UAH

Probabilistic Graphical Models applied to Road Segmentation

Master Degree in Advanced Electronic Systems. Intelligent Systems
Departament of Electronics

Presented by:
Mario Passani Montero

Advisor:
Luis Miguel Bergasa Pascual

Alcalá de Henares, September 23th, 2015

A la memoria de María José Montero

"Hoy es siempre todavia."
Antonio Machado

ACKNOWLEDGEMENTS

En este punto, me gustaría dar las gracias a todas las personas que de una forma u otra han estado a mi lado, ayudándome a cumplir mis metas. Como soy un poco desastre y vosotros sois muchos, seguramente me olvide de alguien, confió en que no me lo tengáis en cuenta.

A las primeras personas a las que debo mostrar mi más profundo agradecimiento son mi familia, especialmente a mi madre. Aunque me pesé bastante tu ausencia, pensar que estarías orgullosa de mi fue el importante aliento para trabajar, ir de un sitio para otro todo el día y estudiar una carrera (aunque me tiré algún que otro año de más y procastiné bastante, siendo sinceros).

Muchas gracias a Luis Miguel Bergasa por lograr que entregue este trabajo y por contar conmigo para formar parte de un grupo de investigación puntero a nivel mundial, permitiéndome publicar en los principales congresos internacionales en el campo de los sistemas de transporte inteligente. Espero que no tengas en cuenta algún que otro *deadline* que me salté... Este agradecimiento se extiende a mis compañeros del grupo *Robesafe*: Roberto, Sergio, Edu y Javi. Vaya lata os he dado durante el año, seguro que ahora mejora la productividad a la par que disminuye el consumo de palmeras. Especial agradecimiento se merece Javi por las recomendaciones que me realizó para el *paper* del *IV 2015*, que fueron clave para que ganáramos el *Best student paper award*. Muchas gracias al resto de profesores del Departamento, especialmente a Felipe Espinosa por ser un crack como profesor, y a Rafa Barea por amenizarnos las mañanas con sus instructivas charlas futboleras.

Este año he tenido varios trabajos, así que también tengo mucha gente a la que mostrar mi agradecimiento. Gracias a mis compañeros de la *OTEC* por el buen trato que siempre nos han dado, espero tener más desayunos y cañas con vosotros; a mis compañeros de *CYTSA* por darme la oportunidad de colaborar en un proyectos de ámbito internacional, por las catas de tartas y por el buen ambiente pese a la de horas y horas que echábamos; gracias a la gente de *Telefónica* por proponerme un trabajo tan interesante y por su buen ojo escogiendo a mis compañeros.

Me queda mucha gente a la que mostrar mi agradecimiento:

- A Andreas Geiger, Justin Domke, Jannik Fristch y Vladimir Haltakov por el intercambio de correos, que me ha permitido realizar un trabajo de mucha más calidad.
- A la gente de IEEE-ITSS por su reconocimiento a mi trabajo en el IV 2015. Muchas gracias también a José Alvarez por proponerme dar una minitalk y por su carta de apoyo.
- A Llorch, Mila, Maroto y demás *universitarios* con los que no coincide en ninguna clase.
- A David, Emilio, Miguel y demás *universitarios* con los que compartí demasiadas clases.
- A Blanca, por hacerme fan de Gran Hermano y a Roberto, por la compañía en las largas noches realizando trabajos (con la de tintos que tomábamos al final no han salido muy mal...).
- A Jacqueline, por seguir acordándose de mi pese estar en la otra punta del globo.
- A Esther, por sacar siempre un rato para quedar conmigo.
- A César por aguantar mis cosas de *ni-ni*.
- A Paco por preocuparte continuamente por mi pese a la distancia.
- A Luis por tus consejos de *escalada* y por el cariño que me ha mostrado siempre tu familia.
- ...

Gracias a todos.

