PERSON DETECTION AND TRACKING USING OMNIDIRECTIONAL CAMERAS, AND RECTANGLE BLANKET PROBLEM

by

Barış Evrim Demiröz

B.S., Computer Engineering, Istanbul Technical University, 2005

M.S., Molecular Biology, Genetics and Biotechnology, Istanbul Technical University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2019

ACKNOWLEDGEMENTS

I would like to thank my beloved partner Aslı Sabancı for her support, guidance and endurance to my endless rantings; she has spent a lot of effort to make this thesis happen. I would also like to thank İsmail Arı for being a true friend and a source of inspiration as a hard worker with lots of talent. I would like to thank my PhD supervisor Lale Akarun for the guidance, she went to great lengths to support me both inside and outside of the PhD program; Kuban Altınel for inspiring me about our work with his diligence, energy and commitment; Yalın Başztanlar for all the advice, sharing the details of their previous work and being humble; Yunus Emre Kara for his friendship and shedding light to complex topics with his simple explanations; my comrades Orhan Sönmez, Deniz Akyıldız, Barış Kurt, Alp Kındıroğlu and Oğulcan Özdemir for their companionship; Cem Ersoy for being an extremely student friendly manager and professor, and increasing everyone's quality of life in the department; Cem Meydan and Onur Babacan for the fruitful and mostly fruitless discussions. Finally, I would like to thank Ali Vahit Şahiner for accepting me as his PhD student in the first place.

This study has been partly supported by the Turkish Ministry of Development under the TAM Project number DPT2007K120610.

ABSTRACT

PERSON DETECTION AND TRACKING USING OMNIDIRECTIONAL CAMERAS, AND RECTANGLE BLANKET PROBLEM

Person detection and tracking can provide the crucial analysis needed to avoid accidents with autonomous machinery, optimize environments for efficiency and assist the elderly. Omnidirectional cameras have a large field of view that allow them to cover more ground at the expense of resolution. Omnidirectional cameras can decrease setup, maintenance and computational costs by reducing the number of cameras and the bandwidth required. Computer vision methods developed for conventional cameras usually fail for omnidirectional cameras due to their different image formation geometry. In this thesis, first, a novel dataset for person tracking in omnidirectional cameras is introduced. The dataset, namely BOMNI, contains 46 videos of persons moving inside a room; where the bounding boxes and the identity of the persons are annotated at every frame. Second, a generative Bayesian framework is developed for coupling person tracking and fall detection. The method is evaluated on BOMNI dataset, producing 93% tracking accuracy and fall detection within a few frames of the event. Third, a similar method for multiple person tracking is developed and evaluated on BOMNI dataset. The method reaches 86% tracking accuracy, increasing a previous approach by 18%. Fourth, a discriminative method for person detection is presented. Also a novel structure called *Radial Integral Image* that speeds up feature extraction step is introduced. This method achieves state of the art detection performance on IYTE dataset: 4.5% miss rate for one false positive per image. Finally, the problem of representing a shape with multiple rectangles, *Rectangle Blanket Problem*, is formulated as an integer programming problem and a branch-and-bound scheme is presented along with a novel branching rule to solve it optimally. This problem is encountered in the earlier sections of this thesis, but it is a general problem that is present in the literature.

ÖZET

TÜMYÖNLÜ KAMERALARDA İNSAN TESPİTİ VE TAKİBİ, VE DİKDÖRTGEN BATTANİYESİ PROBLEMİ

Otonom makinelerle kaza önlenmesi, verimlilik için ortamların iyileştirilmesi ve yaşlılara destek verilmesi için gerekli analizi insan tespiti ve takibi sağlayabilir. Tümyönlü kameralar geniş görüş alanları sayesinde, çözünürlükten feda ederek, daha fazla bir alanı kapsar. Tümyönlü kamera kullanımı gerekli kamera sayısını ve bant genişliğini düşürerek kurulum, işletme ve hesaplama maliyetlerini düşürebilir. Fakat, geleneksel kameralar için geliştirilmiş yapay görü yöntemleri, görüntünün oluşum geometrisi farklı olduğu için genellikle tümyönlü kameralarda başarısız olur. Bu tezde, ilkin, insan takibi için özgün bir veri kümesi, BOMNI, tanıtılmaktadır. BOMNI, insanların bir odanın içinde hareket ettiği 46 video içerir. İnsanların sınırlayıcı kutuları ve kimlikleri her karede işaretlenmiştir. İkinci olarak, insan takibi ve düşme tespitini bağlayan bir üretimsel Bayesian model sunulur. BOMNI veri kümesi üzerinde değerlendirilen bu yöntem, %93 takip isabetliliği ve çoğunlukla bir kaç kare içerisinde düşme tespiti başarımı elde etmiştir. Üçüncü olarak, benzer bir yöntem birden fazla insan takibi için geliştirilmiş ve BOMNI veri kümesi üzerinde değerlendirilmiştir. Geçmiş yönteme kıyasla bu yöntem %18 artış sağlayarak %86 takip isabetliliği elde etmiştir. Dördüncü olarak, insan tespiti için ayrımcı bir yöntem önerilmiştir. Ayrıca, öznitelik çıkarımını hızlandırmak için özgün bir yapı, *Dairesel İntegral Görüntüsü*, sunulmuştur. Bu yöntem IYTE veri kümesi üzerinde bilinen en yüksek insan tespiti başarımını elde etmiştir: görüntü başına bir yanlış pozitif için %4.5 ıska oranı. Son olarak, bir şekli birden fazla dikdörtgen ile ifade etme problemi, *Dikdörtgen Battaniyesi Problemi*, bir tamsayı programlama problemi olarak formüle edilmiş ve en iyi çözümü bulan bir dallan-ve-sınırla yaklaşımı ve yeni bir dallanma kuralı önerilmiştir. Bu problem ile tezin önceki bölümlerinde karşılaşılmıştır fakat literatürde varolan daha genel bir problemdir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A^c	Ideal foreground segmentation obtained from camera c
$A^c_{k,f}$	Binary image generated by putting the human silhouette in fall state f at location k in camera c.
B^c_t	Binary foreground segmentation obtained from camera c at time t
F_t	Fall sate of the person at time t. $F_t = 1$ if the person is fallen, $F_t = 0$ otherwise.
G	Number of discrete physical locations
H	Height of an image
I	An image
$\tilde{I}$	An integral image
I^c_t	The image obtained from camera c at time t
$\ell(\cdot)$	Lower bound for the pricing problem
L_t	Person's location at time t
$\mathbf{r}$	A rectangle
$\mathbf{Q}$	Probabilistic occupancy map
W	Width of an image
x_j	Rectangle selection binary variable. $x_j = 1$ if rectangle j is selected.
X^k	The occupancy state of the location k. $X^k = 1$ if location k is occupied, $X^k = 0$ otherwise.
μ	Dual variables corresponding to the maximum rectangle constraint of the linear program
π	Dual variables corresponding to the overlap constraints of the linear program
$\Psi(\cdot)$	Pseudodistance function between two binary images
$\Phi^{k,s}_t$	Maximum probability of observing the images and the trajectory ending up at location k and fall state s at time $t+1$
$\otimes$	Element-wise multiplication

$\langle\cdot\rangle_{\mathrm{Q}}$	Expectation under Q distribution

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
BIP	Binary Integer Programming
CLEAR	Classification of Events, Activities and Relationships workshop
CNN	Convolutional Neural Networks
CSA	Constrained Simulated Annealing
FAST	Fast Adaptive Silhouette Area Based Template matching
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
ICF	Integral Channel Features
KSP	K-shortest path
LUV	Also known as CIE 1976, a color space defined by International Commission on Illumination
LP	Linear Programming
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
POM	Probabilistic Occupancy Map
RBP	Rectangle Blanket Problem
RICF	Radial Integral Channel Features
SF	Split & Fit
SVM	Support Vector Machine

1. INTRODUCTION

We are entering an era where humans and intelligent systems will co-exist in factory and office environments, smart cities and roads, and home environments. Intelligent systems need to detect and track humans in order to protect them, serve them and communicate with them. Person detection and tracking is a crucial step for surveillance, autonomous vehicles and assisted living applications. Many applications require the detection of multiple persons using video sensors, as well as the tracking of detected persons, their re-identification in different camera views, and at later times, the classification of their actions. The detection of persons and tracking them in an environment using minimal computational resources is a challenging first step.

Many person detection and tracking studies use conventional perspective cameras. In that case, multiple cameras are needed to cover the ground of interest [1, 2]. Omnidirectional cameras have a very wide field of view and might reduce, if not eliminate, the need to use multiple perspective cameras [3–5]. However, the use of omnidirectional cameras for object detection has been limited. This is partly because conventional camera approaches are not directly applicable and need to be modified in a theoretically correct and practical manner to be used with omnidirectional cameras. In this thesis, new methods are proposed for person detection and tracking in omnidirectional cameras.

People detection and tracking problem is computationally expensive because images and videos are very high dimensional data. To decrease the cost of computation, approximations to the data and modifications of the data has to be made along with the proper data structures. For decades, *integral image*s have been one of the most popular approaches to speed up computations for certain tasks [6–8]. They allow querying pixel sums in rectangular regions in the image rapidly. However, when the query region is not rectangular, one might need to decompose the region into multiple rectangles [9] or use modified versions of integral images [10]. In this thesis, rectangular decomposition of irregular regions is also investigated. Furthermore, a new integral image structure for omnidirectional cameras is introduced.

Specifically, the contributions of this thesis are:

- We introduce a new public video dataset that is collected using omnidirectional cameras for fall detection and multiple person tracking. This dataset was first presented in an international conference along with a baseline tracking approach [11]. This dataset is also used in our other works presented in international and national conferences [12–16].
- We develop a new Bayesian framework for tracking and fall detection in omnidirectional cameras. This work was presented in an international and a national conference [13, 15].
- We use Probabilistic Occupancy Map [2] and K-shortest path [17] to do multiple person tracking in omnidirectional cameras. This part of our work was presented in a national conference [16].
- We propose a novel integral image structure, namely *radial integral image*, for omnidirectional cameras.
- We introduce a new state of the art method for person detection in omnidirectional images that uses radial integral image structure. A journal paper is prepared from this part of the thesis [18].
- We present an integer programming formulation and solution to the problem of decomposing a region into multiple rectangles, namely *Rectangle Blanket Problem*.
- We propose a novel branching strategy for branch-and-bound method while solving set packing problems.
- We develop two heuristic solution methods to the Rectangle Blanket Problem. A journal paper is prepared from Rectangle Blanket Problem related part of the thesis [19].

The term *omnidirectional image* will be used throughout this thesis to refer to an image obtained with an omnidirectional camera.

This thesis is structured as follows. In Section 1.1 an introduction to omnidirectional cameras is given along with three widely used camera models. In Section 1.2.1 a review of public datasets related to person detection and tracking in omnidirectional

cameras are given. In the following sections, Section 1.2.2 and Section 1.2.3 a literature survey is given on person detection and person detection in omnidirectional cameras.

In Chapter 2 Bayesian methods with a generative observation model are given for person tracking in omnidirectional cameras. In Section 2.1 a new dataset, BOMNI, for people tracking is introduced. In Section 2.2 fall detection and tracking is coupled via a new Bayesian formulation and experimental results of its performance are given on the BOMNI dataset. In Section 2.3 application of the Probabilistic Occupancy Map and K-shortest path method for multiple person tracking in omnidirectional cameras is given; experimental results that assess the performance of the approach is also presented.

In Chapter 3 a discriminative person detection method in omnidirectional cameras that is based on Integral Channel Features [8] is given. Integral Channel Features is described in Section 3.1. In Section 3.2 a novel integral image structure, Radial Integral Image, is introduced and its application to person detection in omnidirectional cameras is introduced. In Section 3.3 the validity of the approach is shown experimentally.

In Chapter 4 the problem of decomposing an arbitrary region into rectangles, namely Rectangle Blanket Problem (RBP), is investigated. A novel exact integer programming formulation and its solution via branch-and-bound scheme is given in Section 4.1. Two new heuristic and one existing heuristic in the literature to the RBP is presented in Section 4.2. Heuristic methods are compared against the exact solution experimentally in Section 4.3. Finally, thesis is concluded in Section 5.

1.1. Omnidirectional Cameras

Depending on the field of view, cameras are divided into two groups: directional and omnidirectional. Directional cameras can only acquire images through a relatively small solid angle; for omnidirectional cameras, field of view of the camera is much larger. Camera with a fisheye lens is an example of an omnidirectional camera. Although the term omnidirectional also includes systems such as camera networks consisting of multiple perspective cameras and Pan-Tilt-Zoom cameras, usually the term is used to

describe catadioptric (combination of mirror and lenses) or dioptric (only lens) cameras. Examples of such cameras and images acquired with those cameras can be seen in Figure 1.1.

(a)

(b)

(c)

(d)

Figure 1.1. Examples of catadioptric and dioptric systems and acquired images using those cameras [20]. (a) Catadioptric system with a hyperbolic mirror. (b) Image acquired with a catadioptric system. (c) A camera with a fisheye lens. (d) Image acquired with a fisheye camera.

Catadioptric omnidirectional cameras consist of a mirror and a lens, where the mirror is almost always convex. In catadioptric systems, the mirror is placed in front of a conventional perspective camera, so a point in 3D space is first projected onto the mirror surface and then projected onto the image plane. The image formation process of a catadioptric camera can be seen in Figure 1.2. When the projected rays intersect at one point, it is called *central* camera. Central cameras have certain geometric properties that lead to compact mathematical results. Although it is very cheap to construct catadioptric cameras, there is one major drawback; the perspective camera

itself occludes the scene (Figure 1.1.b), resulting in a part of no value or importance usually at the center of the acquired image.

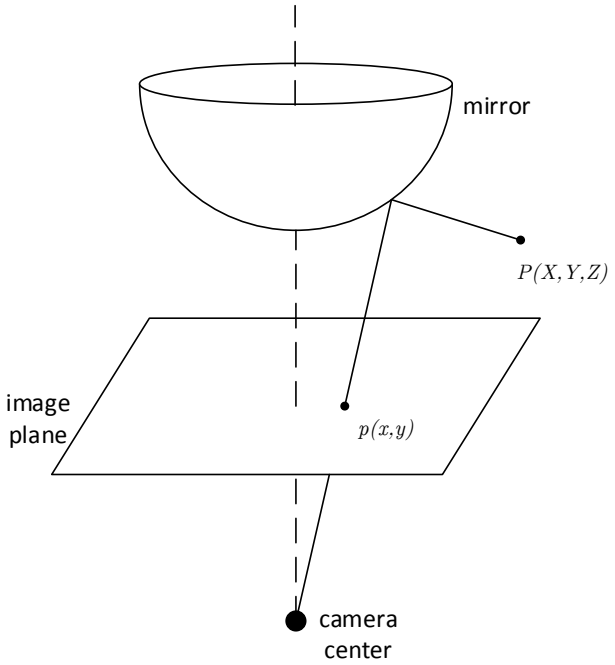


Figure 1.2. Image formation process using catadioptric system.

Image formation process of dioptric cameras also can be described by very similar process to catadioptric ones. A point, instead of being projected to a mirror surface, is projected onto the lens. Catadioptric models are also good approximations to the distortion caused by the fisheye lens; projection for fisheye lenses is usually modeled as a function of the angle between the incoming ray and the optical axis.

1.1.1. Omnidirectional Camera Models

In this section, three widely used omnidirectional camera models are detailed.

1.1.1.1. Generic Polynomial Distortion Model. Using a polynomial for defining the distortion model is a well established method [21, 22]. It can be used to model both dioptric and catadioptric cameras since the underlying mechanism is not explicitly defined in the formulation. Polynomial distortion model became very popular thanks to [23], where Scaramuzza *et al.* introduced a novel toolbox to automatically calibrate

omnidirectional cameras from the images that contain a calibration pattern.

Polynomial distortion models are accurate under certain assumptions:

(i) All the rays reflecting from mirror's surface intersect at one point which is camera center.
(ii) Camera and mirror axis are well aligned.
(iii) Mirror is rotationally symmetric.

Let origin be the camera center, a 3D vector $P = [\, X \; Y \; Z \,]^\intercal$ and a projection of a point on this vector $p = [\, x \; y \,]^\intercal$ onto the image plane (Figure 1.2). Due to the well alignment property (ii), following expression holds:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \alpha \begin{bmatrix} x \\ y \end{bmatrix}$$

Back projection ray of p, i.e. corresponding 3D vector P, can be written as:

$$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \alpha x \\ \alpha y \\ f(p) \end{bmatrix}$$

The calibration of the camera corresponds to finding function f. Note that α can be included in f. Furthermore using mirror's rotational symmetric property (iii), f only depends on the distance of p to the center. Let $r = ||p|| = \sqrt{x^2 + y^2}$, we can write:

$$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ y \\ f(r) \end{bmatrix}$$

If we assume a polynomial $f(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + \cdots + a_n r^n$, calibration corresponds to finding the polynomial's coefficients a_n. n defines the complexity and accuracy of this model, as n increases accuracy gets better. The person detection and tracking work in Chapter 2 uses this camera model.

1.1.1.2. Explicit Image Formation Model. When the exact geometry of the mirror in catadioptric camera is known, it can be used to model the system with a better accuracy [20]. According to this model, first the 3D point is projected onto the mirror surface yielding u, after that u is projected to the image plane and p is obtained (Figure 1.3). Assuming the mirror's shape can be described with g, and keeping $u = \alpha P$ in mind:

$$u = \begin{bmatrix} h(p)p \\ g(p) \end{bmatrix}$$

The equation above can be thought as the back projection of point p to the mirror surface. First p is transformed to $h(p)p$ and its z coordinate is found via mirror shape function. Calibrating the camera corresponds to finding the parameters of h and g. Since g represents mirror's shape, it is easy to define it. Note that when $h = 1$ and $g = 1$ this model corresponds to conventional perspective camera. When only $h = 1$, the projection to the plane is an orthographic projection.

For a catadioptric system with a parabolic mirror and orthographic projection, the following expression can be written:

$$h(p) = 1$$
$$g(p) = \frac{a^2 - ||p||^2}{2a}$$

where a is the parameter defining the mirror shape, which would be the goal of calibration to determine it.

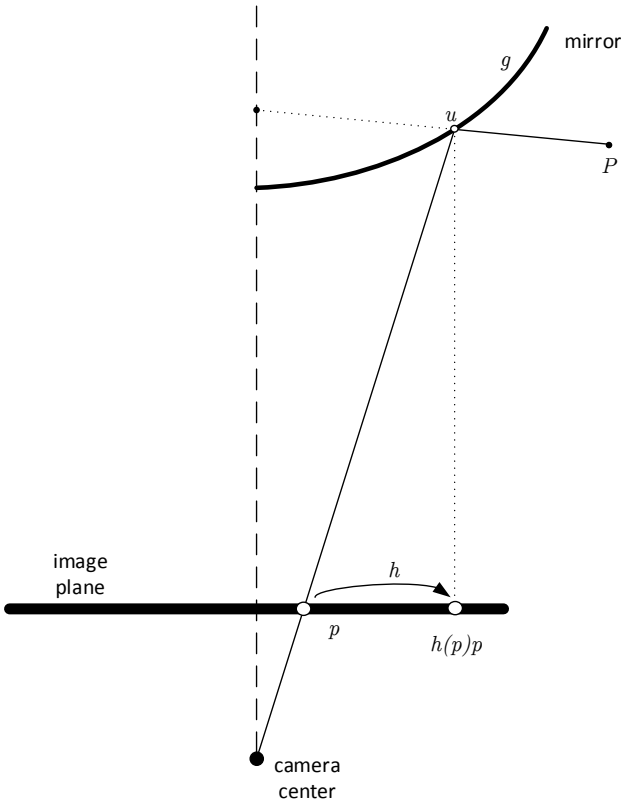


Figure 1.3. Another catadioptric camera model where the image formation process is explicitly defined.

1.1.1.3. Spherical Camera Model. Geyer and Daniilidis introduced an ingenious way of modeling all catadioptric systems under a single formulation [24]. According to this model, all central catadioptric systems can be modeled as projection to a sphere, followed by a secondary projection from the sphere surface to the image plane via a projection point (Figure 1.4). By changing the second projection point via ξ, mirrors with different geometry can be modeled.

The projection point, which acts as the camera center of a virtual camera inside the sphere, is located on the diameter that is perpendicular to the image plane and ξ units away from the center of the sphere (Fig 1.4). We can assume that the sphere is a unit sphere and by changing the position of the image plane, we can scale the image.

Let the z axis be perpendicular and pointing towards the image plane; f be the distance of the image plane to the projection point; p_{world} be an arbitrary 3D

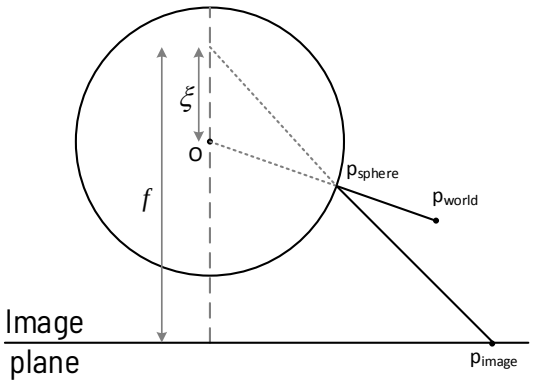


Figure 1.4. In the spherical camera model, a point is projected onto the unit sphere first, then projected onto the image plane. For cameras using parabolic mirrors, $\xi = 1$.

point in the world; (x, y, z) be the coordinates of p_{sphere}, which is the projection of p_{world} on the sphere and p_{image} be the projection of p_{sphere} on the image. A graphical depiction is shown in Figure 1.4. Then, $r = \sqrt{x^2 + y^2 + z^2}$ and the projection from world coordinates to image coordinates can be expressed as:

$$p_{\text{image}} = \left(\frac{fx}{\xi + z}, \frac{fy}{\xi + z} \right)$$

For cameras using parabolic mirrors, $\xi = 1$. In other words, the projection point is located on the sphere. This is a typical situation and also known as stereographic projection. The person detection work in Chapter 3 uses this spherical camera model.

1.2. Related Work

1.2.1. Public Datasets Using Omnidirectional and Wide Angle Cameras

In this section an overview of datasets collected by omnidirectional cameras is given. Omnidirectional camera is a popular sensor in robotics community, and there are several datasets for localization and mapping purposes [25, 26]. There are also datasets for vehicle detection and classification [27, 28]. However, the focus of this section is datasets related to human detection and human actions. These datasets are rather limited compared to the ones collected with perspective cameras.

One of the first datasets for action recognition involving omnidirectional cameras are PETS2001 [29], PETS-ICVS [30] and PETS2004 [31]. In PETS2001 dataset there are five separate sets of training and test sequences of moving vehicles and people. The bounding boxes are given as the ground truth. PETS-ICVS dataset is collected in a smart meeting room. It includes annotation for eye positions, facial expression, gaze estimations and gestures of people. In PETS2004, there are 6 activities performed in 28 videos; the activities are: walking, browsing, collapse, leaving object, meeting, fighting. Example frames from PETS datasets can be seen in Figure 1.5. PETS datasets have not been widely used, because extensive and better designed datasets using conventional cameras has been released the following years.

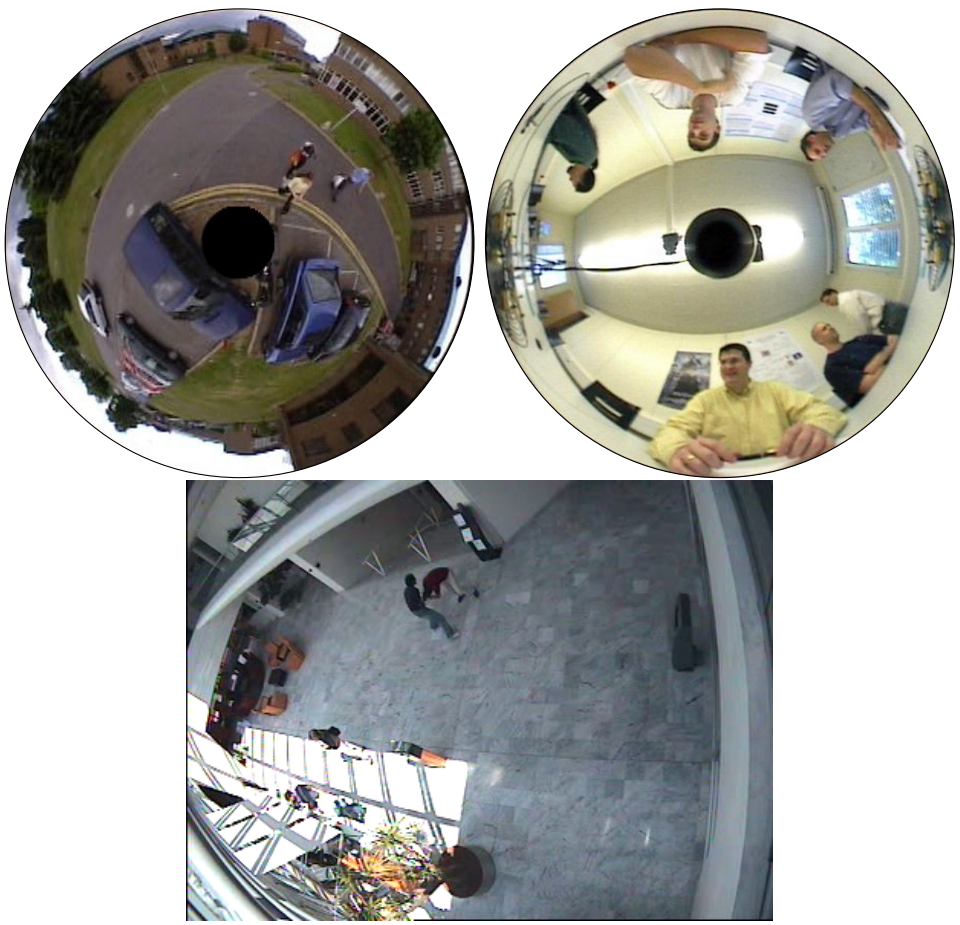


Figure 1.5. Example frames from PETS datasets. Left: PETS2001. Right: PETS-ICVS. Bottom:PETS 2004.

CVBASE '06 dataset contains videos from sport environments recorded using multiple omnidirectional cameras [32]. There are three groups of data in the dataset: team handball, squash and basketball. Team handball subset has both team activity

(e.g. offense, defense, etc.) and individual activity (e.g. pass, shot, etc.) annotations. In squash subset, phases, stroke types, stroke outcome and the types of the strikes (forehand, backhand) are annotated. Basketball subset is not annotated, only the video is available. Example frames from the dataset can be seen in Figure 1.6.

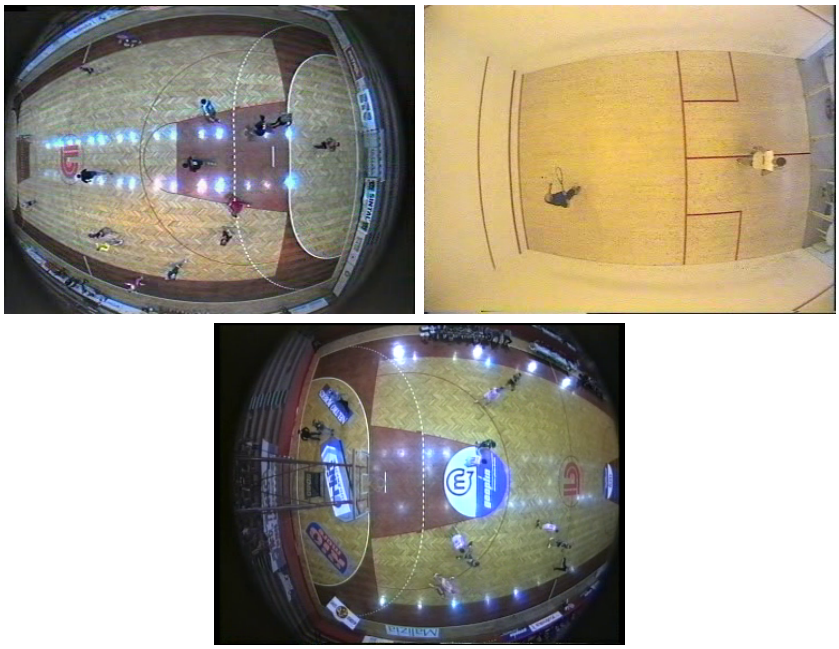


Figure 1.6. Example frames from CVBASE 06 dataset. Left: team handball. Right: squash. Bottom: basketball.

Another dataset using omnidirectional cameras and many different modalities is Opportunity dataset [33]. There are 72 sensors in total of 10 modalities, where 2 of them are omnidirectional cameras. There are 12 subjects performing various morning activities like preparing coffee, eating a sandwich. An example from the dataset can be seen in Figure 1.7. There are four different tracks of annotations, first track determines the high level action like preparing a sandwich, second track contains walking, standing, lying and sitting states. The remaining four tracks define actions of right and left hands, namely, reach, move, release, stir, sip, bite, cut and spread.

In a more recent study, Behera *et al.* introduced new dataset of egocentric actions [34]. A fisheye camera mounted to the chest of the person and two sets of dataset are collected. In the first set there are 27 videos and 9 actions are recorded in total. The actions are take/put box, pick hammer, take/put baton, take nail, hammer nail, put down hammer, pick screwdriver, drive screw and put down screwdriver. The

Figure 1.7. An example from Opportunity dataset.

Figure 1.8. Example frames from COGNITO dataset.

second set is "labeling and packaging bottles" scenario, 9 actions in 23 videos are pick and put bottle, stick label, pick and put box, remove cover, put bottle inside box, take and put cover, write address, take and put tape dispenser, seal the box. Along with the omnidirectional videos there are head mounted depth and RGB recordings of the same scene. At the time of this writing, ground truth was not available. Example images can be seen in Figure 1.8.

Taylor *et al.* developed a dataset generator called ObjectVideo Virtual Video Tool (OVVV) [35]. OVVV generates realistic video from simulated cameras in an interactive virtual world. Although synthetic data cannot fully replace real data, it is important for debugging, early testing, prototyping purposes. OVVV allows various parameters of the scene and cameras like noise, jitter, human trajectories to be altered, and provides

Figure 1.9. Images generated using OVVV. Left: Unwarped, panoramic image from omnidirectional camera. Right: Image from omnidirectional camera.

accurate ground truth. Besides, OVVV can also be used to generate omnidirectional images. An example can be seen in Figure 1.9.

Cinaroglu and Bastanlar created a dataset for human and car detection using omnidirectional camera [5]. There are 30 omnidirectional images containing 66 annotated humans and 50 omnidirectional images containing 65 annotated cars. The same dataset also contains synthetic omnidirectional images of people from INRIA dataset [36]. An example can be seen in Figure 1.10.

1.2.2. Person Detection

In a classical work of object detection [36], Dalal and Triggs extracted histogram of oriented gradient (HOG) features from overlapping rectangular regions from the detection window. They used these features with support vector machine (SVM) classifiers to perform human detection. Consulting such gradient-based features is a well established idea in person detection. Later, Zhu *et al.* used integral image histograms to speed up the feature extraction step in the HOG detector [37]. Felzenszwalb *et al.* developed a similar model with multiple body parts, where the positions of these parts were inferred as the latent variables of an SVM [38]. In [39] authors used the covariance matrix of different image features (i.e. covariance features) for pedestrian detection. Conventional classifiers do not perform well for covariance matrices, because they do not form a vector space. Instead they proposed a method to effectively do classification with covariance features on a Riemannian manifold.

