, vanishing points, etc. All the cues are computed and the classifier decides which ones are actually useful.

Following listed are the cues used -

1. **Location -** For recovering 3D layout of an image, its 2D representation itself(position in the image) is a very strong cue, e.g. Ground tends to be low in the image and sky tends to be high. The x-position of an image can tell a lot about subclasses, e.g. planar surfaces facing left tends to be on the right in an image and vice versa. Thus, pixel locations are normalized by width and height of the image, then cues such as mean, $10th$ and $90th$ percentile of the x and y position as well as number of superpixels and pixels normalized by total area in a segment are computed.

2. **Color -** By modeling color, materials and objects belonging to certain classes can be implicitly identified, e.g. sky is usually blue or white; support segments are often green (trees) and brown (dirt). The color is represented in HSV (to measure hue and "grayness") and RGB color spaces (to extract "blueness" or "greenness").

3. **Texture -** Texture also relates more directly to the geometric class through properties of surfaces in perspective, such as that a vertical plane will tend to have more vertically oriented textures than a horizontal plane. The texture representation includes filter responses and histogram of maximum responses.

4. **Perspective -** Knowledge of the vanishing lines of a plane completely specifies its 3D orientation relative to the viewer [31]. Perspective representation includes a "soft" estimate and an explicit estimate of vanishing points.

To extract above mentioned cues, some knowledge is necessary. The solution is to slowly build structural knowledge of the image — from pixels to superpixels to multiple segmentations. Approach is to compute multiple segmentations is based on simple cues and then use the increased spatial support provided by each segment to better evaluate its quality. The segmentation method groups superpixels into larger continuous segments using pairwise same-label likelihoods learned from training images.

Finally, the likelihoods of each segment being homogeneous and of each possible label is computed. Then, the estimates produced by different segmentations are combined in probabilistic manner. A segment is called *Homogeneous* if all contained superpixels have same label. To find the probabilities, classifiers are trained on various cues mentioned above of training images.

4.3.2 Estimate of Initial Boundaries

Skyline images from the dataset **Skyline-12** are geometrically labeled as described above into various geometric main and sub classes. Intuitionally, these various classes are divided into three initial hori-