RESUMEN

El futuro de los vehículos autónomos y de los sistemas avanzados de asistencia al conductor se sustenta en el desarrollo de sistemas de percepción capaces de proporcionar una detección rápida y precisa del entorno que rodea al vehículo. A pesar del largo trecho recorrido en el campo de detección de carretera, existe todavía un trecho importante en investigación para lograr incorporar capacidades de entendimiento de escena a los vehículos inteligentes. Este trabajo de fin de máster presenta un sistema de segmentación de carreteras a nivel de bit a partir de imágenes monoculares. La propuesta se basa en un modelo gráfico probabilístico y una serie de algoritmos y configuraciones seleccionadas oportunamente para acelerar el proceso de inferencia. En breve, el método propuesto emplea Conditional Random Fields y Uniformly Reweighted Belief Propagation. Por otro lado, el algortimo se valida en el dataset KITTI ROAD, alcanzando resultados en la línea del estado del arte pero con el tiempo de cómputo por imagen más bajo usando un PC estándar.

Palabras clave: Probabilistic Graphical Models, Computer Vision, Conditional Random Fields, Machine Learning, Pater Recognition.

ABSTRACT

The future of autonomous vehicles and driver assistance systems is underpinned by the need of fast and efficient approaches for road scene understanding. Despite the large explored paths for road detection, there is still a research gap for incorporating image understanding capabilities in intelligent vehicles. This Master thesis presents a pixelwise segmentation of roads from monocular images. The proposal is based on a probabilistic graphical model and a set of algorithms and configurations chosen to speed up the inference of the road pixels. In brief, the proposed method employs Conditional Random Fields and Uniformly Reweighted Belief Propagation. Besides, the approach is ranked on the KITTI ROAD dataset yielding state-of-the-art results with the lowest runtime per image using a standard PC

Keywords: Probabilistic Graphical Models, Computer Vision, Conditional Random Fields, Machine Learning, Pattern Recognition.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ADAS	Advanced Driver Assistance Systems.
ANN	Artificial Neural Network.
Back TRW	Back Tree-Reweighted Belief Propagation.
BEV	Bird's Eye View.
BFGS	Broyden Fletcher Goldfarb Shanno algorithm.
BN	Bayesian Network.
CRF	Conditional Random Field.
DGM	Directed Graphical Model.
FFT	Fast Fourier Transform.
FG	Factor Graph.
FN	False Negative.
FP	False Positive.
GPU	Graphical Processor Unit.
HOG	Histogram of Oriented Gradients.
IPM	Inverse Perspective Mapping.
ITS	Intelligent Transport Systems.
IU	Image Understanding.
LASER	Light Amplification by Stimulated Emission of Radiation.
LBP	Local Binary Pattern.

LDWS	Lane Departure Warning System.
LIDAR	LIght Detection And Ranging.
LKAS	Lane Keeping Assist Systems.
LUT	Look Up Table.
MAP	Maximum A Posteriori.
MPM	Maximum Posterior Marginals.
MRF	Markov Random Field.
NP	Non-deterministic Polynomial-time.
PGM	Probabilistic Graphical Model.
RADAR	RAdio Detection And Ranging.
ROI	Region of Interest.
SIFT	Scale Invariant Feature Transform.
SPRAY	Spatial Ray.
SVM	Support Vector Machines.
TN	True Negative.
TP	True Positive.
TRW	Tree-Reweighted Belief Propagation.
UGM	Undirected Graphical Model.
UM	Urban Marked Two-way Road.
UMM	Urban Marked Multi-Lane Road.
UU	Urban Unmarked.

LIST OF SYMBOLS

ψ_i	Unary potential.
ψ_{ij}	Pairwise potential.
A	Log partition function.
$\perp\!\!\!\perp$	Conditional independence.
$\mathcal{D}$	Dataset.
Δ	Loss functions.
$\emptyset$	The empty set.
$\mathbf{F}$	Matrix consist of all observation feature function for nodes.
f_j	Factor j from a factor graph.
$\mathbf{f}_k$	Observation feature function $k-th$ for nodes.
$f_k(y_v, \mathbf{x})$	Node feature function $k-th$ for node y_v.
$\mathbf{G}$	Matrix consist of all observation feature function for edges.
$\mathbf{g}_k$	Observation feature function $k-th$ for edges.
$g_k(y_u, y_v, \mathbf{x})$	Edge feature function $k-th$ for nodes y_u and y_v.
H	Entropy.
$[[\cdot]]$	Function evaluating to 1 if its argument is true, to 0 otherwise.
J	Unit matrix or number or edges features (according to the context).