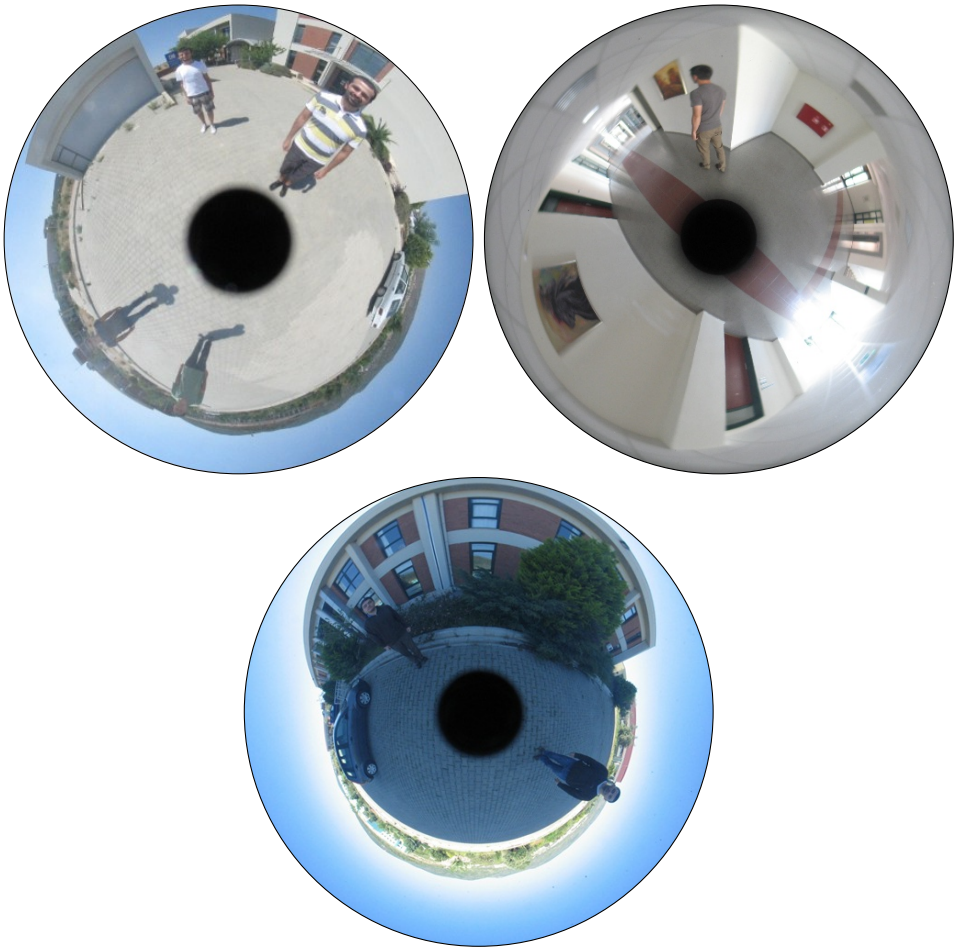


Figure 1.10. Three samples from the IYTE dataset.

In [2], a generative approach is used to detect and track people. Fleuret *et al.* represented the ground plane with discrete, evenly spaced locations, and used a hidden location to handle transitions in and out of the scene. They have formulated a Bayesian framework to track multiple people, which utilizes the distance between the foreground segmentation image and a synthetic model of the human shape at any given location. As a result, a structure is obtained which gives the probability of a human being present at a location. This structure is called *Probabilistic Occupancy Map*. They model the movement of a single person as a hidden Markov model (HMM), where the person's location is the hidden variable and images are observations. The person detection methods introduced in Chapter 2 are based on their generative approach.

Performance is a major concern in person detectors, as the applications typically require real-time computations. In [8], authors combined the idea of boosting multiple

simple features, as in the Viola-Jones detector [7] throughout multiple channels, including the HOG channel. Their work, namely, Integral Channel Features (ICF), has been very successful due to its simplicity, low computational cost and detection performance. ICF forms the backbone of the person detection approach presented in Chapter 3. To deal with the computational performance issues, Dollar *et al.* proposed that feature responses can be used to approximate feature responses at nearby scales [40]. They accelerated person detection by avoiding the building of the full scale space. Following a similar line of work, Benenson *et al.* investigated every component of a rigid (i.e. not part based) detector and improved HOG+SVM miss rate by more than 30%, through adjustment of system components such as feature pooling and normalization [41]. In addition to HOG, local binary patterns (LBP) are also used for detection [42].

In [43], authors investigated the failure cases of top performing pedestrian detectors (most of them being from the ICF family) in detail, and suggested ways to design and improve existing detectors. Besides other adjustments to the system, they also suggest that dataset annotation quality is essential for detection performance.

Introduction of deep learning changed the scene for object detection [44–46]. Although some of the person detection approaches rely on deep neural networks [47] until recently, convolutional neural networks (CNN) failed to catch up with ICF-based methods [48, 49]. In a recent work, Cao *et al.* used handcrafted feature channels (LUV, HOG, etc.) and features from inner layers of a CNN to perform classification with AdaBoost [50]. This can be considered a hybrid approach, as neural networks are not used in an end-to-end fashion. It is possible to fuse CNN features with other detectors, but this represents a trade-off between accuracy and speed. Furthermore, ICF-based methods are still attractive due to computational advantage gained by the simplicity of these features and good detection performance.

For real world computer vision applications, automating people tracking produces more information and value than doing only person detection. Tracking is usually done right after the detection step by associating the detections temporally. In [51], authors tracked people in crowd using multiple cameras. First, they detected heads

using homography between multiple cameras and intensity correlation. Later, these head detections in subsequent frames are transformed into trajectories.

Breitenstein *et al.* tracks multiple people in single view using a generic object detector which outputs confidence values between zero and one [52]. Classifiers are updated at run time for each tracked target and detections are associated with subsequent high confidence detections. This method works real time and it only uses simple components. In other words, this method doesn't rely on use multiple cameras, constant background assumption, scene and depth information.

Berclaz *et al.* formulated a linear program which takes the probability of a person being present on discrete locations as input [17]. In fact, their linear programming formulation corresponds to optimizing flows on a graph. They also show that finding the optimal flows is equivalent to solving K-shortest path on the same graph, where each path corresponds to a trajectory.

Also in [53], person tracking is formulated as a flow optimization problem. Although the approach is very similar to [17], the structure of the graph is different and it uses histogram of oriented gradients (HOG) features for detection. Furthermore, this method forms trajectories in order and allows processing of detection results at the intermediary step.

1.2.3. Person Detection in Omnidirectional Cameras

Regarding object detection studies with omnidirectional cameras, some previous approaches first transform the omnidirectional image into a panoramic image, and then apply conventional detection methods on this image [54–57]. However, this transformation introduces extra parameters for tuning, and brings additional computational effort. Panoramic transformation also distorts objects. Therefore, objects in transformed images differ from those in perspective cameras. Especially for tall objects, this distortion results in a decreased detection performance [5].

Geyer and Daniilidis have shown that every central projection system can be modeled as a projection to a sphere, followed by a backprojection to the image plane [24]. Based on this sphere model, researchers recently proposed methods to compute features directly on omnidirectional images. Puig and Guerrero proposed using differential operators on the sphere to construct a scale space for omnidirectional images [58]. Arıcan and Frossard used the same idea to construct a feature detection and extraction method for catadioptric omnidirectional cameras [59]. The features they used is similar to Lowe's popular scale invariant feature transform (SIFT) [60]. Lourenço *et al.* proposed a similar framework, called sRD-SIFT, for images with radial lens distortions [61]. Their method corrects the gradient using lens distortion coefficient, provided that it is available.

Tracking people with omnidirectional cameras is relevant for both indoor and outdoor settings. Saito *et al.* used template matching in a Bayesian framework to detect and track multiple people in omnidirectional cameras [62]. They generated different templates for people standing at different distances from the camera. Alahi *et al.* used omnidirectional cameras along with perspective cameras for person detection in a basketball game [3, 4]. They used a dictionary of binary human silhouettes for each discrete location and infer the actual occupancies using binary foreground detection as the input. They formulated the problem as a linear inverse problem, and added a constraint on the maximum number of people to enforce the sparsity of the solution. In [63], authors used the same basketball game data to generate ground occupancy maps by backprojecting the foreground maps to ground plane.

Features can be tailored for omnidirectional images. Cinaroglu and Bastanlar took the traditional HOG approach and modified the features according to the Riemannian metric on the sphere camera model [5]. In that way, object detection was done directly on the omnidirectional image. They also proposed rotating annular sectors (doughnut slice shapes) to improve the performance over rotating rectangular windows. Their approach is computationally expensive, since the transformation of HOG features is done separately for each sliding-rotating window. In Chapter 3, we propose a faster and more accurate approach.

2. GENERATIVE PERSON DETECTION AND TRACKING

2.1. BOMNI: Multi-Omnidirectional Video Tracking Dataset

In this section, the BOMNI dataset is introduced and its details are given. BOMNI dataset is collected with two omnidirectional cameras simultaneously. The dataset contains single subject and multisubject interaction scenarios, as well as actions relevant for ambient assisted living, such as falling.

BOMNI dataset includes samples taken by two omnidirectional cameras. The first camera is mounted on the ceiling of the room and the other one is fixed on a side wall. The room is almost square-shaped, approximately 7m×7m. Although there are many objects cluttering the room, only two chairs, two tables and a sink are actively used by the subjects. A rough plan of the room and the location of interacted objects can be seen in Figure 2.1.

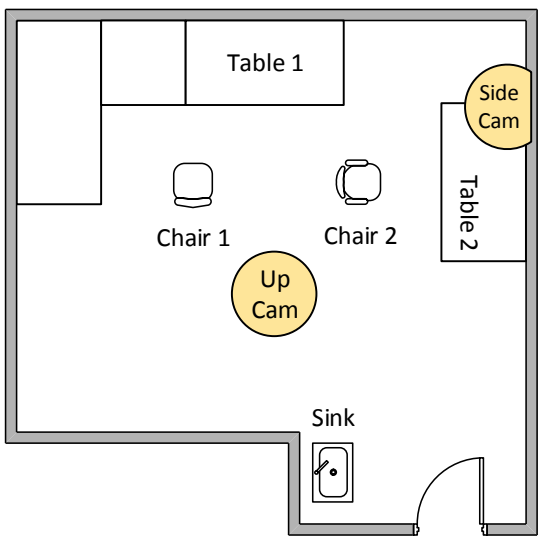


Figure 2.1. Floor plan of the room that is used for the data acquisition.

Videos have been acquired using Oncam IPC, which is a 360° ceiling-mounted IP camera with a 5 megapixel sensor with no moving parts. Calibrated camera parameters are distributed along with the dataset. Although the camera supports higher resolutions, the videos were captured using 640×480 resolution because of the network bandwidth limitations. For the same reason, the frame rate of the original videos are around 8 fps and there are occasional frame drops. Cameras are also configured to work in auto exposure and white balance mode resulting in infrequent intensity changes for the whole captured frames.

The only light source is the natural light entering room through windows located at one side of the room. At the time of video acquisition the light source is not controlled, allowing the illumination to change between videos. To illustrate the illumination variation, the standard deviation of image pixel intensity values are calculated using the first 30 frames of videos, and in HSV color space. The V channel can be interpreted as lightness value of a pixel. The pooled standard deviation for the V channel is 14.99, and 11.73 for top and side view, respectively. Visualization of the variance of the V channel can be seen in Figure 2.2.

Figure 2.2. Illustration of the standard deviation of image pixel intensities for the V channel of HSV color space. Darker colors indicate higher deviation.

Furthermore, from the viewpoint of the side camera, the subjects are occluded by the objects in the room while entering and exiting the room, and in the multi-user scenario there are subjects occluding each other. Low frame rate, frame drops, occlusions and uncontrolled illumination are the usual conditions faced in a real world application, and form the challenges posed by the dataset.

2.1.1. Scenarios

There are two sets of videos in the dataset corresponding to single and multi-user scenarios. Scenario #1 consists of recordings of five subjects from both cameras resulting in a total of 10 videos. In each video, a subject enters to the room, walks across to the table, grabs a bottle, sits down on chair #1, drinks water, stands up, puts the bottle back on to the table, goes to the sink, washes hands, exits the room, enters to the room after a short delay, walks to chair #2, sits down, idles for a short time, stands up, walks across the room to table 1, grabs a bowl, while walking across the room faints and falls down; in this order. In this scenario, there are only minor and rare occlusions caused by the chairs. Sample frames of Scenario #1 can be seen in Figure 2.3.

Figure 2.3. Sample frames from BOMNI Scenario #1. Annotations are depicted in lighter color for clarity.

Scenario #2 presents videos of multiple people interacting with each other. Following actions take place in each video: the first person (A) enters to the room, walks across the room, sits down on chair 1, the second person (B) enters to the room, walks across the room to table #1, A stands up, A and B shake hands, the third person (C) enters the room, A and C meet and shake hands in the middle of the room, B starts walking and exits room, A and C walk to table 2, A and C look at an object for a short duration, A starts walking towards the door, C starts walking, A exits the room, C exits the room. The subjects' appearances on the videos, especially the ones taken with the side camera, suffer from serious occlusions during the performance of the scenario. Five subjects in total formed three groups of size three, and for each group, by changing roles in group, six videos are recorded. As a result, 36 pairs of videos are acquired from both cameras. Samples frames of Scenario #2 can be seen in Figure 2.4.

All videos have been recorded in MJPEG format and annotated using interactive video annotation tool *vatic* [64]. Bounding boxes of the performers and their type of action are annotated for every frame in all of the videos. There are six subjects in total, one of whom is female.

The dataset also contains hand-made annotations for action segments. In the single person scenario, six actions are annotated. These are *sitting, walking, drinking, washing-hands, fainting* (laying on the floor) and the *opening-closing-door* actions, respectively. The three person scenario contains five actions, which are slightly different. Fainting for instance is not relevant for a multi-person scenario, but interactive actions are considered. In particular, *sitting, walking, standing, shaking-hands* and *interested-in-object* actions are annotated. The number of annotated actions for each case and the total number of frames for each action are given together with annotation files on the dataset webpage.

The BOMNI dataset is first presented in [11]. In the same study an evaluation protocol is defined and a baseline method is provided. Baseline method uses interest point matching after foreground-background segmentation to track people. For comparison of methods using this dataset, it is advised to use the CLEAR Multiple Object

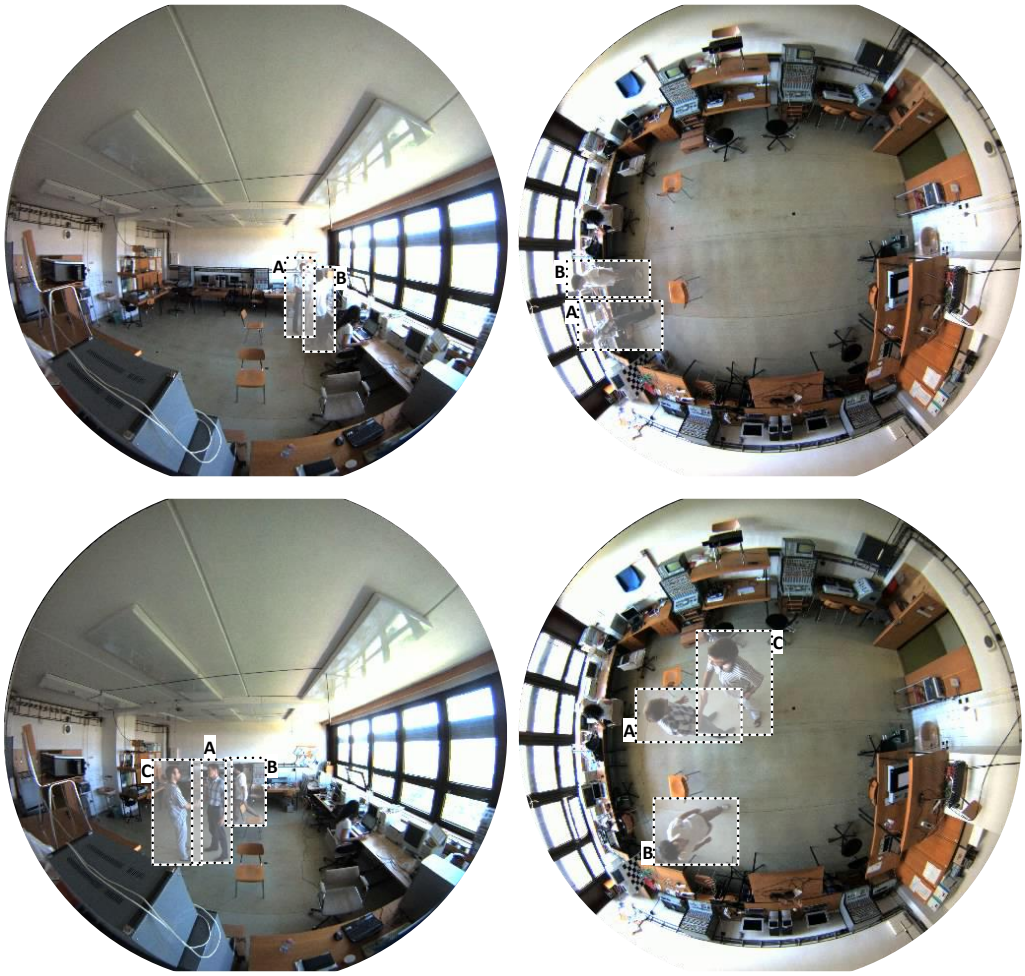


Figure 2.4. Sample frames from BOMNI Scenario #2. Annotations are depicted in lighter color for clarity.

Tracking metrics [65], i.e. multiple object tracking precision and accuracy. The details of these metrics are described in Section 2.1.2.

2.1.2. CLEAR multiple object tracking metrics

In [65], Bernardin and Stiefelhagen proposed two metrics to compare and assess performance of multiple object trackers. These metrics are multiple object tracking precision and accuracy, MOTP and MOTA for short. These metrics have been quickly adapted and widely used by the community since their introduction [66].

MOTP focuses on how well the tracker is able to predict object's location. It is defined as:

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}$$

where d_t^i is the distance between the object o_i and its corresponding hypothesis for frame t, c_t is the number of matches found for the frame t. The distance between object and hypothesis can be defined according to the application. If the targets are represented as points Euclidean distance would suffice. If targets are represented as rectangles, usually some normalized form of non-overlapping area between object and hypothesis is used as distance.

MOTA measures tracker's quality at recognizing object configurations and consistent object labeling. It is defined as:

$$\text{MOTA} = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}$$

where g_t is the number of objects, m_t, fp_t, mme_t are the number of misses, false positives and mismatches respectively, for the frame t. It is a common practice to report these three error metrics separately along with MOTA as ratios:

$$\overline{m} = \frac{\sum_t m_t}{\sum_t g_t} \qquad \overline{fp} = \frac{\sum_t fp_t}{\sum_t g_t} \qquad \overline{mme} = \frac{\sum_t mme_t}{\sum_t g_t} \tag{2.1}$$

Before calculating MOTP and MOTA metrics, hypothesis and objects must be matched to each other. This is achieved by solving the optimal assignment problem via Hungarian algorithm using a distance metric between hypothesis and objects as described above.

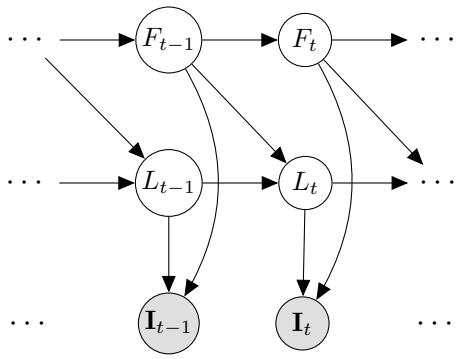


Figure 2.5. The graphical model of the proposed system to detect falls. The variables F_t, L_t, I_t denote whether a person has fallen, location of the person and images obtained from all cameras at time t, respectively.

2.2. COUPLED PERSON TRACKING AND FALL DETECTION

In this section, a method to track the inhabitants of an indoor scene and to detect falls using omnidirectional cameras is described. Multiple omnidirectional cameras are used to deal with occlusions. The approach described in this section is built on Fleuret *et al.*'s foundations [2], and their generative human shape model is extended to fall detection in omnidirectional cameras.

Let L_t, F_t and I_t^c be a person's location, fall state of the person ($F_t = 1$ if fallen) and the image obtained from camera c at time t, respectively. Let bold letters indicate grouping of omitted indices, e.g. $\mathbf{I} = I_{1:T}^{1:C}$. At a given time, a person could be at one of the G locations, or at the hidden location that handles transitions in and out of the room. The aim is to find the most likely value of the trajectory and the sequence of fall states of the person, given the observed images:

$$\{l_{1:T}^*, f_{1:T}^*\} = \underset{l_{1:T}, f_{1:T}}{\arg\max}\ \mathrm{P}(\mathbf{L} = l_{1:T}, \mathbf{F} = f_{1:T} \mid \mathbf{I}) \tag{2.2}$$

The system can be modeled using a Hierarchical HMM, given in Figure 2.5:

$$\mathrm{P}(F_t \mid F_{1:t-1}) = \mathrm{P}(F_t \mid F_{t-1}) \tag{2.3}$$

$$\mathrm{P}(L_t \mid L_{1:t-1}, F_{1:t-1}) = \mathrm{P}(L_t \mid L_{t-1}, F_{t-1}) \tag{2.4}$$

$$\mathrm{P}(\mathbf{I}_t \mid L_{1:t}, F_{1:t}) = \mathrm{P}(\mathbf{I}_t \mid L_t, F_t) \tag{2.5}$$

The problem stated in Equation 2.2 can be solved using the recursive Viterbi algorithm [67]. Let us define the maximum probability of observing the images and the trajectory ending up at location k and fall state s at time $t+1$:

$$\Phi_t^{k,s} = \max_{\substack{l_{1:t-1} \\ f_{1:t-1}}} \mathrm{P}(\mathbf{I}, \mathbf{L} = l_{1:t-1}, \mathbf{F} = f_{1:t-1}, L_t = k, F_t = s) \tag{2.6}$$

The expanded expression yields:

$$\Phi_t^{k,s} = \mathrm{P}(\mathbf{I_t} \mid L_t = k, F_t = s) \max_{r,d} \Big\{ \mathrm{P}(F_t = s \mid F_{t-1} = d) \; \mathrm{P}(L_t = k \mid L_{t-1} = r, F_{t-1} = d)\, \Phi_{t-1}^{r,d} \Big\}$$

The fall model is defined as:

$$\mathrm{P}(F_t = k \mid F_{t-1} = i) = \begin{cases} 1 & \text{if } i = 1 \text{ and } k = 1 \\ z & \text{if } i = 0 \text{ and } k = 1 \\ 1 - z & \text{if } i = 0 \text{ and } k = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

According to the model above, once fallen, the person will remain fallen. z is a small probability that denotes the probability of falling at a given time.

The motion model is defined as:

$$\mathrm{P}(L_t = k \mid L_{t-1} = l, F_{t-1} = f) = \begin{cases} \frac{1}{Z} e^{-\rho \|l-k\|} & \text{if } f = 0 \text{ and } \|l-k\| < v \\ 1 & \text{if } f = 1 \text{ and } l = k \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

The expression above states that if a person has fallen, his location does not change anymore. v defines the upper limit of walking speed, ρ fine tunes the average walking speed.

It is assumed that all the information is encapsulated in the binary foreground mask, and views are independent given the person's location. For a fixed time t, the generative model can be written as:

$$\mathrm{P}(\mathbf{I} \mid L = k, F = f) = \mathrm{P}(\mathbf{B} \mid L = k, F = f) \tag{2.9}$$

$$= \frac{1}{Z} \prod_c e^{-\Psi(B^c, A^c_{k,f})} \tag{2.10}$$

where B^c is the binary foreground segmentation obtained from camera c, $A^c_{k,f}$ is the binary image generated by putting the human silhouette in fall state f at location k and $\Psi(\cdot)$ is a pseudodistance function between two binary images. It is defined for $A, B \in \{0,1\}^{\text{width} \times \text{height}}$ as:

$$\Psi(B, A) = \frac{1}{\sigma} \frac{|B \otimes (1 - A) + (1 - B) \otimes A|}{|A|} \tag{2.11}$$

where $\otimes$ is the element-wise multiplication operation and $|\cdot|$ gives the count of non zero elements. According to this definition as the overlap between ideal and observed foreground segmentation increases as the distance decreases. An example of real foreground segmentation can be seen in Figure 2.6.

Human shapes are represented as cuboids in 3D, instead of a rectangle in the 2D image as proposed in [2]. Example of human silhouettes represented as cuboids can

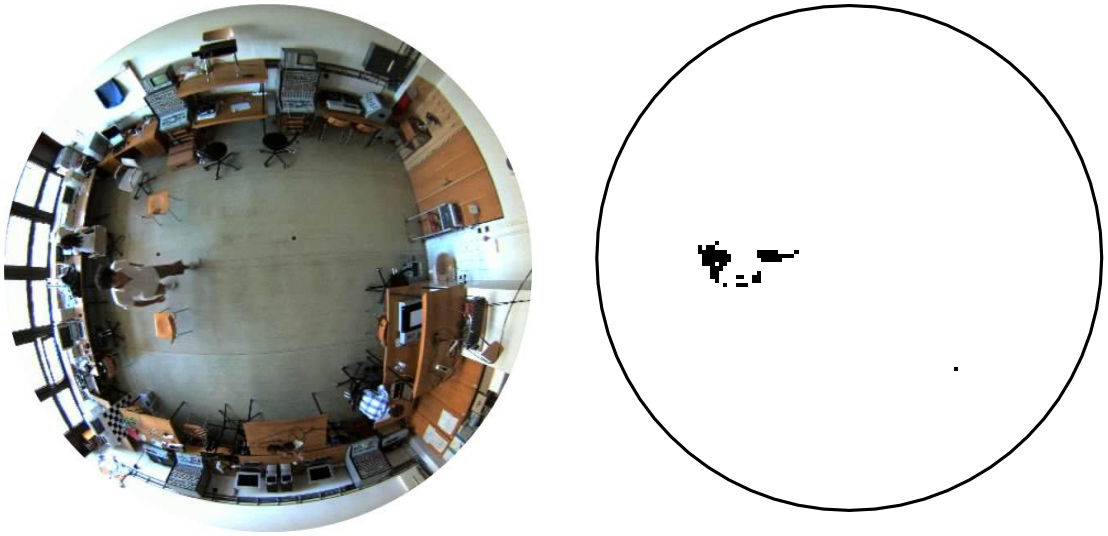


Figure 2.6. Frame obtained from top camera and foreground segmentation of the frame.

be seen in Figure 2.7. However, doing this prevents using integral images directly to speed up computation. Although outlined approach models a single person and the computations are tractable and fast, in the next section a method is described that further speeds up the pseudo-distance calculation.

2.2.1. Fast Integration Over Silhouette Region

The method described in the previous section involves evaluation of $\Psi(B^c, A^c_{k,f})$ for all possible locations and states, hence constitutes the bottleneck of the algorithm. Notice that $\Psi(B^c, A^c_{k,f})$ can be expressed as:

$$\Psi(B^c, A^c_{k,f}) = \frac{1}{\sigma} \frac{|B^c| - 2|A^c_{k,f} \otimes B^c| + |A^c_{k,f}|}{|A^c_{k,f}|} \tag{2.12}$$

If the human silhouettes, $A^c_{k,f}$, are straight rectangles, integral images can be used to speed up the computation of $|A^c_{k,f} \otimes B^c|$. However, in our case due to camera distortion and camera positioning human shapes cannot be represented as axis aligned rectangles. Human shapes are represented as a cuboid in 3D and the silhouettes (their projections to the image plane) are approximated with multiple axis aligned rectangles. By doing this pseudodistance calculation can be carried out in constant time. An example of approximating silhouette with rectangles can be seen in Figure 2.8. Note that, ideal

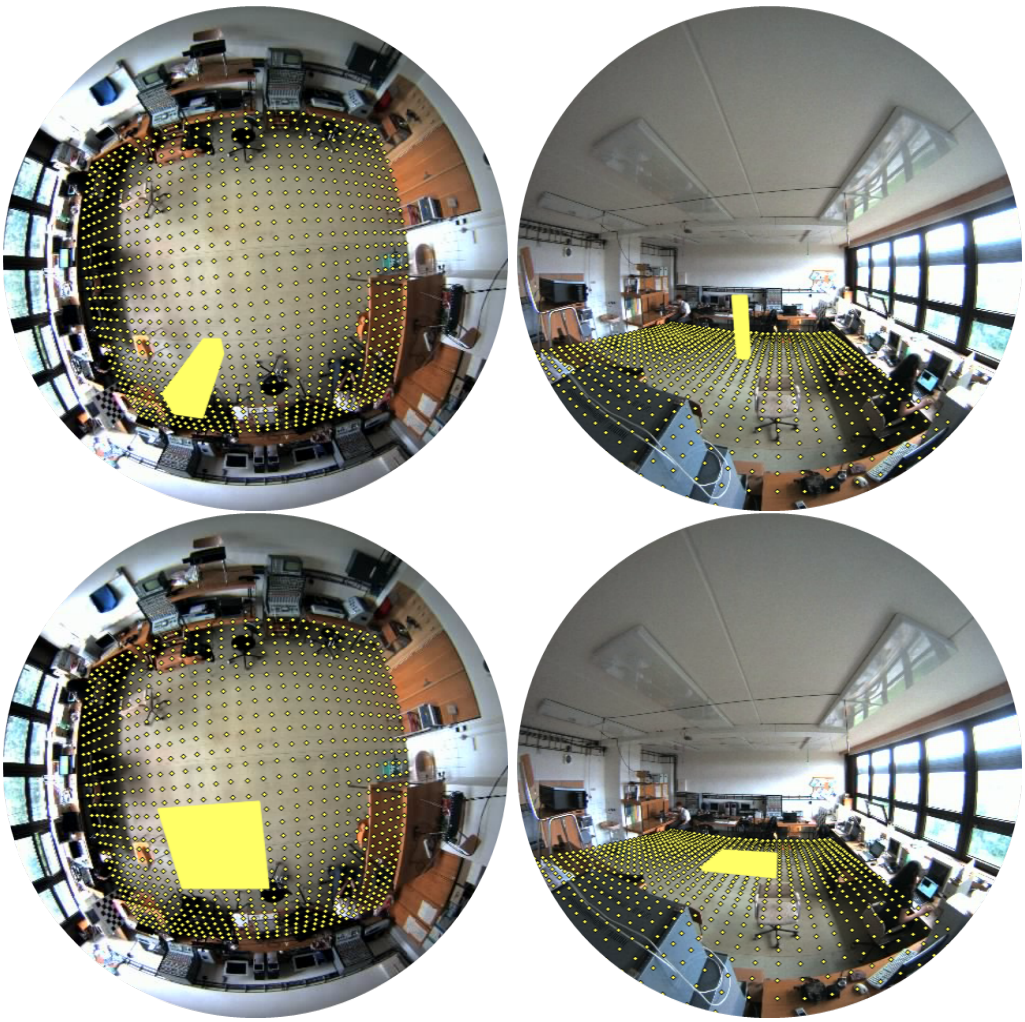


Figure 2.7. Locations in the room and the ideal foreground segmentation (human silhouette) for a human at location $k = 545$. Top: $f = 0$, the person is standing. Bottom: $f = 1$ the person has fallen.

human silhouettes and approximating rectangles need to be generated only once before tracking. In fact, the process of approximating a shape with multiple rectangles is a new type of problem and it is studied in detail in Chapter 4. The particular method used for this computation in this section is Split & Fit, and it is detailed in Section 4.2.1.

2.2.2. Experiments

The proposed fall detection approach was tested on the BOMNI dataset, which involves 5 sets of video pairs obtained using two omnidirectional cameras, containing fainting actions along with other actions of people. See Section 2.1 for the details of the

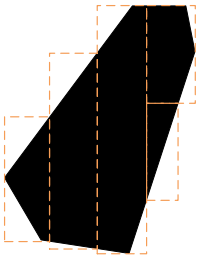


Figure 2.8. Representing a human silhouette with five rectangles.

dataset. Intrinsic and extrinsic camera calibration information is also provided along with the dataset where the camera model described in [23] is used. Videos typically suffer from high noise rates and severe frame loss. Moreover, the person is heavily occluded throughout the video. Keeping these properties of the dataset in mind, the videos can be considered challenging. When processing video pairs, to handle missing frames, a frame in one video is associated with the closest frame in time in the other video.

The proposed approach is compared with a baseline method (denoted as non-HMM here), which ignores temporal relations and maximizes the term in Eq. 2.10 for each frame separately.

The foreground detection is performed using the OpenCV 2.4.6 implementation of [68], where intensity value of each pixel is modeled using a mixture of Gaussians. The ground plane is discretized into 961 locations and in the motion model, the transitions from/to the hidden location are allowed only for the locations just in front of the room.