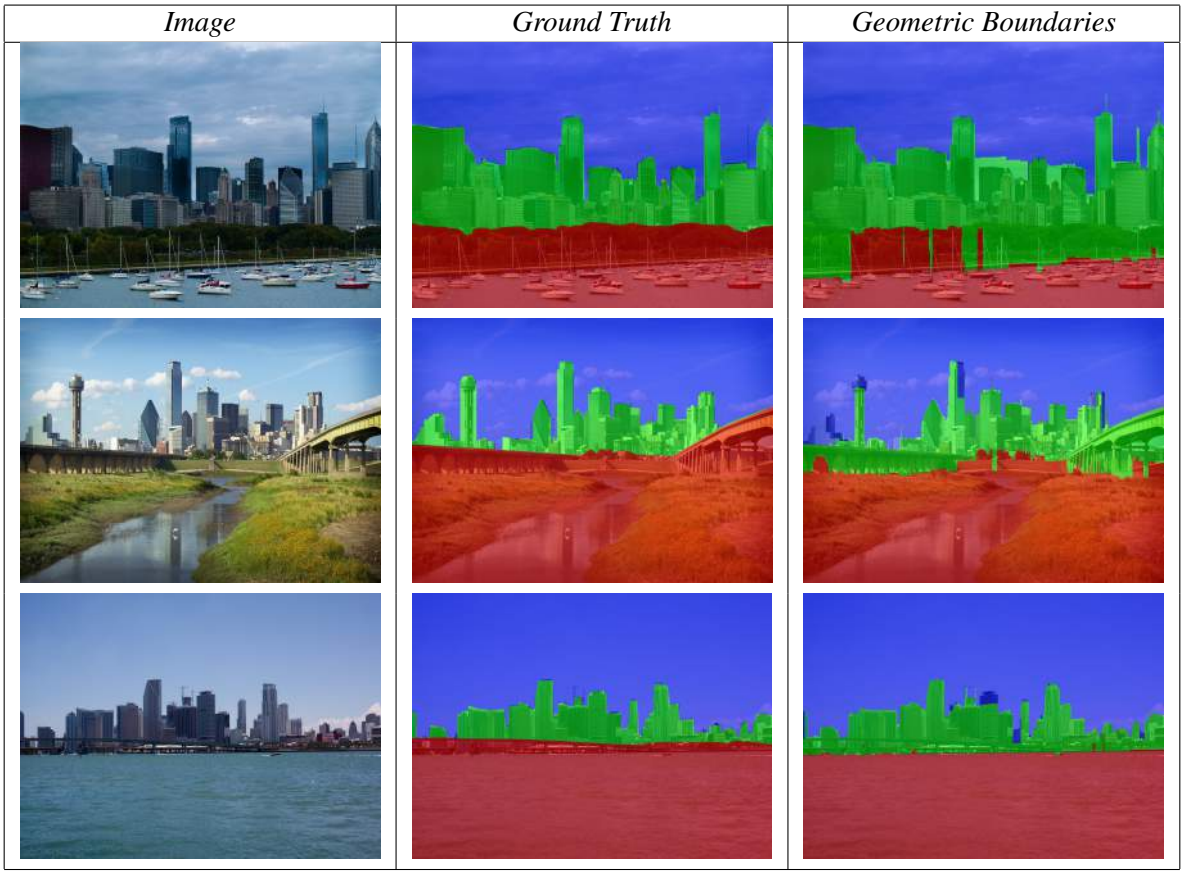

Figure 4.4: Geometric Context Boundaries - In the figure shown are few sample images, Ground truth for the initial upper and lower boundaries and the corresponding Geometric context boundaries obtained using [33, 34].

zontal tiers (Top, Middle and Bottom) as shown in Table 4.2. As is obvious, the geometric label "sky" is assigned to top tier and the geometric label "support" is assigned to bottom tier consisting ground and water bodies. The assignment of geometric label "vertical" is somewhat subjective, as the label itself with its sub classes "left", "right", "center" and "solid" is assigned to the middle tier consisting of buildings facing in all three directions as well as, solid non-planar surfaces for the sake of continuity. The sub class "porous" is assigned to the bottom tier because, we do not consider trees, shrubs and like for the segmentation even thought they are vertical surfaces.

Finally, to get three continuous horizontal layers tiered labeling [24, 66] (described in Section 3.2) is used. The three horizontal tiers obtained in this ways are shown in Figure 4.4 for few images from the dataset along with the ground truth provided with the dataset. As we can see, the geometric boundaries look fairly accurate. The errors occurred are more of a subjective nature, e.g. bridge in the second image, should it be included in middle tier or bottom tier?

To formally evaluate on the dataset, we consider percentage of columns where the row cutting the tier is within $K\%$ margin from the ground truth, i.e. $\%$ of columns where absolute difference between

Tier	Geometric Classes
Top	Sky
Middle	Vertical, Vertical:left, Vertical:right, Vertical:center, Vertical:Solid
Bottom	Vertical:Porous, Support

Table 4.2: The table shows division of Geometric Classes obtained into three horizontal tiers. "sky" is assigned to top tier, "support" is assigned to bottom tier and except "porous" sub-class which includes trees and shrubs, every sub-class of "vertical" is assigned to the middle tier.

ground truth and geometric context boundary is less than $K\%$ of rows. Here, the $\%$ of the total number of rows is considered instead of an absolute number of rows to avoid the biases happening due to varying image sizes. The Results are shown in Table 4.3. As we can see, for tighter values of K, the upper boundary is found much more accurately than the lower boundary.

Boundary	K=0.5%	K=1%	K=3%	K=5%	K=10%
Upper Boundary	41.7%	57.5%	70.0%	74.0%	80.0%
Lower Boundary	14.4%	22.9%	44.5%	59.2%	79.2%

Table 4.3: The table shows the evaluation of the upper and boundary obtained using gerometric context [33, 34] for different values of K.

Chapter 5

Conclusions and Future Work

5.1 Beyond Rectangles and Skylines

Buildings in skyline can be represented by the the "*Rectangular*" shape. Though, in practice many tiered settings can be found where the objects of same overall shape are placed in front of each other, i.e. they are occluding each other. Few of the cases are shown in Figure 5.1 like cars in a parking lot, pine trees in a forest and oranges arranged in a heap. Our approach can be generalized for these cases. These cases are more challenging than the buildings in skylines because -

1. For more complex shapes like cars and circles computing sum inside the shape requires more than $\mathcal{O}(1)$ computations required for rectangles using integral images. Then , we have to compute column-wise integral images to compute sums in each column of the specific shape. Then, the complexity becomes $\mathcal{O}(mn^3)$ for a region of size $m \times n$.