K	Number of observation feature function for nodes.
$\mathfrak{L}$	Local polytope.
$\mathfrak{M}$	Marginal polytope.
$n_{\mathcal{C}}$	Number of cliques in the graph.
$\mathcal{N}_i$	The set of neighbors of X_i.
Ω	Null matrix.
P	Number of samples in a dataset.
π_i	The parents index of node i in a directed graphical model.
ψ_c	Clique potential.
$\mathcal{U}(\mathbf{y}, \mathbf{y}')$	Utility function measuring the satisfaction of a output predicted.
$\boldsymbol{w}$	Parameters of the graphical model.
$\mathcal{X}$	Input domain (a set of images).
X_i	A random variable, or the node for X_i in a graph.
x_i	A realization of random variable X_i.
$\mathbf{x}$	A realization of the random variable $X = (X_1, X_2, \ldots, X_n)$ (observed superpixels in an image).
$\mathcal{Y}$	Output domain (labels assign to the superpixels).
$\mathbf{y}$	A realization of the random variable $Y = (Y_1, Y_2, \ldots, Y_n)$ (classes asigned to the superpixels in an image).
Z	The partition function.

CHAPTER 1

INTRODUCTION

This work is framed with the research line of the *Intelligent Transport Systems (ITS)* developed by the *Robesafe* group at the *University of Alcalá* comprised of professors and researchers ascribed to the Department of Electronics. More specifically, the work is inscribed within the domain of real-time computer vision for *Advanced Driver Assistance Systems (ADAS)* and autonomous driving, being a contribution to initiatives trying to avoid that a vehicle get out of its ego-lane involuntary.

In brief, this project presents an approach to semantic interpretation of scene, *Image Understanding (IU)*, using a embedded system on-board vehicle to classify objects in real time. The core task of our proposal can be regarded as pixel labeling problem. Labeling, just as name implies, labels each pixel (or pixel block) in its category. In our case, the label set has two elements: *road* and *off-road*, so the systems performs real time road detection. Road detection is a key requirement for the successful development and use of intelligent vehicles due to its many potential practical applications, especially in ADAS (example of such ADAS are *Lane Departure Warning System (LDWS)*, *Lane Keeping Assist Systems (LKAS)*, automatic parking, etc.) and autonomous driving.

In order to carry out the proposal, we apply an innovative technique of machine learning, namely *Conditional Random Fields (CRFs)* [1]. Recent advances in discrete optimization and probabilistic graphical models have become CRFs in a standard tool for segmenting and labeling task.

1.1. Motivation

Road transport plays a vital role in the modern society, allowing economic growth, social development and prosperity. According to official data by European Union [2] people travel mainly by road, with private cars accounting for 73% of passenger traffic and about 44% of goods transported in the EU go by road.

However, the road transport is facing a number of challenges. Examples include but no are limited to: traffic accidents, congested roads, the constant rise in the price of fuel, air pollution, etc. Even though there have been advances to address the above-mentioned challenges, the numbers are startling. Thus, there were still more than 31,000 deaths on European Roads in 2010 [3]. In the Figure 1.1 we show the annual number of fatalities, injury accidents and injured people in the European Union [4].