The tracking performance of the system is measured using Multiple Object Tracking Accuracy and Precision (MOTA and MOTP) metrics [65]. Although there is a single person in the videos, this is the protocol suggested by the dataset for all of the videos, including the videos with multiple persons. Unifying the metrics allows researchers to compare their results. The distance between ground truth and prediction

is expressed in terms of the overlap of their bounding rectangles. More formally:

$$d(R^{\mathrm{t}}, R^{\mathrm{p}}) = 1 - \frac{|R^{\mathrm{t}} \cap R^{\mathrm{p}}|}{|R^{\mathrm{p}}|} \tag{2.13}$$

where R^{t} and R^{p} are bounding boxes of the ground truth and prediction, respectively, and $|\cdot|$ is the area operator. The value of the threshold for accepting a prediction is selected as 0.5. Tracking results can be seen in Table 2.1. Since there is only one object to track, the *mismatch* component of MOTA is irrelevant, and therefore omitted from the table. In the table, *MOTP overlap* values are expressed as overlapping score, i.e. 1 − MOTP.

Table 2.1. Tracking results on the BOMNI. Improvements in terms of increased MOTA and decreased error over the baseline are indicated in green, whereas the occasional poorer result is shown in red.

#	Cam	MOTP	change	MOTA	change	miss rate	change	false+	change
1	top	0.52	+0.01	0.96	+0.10	0.02	-0.09	0.02	-0.01
	side	0.57	+0.01	0.81	+0.05	0.09	-0.05	0.09	·
2	top	0.50	-0.01	0.89	+0.01	0.02	-0.05	0.09	+0.04
	side	0.58	·	0.88	+0.04	0.01	-0.06	0.11	+0.02
3	top	0.48	·	0.97	+0.05	0.01	-0.03	0.02	-0.02
	side	0.60	-0.01	0.94	+0.05	0.00	-0.03	0.05	-0.02
4	top	0.53	·	0.99	+0.01	0.00	·	0.00	-0.01
	side	0.53	·	0.94	+0.01	0.01	·	0.05	-0.01
5	top	0.47	·	0.98	+0.02	0.00	-0.01	0.02	·
	side	0.59	·	0.95	+0.01	0.00	-0.01	0.05	·
	OVERALL	0.54	·	0.93	+0.04	0.02	-0.04	0.05	·

The performance of the fall detection is measured by false positive and false negative counts. False positive is a frame where a fall is reported when there is none. False negative is a fall frame reported as a non-fall. If a false positive or false negative is closer than 4 frames (∼half second) to the beginning of the fall it is not counted as an error, since annotations can be subjective at that scale. Fall detection results can be

seen in Table 2.2. An example fall detection result can be watched at [69][1] .

Table 2.2. Fall detection results on BOMNI. FP and FN stands for false positive and negative frame counts respectively.

		non-HMM		HMM	
#	Camera	FP	FN	FP	FN
1	top	156	20	0	0
	side	138	0	0	0
2	top	3	0	0	0
	side	3	0	0	0
3	top	4	0	0	0
	side	3	2	0	2
4	top	3	0	1	0
	side	1	0	0	0
5	top	1	0	0	0
	side	0	1	0	2

MOTP and MOTA values indicate that the proposed system performs tracking very effectively, with few errors. Using temporal information via hierarchical-HMM improves accuracy compared to the baseline method. The advantage of using HMM becomes more apparent in fall detection, in every video, falls are detected correctly. In case of HMM, false positives and negatives arise near the fall event annotation boundary, which may become subjective at sub-second resolution. However, the baseline approach produces false positives and negatives far from the true fall event.

The running time of the algorithm is measured for using multiple rectangles to represent silhouettes and it is compared to the naïve method (using the raw silhouette images). As it can be seen in Table 2.3, using multiple rectangles to represent silhouettes provides up to 7 fold speed up.

[1]Use this qr code to easily access the video:

Table 2.3. Running times of the proposed method using the silhouette image (IMG) and using multiple rectangles (RECT).

#	IMG (sec)	RECT (sec)	speed up
1	340	51	×6.66
2	302	44	×6.86
3	259	38	×6.81
4	253	37	×6.83
6	308	44	×7.00

2.3. MULTIPLE PERSON TRACKING

In this section we describe a method to perform multiple person tracking using omnidirectional cameras. As in Section 2.2, the approach described in this section is built on Fleuret *et al.* and Berclaz *et al.*'s work [2, 17]. In their work they also defined a structure called Probabilistic Occupancy Map (POM) that gives a probability of a person being present at a specific location. They define a generative model as described in Section 2.2 to infer POM via variational Bayesian optimization. In this section, a method to extend POM and K-shortest path for omnidirectional cameras is described. Experimentally, this new method is shown to successfully track multiple people using omnidirectional cameras.

2.3.1. Methodology

The ground plane in the scene is discretized into a grid where humans can be present. A person can only be present on one of these locations at a given time. The objective of multiple person tracking is to successfully infer the location of each person at each time frame correctly.

The multiple person tracking that will be detailed in this section consists of two steps. First, probability of a person being present at a location is obtained using Probabilistic Occupancy Map (POM) method. POM method is modified to work

with omnidirectional cameras. In the next step, a graph is constructed by connecting neighboring locations in subsequent time frames and setting edge weights according to the occupancy probabilities for each location. Each path in the graph corresponds to a person's trajectory. Starting from $K = 1$, K-shortest path is found in the graph. When the sum of the weights in the paths becomes non-negative the algorithm is terminated. In the end, each path corresponds to a person's trajectory and number of people in the scene is also inferred implicitly.

2.3.2. Probabilistic Occupancy Map

In this subsection obtaining the occupancy probabilities for each location using foreground segmentations is described in detail. The main idea is to do inference using the relation between the observed foreground segmentation and ideal foreground segmentation in a Bayesian framework. Similar to the silhouettes in Section 2.2, ideal foreground segmentations are obtained by placing cuboids on the grid locations.

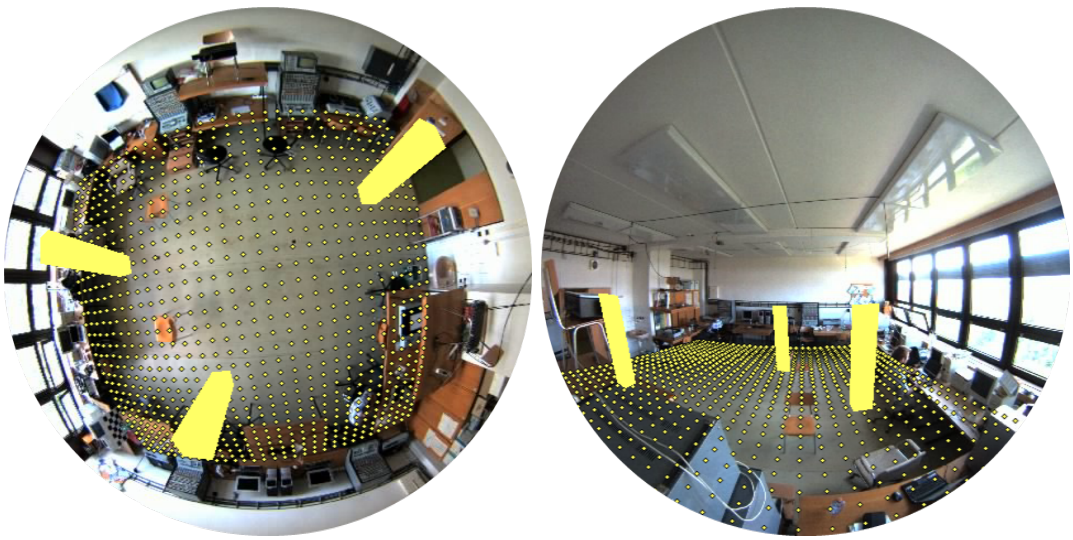


Figure 2.9. Ideal foreground segmentation where three locations are occupied. The ideal foreground segmentation is drawn onto the scene image for clarity. Yellow dots represent discrete locations where human can be present.

Let X^k be a binary variable that represents the occupancy state of the location k; $X^k = 1$ if k^{th} location is occupied. Let A^c be the ideal foreground segmentation obtained from camera c. An example of the ideal foreground segmentation can be seen

in Figure 2.9.

Let B^c be the foreground segmentation obtained from camera c. Also let bold typeface represent grouping over omitted indices. Let us define the probability of observing the foreground segmentations B^c as:

$$\begin{aligned} \mathrm{P}(\mathbf{B} \mid \mathbf{X}) &= \prod_c \mathrm{P}(B^c \mid \mathbf{X}) \\ &= \frac{1}{Z} \prod_c e^{-\Psi(B^c, A^c)} \end{aligned}$$

where $A, B \in \{0,1\}^{\text{width}\times\text{height}}$ and $\Psi(\cdot)$ is the asymmetric distance function between foreground segmentations given in Equation 2.11. According to this definition as the overlap between ideal and observed foreground segmentation increases as the distance decreases.

According to the model defined above it is hard to compute $\mathrm{P}(\mathbf{X} \mid \mathbf{B})$. Instead, $\mathrm{P}(\mathbf{X} \mid \mathbf{B})$ is approximated with a simpler Q distribution which minimizes the Kullback-Leibler divergence between them. Let q_k be $\mathrm{Q}\big(X^k = 1\big)$ and p_k be the prior $\mathrm{P}\big(X^k = 1\big)$, then we should take derivative of $KL(\mathrm{Q}, \mathrm{P})$ with respect to q_k and set it equal to zero

to find the value that minimizes the divergence:

$$
\begin{aligned}
0 &= \frac{\partial}{\partial q_k} KL(\mathrm{Q}, \mathrm{P}) \\
0 &= \frac{\partial}{\partial q_k} \left\langle \log \frac{\mathrm{Q}(\mathbf{X})}{\mathrm{P}(\mathbf{X} \mid \mathbf{B})} \right\rangle_{\mathrm{Q}} \\
0 &= \frac{\partial}{\partial q_k} \left\langle \log \frac{\mathrm{Q}(\mathbf{X})\,\mathrm{P}(\mathbf{B})}{\mathrm{P}(\mathbf{X})\,\mathrm{P}(\mathbf{B} \mid \mathbf{X})} \right\rangle_{\mathrm{Q}} \\
0 &= \frac{\partial}{\partial q_k} \left\langle \sum_l \log \frac{\mathrm{Q}\big(X^l\big)}{\mathrm{P}(X^l)} + \log \mathrm{P}(\mathbf{B}) - \log \mathrm{P}(\mathbf{B} \mid \mathbf{X}) \right\rangle_{\mathrm{Q}} \\
0 &= \frac{\partial}{\partial q_k} \left\langle \log \frac{\mathrm{Q}\big(X^k\big)}{\mathrm{P}(X^k)} - \log \mathrm{P}(\mathbf{B} \mid \mathbf{X}) \right\rangle_{\mathrm{Q}} \\
0 &= \frac{\partial}{\partial q_k} q_k \left(\log \frac{q_k}{p_k} - \big\langle \log \mathrm{P}(\mathbf{B} \mid \mathbf{X}) \mid X^k = 1 \big\rangle_{\mathrm{Q}} \right) \\
&\quad + \frac{\partial}{\partial q_k} (1 - q_k) \left(\log \frac{1 - q_k}{1 - p_k} - \big\langle \log \mathrm{P}(\mathbf{B} \mid \mathbf{X}) \mid X^k = 0 \big\rangle_{\mathrm{Q}} \right) \\
0 &= \log \frac{q_k}{p_k} + 1 - \big\langle \log \mathrm{P}(\mathbf{B} \mid \mathbf{X}) \mid X^k = 1 \big\rangle_{\mathrm{Q}} - \log \frac{1 - q_k}{1 - p_k} + 1 + \big\langle \log \mathrm{P}(\mathbf{B} \mid \mathbf{X}) \mid X^k = 0 \big\rangle_{\mathrm{Q}} \\
0 &= \log \frac{q_k(1 - p_k)}{(1 - q_k)p_k} - \left\langle - \sum_c \Psi(B^c, A^c) \mid X^k = 1 \right\rangle_{\mathrm{Q}} + \left\langle - \sum_c \Psi(B^c, A^c) \mid X^k = 0 \right\rangle_{\mathrm{Q}}
\end{aligned}
$$

where $\langle \cdot \rangle_{\mathrm{Q}}$ denotes the expected value under Q distribution. When we leave the q_k alone we get:

$$
\mathrm{Q}\big(X^k = 1\big) = \frac{1}{1 + \exp\left(\lambda_k + \sum_c \Psi_1^{c,k} - \Psi_0^{c,k}\right)} \tag{2.14}
$$

such that

$$
\begin{aligned}
\lambda_k &= \log \frac{1 - \mathrm{P}(X_k = 1)}{\mathrm{P}(X_k = 1)} \\
\Psi_w^{c,k} &= \Psi\left(B^c, \big\langle A^c \mid X^k = w \big\rangle_{\mathrm{Q}}\right).
\end{aligned}
$$

According to Equation 2.14, if putting a person silhouette to location k decreases the distance $\Psi_1^{c,k}$, in other words, explains the observation better, the probability of person being present at that location, $\mathrm{Q}\big(X^k = 1\big)$, increases. Similarly, if removing the person silhouette completely from location k explains the observation better, i.e. increases the distance $\Psi_0^{c,k}$, the probability $\mathrm{Q}\big(X^k = 1\big)$, increases.

The Equation 2.14 can be solved with a fixed point iteration because it has Q on both sides. Iterations are initialized with small $\mathrm{Q}\big(X^k = 1\big)$ values, first the right hand side of the equation is solved and new $\mathrm{Q}\big(X^k = 1\big)$ value is obtained. This is repeated until convergence. However, the $\mathrm{Q}\big(X^k = 1\big)$ values usually oscillates without convergence. To remedy this, as a common practice, damping is used [70]. At each step values are smoothed by averaging between the next and current one. An example run can be seen in Figure 2.10.

Computing $\Psi_w^{c,k}$ is the bottleneck of the fixed point iterations, it has to be computed for every location. This computation can be accelerated by the following procedure. $\Psi^{c,k} = \Psi\left(B^c, \left\langle A^c \mid X^k = w\right\rangle_{\mathrm{Q}}\right)$ is calculated once at the start of the iteration. Then, for each location k, by using integral images and the trick explained in Section 2.2.1 it takes $O(1)$ time to compute conditioned distance $\Psi_w^{c,k}$ from $\Psi^{c,k}$.

After following the steps described above the probability of a human being present at each location, in other words Probabilistic Occupancy Map, Q, is obtained.

2.3.3. Tracking using K-shortest path

In this section, for multiple person detection using the probabilistic occupancy maps for each time frame along with K-shortest path algorithm [17] is described.

This method is based on doing a flow optimization on a directed graph where each discrete location at each time frame is a node and subsequent neighbor nodes are connected. Let us define *flow* as the value of the edge between two nodes. The value of the flow between subsequent nodes being 1 means one person moved from one node to

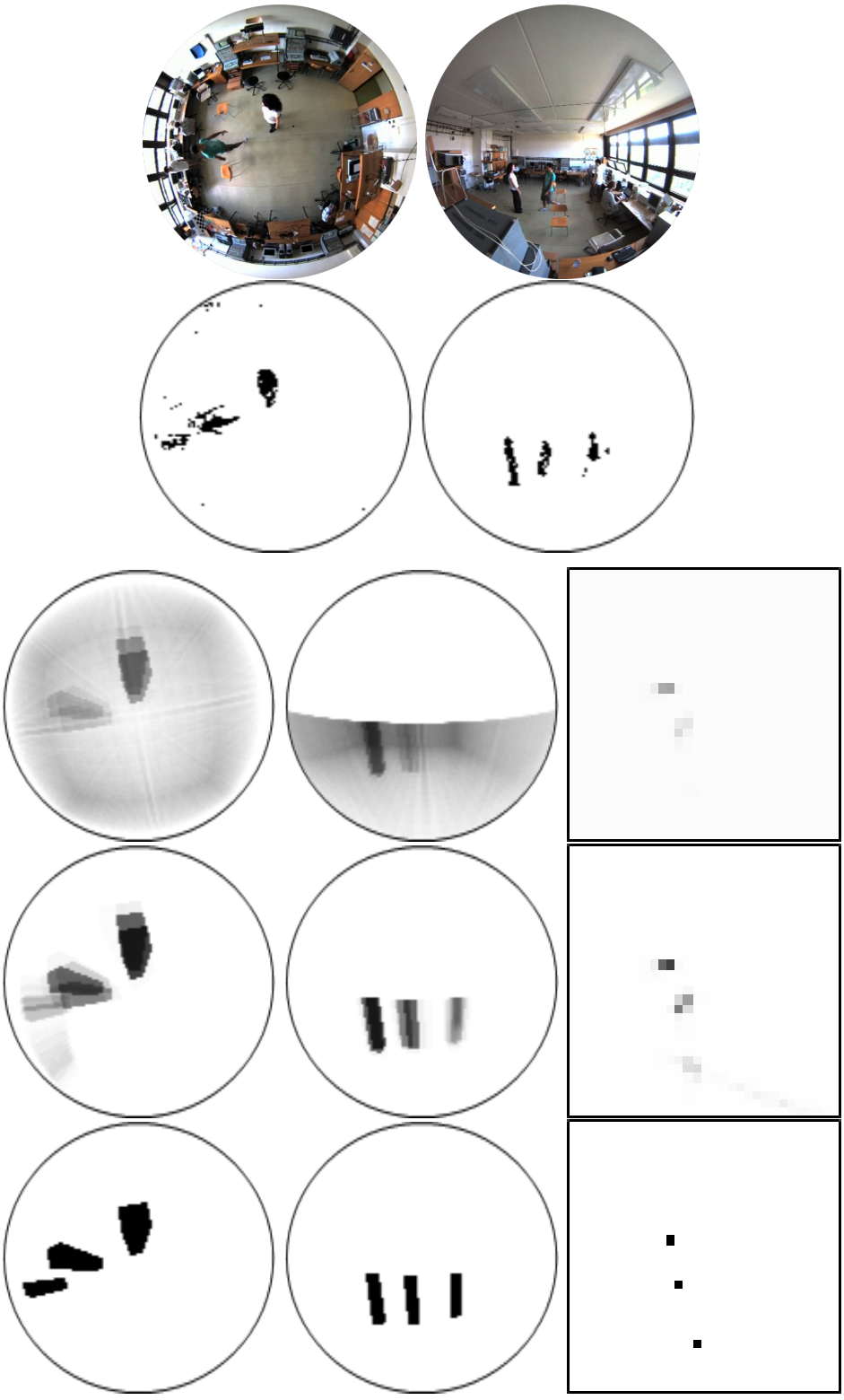


Figure 2.10. A visualization of POM calculation convergence after 63 iterations. Rows in order: Input images, foreground segmentations, $\langle A^c \rangle_Q$ and POM after 15, 20 and 63 iterations. The dark locations in the POM corresponds to the locations where the probability of a human being present in that location is high.

another. In addition to the location nodes let us define *source* and *sink* nodes. All of the flows should start from *source* and end up at *sink*. Entrance and exits into and out of the scene are handled through these nodes; all of the locations that can be used as entry and exit nodes are connected to *source* and *sink* respectively.

Furthermore, there is an edge from *source* to all of the nodes in the first time frame and there is an edge from all of the nodes to the *sink* in the last time frame. This allows to handle the situations where people are present at the beginning and end of the video. An example of the graph described above can be seen in Figure 2.11.

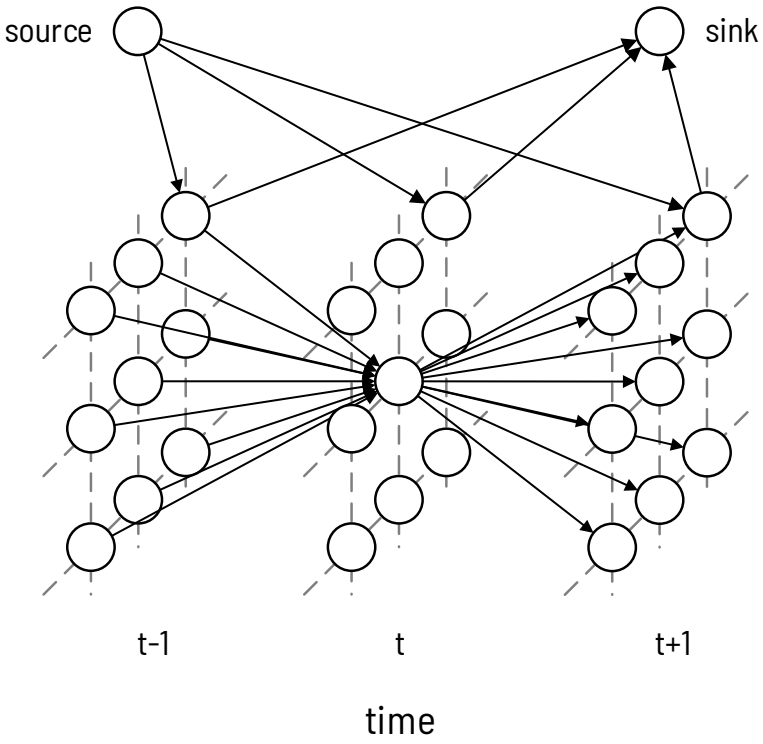


Figure 2.11. A visualization of the problem defined as network flow optimization. Most of the connections are omitted for clarity. It is assumed entrance and exits happen through one location in the figure.

The constraints for the flow in the graph are as follows:

- Flows should not be negative.
- Sum of the flows entering a node should be equal to the sum of the flows leaving that node.

- The sum of the flows leaving *source* should be equal to the sum of the flows entering *sink*.
- At a given time, at most one person can be present at a location. In other words, sum of the flows leaving a node should be less than or equal to 1.

Let X_t^k be a binary variable that represents the occupancy state of the location k at time t. The goal is to find $\mathbf{x}$ that maximizes the probability $\mathrm{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{B})$. Instead of P if we put its approximation Q in place and do the inference the objective function to maximize becomes:

$$\max \sum_{t,k} \alpha_k^t x_k^t$$

where

$$\alpha_k^t = \log \frac{\mathrm{Q}\big(X_t^k = 1\big)}{1 - \mathrm{Q}\big(X_t^k = 1\big)}$$

This objective function along with the constraints can be solved as a linear program. In [17], the authors showed that this linear programming formulation is equivalent to solving a K-shortest path problem where the edge weights leaving location k are $-\alpha_k^t$ at time t. According to this approach, starting from $K = 1$ and increasing K, K-shortest paths are calculated. When the objective value becomes non negative the ideal K value is found. This ideal K value is equal to the number of people that were present in the scene. The obtained shortest paths correspond to the people's trajectories.

2.3.4. Experiments

In our experiments we have used the videos of interacting people in BOMNI dataset. Please refer to the Section 2.1 for the details of the videos.

Note that the frames in a video coming from one camera does not exactly align with the frames from the other camera. During our experiments, we have observed that synchronizing video streams is very critical for tracking performance. When one camera is used, K-shortest path can find reasonable trajectories. However, two unsynchronized videos can produce inconsistent POMs for each step and K-shortest path fails to produce correct trajectories. The noise in the camera calibration and foreground segmentation can lead to slightly different POMs when using frames from one camera or two cameras. When there are many lost frames from one camera, these inconsistencies happen too frequently and K-shortest path algorithm can't find a reasonable trajectory for the locations with different occupancy probabilities. In other words if cameras switch too frequently their results won't be consistent.

For this reason we have processed the videos by pairing frames from one video with the closest frame in the other video.

We have ran the algorithms after resizing the frames to 160×120. For foreground detection we have used the method in [68] that models pixel intensity values as mixture of Gaussians. The locations where a human can be present is defined as a 31×31 uniform spaced grid on the ground plane. The transitions into and out of the room can happen through 7 manually selected locations that are just in front of the door. To generate a human silhouette at a certain location, a cuboid of size 16cm×16cm×180cm on that location is projected to each camera. Visualization of a single frame from tracking can be seen in Figure 2.12. A sample visualization of the trajectories can be seen in Figure 2.13. An example tracking video can be seen at [71][2] .

To assess the tracking performance CLEAR metrics, multiple object tracking precision and accuracy (MOTP and MOTA), are used [65]. See the Section 2.1.2 for the details of these metrics. We have used the bounding box of the ideal human silhouette (i.e. projection of the cuboid) to calculate MOTP. The results obtained are compared with the baseline in [11] on BOMNI dataset. The tracking results can be seen in

[2]Use this qr code to easily access the video:

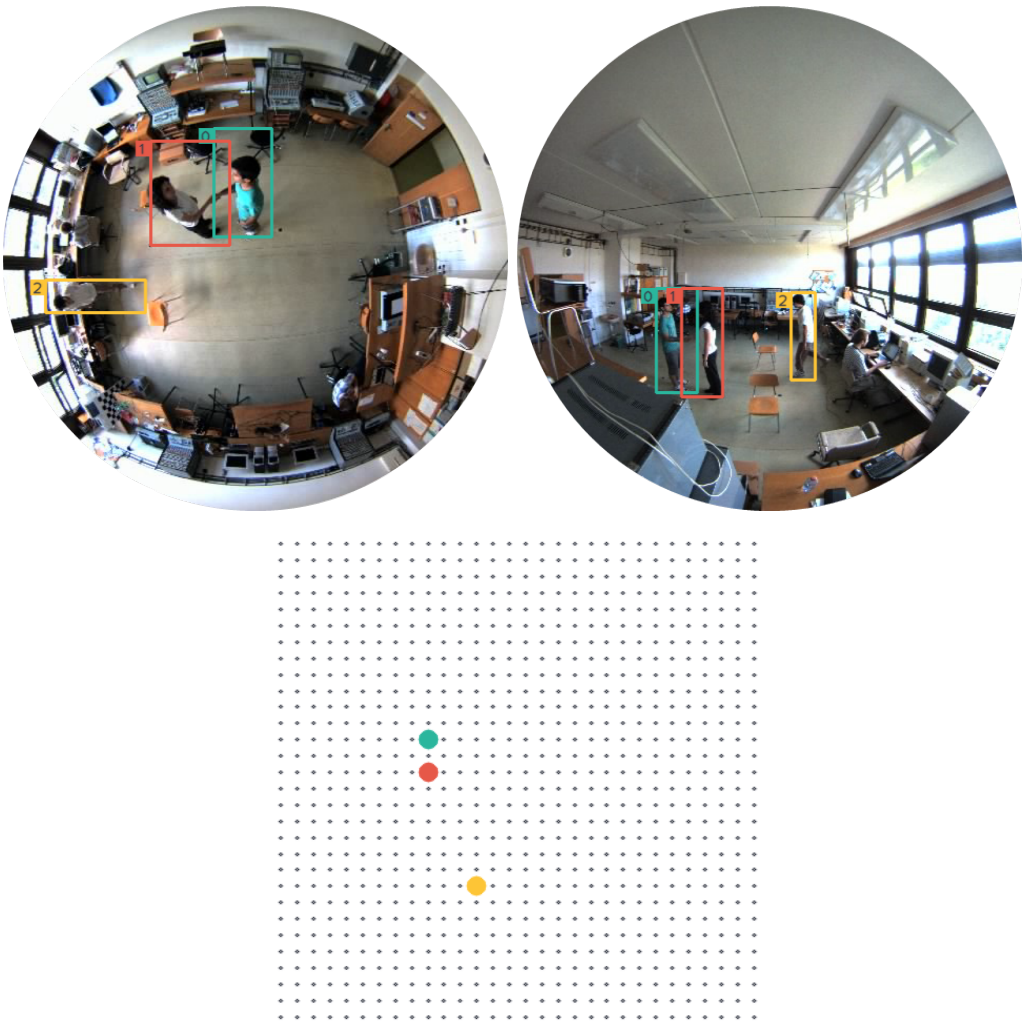


Figure 2.12. A sample frame from tracking on BOMNI dataset. Top: The bounding boxes obtained after tracking. Bottom: Visualization of people's location in top-down view.

Table 2.4.

As it can be seen in Table 2.4 the tracking accuracy is significantly better than the baseline with a margin. The precision is not higher because generally the bounding box of the ideal human silhouettes are larger than the real human bounding boxes.

Using only the side camera yields better precision because in top camera view people appear larger and it allows to generate more precise ground truth; as a result the non-overlap between ground truth and ideal silhouettes become larger. In addition to that, in top view the cuboid used to model the human shape is too coarse. Having said

Figure 2.13. Visualization of the trajectories in a video obtained after tracking. Uniform noise is added to the trajectories to distinguish trajectories in the visualization. Better viewed in color.

that, in multiple object tracking we are interested more in accuracy because precision is guaranteed to be greater than 0.5, which is already good enough, according to the matching criteria as described in Section 2.1.2.

As a result, the method detailed in this section successfully tracks multiple people using binary foreground segmentations as input. This method improves MOTA performance of the baseline method from 68.18% to 86.39%. Note that the conditions are very challenging. The videos heavily suffer from frame losses, the people are occluded frequently and foreground segmentations are usually noisy.

Table 2.4. The results obtained for multiple person tracking on BOMNI dataset with the approach presented in this section and comparison with the baseline in [11].

Camera	**MOTP (overlap)**	**MOTA**	**miss rate**	**false positive**	**mismatch**
[11] Both	0.73	68.18%	23.62%	7.74%	0.44%
Ours Both	0.66	86.39%	10.59%	2.83%	0.18%
[11] Top	0.72	73.52%	18.78%	7.35%	0.32%
Ours Top	0.62	86.27%	11.41%	2.16%	0.15%
[11] Side	0.74	62.21%	28.95%	8.17%	0.66%
Ours Side	0.70	86.52%	9.69%	3.57%	0.22%

3. DISCRIMINATIVE PERSON DETECTION

In the previous section a fall detection and multiple person tracking method is built on top of a person detection that relies on a generative model. When the locations of the people in the scene was given, the ideal foreground segmentation could be generated. In this chapter, a novel discriminative method to detect people in omnidirectional images is presented.

Our contribution in this chapter is twofold. First, a novel integral image scheme for omnidirectional images to speed up feature extraction is introduced. Integral images have been extremely useful for speeding up detection problems, and were popularized by the rapid face detection study proposed by Viola and Jones [7]. However, integral images work with rectangular bounding boxes, an assumption which no longer holds in omnidirectional images. As Figure 3.1 illustrates, the bounding box around the detected person is much narrower in its base. Our proposed solution makes it possible to use integral images directly on omnidirectional images.

As a second contribution, we advance the state of the art in omnidirectional camera based person detection. Using the new integral image structure and the *integral channel features* (ICF) approach [8], we outperform recent omnidirectional camera person detection algorithms [5]. We also compare our method with converting the omnidirectional image to a panoramic image and then applying the standard ICF method, and experimentally show the superior performance of our approach.

To our knowledge, the method we propose in this chapter is the first to compute the integral image for omnidirectional images and therefore, it is the first to apply the state of the art ICF method [72] for object detection with omnidirectional cameras. We

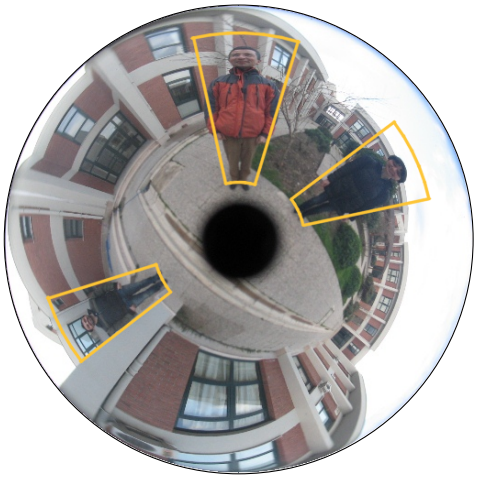


Figure 3.1. Person detection in an omnidirectional camera setting, with the detection results of the proposed approach superimposed.

describe our approach in the next section.

3.1. Integral Channel Features method

This subsection briefly describes the integral channel features method [8], followed by a description of the proposed *radial integral channel features* (RICF).

The integral channel features (ICF) method first computes multiple channels from a single input image. These include individual color channels (e.g. RGB, LUV), gradient magnitude, HOG, and the difference of Gaussian filtered image. Then, summarizing features are extracted from each of these channels. These features are pixel sums over rectangular regions on the channels, which can be computed rapidly using integral images. Finally, boosting is used for classification. Using shallow decision trees as weak classifiers also serves as feature selection process.