2. Buildings in a skyline create predictable occlusion patterns. But, for more complex shapes like cars, the occlusion patterns can be completely unpredictable depending on how much and in which direction one car occludes another. If a car occludes another car almost completely, then the remaining segment of the car behind cannot belong to the generic shape of car leading to smaller or bigger segments than intended.

3. Finally, buildings in a skyline display relatively large variety of colors and textures. Thus, each building can be distinguished by another using their colors and textures almost in all cases. On the contrary, almost all the pines in a pine tree have similar colors and textures making them very hard to perceive individually even by humans. Oranges are all of the same color making them very difficult to distinguish.

4. We use *spatial distance* component while constructing unary potential for a building which emphasizes the rectangular shape of the building in the early stage itself. But, in the case of complex shapes like cars, it becomes very hard to construct such a component accurately affecting the final performance.

Even with all these challenges, we describe in the below section a segmentation enforcing non-rectangular shape prior. In the following section, variation of *rectangle MRF— Triangle MRF* for triangular shape, as the name suggests is described.

Figure 5.1: In the figures, some of the tiered structures for generic cases are shown such as cars, pine trees and oranges.

5.1.1 Triangle MRF

As shown in Figure 5.2, we can define a generic triangle skeleton with two right angles triangles with their corners in *south-east* and *south-west* facing each other along a side.

For triangular regions *se* and *sw*, we can use triangular integral images introduced in [49]. We need to compute south west(*sw*) and south east(*se*) triangular images. We can compute these images in one scan as following -

To compute sum in the sw triangle, ti_{sw}, we traverse the image from left to right and top to bottom in that order. Then, we can compute south west triangular integral image as follows -

$$s_{sw}(x,y) = s_{sw}(x-1,y) + i(x,y) \tag{5.1}$$

$$ti_{sw}(x,y) = ti_{sw}(x-1,y-1) + s_{sw}(x,y) \tag{5.2}$$

Figure 5.2: Left Image is an example of tiered structure of pine trees. As shown in right figure, we assume each pine tree to be composed of two right angle triangles.

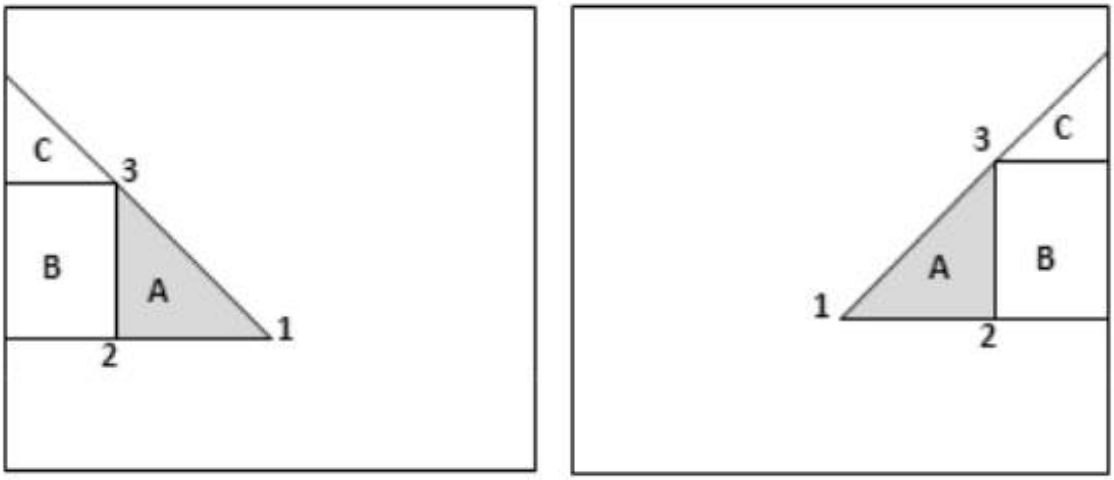

Figure 5.3: In left image, a case of south west triangle and in right image, south east triangle is shown. To get sum of values inside A, we need to subtract the sum of values in B and C from the triangular integral value at A.

Here, $s_{sw}(x, y)$ is a cumulative sum, $i(x, y)$ is image intensity at row y and column x.

Similarly, to compute ti_{se}, we traverse the image from right to left and top to bottom in that order. Then, we can compute south east triangular integral image as follows -

$$s_{se}(x, y) = \quad s_{se}(x + 1, y) + i(x, y) \tag{5.3}$$

$$ti_{se}(x, y) = \quad ti_{se}(x + 1, y - 1) + s_{se}(x, y) \tag{5.4}$$

Finally, as shown in Figure 5.3, sum of values in triangle A in south west case is equal to $ti_{sw}(1) - (ii(2) - ii(3) + ti_{sw}(3))$, where $ii(2)$ is rectangular integral image value at 2.