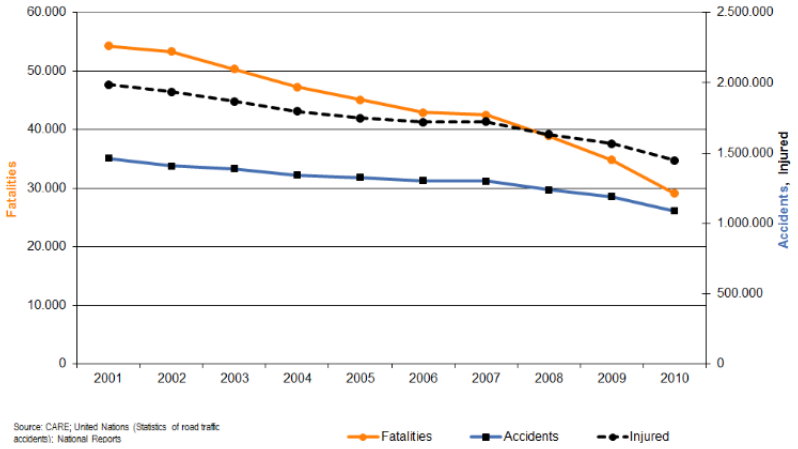


Figure 1.1: Annual number of fatalities, injury accidents and injured people in EU-27, 2001-2010.

According to the Figure 1.2, most accidents do no occur in motorways, but they occur in urban area and rural roads. In both cases, drivers can benefit from a road detection system due to the worse signaling conditions of rural roads and the challenging urban traffic with continuous lane changes.

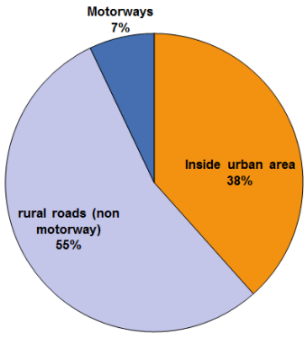


Figure 1.2: Share of fatalities by area type in EU-21, 2010.

In the last years, due to these alarming facts, the automotive industry is introducing different systems, including systems for active and passive safety to improve the security conditions and achieve a more efficient driving. Perhaps one of the most representative examples are the ADAS [5] that provide real time help to the driver. According to several surveys [6], ADAS can prevent up to 40% of traffic accidents, depending on the type of ADAS and the type of accident scenario.

The ADAS can cover a wide range of systems, from systems that provide information or warnings, to system involved in the vehicle control and manoeuvrings tasks. Examples of

ADAS are:

- Adaptive cruise control.
- Adaptive light control.
- Automatic parking.
- Blind spot detection.
- Collision avoidance system, also knows as pre-cash system.
- Driver drowsiness detection.
- Electric vehicle warning sounds.
- Hill descent control.
- Intelligent speed adaptation.
- Lane change assistance.
- Lane departure warning system.
- Night vision.

In the Figure 1.3 we show an example of a car equipped with some of the ADAS listed above.

Nonetheless, the introduction of ADAS on the market is slow. In our opinion, the primary cause of low market penetration is the economic added cost together with the lack of information about these systems.

If we look carefully the ADAS listed above, we note that the most of them use computer vision techniques to detect the presence of possible objects and classify them. Therefore, we can conclude that a way to enhance the use of ADAS can involves to develop cheaper systems capable to perform IU easily installed on different vehicles.

In research and development these systems based on computer vision have grown significantly in importance, due to their advantages over other detection sensors or location technologies. The following are the main advantages identified by Guan [7]:

- They are relatively inexpensive and can be easily installed on a vehicle, and they can detect and identify objects without the need for complementary companion equipment.
- These systems can capture a tremendous wealth of visual information over wide areas, often beyond the longitudinal and peripheral range of other sensors such as *RAdio Detection And Ranging (RADAR)* or *Light Amplification by Stimulated Emission of Radiation (LASER)*.