ICF is not directly applicable to omnidirectional images, because the rectangular feature extraction scheme and the sliding window approach are designed for perspective images, and fail for the omnidirectional image geometry. The irregular distribution of pixels in the omnidirectional image makes it impossible to apply the efficient recursive approaches used for rectangular areas [73]. Our proposed method replaces the sliding window with a rotating annular sector (ring/doughnut slice shape) as in [5], and

rectangular regions with annular sectors. To calculate pixel sums inside annular sectors rapidly, we propose a novel structure, *radial integral image* (Section 3.2). The idea is similar to the conventional integral image, but instead of querying points in the Cartesian coordinate system, polar coordinates are used. See Figure 3.2 for the illustration of the radial integral channel features method.

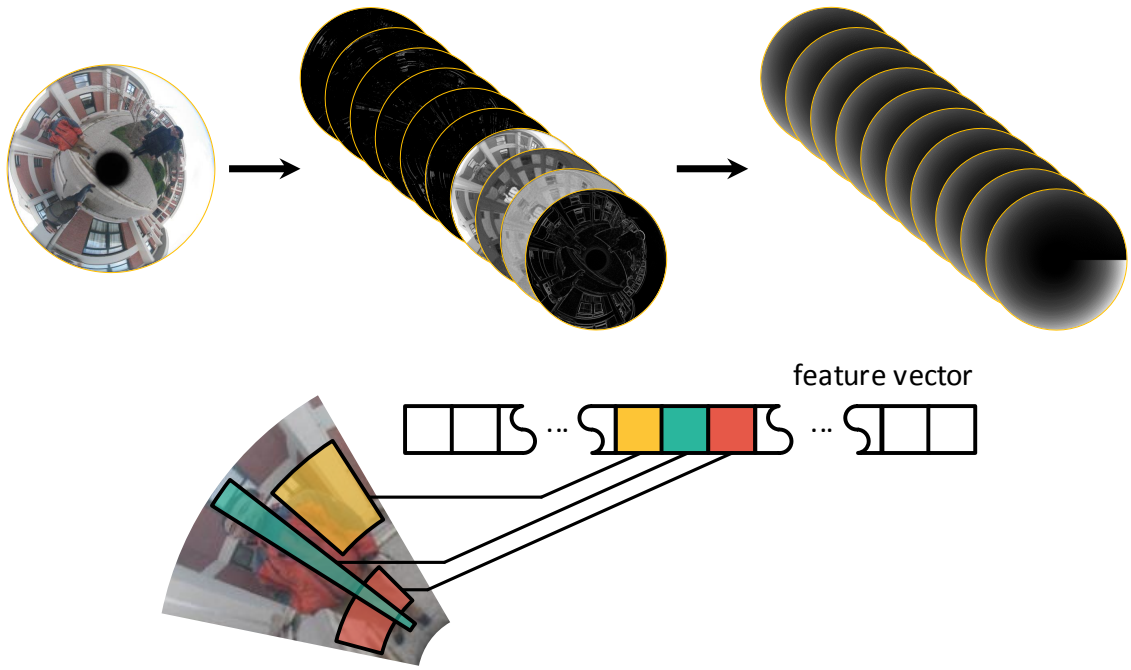


Figure 3.2. Visualization of the radial integral channel features. Top: First, the input image is transformed into various channels (LUV, gradient magnitude etc). Then, radial integral images are formed. Bottom: Lastly, for a given annular sector shaped window, the sum inside the annular sectors (corresponding to a channel) is calculated. Features are shown on the original image for clarity.

3.2. Radial Integral Image

An annulus is a region bounded by two concentric circles. Annular sector (a.k.a. circular ring sector or doughnut slice shape) is a cut from the annulus, which is bordered by two straight lines from its center.

Annular sector features are very similar to the rectangular features used in integral channel features (ICF) [8]. An annular sector feature is the sum of the pixels inside that annular sector. Instead of using Cartesian coordinates to obtain sums inside rectangular

regions, it uses polar coordinates to obtain sums inside circular sectors. Annular sector features can be calculated rapidly using the proposed radial integral image.

Radial integral image, $\tilde{I}$, is defined as:

$$\tilde{I}(p) = \sum_{q:\theta_q \leq \theta_p, r_q \leq r_p} I(q)$$

where I is the input image, p and q are pixels, θ_p and r_p are angular and radial coordinates of pixel p.

According to this definition, we can calculate the sum inside an annular sector $(\theta_{\min}, \theta_{\max}, r_{\min}, r_{\max})$ as:

$$S(\theta_{\min}, \theta_{\max}, r_{\min}, r_{\max}) = \tilde{I}(p_{\theta_{\max}, r_{\max}}) - \tilde{I}(p_{\theta_{\max}, r_{\min}}) - \tilde{I}(p_{\theta_{\min}, r_{\max}}) + \tilde{I}(p_{\theta_{\min}, r_{\min}})$$

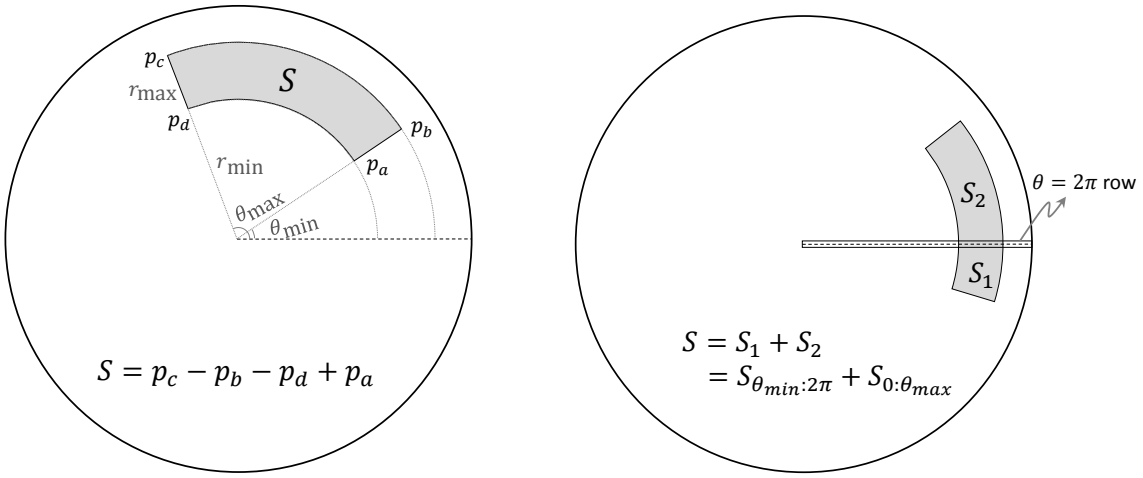
where $p_{\theta,r}$ is the pixel that has the polar coordinates (θ, r). See Figure 3.3a for an illustration. Once the radial integral image is computed, calculating each sum has $O(1)$ complexity (four look-ups and three operations).

Since a given (θ, r) usually corresponds to fractional pixel coordinates in the actual image plane, bilinear interpolation is used to calculate the pixel values in the integral image. Note that, using interpolation to lookup the values instead of rounding the coordinates is critical for the detection performance. Rounding leads to inclusion of unwanted pixel values in the sum, whereas interpolation provides a value much closer to the true sum inside the given range.

If the annular sector crosses the $\theta = 0$ angle, the sum can not be calculated directly, but it can be decomposed into two sums (Figure 3.3b):

$$S(\theta_{\min}, \theta_{\max}, r_{\min}, r_{\max}) = S(\theta_{\min}, 2\pi, r_{\min}, r_{\max}) + S(0, \theta_{\max}, r_{\min}, r_{\max})$$

This requires looking up pixel values corresponding to $\theta = 2\pi$, which can be achieved by storing an extra row for $\theta = 2\pi$ in addition to the radial integral image.



(a) (b)

Figure 3.3. (a) Illustration of rapid computation of the sum inside an annular region using radial integral image. Four look-ups and three operations are sufficient to calculate the sum. (b) If an annular sector query crosses $\theta = 2\pi$ angle, it can be broken up into two annular sectors. In this case, an extra column is needed to look-up values that correspond to $\theta = 2\pi$.

When n denotes the number of pixels in the image, a naïve algorithm to compute the radial integral image has $O(n^2)$ complexity, because for each pixel, θ and r should be compared with a subset of pixels in the image (on average half of the pixels). In the following section, a way to compute the radial integral image in $O(n \log n)$ time is introduced.

3.2.1. Fast computation of the radial integral image

In this section, a multidimensional divide-and-conquer approach [74] is adapted to compute the radial integral image. In this approach, to solve a problem of n points in 2D space, one first recursively solves two problems of $n/2$ points in 2D space, then solves one problem of n points in 1D space in linear time. Let $T(n)$ denote the complexity of calculating the radial integral image, the corresponding recurrence becomes

$$T(n) = 2T(n/2) + O(n)$$

which is solved in $O(n \log n)$ time.

We apply this approach to our case as follows: Along with each pixel, we store radius (i.e. distance from image center), angle, sum and pixel value information. Note that pixels do not have a grid structure in 2D (r, θ) space (Figure 3.4a). We say that point p *dominates* q if all the coordinates (r and θ) of p are greater than that of q. First, the set of all the pixels is divided into two sets A and B using the median radius value (Figure 3.4b). This means that the radius of every pixel in B is greater than the radius of every pixel in A and no pixel in A is dominating a pixel in B. Then, the algorithm is recursively called for A and B (Figure 3.4c). At this point, we know that every pixel in A gives the desired sum, and each pixel in B gives the sum of the dominated pixels in B. Some pixels in B might be dominating particular pixels in A and need to be updated. Finally, two sets are merged and pixels in B are updated by iterating pixels by increasing the angle value, i.e. working in 1D space (Figure 3.4d). At the end of iteration, each pixel value denotes the sum of the pixel values in the original image which are dominated by that pixel.

When the number of pixels in the set is less than a particular value, we stop the recursion and switch to a naïve implementation. This strategy avoids the recursion overhead for small sets and increases CPU cache utilization. The critical set size is observed to be 16 for the hardware architecture used in the experiments. The code for calculating radial integral image and extracting pixel sums is publicly available at [75][3]
.

3.3. Experiments

3.3.1. Experimental setup

The conducted experiments follow the very same setting to the one described in the original ICF paper [8]. For each input image, 10 channels are extracted in total: LUV

[3]Use this qr code to easily access the repository:

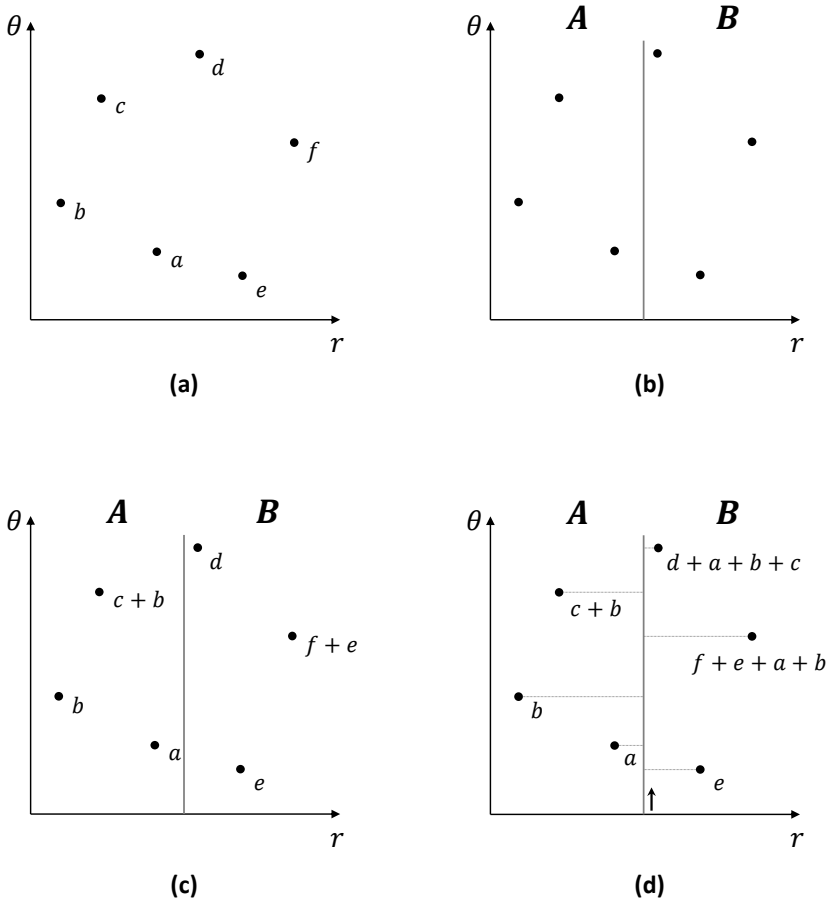


Figure 3.4. Steps of the radial integral image computation. (a) Polar coordinates and values of the pixel are given as the input. (b) Pixels are split from the median radius to form two equally sized sets, A and B. (c) The algorithm is recursively called for each set. (d) Pixels are iterated according to increasing angle value and two sets are merged by updating the values in B.

for color, gradient magnitude and gradient histogram for 6 equally spaced orientations. 20 000 random features are generated and an AdaBoost classifier is trained with 1024 decision trees with a depth of two. This gives a good balance between classification accuracy and computational load.

In the experiments, the validity of the spherical camera model (Section 1.1.1.3) is assumed and a dataset collected with a camera that has a parabolic mirror is used.

Three sets of experiments are presented in this section. For the first experiment, artificial omnidirectional images are generated using the images in the INRIA dataset [36]. For this experiment false positive per window versus miss rate is reported and the validity of our approach is demonstrated. The second experiment was conducted on real omnidirectional images and compares a state of the art approach with the proposed method, illustrating the improvement in accuracy and speed. Lastly, the performance of radial integral channel method is compared against transforming omnidirectional images to panoramic images and using a conventional person detection method. This last set of experiments reveal that working directly on omnidirectional images is a better approach.

3.3.1.1. Gradient correction. It has been shown that modifying gradient magnitudes according to the Riemannian metric on the sphere improves human detection performance using omnidirectional cameras [5]. For cameras with parabolic mirror, the gradient magnitude channels are updated with:

$$|\nabla_{S^2} I| = \frac{(4 + x^2 + y^2)}{4} |\nabla_{\mathcal{R}^2} I|$$

where $|\nabla_{S^2} I|$ and $|\nabla_{\mathcal{R}^2} I|$ are the gradient magnitudes on the sphere and the image, respectively. Observe that at the center of the omnidirectional image, $(x,y)=(0,0)$, gradients are the same. While moving from the center to the periphery of the omnidirectional image, gradients on the sphere are the magnified versions of the gradients on the image.

3.3.2. Evaluation on the INRIA Person Dataset

The first set of experiments are intended as a validation of the method using artificially generated omnidirectional images. For this purpose, a virtual camera is created, INRIA images are placed such that the persons feet are on the ground plane, and INRIA images are projected onto the omnidirectional camera's image plane (Figure 3.5). For each image, a random rotation around the vertical axis is applied to the camera.

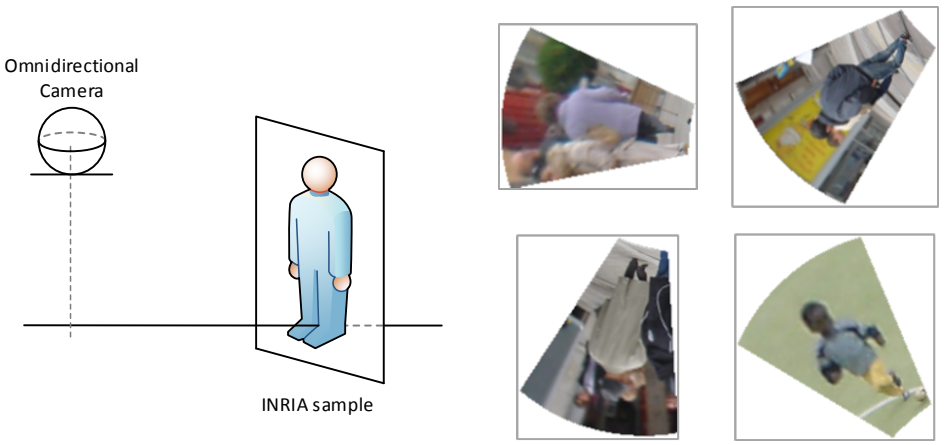


Figure 3.5. Left: Each sample in the INRIA dataset is placed on the ground plane and a virtual omnidirectional image is formed (camera is shown as a sphere since we used the sphere camera model). Right: Some examples of the resulting images.

The classifier is trained with all of the 2416 positive samples and 5000 random windows from negative samples from the INRIA training set.

Using these artificial omnidirectional images, a boosted classifier is trained with features extracted using radial integral images. For performance, a custom radial integral image implementation is used which processes a part of the full omnidirectional image (which corresponds to the projected INRIA image). Feature extraction, training and classification require minimal resources. All of the experiments are run on a PC with an Intel i7 CPU and 4GB RAM.

False positive per window (fppw) versus miss rate is given in Figure 3.6. 85.3% detection rate is observed at the reference point of 10^{-4} fppw. Considering that the detection rate for perspective cameras is around 90% [8], this result shows that our approach is plausible for semi-synthetic omnidirectional images. In Figure 3.6, also the performance of the same classifier is plotted by changing the distance of the INRIA samples to the camera by -20%, -15%, -10%, +10%, +20% and +30%. A single classifier is trained and run on the different samples without scaling the detection window or the test image. The performance decreases rapidly as the samples are placed away from the distance it was trained on. This is due to the image of the INRIA samples being significantly different at different distances and the classifier trained for a particular

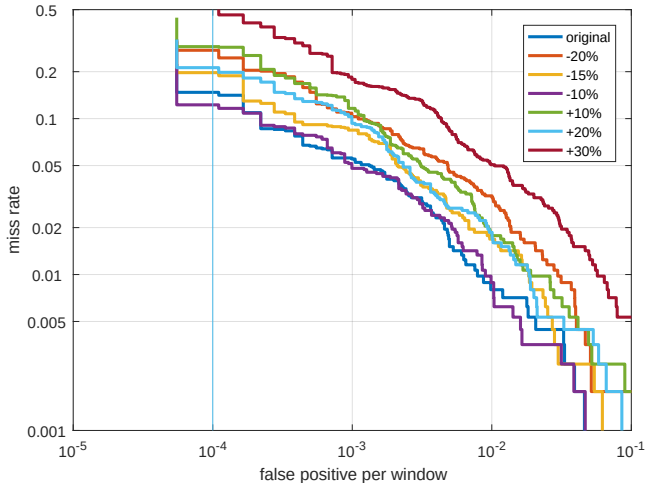


Figure 3.6. The performance of the proposed method (RICF) on the INRIA dataset, where each window is transformed into an omnidirectional image. The performance decreases rapidly as the test images gets away from the distance classifier is trained for.

distance cannot be applied directly to the human shapes at different distances. For detection in real omnidirectional images, we run the detector at multiple scales to detect people at different distances (see Figure 3.1 and 3.9). In the next section (Section 3.3.3) the performance on real omnidirectional images is reported. The visualization of the channels and selected features after boosting can be seen in Figure 3.7. In the figure, top two rows visualize channels generated from the input image: 6 gradient orientations (starting from zero degrees), gradient magnitude and L, U, V channels respectively. Bottom two rows illustrate learned features from the training data. Each feature region is painted and averaged for visualization. Images correspond to the aforementioned channels in the same order. Note that for related gradient orientations, features are clustered around shoulders. For the gradient magnitude and color channels, most of the learned features are from regions covered by the human body.

False positive per window and false positive per image metrics serve different purposes. The false positive per window criterion is better suited for evaluating a binary classifier. On the other hand, for evaluating an end-to-end detection system, using false positive per image (fppi) is a better criterion, because in such a setting ideally all possible windows are considered, resulting potentially in much more false positives than the per-window approach [76]. Besides, not all detectors work on a

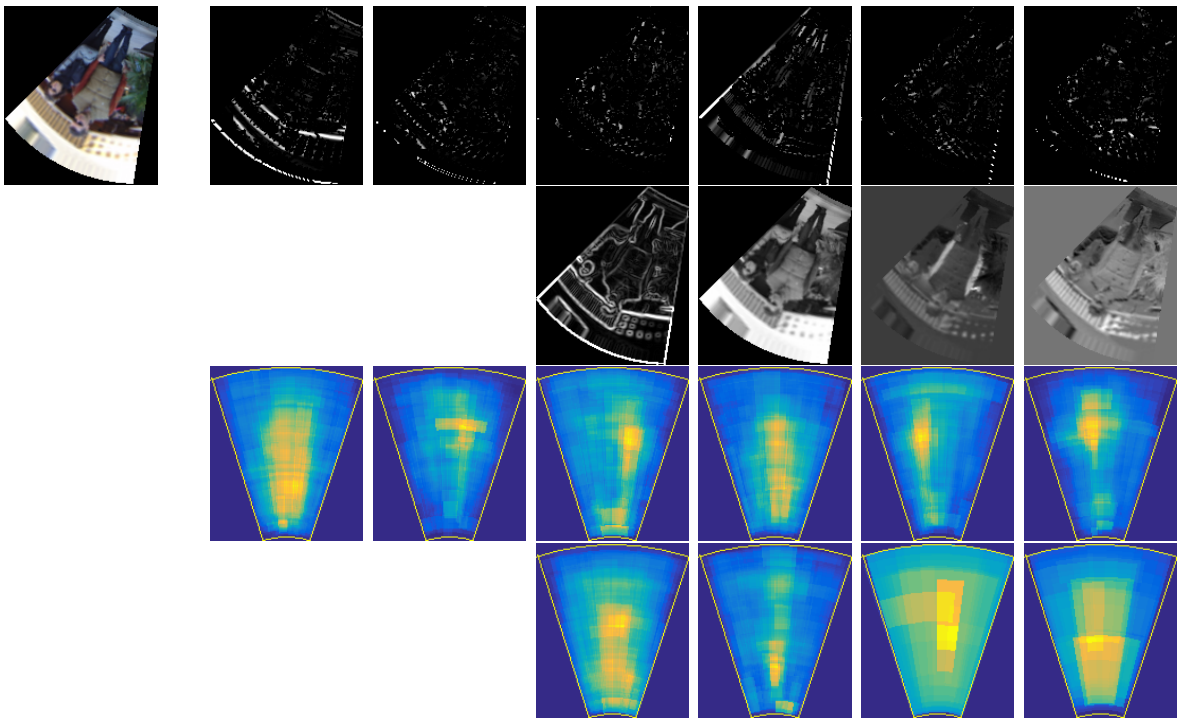


Figure 3.7. Top left: An example image synthesized from the INRIA dataset. Top two rows: Channels generated from the image. Bottom two rows: Visualization of learned features from the training data. See text for details.

per-window basis. Using fppi allows us to compare results of different type of detectors. However, in this section fppw is used because, in this set of experiments human centered omnidirectional images are used and they leave the rest of the image unrealistic, if not empty. Experiments using the false positive per image (fppi) criterion is reported in Section 3.3.3, which uses real omnidirectional images.

3.3.3. Evaluation on the IYTE omnidirectional image dataset

For the second experiment RICF method is compared with Cinaroglu and Bastanlar's recent approach, which is the state of the art in omnidirectional person detection [5]. A detector is trained using artificial omnidirectional images using the INRIA dataset as described in the previous section. For testing, the same dataset with [5] (IYTE dataset - available at [77]) is used. This dataset contains images taken with a real omnidirectional camera (Figure 1.10) and humans are manually annotated for each segmented person as annular sectors. While testing, a scale step of 1.04 and window

step size (i.e. stride) of 6 pixels is used. Note that for rotating annular sectors, the window step size corresponds to varying angle step size for different radii. Experiments with different stride sizes are also conducted (Figure 3.8). We have observed that increasing stride affects the performance only slightly because, in IYTE dataset the humans are large and the detection windows are always dense enough on multiple scales to capture humans. In Figure 3.1 an example result of the RICF method can be seen. In Figure 3.9 example instances can be seen where the RICF method fails.

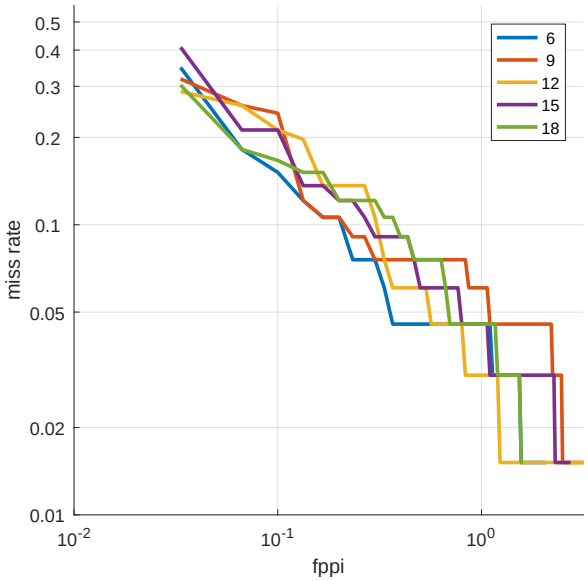


Figure 3.8. The effect of using different stride values are not large due to rotating detection windows being dense enough to capture humans.

In [5], precision-recall curves are reported for their OmniHOG method. To keep the comparison fair, the results are reported using the same metrics, namely precision and recall. Also negative samples are selected with a similar ratio. Furthermore, a false positive per image vs. miss rate plot is provided in Figure 3.10b, which is better suited for the person detection task.

The precision-recall and fppi-missrate curves show that RICF method have surpassed the detection performance of OmniHOG (Figure 3.10). It obtained 11.59% log-average miss rate on the IYTE dataset where the miss rate at fppi = 1 is 4.5% (Figure 3.10b). In [72] it is reported that 10% missrate at fppi=1 is the best result for the datasets evaluated. Although the dataset used in this experiment is different and not as challenging as some of those datasets, our results show that using semi-synthetic omnidirectional images is a viable way to train person detectors for omnidirectional cameras.

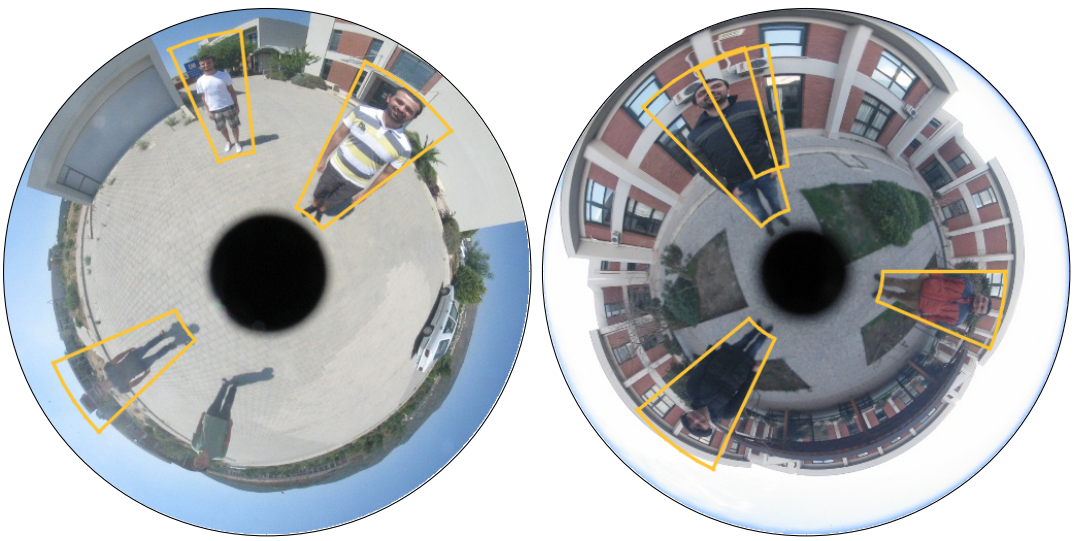
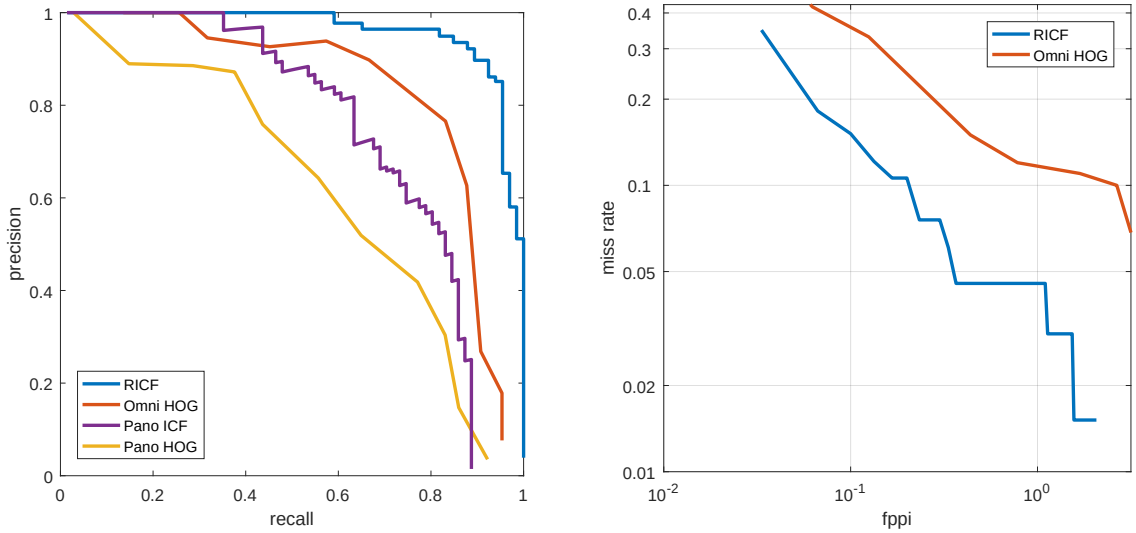


Figure 3.9. Example instances where the RICF method fails. Top: False negative example. Bottom: False positive example.

Also note that, RICF method is much faster than OmniHOG, since in OmniHOG, the transformation of HOG features is done separately for each sliding-rotating window. For a given input image, OmniHOG takes about 17 milliseconds per window, where lightly optimized RICF implementation takes 1.6 milliseconds per window on a similar hardware. In other words, RICF method is more than 10 times faster than the previous

state of the art.



(a) (b)

Figure 3.10. (a) Comparison of our method with [5]. Working directly on omnidirectional images outperforms transforming image to panoramic and using conventional method (Pano ICF). Omni HOG and Pano HOG results are directly taken from the compared previous study [5]. Best viewed in color. (b) Miss rate vs. false positive per image plot of our method.

3.3.4. Evaluation on panoramic images

Lastly, the performance of the RICF method is compared against transforming the omnidirectional image to a panoramic image and using a conventional integral image sliding window approach [8]. The RICF method is compared against this baseline approach, because the baseline approach is usually the most straightforward way to work with omnidirectional images [54–57].

First, omnidirectional images are converted to panoramic images using spherical projection. Spherical projection provides equi-angular representation in the vertical direction of panoramic images and it was shown to provide better performance over cylindrical projection [78]. The resulting image height is set such that that it preserves the 2:1 aspect ratio for humans. Although doing this gives the panoramic method an advantage, the RICF method outperformed the panoramic method (Figure 3.10a).

This shows that working directly on omnidirectional images instead of transforming them into panoramic images has clear benefits for person detection. Besides, since the omnidirectional integral image is computed only once for the input image, the computational complexity of detection on the omnidirectional image is not higher than converting to a panoramic image and applying the standard perspective camera method.

3.4. Conclusion

In this chapter, a novel method is introduced to detect people in images acquired by omnidirectional cameras. This method is called radial integral channel features (RICF). Also a new data structure called *radial integral image* is introduced to speed up feature extraction in omnidirectional images. RICF beats the current state of the art for person detection in omnidirectional cameras and demands less computational resources.

The experiments illustrate that working directly on native omnidirectional images is better than converting them to panoramic images, followed by applying traditional approaches. The distortions caused by such rectification are problematic. If efficient native versions of useful algorithms are introduced, omnidirectional cameras will be more accessible to system developers.

Efficient omnidirectional image processing for detecting humans has great potential for many applications, including indoor scenarios such as smart environments and mobile robot based applications, as well as outdoor scenarios, such as pedestrian detection. We believe advances such as proposed in this chapter will result in more widespread use of omnidirectional cameras.