The output for a sample toy image is shown in Figure 5.4.

5.2 Conclusion

We presented a user-guided approach for extracting the structure of buildings within a skyline image. Our *shape-constrained MRF* approach lets us exploit the shape priors of the buildings and the tiered structure, allowing more accurate parsing. Compared to standard approaches for optimizing MRFs such as α-expansion, our *rectangle MRF* method is significantly faster, taking a few seconds to label a 3 mega-pixel image. Further refinement within the constraints of the rectangle improves accuracy. This coarse-to-fine approach for parsing may be used in other settings where an explicit search over shapes is faster than graph-cuts. Our preliminary results on improving automatic segmentation methods using shape priors are also promising. Finally, the **skyline-12** dataset consisting of 120 high resolution images with detailed annotations, and code for reproducing the results presented, is available for download at the author's website.

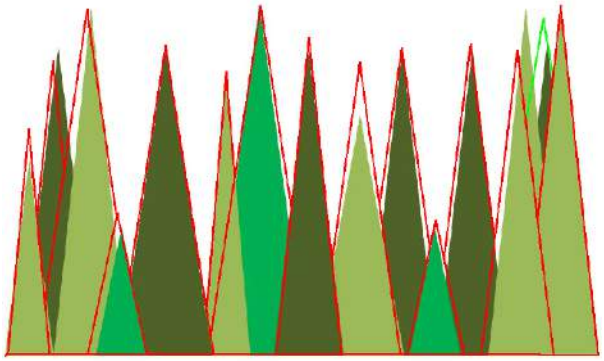

Figure 5.4: In the figure shown is an output of *Triangle MRF* for a sample toy image.

5.3 Future Work

Our method for parsing skylines using shape-constrained MRFs can be extended for generic shapes as explained in the section 5.1.1. As shown in the previous section, *rectangle MRF*can be generalized to generic shapes in various practical situations for shapes “cars”, “pines”, “oranges”, etc. While we have been considering the situations where only one shape is exhibiting ‘tiered’ structure, we can loose our restrictions on this assumption, i.e. we can apply our method to ‘tiered’ structures in which objects of various shapes are occluding each other. In that case, segmentation using shape-constrained MRFs will have to be preceded by shape matching, i.e. cumulative initial segment formed using foreground pixels has to be matched to either of the shapes from the shape library and then the corresponding shape-constrained MRF can be used to segment the object. Similar to the shape library, we can create the library of buildings from the dataset **Skyline-12**. These buildings can be indexed, trained using local features and neighborhood information to segment a skyline followed by recognising each and every building in it.

Until now we have considered possible extensions to our method to achieve various kinds of outputs. We can also use the output of skyline segmentation using our method to various applications. For example, there is an interesting graphics application of our approach. As we know the *depth* of the each building deduced by the order in which the buildings are processed and we have the each building segmented, we can create a 3D pop-up of the skyline using single image itself using methods such as billboarding, pop-up using surface layout [32], etc.

Publications

Publications

- Rashmi Tonge, Subhransu Maji and C. V. Jawahar,
  Parsing World's Skylines using Shape Constrained MRFs,
  In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2014, Columbus, Ohio.

Other Publications

- Nisarg Raval, Rashmi Tonge and C. V. Jawahar,
  Efficient Evaluation of SVM Classifiers using Error Space Encoding *(Oral)*,
  In Proceedings of the 22nd International Conference on Pattern Recognition (ICPR), 2012, Stockholm, Sweden.

- Nisarg Raval, Rashmi Tonge and C. V. Jawahar,
  Image Retrieval using Eigen Queries,
  In Proceedings of the 11th Asian Conference on Computer Vision (ACCV), 2012, Daejeon, Korea.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels compared to state-of-the-art superpixel methods. *IEEE PAMI*, 2012.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE PAMI*, 2011.

[4] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[5] T. Asano, D. Z. Chen, N. Katoh, and T. Tokuyama. Efficient algorithms for optimization-based image segmentation. *Int. J. Comput. Geometry Appl.*, 2001.

[6] G. Baatz, O. Saurer, K. Kser, and M. Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *ECCV*, 2012.

[7] S. A. Barker and P. J. W. Rayner. Unsupervised image segmentation using Markov random field models. *j-LECT-NOTES-COMP-SCI*, 1997.