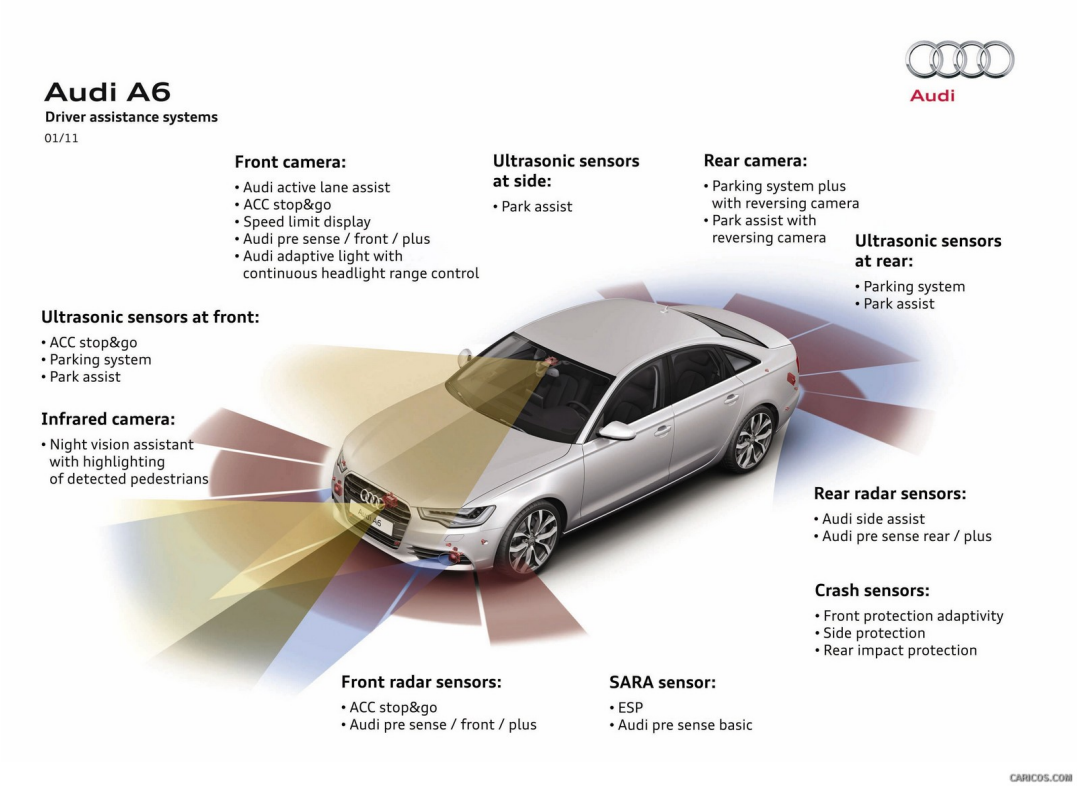


Figure 1.3: The Audi A6 is an example of premium car equipped with several moderns ADAS.

The continuous innovations in computer vision processing algorithms allows to exploit the wealth of visual data captured by cameras, identifying more subtle changes and distinctions between objects, enabling a wide range of ever more driving safety systems. Besides, the use of a monocular system, despite the obvious limitations to geometric reconstruction, has significant advantages over stereoscopic systems due to lower cost, computational requirements and technical complexity.

1.2. Problems Associated

The general problem of detecting objects in images is very complex because it involves the development of a system capable of distinguishing a particular class of objects from the rest.

Our system shall operate correctly over marked roads, like highways, and over unmarked road, that are common in rural areas and inner-city. Therefore, due to the great diversity of environments in which our system must work, we have to deal with a number of issues:

- Low visibility due to inclement weather, including overcast sky, heavy rain, etc.
- Presence of noise in the images.
- Occlusions produced by obstacles such other vehicles.
- Extensive shady zones.

- Variations in the appearance of the material of the road (asphalt, gravel, etc.) and possible wear down.

- The mobile nature of the work platform adds complexity to the problem.

Furthermore, road detection must be not only as accurate as possible but also prompt. In order to have a practical system, classification in real time is mandatory.

1.3. Proposed Objectives

The objectives pursued that we wish to achieve in this work are the followings:

- Development of an algorithm for road segmentation based on CRFs introduced by Lafferty *et al.* [1].

- Selection of the most appropriate model for the CRF.

- Selection of the best visual descriptors for the segmentation road problem.

- Validation of the proposed using the KITTI Road dataset, the most known public dataset for evaluating road area and ego-lane detection approaches [8].

- Documentation of the method developed.

- Report conclusions and propose future works.

A overall description of the system designed is depicted in the Figure 1.4. The remainder of the book explains in detail each one of the different stages.