4. RECTANGLE BLANKET PROBLEM

Summing pixel values over arbitrary regions is a reoccurring theme in various applications, especially in computer vision domain such as template matching [9, 79], people detection and tracking [2, 17, 80], image matching and visual odometry [10, 81], object localization [82], object detection with convolutional neural networks [83]. As described in Chapter 2, the process for person detection and tracking requires generation of human silhouettes. At the inference phase, the pixel sums are computed over each silhouette region corresponding to each location. To speed up computing the sums over these arbitrary shapes, these regions are approximated with multiple non-overlapping axis-aligned rectangles. Then integral images are used to calculate pixel sums rapidly in these rectangular regions.

A rectangle is axis-aligned if its adjacent (orthogonal) edges are parallel to x and y axes. An illustration for an approximation of a shape with three rectangles is provided in Figure 4.1. Here, the rectangles determine a *rectangle blanket*. Blanket is defined as a set of non-overlapping axis-aligned rectangles. Note that neither the shape nor the rectangles need to be fully contained within the other. There can be parts of the target shape that is not covered by the rectangles and also rectangles can exceed the boundaries of the target shape. Nevertheless, it is possible to say that the number of rectangles forming a blanket directly effects the quality of the approximation; the higher it is, the finer the approximation becomes.

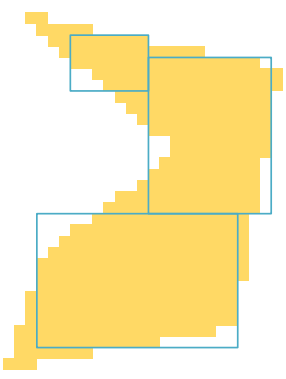


Figure 4.1. An example of a rectangle blanket having $K = 3$ rectangles.

The *Rectangle Blanket Problem* (RBP) is formally defined as determining a rectangle set that contains no more than K non-overlapping axis-aligned rectangles that minimizes the non-overlapping area with the given two-dimensional image. This problem has similarities with *rectangle covering* [84–86], *rectangle partitioning* [87, 88] and *cutting / packing* [89, 90] problems. In this chapter, the focus is on RBP, which can be interpreted as a slightly more relaxed version of partitioning or covering problems. In fact, the binary integer linear programming (BIP) formulation of RBP corresponds to a set packing problem as it will be described in the following section.

Chan *et al.* tackled a very similar problem in the context of integrated circuit manufacturing where the rectangles are allowed to overlap [91]. After transforming the binary input image appropriately, RBP can also be considered as a generalization of Bentley's classic *maximum sum subsequence problem* to two dimensions for multiple subsequences [92]. In fact, Csűrös had previously proposed a method to find a set of K disjoint subsequences of a one dimensional array such that the sum of all elements in the set is maximized [93]. Later, Bengtsson and Chen improved the efficiency of the method to have linear time complexity [94, 95] .

In this chapter, first a BIP formulation of RBP is given and a branch-and-price algorithm is proposed for its exact solution in Section 4.1. However this can be computationally expensive for large instances. To overcome this size limitation, three heuristics are given in Section 4.2. Two of these heuristics are new methods; the method described in Section 4.2.3 is a constrained simulated annealing heuristic and the other one described in Section 4.2.1 is an incremental algorithm where a single rectangle is added to the set at each step. Section 4.3 is essentially a report on the results of a detailed computational study performed for assessing the quality of these methods.

4.1. Problem Formulation

A *polyomino* is a union of unit squares, namely a square with integer coordinates and area of one [96]; and it is possible to represent a shape as a polyomino $\mathcal{P}$, by replacing any pixel $\mathbf{p}$ belonging to its binary image with a unit square. Given a

polyomino $\mathcal{P}$, it is possible to construct a graph $G = (V, E)$ with vertices as all pixels contained in $\mathcal{P}$. Two vertices are the endpoints of an edge if and only if $\mathcal{P}$ contains an axis-aligned rectangle containing both pixels. G is called the *visibility graph* of the polyomino $\mathcal{P}$ [97–99] and has interesting properties. Györi [100] has shown that $\alpha(G)$, the minimum number of cliques that cover G, is equal to $\chi(G)$, the maximum number of independent vertices (i.e. the size of the maximum stable set), if each vertical or horizontal line has a convex intersection with $\mathcal{P}$, namely if $\mathcal{P}$ is *orthogonally convex*. This is a consequence of the fact that G is perfect for orthogonally convex images. Using the correspondence between a clique and a rectangle, and the relation $\alpha(G) \geq \chi(G)$ it is possible to obtain a lower bound on the size of a rectangle cover of $\mathcal{P}$. An intriguing question is whether there is a constant factor approximation algorithm for the rectangle cover problem [101], which is related to whether the ratio $\alpha(G)/\chi(G)$ is bounded by a constant. Although this is still an open question, numerical results [85] report in their study support the fact that such an approximation exists. These results are based on the BIP formulation of a set covering problem. In this chapter, this line of research is followed and a BIP formulation for RBP is proposed.

A target image $\mathcal{I}$ is a union of disconnected polyominos and it is possible to associate a zero-one matrix $I \in \mathbb{B}^{W \times H}$ with the target image $\mathcal{I}$, where W and H are the dimensions of the target image. These dimensions are the dimensions of the smallest rectangle covering the polyominos representing the target image fully. Then, for $p_1 = 1, 2, \ldots, W$ and $p_2 = 1, 2, \ldots, H$

$$I_{p_1 p_2} = \begin{cases} 1 & \text{if pixel } \mathbf{p} \text{ belongs to target image } \mathcal{I} \\ 0 & \text{otherwise,} \end{cases} \tag{4.1}$$

is a binary matrix representing target image $\mathcal{I}$, where $\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$ are the indices of the cell assigned to pixel $\mathbf{p}$.

Let the rectangle r be presented with the binary matrix $\mathbf{r} \in \mathbb{B}^{W \times H}$ such that

$$r_{p_1 p_2} = \begin{cases} 1 & \text{if } r_{\text{left}} \leq p_1 \leq r_{\text{right}} \text{ and } r_{\text{top}} \leq p_2 \leq r_{\text{bottom}} \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

for any pixel $\mathbf{p}$ with cell coordinates $\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$. Here, $r_{\text{left}}, r_{\text{right}}, r_{\text{top}}$ and r_{bottom} are the coordinates of the left, right, top and bottom edges of the rectangle, respectively. Note that according to conventions, y-axis of the coordinate system is oriented downwards therefore, $r_{\text{top}} \leq r_{\text{bottom}}$ for rectangles.

Let $\mathcal{R}$ be the set of all possible rectangles in $\mathbb{B}^{W \times H}$ and has size $|\mathcal{R}| = R$. Then RBP can be formulated as the subset selection problem:

$$\text{RBP: } \min \left\{ z(\mathcal{B}) : \mathcal{B} \subseteq \mathcal{R}, |\mathcal{B}| \leq K, \text{ and } \mathcal{B} \text{ is a blanket} \right\}, \tag{4.3}$$

for a given target image $\mathcal{I}$. A feasible subset $\mathcal{B} = \{\mathbf{r}^1, \mathbf{r}^1, \ldots, \mathbf{r}^B\}$ represents a blanket with size $B = |\mathcal{B}| \leq K$ and has cost

$$\begin{aligned} z(\mathcal{B}) &= \sum_{\mathbf{r} \in \mathcal{B}} \left(\sum_{p_1=1}^{W} \sum_{p_2=1}^{H} r_{p_1 p_2} - \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} I_{p_1 p_2} \times r_{p_1 p_2} \right) \\ &+ \left(\sum_{p_1=1}^{W} \sum_{p_2=1}^{H} I_{p_1 p_2} - \sum_{\mathbf{r} \in \mathcal{B}} \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} I_{p_1 p_2} \times r_{p_1 p_2} \right). \end{aligned} \tag{4.4}$$

The first term is the area (i.e. the number of pixels) of blanket $\mathcal{B}$ overflowing the target image and the second term is the area of the target image not covered by blanket $\mathcal{B}$, since rectangles are non-overlapping. The product $I_{p_1 p_2} \times r_{p_1 p_2}$ is 1 if $\mathbf{p}$ belongs to both the target image and rectangle r; otherwise it is 0. Besides, the first term of the second line equals to the size of the target region, namely $\sum_{p_1=1}^{W} \sum_{p_2=1}^{H} I_{p_1 p_2} = |\mathcal{I}|$ and the objective can be rearranged as:

$$z(\mathcal{B}) = |\mathcal{I}| + \sum_{\mathbf{r} \in \mathcal{B}} \left(\sum_{p_1=1}^{W} \sum_{p_2=1}^{H} r_{p_1 p_2} - 2 \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} I_{p_1 p_2} \times r_{p_1 p_2} \right). \tag{4.5}$$

Here, $|\mathcal{I}|$ is constant and thus RBP has the equivalent form

$$\text{RBP: } \min \left\{ \sum_{\mathbf{r} \in \mathcal{B}} c(\mathbf{r}) : \mathcal{B} \subseteq \mathcal{R}, |\mathcal{B}| \leq K, \text{ and } \mathcal{B} \text{ is a blanket} \right\}, \tag{4.6}$$

with

$$c(\mathbf{r}) = \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} r_{p_1 p_2} - 2 \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} I_{p_1 p_2} \times r_{p_1 p_2}. \tag{4.7}$$

As many of the combinatorial optimization problems, the above formulation can be re-expressed using binary variables and a slight change in the notation with $c(\mathbf{r}^j)$ denoting the value (cost) of rectangles $\mathbf{r}^j \in \mathcal{R}$, $j = 1, 2, \ldots, R$, as the constrained set packing problem

$$\text{RBP:} \quad \min \quad \sum_{j=1}^{R} c(\mathbf{r}^j) x_j \tag{4.8}$$

$$\text{s.t.} \quad \sum_{j=1}^{R} x_j \leq K \tag{4.9}$$

$$\sum_{j=1}^{R} r^j_{p_1 p_2} x_j \leq 1 \qquad p_1 = 1, 2, \ldots, W; p_2 = 1, 2, \ldots, H \tag{4.10}$$

$$x_j \in \{0, 1\} \qquad j = 1, 2, \ldots, R. \tag{4.11}$$

Here, the decision variable x_j is set to 1 if rectangle j is selected for the blanket; otherwise it is set to 0. Inequality (4.9) restricts the number of rectangles in an optimal blanket to be less than or equal to a given positive integer K. The given integer upper bound K is actually an implicit parameter for the approximation quality: the larger it is, the better the blanket $\mathcal{B}$ approximates the target image $\mathcal{I}$. Set packing inequalities (4.10) allow each pixel $\mathbf{p}$ to be in at most one of the rectangles. The number of all possible configurations of K rectangles is $\mathcal{O}((W \times H)^{2K})$, which can be very large depending on the size of the target image.

RBP is an NP-complete problem. To prove that, we can reduce weighted set packing problem, which is known to be NP-complete, to RBP by adding the constraint $\sum_j x_j \leq M$ to weighted set packing problem where M is the total number of subsets. Adding this constraint does not change the optimum solution of weighted set packing.

Mohr and Zachmann used a fitness function while assessing the quality of a given rectangle set based on the total number of non-overlapping pixels [9, 79]. They consider RBP within the context of silhouette matching in particular, and propose a dynamic programming approach [79] and a recursive search heuristic [9]. The method described in Chapter 2 uses a similar function in context of silhouette matching for person tracking and fall detection. More general criteria are also suggested in the literature for analytical description of relations between a given target image and the finite set of rectangles that is supposed to approximate it; Φ function applied as a fitness criterion for object packing [102] and Γ function for polygonal region covering [86]. Unfortunately, their evaluation requires more computational effort and they do not serve our purpose better than (4.7) in the BIP formulation (4.8)–(4.11) of RBP.

Since the number of all possible rectangles can be large depending on the size of the image, column generation procedure can be applied to generate new rectangles as long as they improve the objective function [103]. However, column generation is not directly applicable when the variables are integer. So, the whole column generation problem is wrapped into a branch-and-bound scheme, which is also known as branch-and-price [104]. Let $\text{RBP}^{(t)}$ denote the restricted integer programming master problem (i.e. rectangle blanket problem) at step t. Then, the LP relaxation of the restricted

integer programming master at step t (RLPM$^{(t)}$) is the linear programming problem

$$\text{RLPM}^{(t)}: \quad \min \quad \sum_{j=1}^{R^{(t)}} c(\mathbf{r}^j) x_j \tag{4.12}$$

$$\text{s.t.} \quad \sum_{j=1}^{R^{(t)}} x_j \leq K \tag{4.13}$$

$$\sum_{j=1}^{R^{(t)}} r^j_{p_1 p_2} x_j \leq 1 \quad p_1 = 1, 2, \ldots, W; p_2 = 1, 2, \ldots, H \tag{4.14}$$

$$x_j \geq 0 \qquad j = 1, 2, \ldots, R^{(t)}, \tag{4.15}$$

after relaxing binary restrictions on the variables. Here, $R^{(t)} \leq R$ is the number of columns (variables) and $\mu^{(t)}$ and $\{\pi^{(t)}_{p_1 p_2} : p_1 = 1, 2, \ldots, W; p_2 = 1, 2, \ldots, H\}$ are the optimal values of the dual variables μ and $\{\pi_{p_1 p_2} : p_1 = 1, 2, \ldots, W; p_2 = 1, 2, \ldots, H\}$ at step t. Inequalities $x_j \leq 1$ are not included since they are implied by constraints (4.14). Then, the dual of RLPM$^{(t)}$ can be written as

$$\text{DRLPM}^{(t)}: \quad \min \quad \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \pi_{p_1 p_2} + K\mu \tag{4.16}$$

$$\text{s.t.} \quad \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} r^j_{p_1 p_2} \pi_{p_1 p_2} + \mu \geq -c(\mathbf{r}^j) \qquad j = 1, 2, \ldots, R^{(t)} \tag{4.17}$$

$$\mu, \pi_{p_1 p_2} \geq 0 \qquad p_1 = 1, 2, \ldots, W; p_2 = 1, 2, \ldots, H, \tag{4.18}$$

and the reduced cost can be expressed as

$$\overline{c}(\mathbf{r}^j) = c(\mathbf{r}^j) - \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} r^j_{p_1 p_2} \pi_{p_1 p_2} - \mu \qquad j = 1, 2, \ldots, R^{(t)}, \tag{4.19}$$

where the dual variables $\boldsymbol{\pi}$ and μ are restricted to be nonpositive.

We provide a formal pseudocode of the column generation algorithm that solves linear programming relaxation of the integer programming formulation of RBP as Algorithm 4.2. This is the most generic form and does not include improvements and implementation details such as the multiple column generation, lower bounding, and

stabilization. Notice that $\mathbf{x} = \mathbf{0}$ is always a trivial solution and RLPM$^{(t)}$ is always feasible. This requires the setting of the $W \times H + 1$ slack variables to the right hand sides of the inequalities (4.13) and (4.14) as the initial basic feasible solution. Hence, the $(W \times H + 1) \times (W \times H + 1)$ identity matrix, which corresponds to the slack variables can always be selected as the initial basic matrix.

1: **procedure** SolveLinearProgrammingMaster(LPM)
2:(Initialization): Set $t = 0$. Given an initial subset of columns solve RLPM$^{(0)}$. Since $\mathbf{x} = \mathbf{0}$ is always a trivial feasible solution, the $(W \times H + 1) \times (W \times H + 1)$ identity matrix can be selected as the initial basis.
3:(Pricing): Check whether $\boldsymbol{\pi}^{(t)}$ and $\mu^{(t)}$ are dual feasible for LPM. This requires the determination of the minimum reduced cost value $\overline{c}(\mathbf{r}^{(t+1)})$, namely the solution of the pricing subproblem. Here, $\mathbf{r}^{(t+1)}$ is an optimal rectangle (i.e. column) with the minimum reduced cost value. For this purpose, call procedure SolvePricingSubproblem(PSP$^{(t)}$) given as Algorithm 4.4.
4:(Optimality check): If $\overline{c}(\mathbf{r}^{(t+1)}) = 0$, then STOP; $\boldsymbol{\pi}^{(t)}$ and $\mu^{(t)}$ are dual feasible and $\mathbf{x}^{(t)}$ is primal optimal solution $\mathbf{x}_{LPM}$ for LPM with optimal value $z_{LPM} = z_{RLPM^{(t)}} = \sum_{j=1}^{R^{(t)}} c(\mathbf{r}^j) x_j^{(t)}$ and go to Step 6. Otherwise, go to Step 5.
5:(Generating a new column): If $\overline{c}(\mathbf{r}^{(t+1)}) < 0$, then the column corresponding to the optimal solution $\mathbf{r}^{(t+1)}$ of the pricing subproblem has negative reduced cost. Introducing the column

$$\begin{pmatrix} 1 \\ \mathbf{r}^{(t+1)} \end{pmatrix}$$

with variable $x_{R^{(t)}+1}$ and unit cost $c(\mathbf{r}^{(t+1)})$ leads to RLPM$^{(t+1)}$, which can be easily re-optimized. Here, $R^{(t+1)} = R^{(t)} + 1$. Set $t \leftarrow t + 1$, and go to Step 3.
6: **return** $\mathbf{x}_{LPM}$ and z_{LPM}.
7: **end procedure**

Figure 4.2. Column generation algorithm that solves the LP relaxation of RBP.

The solution of the RLPM using column generation yields an optimal solution $\mathbf{x}^* = \mathbf{x}^{(t^*)}$ (at some node of the branch-and-bound tree) at step t^* after solving RLPM$^{(t^*)}$ (i.e. restricted master LP with $R^{(t^*)}$ columns). If it is fractional and it is not possible to prune that node by bound or infeasibility. New branches can be created by partitioning the solution set of the integer programming master RBP, which can be done by adding constraint(s) either to the master problem or the pricing subproblems. In the branch-and-price algorithm, we let $\mathcal{L}$ be a collection of rectangle blanket problems RBP^i of the form $z^i = \min\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in \mathcal{S}^i\}$ where $\mathcal{S}^i$ is a subset of the original feasible solution set $\mathcal{S}$ (i.e. the feasible solution set of the original problem RBP described with constraints (4.9)–(4.11)). Associated with each subproblem of $\mathcal{L}$ is a lower bound

$\underline{z}^i \leq z^i$. However, there are some special difficulties in combining column generation with integer programming techniques such as selecting a branching strategy and dealing with slow convergence.

4.1.1. Pricing subproblem

As a consequence of the reduced cost expression (4.19) the pricing subproblem can be formulated as

$$\overline{c}(\mathbf{r}^{(t+1)}) = \min\{c(\mathbf{r}) - \sum_{p_1=1}^{W}\sum_{p_2=1}^{H} r_{p_1p_2}\pi_{p_1p_2}^{(t)} - \mu^{(t)} : \mathbf{r} \in \mathcal{R}\}, \tag{4.20}$$

which can be restated as

$$\overline{c}(\mathbf{r}^{(t+1)}) + \mu^{(t)} = \min\{\sum_{p_1=1}^{W}\sum_{p_2=1}^{H}(1 - 2I_{p_1p_2} - \pi_{p_1p_2}^{(t)})r_{p_1p_2} : \mathbf{r} \in \mathcal{R}\} \tag{4.21}$$

by using the definition of $c(\mathbf{r})$ given with expression (4.7) and the fact that $\mu^{(t)}$ is constant. Here, $\mathbf{r}^{(t+1)} = \arg\min\{\overline{c}(\mathbf{r}^{(t+1)}) - \mu^{(t)}\}$, namely an optimal solution of the minimization problem given as the right-hand side of expression (4.21). Hence, if $\overline{c}(\mathbf{r}^{(t+1)}) = z(\mathbf{r}^{(t+1)}) + \mu^{(t)} = 0$ where $z(\mathbf{r}^{(t+1)})$ is its optimal value, then the current dual solution is feasible and an optimal solution of the LP relaxation of RBP is reached. Otherwise, if $\overline{c}(\mathbf{r}^{(t+1)}) < 0$, a new column is added with the following entries: $c(\mathbf{r}^{(t+1)})$ for objective function (4.12), 1 for inequality (4.13) and $\{r_{p_1p_2}^{(t+1)} : p_1 = 1, 2, \ldots, W; p_2 = 1, 2, \ldots, H\}$ for inequalities (4.14) corresponding to the $R^{(t+1)} = (R^{(t)} + 1)^{\text{th}}$ variable. Observe that $c(\mathbf{r}^{(t+1)})$ is calculated using expression (4.7) with $\mathbf{r}^{(t+1)}$. As a consequence of this discussion the pricing subproblem becomes

$$\text{PSP}^{(t)}: \; z(\mathbf{r}^{(t+1)}) = \min\left\{\sum_{p_1=1}^{W}\sum_{p_2=1}^{H}(1 - 2I_{p_1p_2} - \pi_{p_1p_2}^{(t)})r_{p_1p_2} : \mathbf{r} \in \mathcal{R}\right\}. \tag{4.22}$$

For the solution of the subproblems created at each node of the RBP's search tree, we adapt a branch-and-bound method that geometrically searches within a rectangle for

a subrectangle with minimum cost (i.e. such that the sum of the weights of the pixels included in the subrectangle is the smallest). This method is originally proposed for determining the subwindow for localizing an object of interest within a given image [82]. This method is summarized in this section for the sake of completeness. The idea is to start with an initial rectangle set containing all possible rectangles and dividing it into two disjoint subsets at each branch. Notice that this geometric algorithm, which is essentially a two-dimensional interval bracketing (bisection) method, eventually computes an optimal solution of the pricing subproblem $\text{PSP}^{(t)}$ at iteration t.

Using the definition given in (4.2) we can represent any rectangle $\mathbf{r}$ with the quadruplet as $(r_{\text{top}}, r_{\text{left}}, r_{\text{bottom}}, r_{\text{right}})$ denoting respectively its top, left, bottom and right edge coordinates. As we have already mentioned, because of the conventions we adopt from image processing and graphic related fields the orientation of y-axis is downward, and $r_{\text{top}} \leq r_{\text{bottom}}$ as a result.

Similarly, we represent a set $\mathcal{R}^k$ of rectangles associated with node k of the search tree as $\mathcal{R}^k = (\mathbf{T}^k, \mathbf{L}^k, \mathbf{B}^k, \mathbf{R}^k)$. Here, $\mathbf{T}^k, \mathbf{L}^k, \mathbf{B}^k$ and $\mathbf{R}^k$ are the range of values that top, left, bottom and right edge's coordinates can take. Each of these ranges have lower and upper bounds. For example $\mathbf{T}^k = [\text{top}^k_{\text{low}}, \text{top}^k_{\text{high}}]$ means that the top edge coordinate of the rectangles of set $\mathcal{R}^k$ are between $\text{top}^k_{\text{low}}$ and $\text{top}^k_{\text{high}}$, inclusive. This representation is illustrated in Figure 4.3 with $\mathbf{r}_\cup$ and $\mathbf{r}_\cap$ denoting respectively the subsets corresponding to the union and intersection of all rectangles in the set.

At the beginning, the initial node (i.e. node 0) of the branch-and-bound tree includes the whole set of rectangles $\mathcal{R}^0 = ([1, H], [1, W], [1, H], [1, W])$. At each step, the largest range is bisected to form two branches such that rectangle subsets form a partition of their parent node's rectangle set. Binary search proceeds according to the best-first branching rule using a lower bound on the sum of the weights of the pixels the rectangles include for pruning, until a node representing a subset consisting of a single rectangle with the minimum cost is reached.

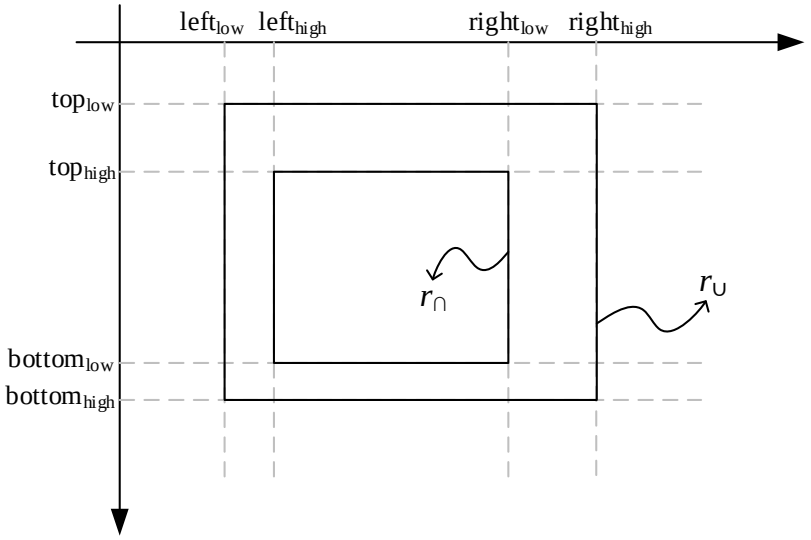


Figure 4.3. Representation of a rectangle set.

The sum of the negative weights inside a rectangle, is a lower bound for the sum of all the weights inside it. We are going to use a similar principle to define a lower bound for a given rectangle set. Let $z(\mathbf{r})$ be the cost associated with a given rectangle $\mathbf{r}$ and $\ell(\mathcal{R})$ be the lower bound on the costs of the rectangles belonging to the rectangle set $\mathcal{R}$. Hence,

$$z(\mathbf{r}) = \sum_{p_1=r_{\text{left}}}^{r_{\text{right}}} \sum_{p_2=r_{\text{bottom}}}^{r_{\text{top}}} (1 - 2I_{p_1p_2} - \pi^{(t)}_{p_1p_2}) r_{p_1p_2}, \tag{4.23}$$

is the objective function for rectangle $\mathbf{r} = (r_{\text{left}}, r_{\text{right}}, r_{\text{top}}, r_{\text{bottom}})$. Then, $\ell(\mathcal{R})$ should satisfy the following properties as a lower bound:

$$\ell(\mathcal{R}) \leq \min_{\mathbf{r}\in\mathcal{R}} z(\mathbf{r}) \tag{4.24}$$

$$\ell(\mathcal{R}) = z(\mathbf{r}) \text{ if set } \mathcal{R} \text{ is the singleton } \{\mathbf{r}\}. \tag{4.25}$$

A particular candidate for a lower bound is:

$$\ell(\mathcal{R}) = \ell^{-}(\mathbf{r}_{\cup}) + \ell^{+}(\mathbf{r}_{\cap}). \tag{4.26}$$

Here, $\ell^{-}(\mathbf{r})$ and $\ell^{+}(\mathbf{r})$ are the sum of the negative and positive weights of the pixels of rectangle $\mathbf{r}$. The inequalities

$$\ell^{+}(\mathbf{r}_{\cap}) \leq \ell^{+}(\mathbf{r}), \tag{4.27}$$

$$\ell^{-}(\mathbf{r}_{\cup}) \leq \ell^{-}(\mathbf{r}) \tag{4.28}$$

hold for any rectangle $\mathbf{r}$ of set $\mathcal{R}$ because of two reasons. First, every rectangle $\mathbf{r} \in \mathcal{R}$ includes $\mathbf{r}_{\cap}$ so the sum of the positive pixels inside $\mathbf{r}$ is at least $l^{+}(\mathbf{r})$. Second, every rectangle $\mathbf{r} \in \mathcal{R}$ is included in $\mathbf{r}_{\cup}$, and following the same reasoning, it can be seen that the second inequality also holds. If we sum the inequalities we can see that $\ell(\mathcal{R}) \leq \min_{\mathbf{r}\in\mathcal{R}} z(\mathbf{r})$.

The rectangle set representation allows $\mathbf{r}_{\cup}$ and $\mathbf{r}_{\cap}$ to be calculated very quickly since,

$$\mathbf{r}_{\cup} = [\text{top}_{\text{low}}, \text{left}_{\text{low}}, \text{bottom}_{\text{high}}, \text{right}_{\text{high}}] \text{ and}$$
$$\mathbf{r}_{\cap} = [\text{top}_{\text{high}}, \text{left}_{\text{high}}, \text{bottom}_{\text{low}}, \text{right}_{\text{low}}].$$

When $\mathbf{r}_{\cap}$ does not define a valid rectangle, then the intersection is empty. Furthermore, $\ell^{-}(\mathbf{r})$ and $\ell^{+}(\mathbf{r})$ can be evaluated in constant time using integral images (summed area tables) of only negative and only positive values [6], which makes the computation of the lower bounds very efficient.

Finally, all the ingredients of a branch-and-bound algorithm that can solve the pricing subproblem is in place. Let $\mathcal{L}$ be a collection of subproblems PSP^i of the form $z(\mathcal{R}^i) = \min\left\{\sum_{p_1=1}^{W}\sum_{p_2=1}^{H} w_{p_1p_2}r_{p_1p_2} : \mathbf{r} \in \mathcal{R}^i\right\}$; where $w_{p_1p_2} = (1 - 2I_{p_1p_2} - \pi^{(t)}_{p_1p_2})$ is the weight of pixel $\mathbf{p}$. Associated with each subproblem PSP^i of $\mathcal{L}$ is a lower bound

$\ell(\mathcal{R}^i) \leq z(\mathcal{R}^i)$ computed according to (4.26). Steps of the branch-and-bound algorithm is listed as Algorithm 4.4.

1: **procedure** SolvePricingSubproblem(PSP)
2:(Initialization): Set $\text{PSP}^0 = \text{PSP}$, $\mathcal{R}^0 = \mathcal{R}$, $\underline{z}^0 = \ell(\mathcal{R}^0)$, $\mathcal{L} = \{\text{PSP}^0\}$, $\mathbf{r}^* \leftarrow (1, W, 1, H)$ and $\overline{z} = \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \max(0, 1 - 2I_{p_1p_2} + \pi^{(t)}_{p_1p_2}) r^*_{p_1p_2}$.
3:(Termination test): If $\mathcal{L} = \emptyset$, then rectangle $\mathbf{r}^*$ that yields $\overline{z}$ is an optimal solution of the pricing subproblem.
4:(Problem solution and lower bound computation): Select and delete a problem from $\mathcal{L}$, say PSP^i.
5:(Pruning):
i. If $\underline{z}^i \geq \overline{z}$, then go to Step 3.
ii. If rectangle set $\mathcal{R}^i$ does not consist of a single rectangle, then go to Step 6.
iii. If rectangle set $\mathcal{R}^i$ consists of a single rectangle, say $\mathbf{r}^i$, and $z(\mathcal{R}^i) < \overline{z}$, then set $\overline{z} = z(\mathcal{R}^i)$ and $\mathbf{r}^* \leftarrow \mathbf{r}^i$, go to Step 3. Otherwise go to Step 6.
6:(Division): Let $\{\mathcal{R}^{ij}\}_{j=1}^2$ be a bi-partition of $\mathcal{R}^i$ and $\mathcal{R}^{i1} \cap \mathcal{R}^{i2} = \emptyset$, which is obtained by bisecting the largest range of rectangle $\mathbf{r}^i$ of set $\mathcal{R}^i$. Add problems $\{\text{PSP}^{ij}\}_{j=1}^2$ to $\mathcal{L}$ with lower bounds $\underline{z}^{ij} = \ell(\mathcal{R}^{ij})$ $j = 1, 2$; go to Step 3.
7**return** $\mathbf{r}^*$ and $\overline{z}$.
8: **end procedure**

Figure 4.4. Branch-and-bound algorithm that solves PSP for computing $\mathbf{r}^*$.

4.1.2. Branching

First difficulty one can face in combining column generation with integer programming techniques is related to the branching rule. A scheme suitable for column generation must be devised at some node, say node h, of the branch-and-bound tree.

First, in section 4.1.2.1, we describe conventional branching on variables of the master problem. Then in section 4.1.2.2 we propose a novel branching rule. Throughout this section, index h is omitted in the derivations to increase the readability. For example, we use $\mathcal{S}$, $\mathcal{R}$, $\mathcal{S}^i$, $\mathcal{R}^i$, $\text{RLPM}_i^{(t)}$ and $\text{PSP}_i^{(t)}$ instead of $\mathcal{S}^h$, $\mathcal{R}^h$, $\mathcal{S}^{hi}$, $\mathcal{R}^{hi}$, $\text{RLPM}_{hi}^{(t)}$ and $\text{PSP}_{hi}^{(t)}$.