[8] S. Belongie and J. Malik. Finding boundaries in natural images: A new method using point descriptors and area completion. In *In Proc. 5th Euro. Conf. Computer Vision*, 1998.

[9] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B"*, 1986.

[10] A. Blake, C. Rother, M. Brown, P. Pérez, and P. H. S. Torr. Interactive image segmentation using an adaptive GMMRF model. In *ECCV*, 2004.

[11] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *IEEE PAMI*, 2004.

[12] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998.

[13] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE PAMI*, 2001.

[14] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1986.

[15] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *CVPR*, 2001.

[16] D. Cremers, S. J. Osher, and S. Soatto. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. *Int. J. Comput. Vision*, 2006.

[17] F. C. Crow. Summed-area tables for texture mapping. In *SIGGRAPH*, 1984.

[18] P. Das, O. Veksler, V. Zavadsky, and Y. Boykov. Semiautomatic segmentation with compact shape prior. *Image Vision Comput.*, 2009.

[19] A. Delong and Y. Boykov. Globally optimal segmentation of multi-region objects. In *ICCV*, 2009.

[20] M. B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *J. ACM*, 1992.

[21] C. Elkan. Using the triangle inequality to accelerate k-means. In *In proceedings International Conference of Machine Learning*, 2003.

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[23] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.

[24] P. F. Felzenszwalb and O. Veksler. Tiered scene labeling with dynamic programming. In *CVPR*, 2010.

[25] L. R. Ford and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *CANADIAN JOURNAL OF MATHEMATICS*, 1957.

[26] C. Fowlkes and J. Malik. How much does globalization help segmentation. Technical report, UC Berkeley, 2004.

[27] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: combining patch-based and gradient-based approaches. In *In Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 2003.

[28] D. Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. In *CVPR*, 2005.

[29] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1984.

[30] D. Greig, B. Porteus, and H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B*, 1989.

[31] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[32] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 2005.

[33] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005.

[34] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 2007.

[35] H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2003.

[36] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *In European Conference on Computer Vision*, 1998.

[37] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. 2004.

[38] P. Kohli, L. Ladický, and P. H. Torr. Robust higher order potentials for enforcing label consistency. *IJCV*, 2009.

[39] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2007.

[40] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[41] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time (1+)-approximation algorithm for k-means clustering in any dimensions. In *FOCS*, 2004.

[42] T. Leung and J. Malik. Contour continuity in region based image segmentation. In *In Proc. ECCV, LNCS 1406*, 1998.

[43] M. E. Leventon, W. E. L. Grimson, and O. D. Faugeras. Statistical shape influence in geodesic active contours. In *CVPR*, 2000.

[44] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.

[45] S. P. Lloyd. Least squares quantization in pcm. Technical report, Bell Laboratories, 1957.

[46] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 2001.

[47] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *BMVC*, 2007.

[48] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE PAMI*, 2004.

[49] H. Proenca and S. Filipe. Combining rectangular and triangular image regions to perform real-time face detection. In *In the proceedings of 9th International Conference on Signal Processing*, 2008.

[50] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004.

[51] S. Roy and I. J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *In Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[52] M. A. Ruzon and C. Tomasi. Alpha estimation in natural images. In *CVPR*, 2000.

[53] A. Savitzky and M. J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 1964.

[54] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 2002.

[55] D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical report, 2006.

[56] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2007.

[57] R. H. Swendsen and J. S. Wang. Nonuniversal, critical dynamics in monte carlo simulations. *Phys. Rev. Lett.*, 1987.

[58] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2008.

[59] O. Veksler. *Efficient Graph-based Energy Minimization Methods in Computer Vision*. PhD thesis, 1999.

[60] O. Veksler. Star shape prior for graph-cut image segmentation. In *ECCV*, 2008.

[61] O. A. Verevka and J. W. Buchanan. Local k-means algorithm for color image quantization. In *Graphics/Vision Interface*, 1995.

[62] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 2004.

[63] G. Wyszecki and W. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley and Sons, New York, NY, 1982.

[64] J. Xu, M. D. Collins, and V. Singh. Incorporating topological constraints within interactive segmentation and contour completion via discrete calculus. In *CVPR*, 2013.

[65] C. Zhang, L. Wang, and R. Yang. Semantic segmentation of urban scenes using dense depth maps. In *ECCV*, 2010.

[66] Y. Zheng, S. Gu, and C. Tomasi. Fast tiered labeling with topological priors. In *ECCV*, 2012.

[67] C. L. Zitnick and S. B. Kang. Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision*, 2007.