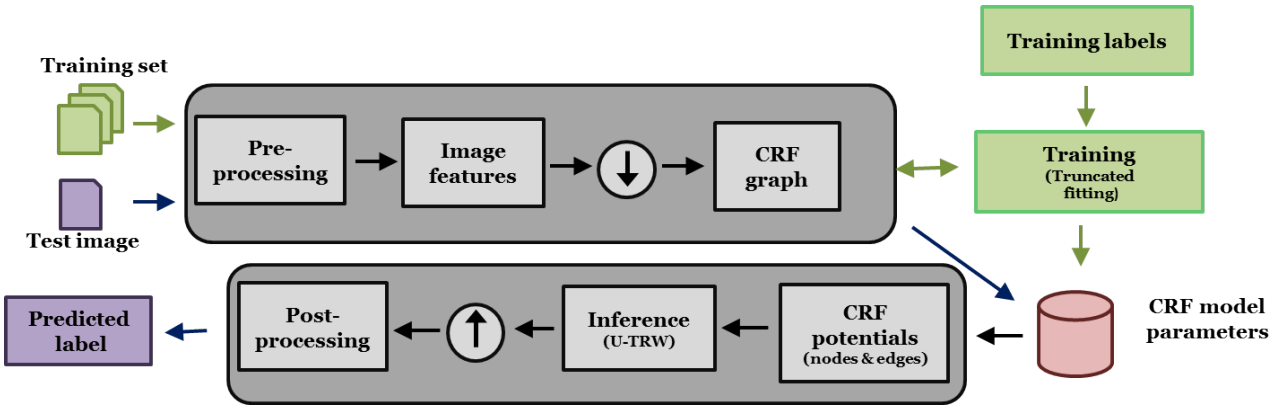


Figure 1.4: System block diagram showing the main steps in our road extractor. Down and up arrows correspond with the downsample and upsample performing with superpixels

1.4. Organization

This dissertation is structured as follow:

Chapter 2 : Reviews related state of art on road segmentation.

Chapter 3: This chapter presents a basic introduction to *graphical models*. Directed and undirected graphical models are described, giving particular emphasis to CRFs. It also described the tool *factor graphs*.

Chapter 4: This chapter describes the preprocessing followed to obtain the *Region of Interest (ROI)* in where the CRF will be built.

Chapter 5: We justify the choice of the implemented model and describe in detail the skeleton of our CRF, showing some of the required structures for its implementation.

Chapter 6: This chapters concerns the necessary process to inference the road in a given image. In our case, the graph of the CRF is not tree-structured, which implies that probabilistic inference is NP-hard. However, we can effectively approximate by some message passing algorithm.

Chapter 7: Here we detailed the learning task. That is, the process required to obtain the parameters of the model. In this work we use a recent approach presented by Justin Domke [9] to do parameter learning using approximate marginal inference instead the usual approach based in approximations of the likelihood.

Chapter 8: Describes the feature functions used in our model distinguishing between node and edge features. This chapter is key due the choice of appropriate features that determine the success or failure of the classifier.

Chapter 9: This chapter describes the post-processing based on morphological operations to slightly increase the overall accuracy.

Chapter 10: We present and discuss the results of the segmentation on the KITTI Road datasets presented above, carrying out an comparative evaluation against the state of art.

Chapter 11: Concludes the dissertation and discusses possible future directions that are not covered by this work.

CHAPTER 2

STATE OF ART

Road segmentation is a well-known problem in ITS that has been studied for decades [10]. However, the emergence of new systems such as ADAS (e.g., lane departure warning, adaptative cruise control, lane keeping, lane centering, turn assist), personal navigators, autonomous driving, etc. has caused a renewed interest in this issue because most of these systems need to detect the road surface ahead the ego-vehicle.

The potential uses for road scene segmentation are very varied [11] such as discard large image areas, impose geometrical constraints on objects in the scene, etc. Thus, a variety of systems has been developed to detect the road in some kind of environment. They used different sensors to acquire the information of the environment, such as, monocular vision, stereo vision, laser range finders and fusion of some of them [12].

Although the road detection problem, does not look like a hard one, this impression is misleading. The significant gaps in research, high reliability demands and large diversity in case conditions make that the building a useful road segmentation system is a large scale research and development effort [10].