4.1.2.1. Branching on the master problem's variables. A conventional scheme is to consider one of the fractional entries of $\mathbf{x}^*$, say x_j^*, and set $x_j = 0$ in one branch and $x_j = 1$ on the other. These are usually introduced as new constraints to the master problem. Unfortunately, this is not a good choice since it yields an unbalanced branch-and-bound tree and the optimum is usually reached after many branchings as explained in [105].

We have observed that the use of column generation in the solution of the linear programming master LPM produces integral or close to integral optimal solutions in particular for RBP, which is an important advantage and results in very few branchings. Hence, one can prefer branching on the master problem's variables, since it is simple to implement and define a partition of the feasible solution set $\mathcal{S}$ as $\mathcal{S}^i = \{\mathcal{S} \cap \{\mathbf{x} : x_q = i\}\}$ for $i = 0, 1$ given that $x_q^* \notin \mathbb{Z}$ is the fractional entry of $\mathbf{x}^*$ and x_q is selected as the branching variable.

4.1.2.2. Branching on the pricing subproblem variables. This type of branching is preferred to the previous one since it is more likely to produce a balanced search tree as pointed by [106]. A well-known branching rule for the set partitioning problems is due to [107]. It is possible to adopt it for the solution of the set packing problem after adding slack variables and transforming packing inequalities to partitioning equalities. However, this causes a considerable increase in the number of variables. Instead, we propose a new branching scheme for RBP without considering the cardinality inequality (4.9). It is a consequence of the next proposition and suitable for the solution of RBP by branch-and-price.

Proposition 4.1. *Let* $\mathbf{x}$ *be the fractional optimal solution to the LPM, i.e.* $0 < x_k < 1$ *for some* $k = 1, 2, \ldots, R$. *Then, there exists two pixels* $\begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$ *and* $\begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$ *and rectangles* $j \neq k$ *such that* $r^k_{e_1e_2} \neq r^j_{e_1e_2}$, $r^k_{f_1f_2} = r^j_{f_1f_2} = 1$, $0 < x_j \leq 1$.

Proof. Assume that $r^j_{f_1f_2} = 0$ for all $f = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$ such that $r^k_{f_1f_2} = 1$, $x_j > 0$ and $j \neq k$. Then, we can improve the objective value by setting $x_k = 1$ since doing so does not violate any constraints. This contradicts $\mathbf{x}$ is an optimal solution. Hence, there must exist a rectangle j and pixel $f = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$ with $r^k_{f_1f_2} = r^j_{f_1f_2} = 1$. Then, since there is no duplicated column in the basis, there must exist one pixel $e = \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$ such that $r^k_{e_1e_2} \neq r^j_{e_1e_2}$. □

After identifying rows e and f we can impose the branching constraints

$$\sum_{k:r^k_{e_1e_2}=r^k_{f_1f_2}=1} x_k \leq 0 \quad \text{and} \quad \sum_{k:r^k_{e_1e_2}=r^k_{f_1f_2}=1} x_k \geq 1 \tag{4.29}$$

for respectively left and right branches. The pixels e and f have to be covered by different rectangles on the left branch and by the same rectangle on the right branch.

As a consequence of this proposition, if it is not possible to identify any (e, f) pairs, then the solution of the master problem must be integer. The algorithm terminates in a finite number of branchings since there are only a finite number of row pairs. Besides, a large number of variables are eliminated at each branch. Notice that this rule eliminates the submatrix

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \tag{4.30}$$

which is precisely the excluded submatrix in the characterization of the totally balanced matrices [108]. Total balancedness of the coefficient matrix is a sufficient condition for the LP relaxation of a set packing problem to have integer optimal solutions.

RBP includes a cardinality constraints for the number of sets to be packed in addition to the ordinary set packing constraints. The new branching rule still remains valid for its solution by branch-and-price.

4.1.2.3. Implementing the branching rules. Notice that, new constraints added to the pricing subproblem should be compatible with the geometric solution procedure presented previously as Algorithm 4.4. Branching schemes that can be implemented by modifying the objective function (4.22) are appropriate.

However, as can be observed, any branching scheme that can be handled by only modifying the objective function (4.8) is appropriate. We restrict a pixel coverage

by making its weight arbitrarily large (i.e. $w_{u_1u_2} = +\infty$) and arbitrarily small (i.e. $w_{u_1u_2} = -\infty$), respectively for $r_{u_1u_2} = 0$ and $r_{u_1u_2} = 1$.

This is what we do for $x_j = 1$ branch of the first branching rule: the weights of the pixels belonging to rectangle j are set to very large numbers in the image of the pricing subproblem so that no rectangle overlapping with rectangle j is generated. The right hand side of the constraints of the master problem are modified accordingly. However, for $x_j = 0$ branch the list of prohibited rectangles belonging to the ancestors is passed to the child. So a rectangle generated by the pricing subproblem is discarded if it belongs to this list and the second best rectangle is considered. This procedure continues until the best column (i.e. the column with the smallest reduced cost) that is not included in the list is generated.

Another approach is to find the geometric representation of the new restrictions at each branch. First, columns violating the branching constraints are removed from the master problem. When the second branching rule is used, at a left branch where two pixels are forced to reside in different rectangles, we prune the node by deleting the associated rectangle set if their intersection contains these two pixels as illustrated in Figure 4.5a. In case a right branch is visited we prune the node if the union of rectangles contains only one of these two pixels, since both of the pixels must reside in the same rectangle. This is illustrated in Figure 4.5d and Figure 4.5e. Remaining alignments, which are illustrated with Figure 4.5b, Figure 4.5c and Figure 4.5f are valid for both branches.

4.1.3. Improvements for the tailing-off effect

The next difficulty is the efficiency of the branch-and-bound algorithm. It can decrease remarkably because of the tailing-off effect of the column generation algorithm, which can be prevented by means of effective lower bounds on the optimal value of RLPM and dual smoothing.

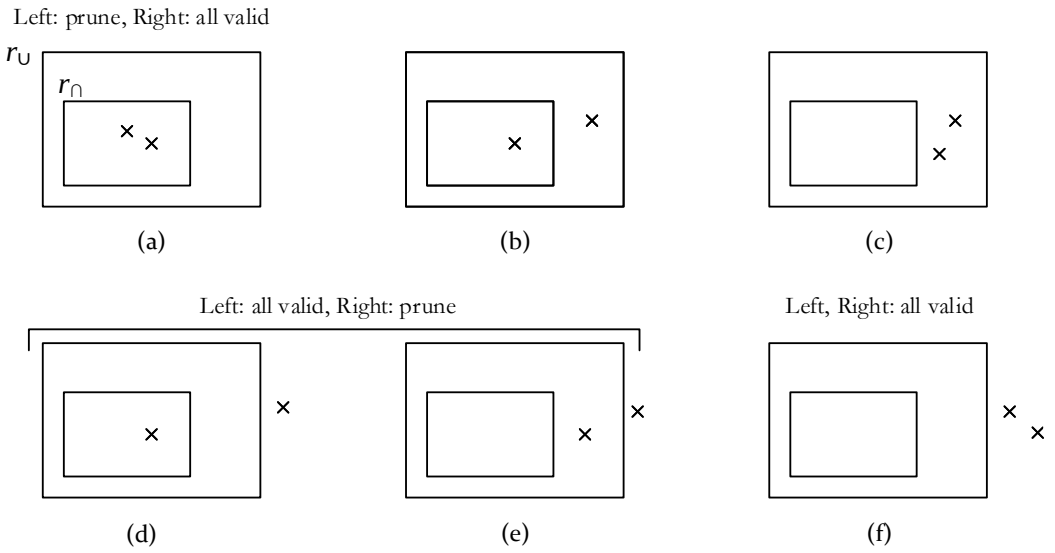


Figure 4.5. Possible alignments of two pixels with respect to the rectangle set during pricing.

4.1.3.1. Lower bound for early stopping column generation. The Lagrange function of the restricted master problem at step t, i.e. RLPM$^{(t)}$ can be written as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\pi}, \mu) = \sum_{j=1}^{R^{(t)}} c(\mathbf{r}^j) x_j + \mu(\sum_{j=1}^{R^{(t)}} x_j - K) + \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \pi_{p_1 p_2} (\sum_{j=1}^{R^{(t)}} r_{p_1 p_2}^j x_j - 1) \tag{4.31}$$

for nonnegative Lagrange multipliers $\mu \geq 0$ and $\boldsymbol{\pi} \geq 0$. It is clearly equivalent to

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \boldsymbol{\pi}, \mu) &= \sum_{j=1}^{R^{(t)}} \left(c(\mathbf{r}^j) + \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \pi_{p_1 p_2} \sum_{j=1}^{R^{(t)}} r_{p_1 p_2}^j + \mu \right) x_j - \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \pi_{p_1 p_2} - \mu K \qquad (4.32) \\ &= \sum_{j=1}^{R^{(t)}} \overline{c}(\mathbf{r}^j) x_j - \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \pi_{p_1 p_2} - \mu K. \qquad (4.33) \end{aligned}$$

Then, for the dual optimal solution $\boldsymbol{\pi}^{(t)}$ and $\mu^{(t)}$ of RLPM$^{(t)}$

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \boldsymbol{\pi}^{(t)}, \mu^{(t)}) &= \sum_{j=1}^{R^{(t)}} \overline{c}(\mathbf{r}^j) x_j - \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \pi_{p_1 p_2}^{(t)} - \mu^{(t)} K \qquad (4.34) \\ &\geq \overline{c}(\mathbf{r}^{(t+1)}) \sum_{j=1}^{R^{(t)}} x_j - \sum_{p_1=1}^{W} \sum_{p_2=1}^{H} \pi_{p_1 p_2}^{(t)} - \mu^{(t)} K. \qquad (4.35) \end{aligned}$$

The last inequality follows from the fact that $\overline{c}(\mathbf{r}^{(t+1)}) = \min_j \{\overline{c}(\mathbf{r}^j)\}$. Moreover, since $\sum_{j=1}^{R^{(t)}} x_j \leq K$ and $\overline{c}(\mathbf{r}^{(t+1)}) \leq 0$ we can write

$$z_{LPM} \geq \min_{\mathbf{x}\geq 0} \mathcal{L}(\mathbf{x}, \boldsymbol{\pi}^{(t)}, \mu^{(t)}) \geq K\overline{c}(\mathbf{r}^{(t+1)}) - \sum_{p_1=1}^{W}\sum_{p_2=1}^{H} \pi_{p_1p_2}^{(t)} - \mu^{(t)}K = K\overline{c}(\mathbf{r}^{(t+1)}) + z_{RLPM^{(t)}}. \tag{4.36}$$

The last inequality is obtained using the LP duality and the fact that $z_{DRLPM^{(t)}} = -\sum_{p_1=1}^{W}\sum_{p_2=1}^{H} \pi_{p_1p_2}^{(t)} - \mu^{(t)}K$. Besides, $z_{RLPM^{(t)}} \geq z_{LPM}$. This discussion can be summarized with the following proposition.

Proposition 4.2. *Let* $LB(\boldsymbol{\pi}^{(t)}, \mu^{(t)}) = K\overline{c}(\mathbf{r}^{(t+1)}) + z_{RLPM^{(t)}}$. *Then,*

$$z_{RLPM^{(t)}} \geq z_{LPM} \geq LB(\boldsymbol{\pi}^{(t)}, \mu^{(t)}).$$

As a direct consequence of this proposition, column generation can be stopped at step t if $\lceil LB \rceil \geq z_{RLPM^{(t)}}$ or $\lceil LB \rceil \geq \overline{z}$ where LB is the maximum of the previously computed lower bounds and $\overline{z}$ is the current integer upper bound (i.e. objective value of the incumbent). The first case implies that the restricted master is solved to optimality and the second one implies that the current node can be pruned. For the first case, we use the fact that the optimal value of RBP is integer (i.e. it is the sum of the product of a binary variable with integer coefficients).

Because of the simplified notation we use (i.e. no explicit index denoting the node of the search tree) the derivation given above may seem to be valid only for the root node. However, it is possible to derive similar stopping conditions for any node of the branch-and-bound tree following a similar path.

4.1.3.2. Dual smoothing. One of the reasons behind the tailing-off effect is the oscillation of the dual variables erratically during the iterations of the column generation. One remedy is to stabilize the dual variables using the convex combination of the current

and incumbent dual values as [109] and [110] suggest:

$$\tilde{\boldsymbol{\pi}}^{(t)} = \alpha \boldsymbol{\pi}^{(t)}_{\text{best}} + (1-\alpha)\boldsymbol{\pi}^{(t)}. \tag{4.37}$$

Here, $\boldsymbol{\pi}^{(t)}_{\text{best}}$ are the dual variable values that give the largest of the lower bounds (i.e. incumbent dual) calculated up to iteration t. Then, the pricing subproblem is solved with the modified duals $\tilde{\boldsymbol{\pi}}^{(t)}$, instead of the current duals $\boldsymbol{\pi}^{(t)}$. If the reduced cost of the generated column is nonnegative, it is a misprice and the column is not added to RLPM$^{(t)}$. Since we are not working on a Lagrangian relaxation of RBP, there is no available subgradient to benefit from the automatic parameter tuning described in [110]. Nevertheless, we have experimentally noticed that setting $\alpha = 0.8$ increases the efficiency considerably.

4.2. Heuristics

Solving RBP exactly can be computationally demanding, especially for instances of realistic size. Hence, efficient heuristics are very valuable. In this section, we introduce three of them. The first two are simple heuristics that enlarge the blanket, one rectangle at a time. One of these incremental method is based on the previous work by [9]. The third one is a novel application of the constrained simulated annealing. For the sake of completeness we describe each of them in this section.

4.2.1. Split & Fit

Split and Fit heuristic (SF) iteratively splits rectangles and decreases the objective function using their *fitness scores*. The fitness score of rectangle $\mathbf{r}$ is defined to be inversely proportional to the area of the non-overlapping region of the rectangle with the target image and can be formulated as

$$g(\mathbf{r}) = \frac{1}{\sum_{p_1=r_{left}}^{r_{right}} \sum_{p_2=r_{top}}^{r_{bottom}} (1 - I_{p_1 p_2})}. \tag{4.38}$$

It can be computed in constant time using the integral image of the given image. The blanket determined by SF is represented by the rectangle set $\mathcal{R}^*$, which is initialized with the axis-aligned initial rectangle $(1, W, 1, H)$ for a given object image. Another structure $\mathcal{Q}$ is maintained for keeping all possible split pairs along with their fitness scores. At each step, a rectangle $\mathbf{r}$ with the lowest fitness score and its complement rectangle $\mathbf{r}'$ (i.e. the rectangle with the higher fitness score) are removed from $\mathcal{Q}$. Its parent, namely the rectangle that is split to obtain $\mathbf{r}$ is removed from $\mathcal{R}^*$ and all the siblings are removed from $\mathcal{Q}$. Finally, $\mathbf{r}$ and $\mathbf{r}'$ are added to $\mathcal{R}^*$, and their possible split pairs are added to $\mathcal{Q}$. This splitting process is repeated until K rectangles are obtained. In the original work, [111] used a tolerance value to enable early stopping. Unfortunately, although early stopping increases the efficiency, the solutions become less accurate. Therefore, we have run SF without early stopping (i.e. with zero tolerance) in our experiments.

Possible split pairs of a rectangle are generated by dissecting it vertically and horizontally in different proportions. This process is controlled with parameter ρ. Formally, possible split ratios of a rectangle pair are $(\frac{i}{\rho}, \frac{\rho-i}{\rho})$ for $i = 1, 2, \ldots, \rho - 1$. For example, for $\rho = 4$ possible vertical splits of a rectangle with width W have widths $\{(\frac{W}{4}, \frac{3W}{4}), (\frac{W}{2}, \frac{W}{2}), (\frac{3W}{4}, \frac{W}{4})\}$. Note that $\rho = 2$ corresponds to splitting the rectangles exactly in half as it is in the original due to [111]. We have experimentally observed that setting $\rho = 3$ performs better in our experiments.

This procedure yields a set of rectangles covering the target image, i.e. the pixels are fully contained in the rectangles. To improve the solution further, as a final step, each rectangle is shrunk until $c(\mathbf{r})$ increases. Growing / shrinking operation moves one of the edges of a rectangle by one pixel outside/ inside, as shown in Figure 4.7. The steps of the heuristic are illustrated in Figure 4.6 for $\rho = 2$.

4.2.2. Fast Adaptive Silhouette Area Based Template Matching

Fast Adaptive Silhouette Area Based Template matching (FAST) heuristic is proposed by [9]. It eventually determines an approximation of a given image by means

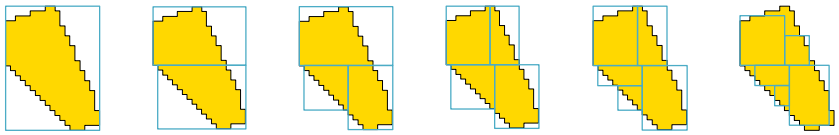


Figure 4.6. Illustration for Split and Fit for $\rho = 2$. The last image shows the output of the shrinking step.

of rectangles, which makes it a potential solution method for the solution of RBP. Unfortunately, the output set of rectangles do not exactly form a blanket since they are allowed to overlap, which means FAST can produce infeasible solutions for RBP in its original form. Based on our experiments, we can say that the size of the overlaps is usually very small and does not have a serious impact on the application performance of the heuristic. Nevertheless, we have modified FAST to guarantee the feasibility of its solutions, i.e. sets of non-overlapping rectangles, at the end.

Like SF, FAST finds rectangles iteratively, considering a *benefit function*, which is similar to the fitness score (4.38),

$$f(\mathbf{r}) = \sum_{p_1=r_{left}}^{r_{right}} \sum_{p_2=r_{bottom}}^{r_{top}} (I_{p_1 p_2} - \tau). \tag{4.39}$$

Here, $\tau \in [0, 1]$ is a parameter that controls total area that is not covered by penalizing covering a zero valued pixel. The method consists of two main steps. First, a rectangle of size $W/2 \times H/2$ that maximizes $f(\mathbf{r})$ is found. W and H denote respectively the width and the height of the image, again. If this rectangle has uncovered pixels, then the size is halved. This procedure is repeated until a rectangle that completely lies inside the binary image is found. Second, the rectangle is enlarged until $f(\mathbf{r})$ starts to decrease. The rectangle area is erased from the target image. These two steps are repeated until K rectangles are obtained.

4.2.3. Constrained Simulated Annealing

Constrained Simulated Annealing CSA [112] is originally proposed for solving a nonlinear programming problem consisting of the minimization of $f(\mathbf{x})$ subject to the

equality constraints $h_i(\mathbf{x}) = 0$, $i = 1, 2, \ldots, m$. CSA considers the Lagrange function

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x}). \tag{4.40}$$

obtained by moving the constraints into the objective function with multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m$.

The goal is to find $\mathbf{x}^*$ that minimizes $f(\mathbf{x})$ subject to the equality constraints $h_i(\mathbf{x}) = 0$, $i = 1, 2, \ldots, m$ by finding $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ that minimizes (4.40). Here, $\boldsymbol{\lambda}^*$ is the vector of Lagrange multipliers at which optimum solution $\mathbf{x}^*$ is obtained. In other words an equality constrained nonlinear minimization problem is solved in $\mathbf{x}$ by solving an unconstrained minimization problem in $\mathbf{y} = (\mathbf{x}, \boldsymbol{\lambda})$, which is one of the classical research problems of nonlinear optimization.

In this outline, $\mathcal{N}(\mathbf{y})$, $G(\mathbf{y}' \mid \mathbf{y})$ and $A(\mathbf{y}, \mathbf{y}', T)$ denote respectively the neighborhood of $\mathbf{y}$, generation probability of $\mathbf{y}'$ given $\mathbf{y}$ and the acceptance probability of the new point $\mathbf{y}'$. Then the new point $\mathbf{y}'$ can be obtained by changing $\mathbf{x}$ to $\mathbf{x}'$, i.e. at $\mathbf{y}' = (\mathbf{x}', \boldsymbol{\lambda})$, or by changing $\boldsymbol{\lambda}$ to $\boldsymbol{\lambda}'$, i.e. at $\mathbf{y}' = (\mathbf{x}, \boldsymbol{\lambda}')$. The algorithm is very similar to the conventional simulated annealing procedure, e.g. convergence condition can be extended to take the unchanged $\mathbf{y}$ into account in successive iterations. At each iteration, a random point is generated by fixing $\mathbf{x}$ or $\boldsymbol{\lambda}$. The neighborhood $\mathcal{N}(\mathbf{y})$ and the acceptance probability $A(\mathbf{y}, \mathbf{y}', T)$ for $\mathbf{y} = (\mathbf{x}, \boldsymbol{\lambda})$ are defined as

$$\mathcal{N}(\mathbf{y}) = \{(\mathbf{x}', \boldsymbol{\lambda}) : \mathbf{x}' \in \mathcal{N}_1(\mathbf{x})\} \cup \{(\mathbf{x}, \boldsymbol{\lambda}') : \boldsymbol{\lambda}' \in \mathcal{N}_2(\boldsymbol{\lambda})\}, \tag{4.41}$$

and

$$A(\mathbf{y}, \mathbf{y}', T) = \begin{cases} \exp\left(-\frac{L(\mathbf{y}') - L(\mathbf{y})}{T}\right) & \text{if } \mathbf{y}' = (\mathbf{x}', \boldsymbol{\lambda}) \\ \\ \exp\left(-\frac{L(\mathbf{y}) - L(\mathbf{y}')}{T}\right) & \text{if } \mathbf{y}' = (\mathbf{x}, \boldsymbol{\lambda}'). \end{cases} \tag{4.42}$$

CSA can be applied to the solution of RBP. The objective function defined as (4.8) is $f(\mathbf{x})$ in the Lagrange function (4.40). There are two constraints in RBP: the maximum number of rectangles is fixed and the solution should not contain overlapping rectangles. These constraints can be incorporated into the Lagrange function as

$$\begin{aligned} h_1(\mathcal{R}) &= \max(0, |\mathcal{R}| - K), \\ h_2(\mathbf{r}) &= \sum_{\substack{\mathbf{r}^i, \mathbf{r}^j \in \mathcal{R} \\ \mathbf{r}^i \neq \mathbf{r}^j}} \sum_{p_1} \sum_{p_2} r^i_{p_1 p_2} \times r^j_{p_1 p_2}, \end{aligned}$$

with multipliers λ_1 and λ_2. $h_1(\mathcal{R})$ is for penalizing the excess in the number of rectangles; it increases as the number of rectangles in the blanket exceeds the upper bound K. $h_2(\mathbf{r})$ is for penalizing overlapping rectangles; its value is obtained by counting pixels in overlapping rectangle pairs.

In this work, five primitive operations are realized using $G(\mathbf{y}' \mid \mathbf{y})$: *grow*, *shrink*, *split*, *delete* and *create*. Growing/shrinking operation moves one of the edges of a rectangle $\mathbf{r} \in \mathcal{R}$ by one pixel outside/inside. Splitting operation partitions a rectangle $\mathbf{r} \in \mathcal{R}$ into two rectangles $\mathbf{r}^1$ and $\mathbf{r}^2$ ($\mathbf{r} = \mathbf{r}^1 \cup \mathbf{r}^2$ and $\mathbf{r}^1 \cap \mathbf{r}^2 = \emptyset$). Deleting simply removes a rectangle $\mathbf{r}$ from $\mathcal{R}$. Creating adds a rectangle $\mathbf{r}$ to $\mathcal{R}$ that resides in the image region (i.e. $1 \leq r_{\text{left}} < r_{\text{right}} \leq W$, $1 \leq r_{\text{top}} < r_{\text{bottom}} \leq H$). These operations are illustrated in Figure 4.7.

4.3. Experiments

In this section we report the results obtained in the computational tests made for assessing the performance of the solution methods.

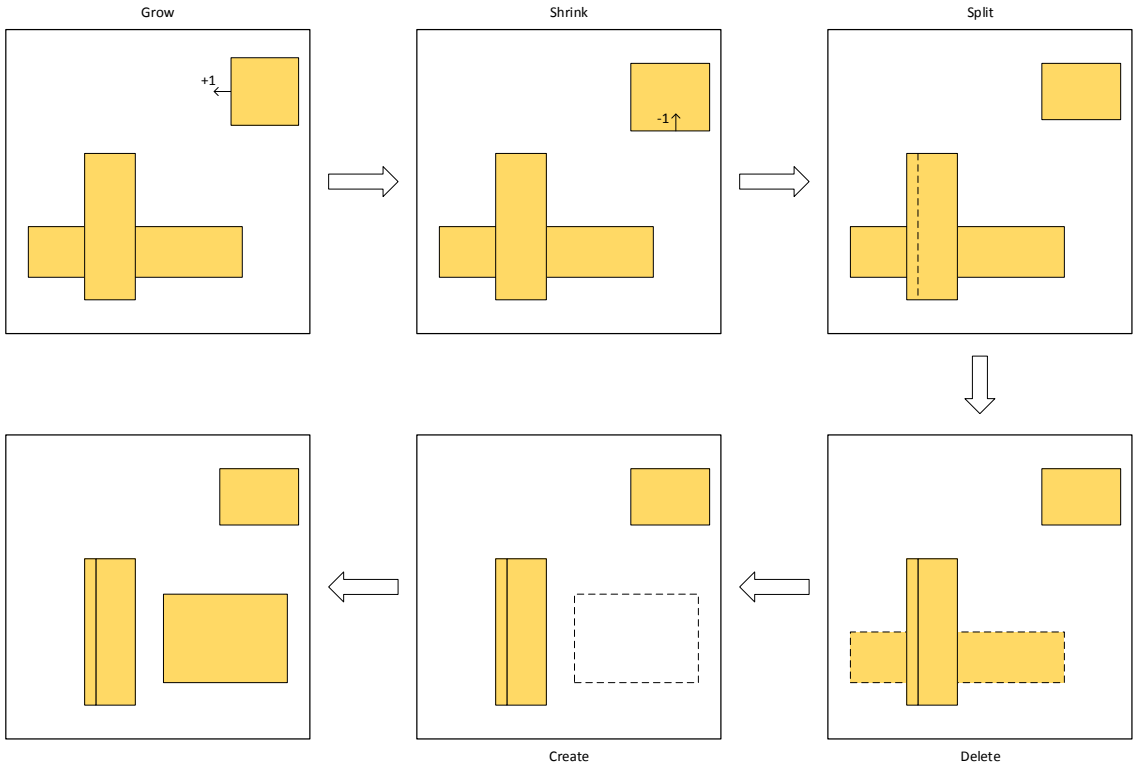


Figure 4.7. Illustration for grow, shrink, split, delete and create operations.

4.3.1. Test bed

For benchmarking, five different groups of binary images are generated; they all have different properties. The benchmark instances can be seen in Figure 4.8–Figure 4.12. The labels and resolutions (i.e. the number of pixels in each dimension) are given below the images. The first group is *ideal human silhouette*s selected from the work of [111] on fall detection and tracking (Figure 4.8). They are simple nonconvex polygons and the size of the images are small compared to the other groups. Since the structure of all shapes used in the study are very similar, only four representative shapes are selected. The second group of benchmark images (Figure 4.9) are taken from [91] where the authors created a similar benchmark for *mask fracturing*, a process where complex shapes are translated into the union of simpler shapes called *shots* during integrated circuit layout production. This group has the largest images and each image represents a single region. The third group (Figure 4.10) contains images generated artificially to capture certain shape properties that the first two groups do not have.

This group contains convex regions, disconnected regions, regions with holes and nested disconnected regions. The fourth group of images (Figure 4.11) is a subset of MPEG-7 shape dataset [113], which are cropped and resized for our experiments. Each category in this group has 20 instances. Because the purpose of the MPEG-7 shape dataset is to evaluate shape similarity measures, we have found that using only a subset of the MPEG-7 dataset is sufficient for our study. The selected categories also capture different shape properties like the previous group. Finally, the last group (Figure 4.12) consists of images belonging to 10 realistic nesting problems. They were obtained from leather garment and furniture with defects on the master surfaces and have been used to generate the results given in Table 4 of [114]. We rasterized the images included in the files in DXF format that were sent by [115], using KABEJA library written in JAVA [116]. The curated benchmark instances can be found under [117][4] . In short, the test bed consists of 147 instances and each one is solved for five distinct blanket sizes (i.e. the number of rectangles in the blanket); $K \in \{3, 5, 10, 15, 20\}$. This results in $147 \times 5 = 735$ test runs for each solution method.

4.3.2. Implementation details

In the actual implementation of the heuristics, rectangles are presented as quadruplets, (x, y, w, h) where x and y represent the top left coordinates, and w and h are the width and height of the rectangle. However, in BP, rectangles are presented as matrices (or vectors) with entries set according to expression (4.2).

avatar1 18×15 | avatar2 19×25 | avatar3 28×17 | avatar4 21×19

Figure 4.8. Selected binary masks of *ideal human silhouettes* from [111].

[4]Use this qr code to easily access the benchmark images:

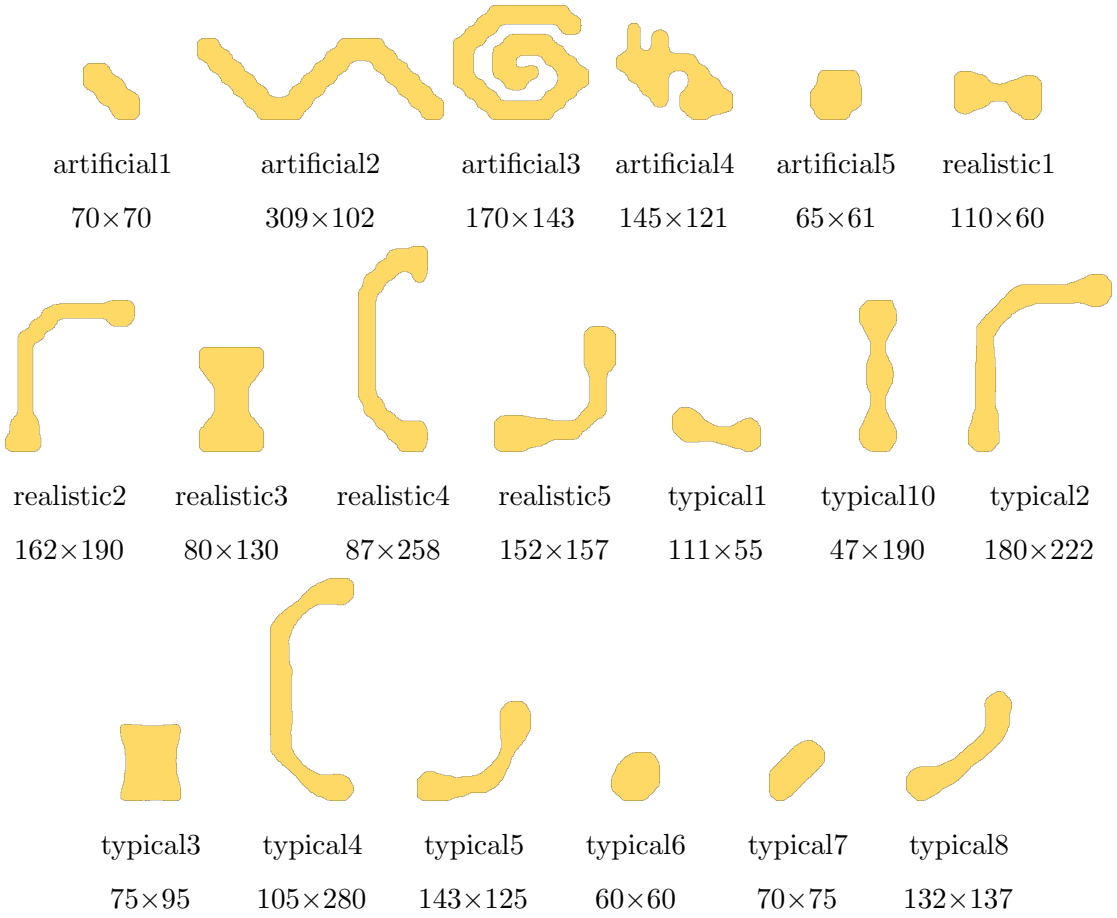


Figure 4.9. Binary masks used for *mask fracturing* in [91].