On well marked roads, especially highways, road detection can easily been done by detecting the lane marking. However, general road detection is much more challenging due to many reasons, some of which are:

- Arbitrary road shape.
- Absence of lane markings.
- Occlusions with other vehicles and objects.
- Variations in the type and shape of the road.
- Variations in lighting conditions with the daytime. It also can occur when the vehicle is passing through a tunnel.

- Variations in the appearance of the material asphalt, gravel, etc. Its includes phenomenons like wear down and switch from one road to another.
- Variations depending on acquisition conditions.
- The clutter of background.

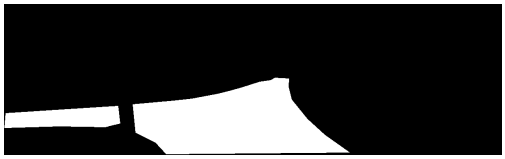
In order to show the variability of some of the conditions mentioned above we show some road scenes in Figures 2.1, 2.2 and 2.3. The road segmentation, as we can see, is an easy task only in some specific cases.

(a) Road urban scene image

(b) Ground truth

Figure 2.1: Example of a urban marked road and its associated ground truth taken from [8]. In this case the road segmentation is relatively straightforward.

(a) Road urban scene image



(b) Ground truth

Figure 2.2: Example of a "harder" urban marked road and its associated ground truth taken from [8]. In this case the road segmentation is further complicate.

(a) Road urban scene image

(b) Ground truth

Figure 2.3: Example of a urban unmarked road and its associated ground truth taken from [8]. The case is more complex due to the absence of lane marks and the presence of shadows.

For these reason, when addressing the study of state of art in road segmentation is appropriate to distinguish two types of roads because there are different approaches most appropriated for each case, namely:

1. *Marked*, like highways and highway-like roads (Figure 2.4a).
2. *Unmarked*, that are common in rural areas and inner-city (Figure 2.4b).

A particular case of road segmentation is the *ego-lane segmentation* that involves detect and extract the lane in which the vehicle is currently driving on. This is an important question in

(a) Marked road

(b) Unmarked road

Figure 2.4: Marked road vs. unmarked road.

the design of ADAS because these systems should avoid to invade the opposite lane. In the Figure 2.5 we show the difference between lane and road detection for the same road scene.

2.1. Marked Road Segmentation

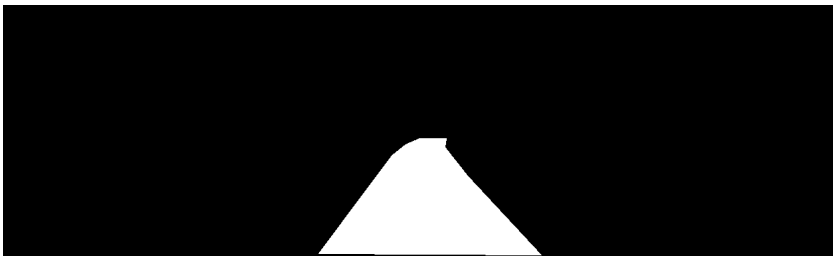
In order to segment the marked road scenes there are highly diverse techniques. However, the localization of road markings is the most used approach [13–16]. Methods based upon segmenting the road using the color cue have also been proposed but they do not work well for general road image, specially when the roads have little difference in colors between their surface and the environment. In addition, laser, radar and stereovision have also been used for structured-road detection.

For example, [14] Sun et al. propose a method for lane-marking detection which is carried out by color analysis of road scene images using hue-saturation-intensity [17] color model achieving better results that using RGB color model. A example is depicted in Figure 2.6.