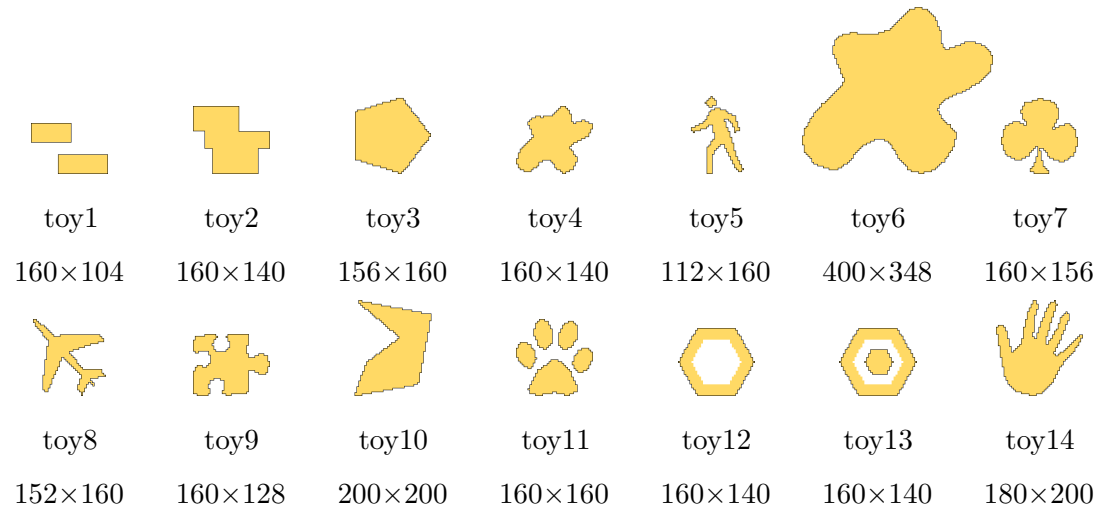


Figure 4.10. Generated binary masks.

Figure 4.11. Subset of MPEG7 shape dataset categories.

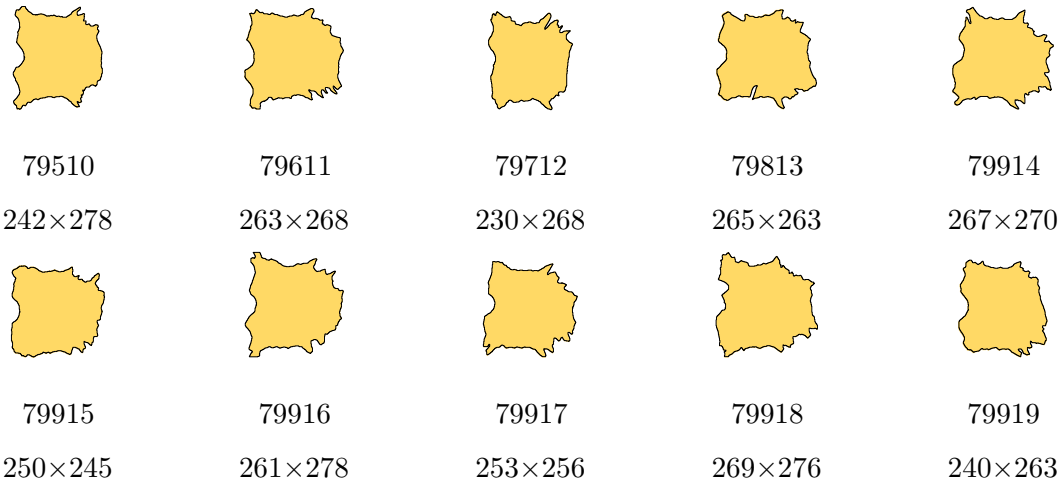


Figure 4.12. Leather master surfaces of industrial nesting problems.

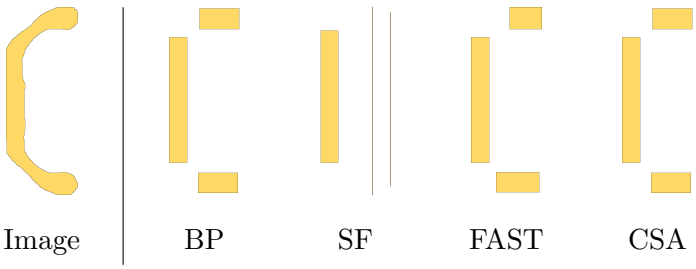


Figure 4.13. Sample nonconvex image, *typical4*, where SF performs poorly for $K = 3$.

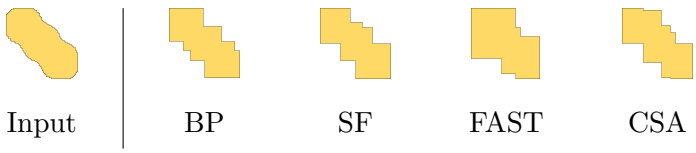


Figure 4.14. Sample image, *artificial1*, where FAST performs poorly for $K = 5$.

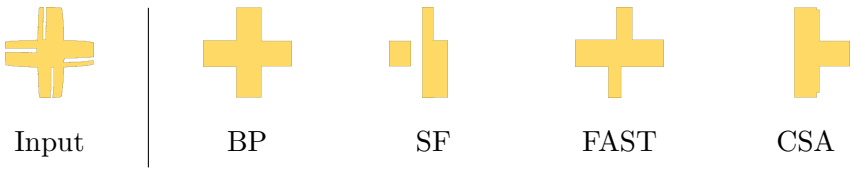


Figure 4.15. Sample image, *device5-9*, where BP finds the optimum for $K = 3$.

We have noticed that starting column configurations do not affect the number of generated columns and convergence behavior significantly in BP. Our column generation process shows rapid progress in the early iterations, especially with the dual smoothing scheme we have implemented.

We have considered various operations to speed up the optimization procedure. Whenever RLPM is modified via column generation or branching, instead of solving it from scratch we start the optimization from the basis obtained in the previous step (i.e. warm start). Multiple columns (up to 10) are generated during pricing in order to speed up the computations. We also keep track of how long each column has been in RLPM. A column that does not enter into the basis for a certain amount of time, is discarded. If a previously discarded column is regenerated, we double the lifespan of that column when we add it back to RLPM. This technique limits the amount of columns present in RLPM and speeds up the optimization dramatically.

In the pricing problem, we have used priority queue data structure, as usually done, to get the next promising rectangle set. However, rectangle sets that contain few rectangles dominate the overall cost of the algorithm. This is because of the overhead associated with the priority queue and the data structure that holds the rectangle set. When the number of rectangles are sufficiently small, we switch to a naive algorithm that iterates through all rectangles in the set. Iterated rectangles are added to another priority queue with given capacity that determines the number of columns to return from the pricing problem. We have experimentally found that the best threshold is 256 for switching to the naive algorithm for our hardware. To make this hybrid algorithm work with the branching constraints, we also check if all the rectangles in the rectangle set satisfy the constraints before switching to the naive algorithm. This can be performed quickly by checking the pixels involved in the constraints against the rectangle set (see Figure 4.5 for valid configurations).

Experiments are carried out on a computer with Core i7 3.07 GHz CPU and 8 GB RAM. BP and CSA are implemented using Java; SF and FAST are implemented in C++. Three software libraries are used: OpenCV [118] for operations on images, [116] for

rasterization of the images given in DXF format and Gurobi [119] for BP. The running time of all tests are limited to 1 hour, except for the runs related to the performance of the branching rules (i.e. Table 1), for which the run time limit is 1.5 hours.

4.3.3. Observations

The computational results can be grouped in two major categories. The first one is for inspecting the effect of branching rules on the efficiency of BP. The second one is for analyzing the performance of the solution methods with respect to their efficiency and accuracy.

4.3.3.1. Branching rules. We should point that in 652 out of the 685 runs of BP on the first four data sets, the initial LP relaxation gives the integer optimal solution. So we only have 33 cases which we can use for the comparison of the branching rules. The results are reported in Table 4.1, where the first two columns list data sets and the corresponding blanket size. In the next two columns, we report the number of nodes visited for each rule. RULE 1 and RULE 2 denote branching explicitly or implicitly in the master, respectively. The numbers in the last two columns are simply the percent relative deviations for the LP relaxation lower bound from the best feasible solution computed in 1.5 hours CPU time limit using BP. As it can be observed, they are quite small, indicating the tightness of the LP relaxation. Besides, "none" means 0.00 % deviation. This occurs when the LP relaxation have also an integer alternative optimal solution. It is detected by BP later without changing the optimum objective value. Observe that there is a single column for reporting the deviations in the table, because in all instances, both methods find a solution with the same objective value even if they reach the time limit and are forced to stop. Besides, when the methods finish running within the time limit, they both find an optimal solution which makes the reported objective values equal.

We have measured the number of nodes visited in branch-and-price using both rules. RULE 2 visited fewer nodes for 15 instances and more nodes for 11 instances.

We can claim that, although there is no clear winner, RULE 2 is more efficient than RULE 1 based on the averages reported on the last row of the table. This is expected since RULE 2 is more likely to produce a balanced search tree as mentioned earlier.

Table 4.1. Efficiency of the branching rules and the improvement on the initial LP relaxation.

		num. of visited nodes		**CPU time (secs)**		
name	**K**	**RULE1**	**RULE2**	**RULE1**	**RULE2**	**Rel. dev. (%)**
dog-12	3	73	115	16	12	1.54
dog-13	3	79	55	18	8	1.54
key-16	3	3	9	210	310	none
key-18	3	3	3	4125	4398	none
toy13	3	3	351	1	17	none
toy4	3	89	43	36	20	0.75
avatar4	5	7	9	1	2	none
device5-13	5	27	27	853	1416	0.28
device5-8	5	18	41	5411	5402	0.27
key-17	5	51	47	2736	3666	0.09
toy12	5	5	5	2	3	none
toy14	5	25	29	31	60	0.25
bat-2	10	48	37	5405	5402	0.05
bat-6	10	179	61	1920	1524	0.13
device5-18	10	7	3	1289	952	0.02
dog-12	10	37	5	20	6	0.24
dog-13	10	7	17	8	12	0.24
toy11	10	3	5	7	11	0.15
typical4	10	6	6	5407	5421	none
dog-6	15	5	7	46	62	none
toy1	15	3	23	4	19	none
toy10	15	33	13	257	175	0.09
toy11	15	7	13	14	13	0.14
toy14	15	9	3	95	68	none
avatar4	20	9	3	2	1	none
dog-12	20	17	17	17	20	0.22
dog-13	20	15	9	12	14	0.22
dog-16	20	7	3	7	6	none
dog-17	20	61	3	6	1	0.44
dog-7	20	3	3	86	90	none
toy2	20	958	5	5428	99	none
toy4	20	15	11	97	126	none
toy9	20	3	3	61	66	none
Mean		55.00	29.82	1019.04	890.98	0.20

4.3.3.2. Solution time. According to the computational results, we can say that the heuristics are very efficient. Because the order of magnitude of running times are different, we do not provide a detailed running time comparison between BP and heuristics. FAST and SF heuristics are the fastest; and they run in milliseconds for a typical input. CSA takes about 2.5 seconds to converge on the average. We force BP to stop running within 1 CPU hour.

As for the BP, column generation shows rapid progress in the early iterations, especially with the dual smoothing scheme. We illustrate this typical behavior of the optimality gap in Figure 4.16 by exposing the progress in the best lower bound and objective value our algorithm calculates on artificial3 data set for $K = 5$. This allows stopping the process earlier using lower and upper bounds on the optimal value.

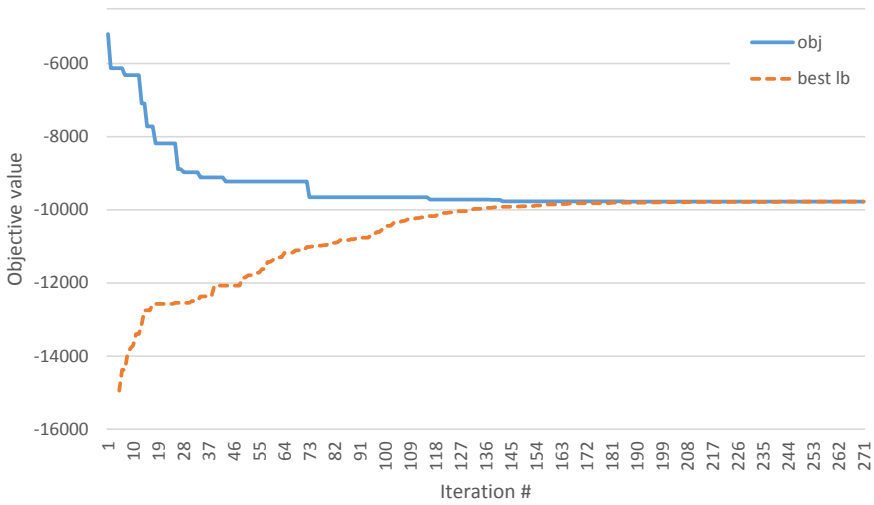


Figure 4.16. The behavior of objective value and best lower bound.

Three more graphs are provided related to the performance of the BP algorithm. They illustrate typical behavior observed on almost all test instances. The first one, Figure 4.17, uses data collected on *artificial3* sample for $K = 15$ and plots the number of columns in the master problem during the iterations. There is a sharp increase at the very beginning because, column discarding scheme starts working after columns get stale enough. Note that, the column discarding scheme keeps the number of columns relatively lower through out the run.

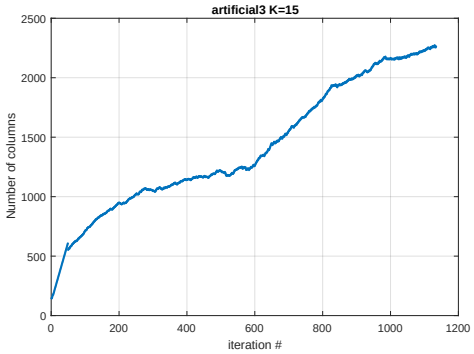


Figure 4.17. Typical change of the number of columns in the master problem.

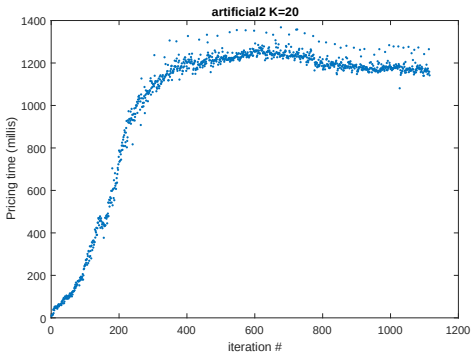


Figure 4.18. Typical change of the solution time of the pricing subproblem.

The second one, Figure 4.18, is obtained on *artificial2* data set for $K = 20$. It illustrates the typical behavior of the time devoted to the solution of the pricing subproblem throughout the iterations. It follows a regular behavior. The curve behaves parabolically at the beginning with a sharp increase. Then, it reaches a peak and settles down asymptotically. In short, pricing subproblems are easy in early iterations, and become harder as iterations progress. The hardness is capped at some point.

The third one, Figure 4.19, is the plot of the CPU time spent to solve the RLPMs throughout the iterations for *realistic4* sample for $K = 10$. With each iteration the solution of the LP becomes harder. After a while, a second trend emerges, and occasional solutions take significantly longer times.

Finally, for the first test problem group for $K = 3, 5, 10, 15, 20$, Table 4.2 and Table 4.3 presents the number of generated columns, time spent for the solution of the RLPMs (t_{RLPM}) and PSPs (t_{PSP}) in milliseconds. The first column includes the instances as usual. The first column of each group includes the median values for the

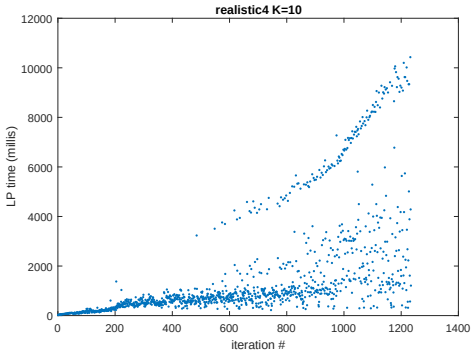


Figure 4.19. Typical change of the LP time.

number of columns added at each iteration (Δ_{col}). The second and third columns are the median values of t_{RLPM} and t_{PSP}. They all increase with K as expected. If time spent is zero, it means the median time spent is less than a millisecond.

4.3.3.3. Solution quality. As for the accuracy of the methods, the results are given in Table 4.4, Table 4.5, Table 4.10 and Table 4.11. The first column of the tables lists the instances. The accuracies of each algorithm are listed for different K values. The first column of each group consists of the objective values BP calculates, i.e. the value of the best feasible solution, total non-overlapping area between the rectangles and the region as in (4.4), in 1 hour CPU time limit. The next three are the percent relative deviations of the values computed using SF, FAST and CSA, respectively. They are calculated according to the formula

$$\text{PD} = 100 \times \frac{z_{\text{H}} - z_{\text{BP}}}{z_{\text{BP}}},$$

where z_{BP} and z_{H} for $\text{H} \in \{\text{SF}, \text{FAST}, \text{CSA}\}$ are the objective values obtained by BP and the heuristics SF, FAST and CSA, respectively. z_{H} values are given within parentheses in case the deviation is undefined, i.e. $z_{\text{BP}} = 0$. Column arithmetic averages and standard deviations of the relative percent deviations are also provided on the last row for a rough comparison of the methods. BP guarantees the optimal solution. However, it can take remarkably longer.

Table 4.2. The results presenting the efficiency of BP on easier samples.

	$K = 3$			$K = 5$			$K = 10$			$K = 15$			$K = 20$		
Image	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}
avatar1	1	0	0	2	1	0	2	1	0	3	1	0	1	1	0
avatar2	4	3	1	7	5	1	6	8	1	8	11	1	9	8	1
avatar3	9	2	1	3	3	1	4	6	1	7	10	1	9	6	1
avatar4	3	2	0	1	3	0	6	6	0	7	6	0	8	4	0
toy1	10	14	2	10	15	2	9	16	1	1	18	0	0	15	0
toy2	3	5	5	10	160	8	9	230	5	10	112	9	10	118	9
toy3	6	12	11	8	34	12	7	140	13	6	301	13	7	362	13
toy4	0	13	7	6	31	9	6	63	10	7	100	10	5	103	10
toy5	6	2	3	10	2	3	5	3	4	7	5	4	7	6	4
toy6	3	304	145	5	322	292	7	2197	323	8	1856	329	8	2358	335
toy7	6	9	10	7	22	12	7	46	12	7	100	12	7	112	12
toy8	2	2	4	5	3	6	7	7	6	7	12	6	7	14	6
toy9	0	7	2	2	5	6	8	37	8	7	64	8	6	85	8
toy10	6	18	16	6	37	19	6	74	20	4	81	20	6	125	20
toy11	3	2	5	6	3	10	4	8	9	5	14	9	7	13	9
toy12	0	1	1	4	3	4	6	7	5	7	8	5	7	11	5
toy13	0	4	2	2	3	6	7	6	9	6	10	8	7	12	8
toy14	9	17	18	0	21	7	7	40	21	6	93	21	6	77	22

Table 4.3. The results presenting the efficiency of BP on harder samples.

	$K = 3$			$K = 5$			$K = 10$			$K = 15$			$K = 20$		
Image	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}	Δ_{col}	t_{LP}	t_{PSP}
artificial1	5	83	42	4	228	90	5	1263	95	5	2371	92	6	4590	91
artificial2	3	151	25	2	210	153	3	426	410	3	378	717	3	842	1180
artificial3	2	402	78	2	332	362	2	320	743	2	442	934	3	1500	1380
artificial4	2	159	83	2	146	190	3	454	647	7	1431	677	8	1750	671
artificial5	5	119	60	6	1200	97	7	4114	102	9	5117	103	9	6341	103
realistic1	6	88	62	5	214	163	7	1376	185	7	2062	193	8	2871	186
realistic2	3	123	45	2	115	142	4	1009	396	5	1465	426	8	2300	440
realistic3	3	215	85	8	2620	499	9	2447	507	9	2349	523	9	2797	521
realistic4	10	128	6	3	130	185	3	743	504	3	1433	538	4	1905	547
realistic5	2	121	45	2	165	235	4	1070	488	6	1345	508	7	1828	506
typical1	3	66	50	4	152	111	5	1795	125	5	3521	125	7	3086	126
typical2	2	148	171	2	191	269	3	1569	647	4	2110	734	7	3255	752
typical3	4	3128	307	8	4871	313	9	6079	315	9	6940	317	10	6825	318
typical4	2	165	10	2	150	277	3	680	587	3	1157	656	3	2061	711
typical5	2	84	49	3	165	131	4	474	259	6	1615	295	8	996	296
typical6	5	99	53	6	685	74	6	4639	81	8	5887	82	8	4400	82
typical7	3	91	73	5	226	98	5	842	118	4	1283	117	4	3079	114
typical8	3	72	29	3	131	116	4	364	276	5	1562	298	7	704	310
typical10	4	153	201	4	761	329	7	1114	365	7	1465	376	7	2184	377

Table 4.8 is structured similarly but reports results on MPEG 7 images. Because each category contains 20 instances, only the arithmetic averages and standard deviations of the relative percent deviations are reported. However, we should point out that for the configuration $K = 20$ and *device5*, BP performed extremely well for a particular test instance, producing huge errors for other methods. Therefore, we have treated that result as an outlier and removed it from the sample. In other words, reported arithmetic averages and standard deviations are calculated using 19 values out of 20.

We have observed that the restricted LP master tends to yield integer solutions. Approximately 5% of the runs yielded fractional solutions. Besides, most of the time an integer solution is acquired in very few branchings. We have also observed that, some problems with fractional optimal solutions had also alternative optimal integer solutions. This explains why only a few branchings are sufficient mostly.

In the early iterations of the column generation, the objective value decreases rapidly and approaches the optimum value. For the aforementioned reasons, the 1-hour CPU time limit does not effect its performance dramatically for the instances of the first four sets. However, this is not true for the last set.

For the most of the instances, BP is able to find an optimal solution within the 1-hour CPU time limit except the fifth group of test problems. These cases are marked with an asterisk in the tables. Optimal rectangle blankets obtained for the artificial1 - artificial5 data sets are illustrated in Table 4.9. They can be compared with the original ones given in Figure 4.9. Observe the increase in the quality of the approximation with the increasing K values. It is observed in Table 4.10 and Table 4.11, that BP cannot find a proven optimal solution in 1-hour CPU time limit. They turned out to be the most challenging test problems of the test instances. Still, BP gives the best results in one hour. Similarly, considerable decrease in the relative performances of the heuristics is observed. This points to the decrease in the performance of the BP. Also, it is possible to observe the considerable increase in the performance of SF. BP may have produced an optimum solution for certain instances. Nevertheless, necessary columns are not generated within the time limit to prove optimality. For example for *toy2* instance (see

Table 4.4. Performance comparison of the algorithms for $K = 3, 5, 10$ on easier samples.

(*) Proven optimal solution in 1 hour running time.

	$K = 3$				$K = 5$				$K = 10$			
Image	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)
avatar1	27*	25.9	22.2	25.9	18*	44.4	16.7	88.9	7*	85.7	128.6	385.7
avatar2	47*	4.3	19.1	55.3	31*	45.2	25.8	135.5	13*	130.8	123.1	461.5
avatar3	47*	42.6	12.8	42.6	33*	33.3	51.5	103.0	13*	69.2	184.6	415.4
avatar4	44*	38.6	6.8	56.8	28*	25.0	10.7	146.4	10*	100.0	180.0	590.0
toy1	0*	(70)	(0)	(0)	0*	(44)	(0)	(0)	0*	(0)	(0)	(0)
toy2	0*	(86)	(52)	(182)	0*	(51)	(32)	(182)	0*	(13)	(32)	(182)
toy3	107*	60.7	29.9	93.5	72*	26.4	62.5	187.5	35*	45.7	191.4	491.4
toy4	143*	18.9	14.7	74.8	95*	68.4	63.2	163.2	59*	78.0	116.9	323.7
toy5	158*	41.1	3.8	10.1	122*	68.0	22.1	20.5	71*	109.9	63.4	94.4
toy6	921	21.4	22.1	66.9	601*	72.0	68.6	118.0	369	82.7	83.2	244.7
toy7	136*	119.9	38.2	20.6	113*	61.1	46.0	15.0	72*	138.9	63.9	80.6
toy8	212*	16.5	9.4	19.3	148*	42.6	19.6	48.6	89*	92.1	60.7	109.0
toy9	186*	83.3	37.6	15.1	75*	130.7	108.0	105.3	42*	169.0	214.3	266.7
toy10	232*	16.8	14.2	23.7	174*	49.4	24.7	64.9	96*	63.5	82.3	199.0
toy11	297*	17.5	38.4	18.2	177*	92.7	110.7	13.0	115*	63.5	39.1	59.1
toy12	259*	42.9	0.0	28.2	151*	29.8	7.9	77.5	72*	98.6	87.5	163.9
toy13	328*	44.2	0.0	17.7	193*	125.9	30.6	40.9	101*	240.6	66.3	111.9
toy14	366*	26.5	11.2	23.5	304*	41.4	24.7	40.8	197*	94.9	57.4	93.9

Table 4.5. Performance comparison of the algorithms for $K = 3, 5, 10$ on harder samples.

(*) Proven optimal solution in 1 hour running time.

	$K = 3$				$K = 5$				$K = 10$			
Image	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)
artificial1	512*	44.5	9.6	49.2	285*	77.5	33.3	154.4	150*	142.7	60.7	383.3
artificial2	6973*	59.5	0.0	19.7	5287*	46.4	2.2	28.7	2769*	45.1	32.9	70.5
artificial3	6359	23.3	11.9	67.5	4741*	38.8	7.5	59.8	2787*	76.5	44.1	31.3
artificial4	2301*	48.8	16.0	32.8	1479*	66.7	59.2	47.5	578*	227.7	120.2	200.3
artificial5	180*	87.8	1.1	92.8	122*	26.2	74.6	184.4	56	78.6	305.4	519.6
realistic1	499*	29.9	38.5	78.0	325*	17.2	38.2	135.4	184	71.2	160.9	315.8
realistic2	1430*	27.5	0.8	106.7	725*	82.8	23.7	229.2	373*	144.5	133.8	480.4
realistic3	538*	21.2	77.0	191.1	294	61.9	87.8	432.7	185	38.9	95.7	746.5
realistic4	2023*	5.4	0.0	12.6	1255*	52.1	31.4	23.9	593*	176.7	133.7	147.0
realistic5	1431*	23.5	12.2	62.6	679*	86.0	44.8	242.7	360*	108.1	141.1	546.4
typical1	535*	11.4	12.7	43.9	365*	16.7	40.5	99.5	206	41.7	75.2	253.4
typical2	1811*	31.8	7.6	54.1	1180*	51.5	36.2	96.0	716	64.0	79.2	204.3
typical3	202	9.4	19.8	88.1	162	24.7	28.4	134.6	109	64.2	41.3	248.6
typical4	2203*	5.9	0.0	7.4	1590*	33.1	24.7	17.4	938*	77.9	92.6	54.8
typical5	1318*	31.4	8.2	19.9	861*	47.9	22.9	69.7	482*	28.0	74.1	203.1
typical6	283*	51.6	17.3	64.7	190*	63.7	41.6	135.8	95	38.9	107.4	371.6
typical7	579*	4.8	1.2	25.7	380*	16.8	21.8	67.4	193*	57.0	96.9	215.0
typical8	1577*	33.5	0.0	10.3	986*	45.4	16.9	51.6	573*	73.8	67.5	105.1
typical10	934*	30.6	11.6	36.3	681*	43.0	37.6	48.0	377	100.5	127.1	153.8

Table 4.6. Performance comparison of the algorithms for $K = 15, 20$ on easier samples. (*) Proven optimal solution in 1 hour running time.

	$K = 15$				$K = 20$			
Image	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)
avatar1	0*	(12)	(8)	(34)	0*	(12)	(8)	(34)
avatar2	4*	475.0	625.0	1725.0	0*	(22)	(20)	(73)
avatar3	3*	566.7	800.0	2133.3	0*	(20)	(25)	(67)
avatar4	2*	600.0	1050.0	3350.0	0*	(12)	(17)	(69)
toy1	0*	(0)	(0)	(0)	0*	(0)	(0)	(0)
toy2	0*	(0)	(9)	(182)	0*	(0)	(11)	(182)
toy3	20*	90.0	330.0	935.0	8*	262.5	900.0	2487.5
toy4	40*	77.5	120.0	525.0	26*	165.4	280.8	861.5
toy5	47*	148.9	136.2	193.6	30*	216.7	196.7	360.0
toy6	267	58.1	108.2	376.4	210	93.3	158.6	505.7
toy7	50*	152.0	132.0	160.0	32*	250.0	203.1	306.3
toy8	63*	71.4	81.0	195.2	46*	95.7	137.0	304.3
toy9	27*	170.4	385.2	470.4	17*	252.9	282.4	805.9
toy10	62*	106.5	141.9	362.9	37*	175.7	256.8	675.7
toy11	86*	77.9	66.3	112.8	63*	141.3	119.0	190.5
toy12	48*	106.3	118.8	295.8	28*	196.4	307.1	578.6
toy13	73*	305.5	97.3	193.2	51*	374.5	103.9	319.6
toy14	146*	147.9	90.4	161.6	111*	184.7	118.9	244.1

Table 4.7. Performance comparison of the algorithms for $K = 15, 20$ on harder samples. (*) Proven optimal solution in 1 hour running time.

	$K = 15$				$K = 20$			
Image	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)
artificial1	95*	146.3	148.4	663.2	65	227.7	240.0	1015.4
artificial2	1510*	96.6	135.4	152.0	1147	127.0	129.2	194.8
artificial3	1631*	138.6	91.8	86.8	1102	193.7	144.9	135.1
artificial4	459	155.3	199.6	278.2	387	146.5	198.7	348.6
artificial5	33	130.3	581.8	951.5	22	218.2	709.1	1477.3
realistic1	127	112.6	184.3	502.4	92	158.7	201.1	731.5
realistic2	262	97.7	168.3	726.3	210	90.0	379.5	931.0
realistic3	127	33.1	203.1	1133.1	137	-1.5	243.1	1043.1
realistic4	431	103.2	183.5	239.9	333	113.2	270.6	339.9
realistic5	259	114.7	132.0	798.5	190	123.2	254.7	1124.7
typical1	138	51.4	153.6	427.5	100	57.0	156.0	628.0
typical2	462	122.1	116.0	371.6	372	113.2	159.7	485.8
typical3	65	129.2	256.9	484.6	55	167.3	280.0	590.9
typical4	639	182.8	134.3	115.3	482	127.2	138.0	185.5
typical5	329	48.0	91.5	344.1	253	63.2	131.6	477.5
typical6	58	46.6	243.1	672.4	38	105.3	326.3	1078.9
typical7	122	71.3	153.3	398.4	86	129.1	255.8	607.0
typical8	397	76.8	91.9	183.9	300	94.3	157.7	275.7
typical10	267	165.2	151.7	258.4	200	177.0	188.0	378.5

Table 4.8. Performance comparison of the heuristics: MPEG 7 benchmark category.

	$K = 3$			$K = 5$		
Category	SF (%)	CSA (%)	FAST (%)	SF (%)	CSA (%)	FAST (%)
bat	51.34 ± 20.72	8.99 ± 7.47	27.61 ± 14.22	55.94 ± 17.94	30.10 ± 10.48	50.30 ± 16.46
device5	182.53 ± 147.85	40.29 ± 57.90	36.39 ± 39.29	114.10 ± 68.33	37.42 ± 38.25	49.83 ± 40.61
dog	28.66 ± 17.02	10.90 ± 7.81	26.12 ± 15.09	47.48 ± 23.08	23.04 ± 11.40	43.36 ± 21.43
key	52.82 ± 37.96	25.99 ± 22.07	52.81 ± 50.12	58.38 ± 25.59	43.04 ± 20.21	112.02 ± 74.67
Misk	43.15 ± 9.05	22.67 ± 9.13	39.68 ± 15.75	60.70 ± 14.77	36.54 ± 11.32	70.23 ± 33.92
	$K = 10$			$K = 15$		
Category	SF (%)	CSA (%)	FAST (%)	SF (%)	CSA (%)	FAST (%)
bat	76.70 ± 34.79	62.67 ± 13.53	110.94 ± 39.89	94.75 ± 40.39	103.59 ± 18.35	185.30 ± 63.71
device5	166.01 ± 77.71	84.06 ± 50.95	125.26 ± 91.08	170.57 ± 84.24	114.53 ± 80.65	190.48 ± 118.67
dog	61.99 ± 23.58	47.22 ± 25.55	84.88 ± 45.67	72.69 ± 22.92	71.08 ± 38.58	136.96 ± 80.49
key	74.37 ± 30.53	110.26 ± 40.57	282.73 ± 131.24	97.35 ± 47.52	195.41 ± 79.82	470.81 ± 195.04
Misk	69.96 ± 34.39	83.76 ± 16.23	160.79 ± 69.50	65.84 ± 51.69	102.41 ± 16.98	225.72 ± 85.18
	$K = 20$					
Category	SF (%)	CSA (%)	FAST (%)			
bat	102.83 ± 43.50	140.20 ± 23.90	268.72 ± 89.28			
device5	167.56 ± 105.03	155.67 ± 122.09	247.48 ± 144.18			
dog	91.21 ± 30.76	100.67 ± 61.98	195.58 ± 118.57			
key	101.71 ± 55.17	277.82 ± 137.49	668.53 ± 290.36			
Misk	57.20 ± 50.65	110.09 ± 24.90	269.25 ± 101.10			

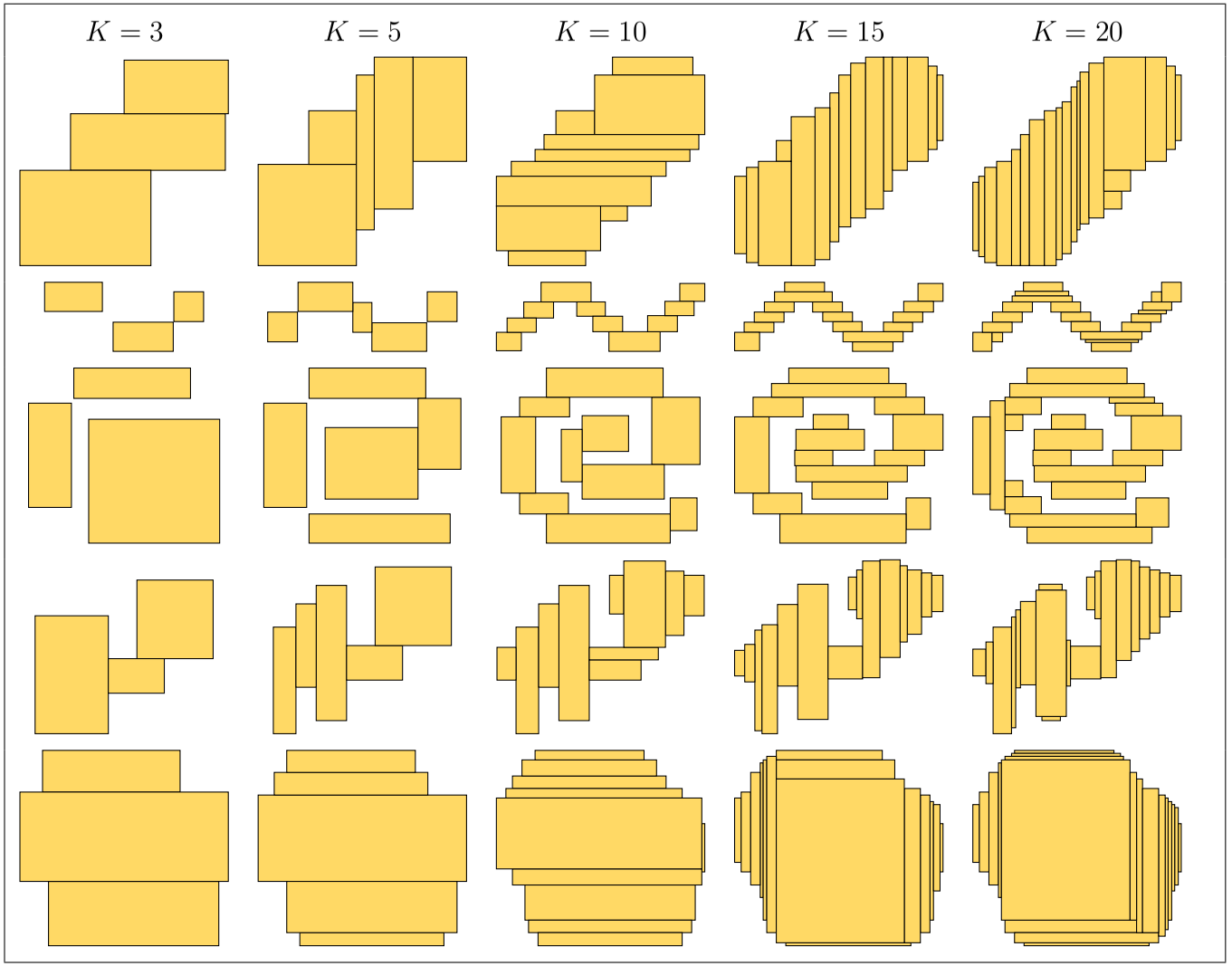
Figure 4.9), it is easy to come up with the optimum for $K > 2$ manually where the objective value is zero.

As the number of rectangles, K increases, the non-overlapping area decreases and CPU time increases for all methods, as expected. However, the amount of decrease in the non-overlapping area is much larger for BP, which is also expected since it is an exact solution method.

FAST performs slightly better than SF for small K values. As mentioned earlier, SF performs poorly on nonconvex shapes as illustrated in Figure 4.13 for a particular example. On some simple instances FAST may perform worse than other heuristics as a consequence of its greedy nature, as illustrated in Figure 4.14. FAST has another drawback: the maximum number of rectangles that can be placed may be limited depending on the image. After placing fewer than K rectangles, all the pixels of the target region might be covered with rectangles preventing FAST to add a new rectangle, because FAST does not modify rectangles placed in previous steps. For example, for a simple small shape like *avatar1*, it cannot place rectangles to improve the objective value and has the same value for increasing K. Although SF performs poorly for small K, the quality of the approximation increases as K increases. It even outperforms CSA for simple shapes when K is large.

BP always outperforms other methods in exchange for increased running time. Other methods are prone to being trapped in a local minimum, as illustrated in Figure 4.15. However BP might become computationally infeasible if the input is too large. BP's inefficiency is not a problem for real world computer vision applications. Although they are mostly based on online scenarios, a rectangle blanket is first computed offline, which is then repeatedly used online. For example in [79] and [9] authors first precompute rectangle blankets for different hand shapes and stores them for real time use to speed up hand shape matching. In Chapter 2, rectangle blankets are determined for crude human silhouettes at different locations offline before using them in a generative model to compute occupancy probability at a location. In a slightly different work [91], authors find the minimum number of rectangles that best fit an integrated circuit layout

Table 4.9. Optimum rectangle blankets for artificial1 - artificial5 with $K = 3, 5, 10, 15, 20$.



offline prior to the minimization of the mass production time.

Table 4.10. Performance comparison of the algorithms for $K = 3, 5, 10$ on hide samples.

	$K = 3$				$K = 5$				$K = 10$			
Image	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)
79510	4678	49.9	29.2	41.2	3940	22.8	38.0	47.1	2912	37.8	49.7	65.0
79611	4952	21.4	5.2	27.4	3834	23.8	17.6	57.0	2664	49.2	63.1	105.3
79712	3981	26.9	34.3	30.4	3420	23.4	37.0	33.5	2559	39.3	56.2	47.6
79813	4806	30.5	11.6	20.7	4045	21.5	22.6	31.9	3339	31.9	33.1	42.9
79914	4879	74.1	29.0	51.0	4259	33.6	28.2	57.0	3513	8.5	48.2	64.6
79915	4375	19.8	13.9	51.5	3406	28.3	36.4	79.7	2383	37.1	59.9	138.9
79916	5639	31.0	14.1	49.6	4487	18.8	43.0	71.5	3334	14.1	43.7	102.0
79917	4676	54.0	31.7	34.4	3689	33.6	24.2	57.1	2889	35.1	30.3	79.3
79918	5160	38.6	17.8	55.0	4336	19.4	26.3	66.4	3509	25.0	39.3	89.2
79919	4288	35.4	14.2	18.7	3369	24.8	39.1	34.6	2486	39.0	70.9	47.5

Table 4.11. Performance comparison of the algorithms for $K = 15, 20$ on hide samples.

	$K = 15$				$K = 20$			
Image	BP	SF (%)	CSA (%)	FAST (%)	BP	SF (%)	CSA (%)	FAST (%)
79510	2090	76.0	94.4	115.4	2492	23.3	80.0	76.8
79611	2505	22.5	56.2	117.4	2166	28.6	93.9	151.5
79712	2772	17.9	38.0	34.9	2186	17.1	91.2	71.1
79813	2668	40.3	71.0	78.9	2587	25.0	57.0	84.5
79914	2666	23.7	76.6	105.4	2540	21.2	82.0	111.5
79915	2736	8.6	53.2	108.0	1759	49.9	143.0	223.6
79916	2255	49.4	112.6	187.5	2671	14.9	73.0	142.7
79917	2460	22.0	49.1	100.4	2329	20.0	72.4	111.7
79918	3274	22.1	36.2	102.7	3099	15.0	50.2	114.2
79919	1905	22.2	108.2	82.6	2044	3.4	93.0	70.2

5. CONCLUSION

In this thesis first, a novel dataset, namely BOMNI, for person tracking in omnidirectional cameras is introduced. It contains 46 videos of persons moving inside a room; where the bounding boxes and the identity of the persons are annotated at every frame. Then, a generative Bayesian framework is developed for coupling person tracking and fall detection. The method is evaluated on BOMNI dataset, producing 93% tracking accuracy. Fall detection performance is measured as frame differences between the actual event and the detection. Proposed system usually detects the falls withing a couple of frames of the event, without any false positives. Third, a similar method for multiple person tracking is developed and evaluated on BOMNI dataset. The method reaches 86% tracking accuracy, improving upon a previous approach by 18%.

Fourth, a discriminative method for person detection is presented along with a novel structure called *Radial Integral Image*. Radial Integral Image speeds up feature extraction step from omnidirectional images. This method achieves state of the art detection performance on IYTE dataset: 4.5% miss rate for one false positive per image. Also presented approach is about 10 times faster than the previous state of the art.

Finally, the problem of representing a shape with multiple rectangles, *Rectangle Blanket Problem*, is formulated as an integer programming problem and a branch-and-bound scheme is presented to solve it optimally. Also a novel branching rule is proposed for branch-and-bound solutions of set packing problems. This novel branching rule results in a balanced search tree and visits fewer nodes to reach to an optimal solution. In addition to branch-and-bound scheme, two other novel heuristics, *Split and Fit* and constrained simulated annealing based one, are presented to solve the Rectangle Blanket Problem. The performance of these novel methods and another method from the literature are compared on a curated dataset. Branch-and-bound approach provides the best results with a margin when the images are not too large. Branch-and-bound approach becomes computationally infeasible to solve when the instances are too large,

heuristics usually provides good solutions that might be useful in practical applications.

There are several future research directions this thesis enables:

- Extracting features, i.e. pixel sums, from omnidirectional images using Radial Integral Image requires an interpolation operation. Instead of relying on the interpolation, inventing a new type of data structure and algorithm might enable calculating sums exactly and rapidly.
- Radial Integral Image uses polar coordinates to represent pixels and allows querying annular sector shaped regions. It might be possible to construct another integral image structure that takes camera parameters, especially radial distortion, into account. Incorporating more information of this kind into the integral image structure might result in increased detection performance.
- The novel branching strategy presented in Section 4.1.2 seems promising, it yields fewer branchings for most of the RBP cases. It would be interesting to investigate the impact of this novel branching rule on other set packing problems.
- The proposed solution to RBP uses rasterized representation of shapes and rectangles. Formulating the RBP using polygonal representations remains open for exploration.

REFERENCES

1. Aghajan, H. and A. Cavallaro, *Multi-camera networks: principles and applications*, Academic press, 2009.

2. Fleuret, F., J. Berclaz, R. Lengagne and P. Fua, "Multicamera people tracking with a probabilistic occupancy map", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 2, pp. 267–282, February 2008.

3. Alahi, A., Y. Boursier, L. Jacques and P. Vandergheynst, "Sport players detection and tracking with a mixed network of planar and omnidirectional cameras", *International Conference on Distributed Smart Cameras (ICDSC)*, pp. 1–8, IEEE, August 2009.

4. Alahi, A., L. Jacques, Y. Boursier and P. Vandergheynst, "Sparsity Driven People Localization with a Heterogeneous Network of Cameras", *Journal of Mathematical Imaging and Vision*, Vol. 41, No. 1-2, pp. 39–58, January 2011.

5. Çınaroğlu, I. and Y. Baştanlar, "A Direct Approach for Object Detection With Omnidirectional Cameras", *Signal, Image and Video Processing*, Vol. 10, No. 2, pp. 413—-420, 2016.

6. Crow, F. C., "Summed-area tables for texture mapping", *ACM SIGGRAPH computer graphics*, Vol. 18, pp. 207–212, ACM, 1984.

7. Viola, P. and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features", *Computer Vision and Pattern Recognition*, Vol. 1, pp. 511—-518, 2001.

8. Dollár, P., Z. Tu, P. Perona and S. Belongie, "Integral Channel Features", *British Machine Vision Conference*, pp. 1–11, 2009.

9. Mohr, D. and G. Zachmann, "FAST: Fast Adaptive Silhouette Area based Template Matching", *Procedings of the British Machine Vision Conference*, pp. 39.1–39.12, 2010b.

10. Agrawal, M., K. Konolige and M. R. Blas, "Censure: Center surround extremas for realtime feature detection and matching", *European Conference on Computer Vision*, pp. 102–115, Springer, 2008.

11. Demiröz, B. E., . Ari, O. Eroğlu, A. A. Salah and L. Akarun, "Feature-based tracking on a multi-omnidirectional camera dataset", *5th International Symposium on Communications, Control and Signal Processing*, pp. 1–5, May 2012.

12. Demiröz, B. E., A. A. Salah and L. Akarun, "Multi-omnidirectional cameras for ambient intelligence", *20th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, April 2012.

13. Demiröz, B. E., A. A. Salah and L. Akarun, "Fall detection using multi-omnidirectional cameras", *21st Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, April 2013.

14. Karpov, A., L. Akarun, H. Yalçın, A. Ronzhin, B. E. Demiröz, A. Çoban and M. Železnỳ, "Audio-visual signal processing in a multimodal assisted living environment", *15th Annual Conference of the International Speech Communication Association*, 2014.

15. Demiröz, B. E., A. A. Salah and L. Akarun, "Coupling fall detection and tracking in omnidirectional cameras", *International Workshop on Human Behavior Understanding (HBU@ECCV)*, pp. 73–85, Springer, 2014.

16. Demiröz, B. E., A. A. Salah and L. Akarun, "Multiple person tracking using omnidirectional cameras", *22nd Signal Processing and Communications Applications Conference (SIU)*, pp. 1231–1234, April 2014.

17. Berclaz, J., F. Fleuret, E. Türetken and P. Fua, "Multiple object tracking using k-shortest paths optimization", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 33, No. 9, pp. 1806–1819, 2011.

18. Demiröz, B. E., A. A. Salah, Y. Baştanlar and L. Akarun, "Affordable Person Detection in Omnidirectional Cameras Using Radial Integral Channel Features", *Machine Vision And Applications*, submitted.

19. Demiröz, B. E., K. Altınel and L. Akarun, "Rectangle Blanket Problem: Binary integer linear programming formulation and solution algorithms", *European Journal of Operational Research*, submitted.

20. Micušik, B., *Two-view geometry of omnidirectional cameras*, Ph.D. Thesis, Czech Technical University, 2004.

21. Shah, S. and J. K. Aggarwal, "Intrinsic parameter calibration procedure for a (high-distortion) fish-eye lens camera with distortion model and accuracy estimation", *Pattern Recognition*, Vol. 29, No. 11, pp. 1775—-1788, 1996.

22. Weng, J., P. Cohen and M. Herniou, "Camera calibration with distortion models and accuracy evaluation", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, Vol. 14, No. 10, pp. 965–980, 1992.

23. Scaramuzza, D., A. Martinelli and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras", *International Conference on Intelligent Robots and Systems*, pp. 5695–5701, IEEE, October 2006.

24. Geyer, C. and K. Daniilidis, "A unifying theory for central panoramic systems and practical implications", *European conference on computer vision*, pp. 445–461, Springer, 2000.

25. Schönbein, M. and A. Geiger, "Omnidirectional 3D Reconstruction in Augmented Manhattan Worlds", *International Conference on Intelligent Robots and Systems*

(IROS), 2014.

26. Caruso, D., J. Engel and D. Cremers, "Large-Scale Direct SLAM for Omnidirectional Cameras", *International Conference on Intelligent Robots and Systems (IROS)*, September 2015.

27. Karaimer, H. C. and Y. Baştanlar, "Detection and Classification of Vehicles from Omnidirectional Videos using Temporal Average of Silhouettes", *International Conference on Computer Vision Theory and Applications*, Berlin, Germany, 2015.

28. Barış, İ. and Y. Baştanlar, "Classification and Tracking of Traffic Scene Objects with Hybrid Camera Systems", *IEEE International Transportation Systems Conference*, Yokohama, Japan, 2017.

29. *PETS2001 Dataset*, 2001, `http://www.cvg.reading.ac.uk/PETS2001/pets2001-dataset.html`, accessed at January 2019.

30. *PETS-ICVS Dataset*, 2003, `http://www-prima.inrialpes.fr/FGnet/data/08-Pets2003/pets-icvs-db.html`, accessed at January 2019.

31. Fisher, R. B., "The PETS04 surveillance ground-truth data sets", *6th IEEE international workshop on performance evaluation of tracking and surveillance*, 2004.

32. Pers, J., M. Bon and G. Vuckovic, *CVBASE 06 Dataset*, 2006, `http://vision.fe.uni-lj.si/cvbase06/dataset.html`, accessed at January 2019.

33. Roggen, D., A. Calatroni and M. Rossi, "Collecting complex activity datasets in highly rich networked sensor environments", *Networked Sensing Systems*, 2010.

34. Behera, A., D. C. Hogg and A. G. Cohn, "Egocentric Activity Monitoring and Recovery", *Asian Conference on Computer Vision*, pp. 7–9, 2012.

35. Taylor, G. R., A. J. Chosak and P. C. Brewer, "OVVV: Using Virtual Worlds to

Design and Evaluate Surveillance Systems", *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, June 2007.

36. Dalal, N. and B. Triggs, "Histograms of oriented gradients for human detection", *CVPR*, Vol. I, pp. 886–893, IEEE, San Diego, 2005.

37. Zhu, Q., S. Avidan, M.-C. Yeh and K.-T. Cheng, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients", *Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 1491–1498, 2006.

38. Felzenswalb, P. F., R. B. Girshick, D. McAllester and D. Ramanan, "Object detection with discriminatively trained part based models", *PAMI*, Vol. 32, No. 9, pp. 1627–1645, 2010.

39. Tuzel, O., F. Porikli and P. Meer, "Pedestrian Detection via Classification on Riemannian Manifolds", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30, No. 10, pp. 1713–1727, October 2008.

40. Dollar, P., S. Belongie and P. Perona, "The Fastest Pedestrian Detector in the West", *BMVC*, p. 7, 2010.

41. Benenson, R., M. Mathias, T. Tuytelaars and L. Van Gool, "Seeking the strongest rigid detector", *CVPR*, pp. 3666–3673, 2013.

42. Trichet, R. and F. Bremond, "LBP Channels for Pedestrian Detection", *Winter Conference on Applications of Computer Vision*, Lake Tahoe, 2018.

43. Zhang, S., R. Benenson, M. Omran, J. Hosang and B. Schiele, "How Far are We from Solving Pedestrian Detection?", *CVPR*, 2016.

44. Girshick, R., J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

45. Redmon, J., S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection", *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

46. Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "Ssd: Single shot multibox detector", *European conference on computer vision*, pp. 21–37, Springer, 2016.

47. Ouyang, W. and X. Wang, "Joint deep learning for pedestrian detection", *ICCV*, pp. 2056–2063, 2013.

48. Zhang, L., L. Lin, X. Liang and K. He, "Is faster R-CNN doing well for pedestrian detection?", *European Conference on Computer Vision*, pp. 443–457, Springer, 2016.

49. Hu, Q., P. Wang, C. Shen, A. van den Hengel and F. Porikli, "Pushing the Limits of Deep CNNs for Pedestrian Detection", *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2017.

50. Cao, J., Y. Pang and X. Li, "Learning Multilayer Channel Features for Pedestrian Detection", *IEEE Transactions on Image Processing*, Vol. 26, No. 7, pp. 3210–3220, July 2017.

51. Eshel, R. and Y. Moses, "Homography based multiple camera detection and tracking of people in a dense crowd", *CVPR*, pp. 1–8, June 2008.

52. Breitenstein, M. D., F. Reichlin, B. Leibe, E. Koller-Meier and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 9, pp. 1820–1833, 2011.

53. Pirsiavash, H., D. Ramanan and C. C. Fowlkes, "Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects", *CVPR*, pp. 1201–1208, June

2011.

54. Kobilarov, M., G. Sukhatme, J. Hyams and P. Batavia, "People tracking and following with mobile robot using an omnidirectional camera and a laser", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 557—-562, 2006.

55. Zivkovic, Z. and B. Krose, "Part based people detection using 2D range data and images", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 214–219, 2007.

56. Wang, M. L. and H. Y. Lin, "Object recognition from omnidirectional visual sensing for mobile robot applications", *IEEE International Conference on Systems, Man and Cybernetics*, October 2009, pp. 1941–1946, 2009.

57. Kang, S., A. Roh, B. Nam and H. Hong, "People detection method using graphics processing units for a mobile robot with an omnidirectional camera", *Optical Engineering*, Vol. 50, No. 12, 2011.

58. Puig, L. and J. J. Guerrero, "Scale space for central catadioptric systems: Towards a generic camera feature extractor", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1599–1606, 2011.

59. Arican, Z. and P. Frossard, "OmniSIFT: Scale invariant features in omnidirectional images", *International Conference on Image Processing*, pp. 3505–3508, IEEE, 2010.

60. Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 91–110, 2004.

61. Lourenço, M., J. P. Barreto and F. Vasconcelos, "sRD-SIFT: Keypoint Detection and Matching in Images With Radial Distortion", *IEEE Transactions on Robotics*, Vol. 28, No. 3, pp. 752–760, 2012.

62. Saito, M., K. Kitaguchi, G. Kimura and M. Hashimoto, "People Detection and Tracking from Fish-eye Image Based on Probabilistic Appearance Model", *Sice*, 1, pp. 435–440, 2011.

63. Delannay, D., N. Danhier and C. De Vleeschouwer, "Detection and recognition of sports (wo) men from multiple views", *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pp. 1–7, IEEE, 2009.

64. Vondrick, C., D. Ramanan and D. Patterson, "Efficiently Scaling Up Video Annotation with Crowdsourced Marketplaces", *ECCV 2010*, pp. 610–623, 2010.

65. Bernardin, K. and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics", *Journal on Image and Video Processing*, Vol. 2008, pp. 1–10, 2008.

66. Smeulders, A. W., D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan and M. Shah, "Visual tracking: An experimental survey", *IEEE transactions on pattern analysis and machine intelligence*, Vol. 36, No. 7, pp. 1442–1468, 2014.

67. Forney Jr, G. D., "The Viterbi Algorithm", *Proceedings of the IEEE*, Vol. 61, No. 3, pp. 268–278, 1973.

68. KaewTraKulPong, P. and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection", *Video-based surveillance systems*, pp. 135–144, Springer, 2002.

69. Demiröz, B. E., *An example fall detection video*, `http://goo.gl/4KjkDx`, accessed at January 2019.

70. Baqué, P., T. Bagautdinov, F. Fleuret and P. Fua, "Principled parallel mean-field inference for discrete random fields", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5848–5857, 2016.

71. Demiröz, B. E., *An example multiple person tracking video*, `http://goo.gl/k8KdUL`, accessed at January 2019.

72. Dollár, P., C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 4, pp. 743–761, 2012.

73. Ehsan, S., A. F. Clark, N. ur Rehman and K. D. McDonald-Maier, "Integral images: efficient algorithms for their computation and storage in resource-constrained embedded vision systems", *Sensors*, Vol. 15, No. 7, pp. 16804–16830, 2015.

74. Bentley, J. L., "Multidimensional Divide-and-Conquer", *Commnunications of the ACM*, Vol. 23, No. 4, pp. 214—-229, 1980.

75. Demiröz, B. E., *Radial integral image implementation*, `https://github.com/barisdemiroz/radial_integral_image`, accessed at January 2019.

76. Dollar, P., C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: A benchmark", *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 304–311, 2009.

77. *IYTE Omnidirectional and panoramic image dataset*, `http://cvrg.iyte.edu.tr`, accessed at January 2019.

78. Karaimer, H. C. and Y. Başţanlar, "Car detection with omnidirectional cameras using Haar-like features and cascaded boosting", *IEEE Signal Processing and Communications Applications Conference*, pp. 301–304, 2014.

79. Mohr, D. and G. Zachmann, "Silhouette area based similarity measure for template matching in constant time", *Articulated Motion and Deformable Objects*, pp. 43–54, 2010a.

80. Wang, X., E. Türetken, F. Fleuret and P. Fua, "Tracking Interacting Objects

Optimally Using Integer Programming", *Computer Vision–ECCV 2014*, pp. 17–32, 2014.

81. Bay, H., T. Tuytelaars and L. Van Gool, "SURF: Speeded up robust features", *European Conference on Computer Vision*, pp. 404–417, Springer, 2006.

82. Lampert, C. H., M. B. Blaschko and T. Hofmann, "Efficient subwindow search: a branch and bound framework for object localization.", *PAMI*, Vol. 31, No. 12, pp. 2129–42, Dec. 2009.

83. Tolias, G., R. Sicre and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations", *arXiv preprint arXiv:1511.05879*, 2015.

84. Chaiken, S., D. Kleitman, M. Saks and J. Shearer, "Covering regions by rectangles", *SIAM J. Algebraic Discrete Methods*, Vol. 2, pp. 394–410, 1981.

85. Heinrich-Litan, L. and M. E. Lubbecke, "Rectangle covers revisited computationally", *ACM J. Experimental Algorithmics*, Vol. 11, pp. 1–21, 2006.

86. Stoyan, Y., T. Romanova, G. Scheithauer and A. Krivulya, "Covering a polygonal region by rectangles", *Comput. Optim. Appl.*, Vol. 48, pp. 675–695, 2011.

87. Ohtsuki, T., "Minimum dissection of rectilinear regions", *Proc. 1982 IEEE Symp. on Circuits and Systems*, pp. 1210–1213, Rome, 1982.

88. O'Rourke, J. and G. Tewari, "Partitioning orthogonal polygons into fat rectangles in polynomial time", *Proc. 13th Canadian Conference on Computational Geometry*, pp. 133–136, 2001.

89. Dyckhoff, H., G. Scheithauer and J. Terno, *Annotated Bibliographies in Combinatorial Optimization*, chap. Cutting and Packing, pp. 393–413, Wiley, Chichester, 1997.

90. Wascher, G., H. Haussner and H. Schumann, "An improved typology of cutting

and packing problems", *European Journal of Operational Research*, Vol. 183, pp. 1109–1130, 2007.

91. Chan, T. B., P. Gupta, K. Han, A. A. Kagalwalla, A. B. Khang and E. Sahouria, "Benchmarking of Mask Fracturing Heuristics", *International Conference on Computer-Aided Design*, San Jose, 2014.

92. Bentley, J., "Programming pearls: algorithm design techniques", *Communications of the ACM*, Vol. 27, No. 9, pp. 865–873, 1984.

93. Csurös, M., "Maximum-scoring segment sets", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 1, No. 1, pp. 139–150, 2004.

94. Bengtsson, F. and J. Chen, "Computing Maximum-Scoring Segments in Almost Linear Time", *12th Annual International Computing and Combinatorics Conference*, pp. 255–264, Springer Berlin Heidelberg, Taipei, 2006.

95. Bengtsson, F. and J. Chen, *Computing maximum-scoring segments optimally*, Luleå tekniska universitet, 2007.

96. Schrijver, A., *Combinatorial Optimization: Polyhedra and Efficiency*, Vol. B, Springer-Verlag, 2003.

97. Maire, F., "Polyominos and perfect graphs", *Information Processing Letters*, Vol. 50, pp. 57–61, 1994.

98. Motwani, R., A. Raghunathan and H. Saran, "Perfect graphs and orthogonally convex covers", *SIAM J. Discrete Math.*, Vol. 2, pp. 371–392, 1989.

99. Motwani, R., A. Raghunathan and H. Saran, "Covering orthogonal polygons with star polygons: the perfect graph approach", *J. Comput. System Sci.*, Vol. 40, pp. 19–48, 1990.

100. Gyori, E., "A minimax theorem on intervals", *J. Combinatorial Theory, Series B*,

Vol. 37, pp. 1–9, 1984.

101. Bern, M. and D. Eppstein, "Approximation algorithms for geometric problems", D. Hochbaum (Editor), *Approximation Algorithms*, pp. 296–345, 1997.

102. Bennell, J., G. Scheithauer, Y. Stoyan and T. Romanova, "Tools of mathematical modeling of arbitrary object packing problems", *Annals of Operations Resarch*, Vol. 179, pp. 343–368, 2010.

103. Desrosiers, J. and M. E. Lübbecke, "A primer in column generation", *Column generation*, pp. 1–32, Springer, 2005.

104. Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs", *Operations research*, Vol. 46, No. 3, pp. 316–329, 1998.

105. Vanderbeck, F., "Implementing mixed integer column generation", G. Desaulniers, J. Desrosier and M. M. Solomon (Editors), *Column Generation*, pp. 331–358, 2005.

106. Wolsey, L. A., *Integer Programming*, Wiley-Interscience, 1998.

107. Ryan, D. M. and B. A. Foster, "An integer programming approach to scheduling", A. Wren (Editor), *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pp. 269–280, North-Holland, Amsterdam, 1981.

108. Hoffman, A. J., A. Kolen and M. Sakarovitch, "Totally Balanced and Greedy Matrices", *SIAM J. Algebraic and Discrete Methods*, Vol. 6, pp. 721–730, 1985.

109. Wentges, P., "Weighted Dantzig-Wolfe Decomposition of Linear Mixed-integer Programming", *Int. Trans. Opl. Res.*, Vol. 17, pp. 151–162, 1997.

110. Pessoa, A., R. Sadykov, E. Uchoa and F. Vanderbeck, "Automation and combination of linear-programming based stabilization techniques in column generation", *INFORMS Journal on Computing*, Vol. 30, No. 2, pp. 339–360, 2018.

111. Demiröz, B. E., A. A. Salah and L. Akarun, "Coupling Fall Detection and Tracking in Omnidirectional Cameras", *International Workshop on Human Behavior Understanding, European Conference on Computer Vision*, pp. 73–85, 2014.

112. Wah, B. W., Y. Chen and T. Wang, "Simulated annealing with asymptotic convergence for nonlinear constrained optimization", *Journal of Global Optimization*, Vol. 39, No. 1, pp. 1–37, Dec. 2006.

113. Latecki, L. J., R. Lakämper and U. Eckhardt, "Shape descriptors for non-rigid shapes with a single closed contour", *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, Vol. 1, pp. 424–429, IEEE, 2000.

114. Baldacci, R., M. A. Boschetti, M. Ganovelli and V. Maniezzo, "Algorithms for nesting with defects", *Discrete Applied Mathematics*, Vol. 163, pp. 17–33, 2014.

115. Ganovelli, M. and V. Maniezzo, *private communications*, 2018.

116. *Kabeja*, `https://github.com/fuzziness/kabeja`, accessed at January 2019.

117. Demiröz, B. E., *Rectangle Blanket Problem benchmark*, `https://github.com/barisdemiroz/rbp_benchmark`, accessed at January 2019.

118. Kaehler, A. and G. Bradski, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, O'Reilly Media, 2015.

119. Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*, 2015, `http://www.gurobi.com`, accessed at January 2019.