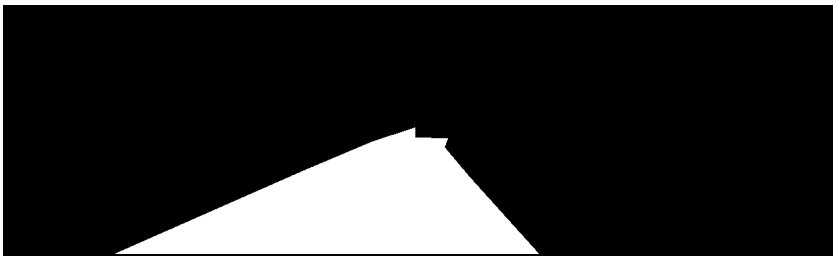
The approach of Felisa and Zani [13] produces reliable results exploiting a robust polyline matching technique capable of running at soft real-time rates. They employs an *Inverse Perspective Mapping (IPM)* transformation [18]. The Figure 2.7 shows a example of lane-marking using this approach.

Kuo-Yu and Sheng-Fuu Lin [19] use similar ideas. Firstly, they choose a ROI to find out a threshold using statistical method in a color image. Then, this threshold will be used to distinguish possible lane boundaries from the road. They use a color-based segmentation to find out the lane boundary. Since in the real world, lane marking is extensional vertically, they propose use a quadratic function to approach the lane marking because it may be interpreted as a kind of parabolic. This algorithm can deal with solid or broken line, straight or curved line, obstacle on lane marking, other traffic signs drawn on the road, road pavement, shadow,

(a) Road scene image.



(b) Lane ground truth.



(c) Road ground truth.

Figure 2.5: Images illustrating an example of the difference between lane and road detection.

(a) Example of input image.

(b) Final result.

Figure 2.6: Example of lane-marking detection using HSV color model taken from [14].

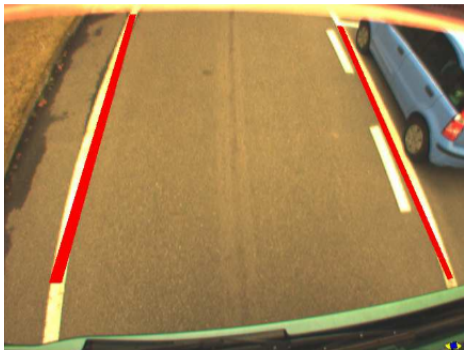


Figure 2.7: Example of lane-marking algorithm output using a camera mounted on top of a truck cabin taken from [13].

and sun light reflection. The system proposed demands low computational cost and memory requirements and is robust in the presence of noise, shadows, pavement and obstacles like cars, motorcycles and pedestrians.

Yingua He et al. [20] present a road-area detection algorithm based on color images. Their algorithm is composed of two modules:

- In the first module, an edge image of the road scene is analyzed to obtain the candidates for road borders and to delimit the area that will subsequently be used to compute the mean and variance of the Gaussian distribution, assumed to be obeyed by the color components of road surfaces.
- The second module effectively extracts the road area and reinforces boundaries that most appropriately fit the road-extraction result.

The combination of these modules can overcome basic problems due to inaccuracies in edge detection based on the intensity image alone and due to the computational complexity of segmentation algorithms based on color images. Figure 2.8 depicts an example of road detection.

Figure 2.8: Result of road detection in a curved roads in color-based road segmentation [20].

Southall and Taylor [21] present a method for estimating road shape using a single on board color camera, together with inertial and velocity information. They use a six-dimensional state vector $s[k]$ to describe both the position of the vehicle and the geometry of the road:

$$s(t) = [y_0(t),\, \tan\epsilon(t),\, C_0(t),\, C_1(t),\, W(t),\, \theta(t)]^T \tag{2.1}$$

where y_0 denotes the lateral offset and ϵ the bearing of the vehicle with respect to the centre-line of the lane, C_0 and C_1 the curvature and rate of change of curvature of the lane ahead of the vehicle, W the width of the lane, and θ the pitch of the camera to the road surface, which is assumed to be locally flat. Then, given a state $s(t)$ Equation (2.2) describes the shape of the road ahead of the vehicle:

$$y(x) = y_0 + \tan(\epsilon)x + \frac{C_0}{2}x^2 + \frac{C_1}{6}x^3 \tag{2.2}$$

where y is the lateral position of the road centre with respect to the vehicle, and x the distance ahead, as illustrated in Figure 2.9.

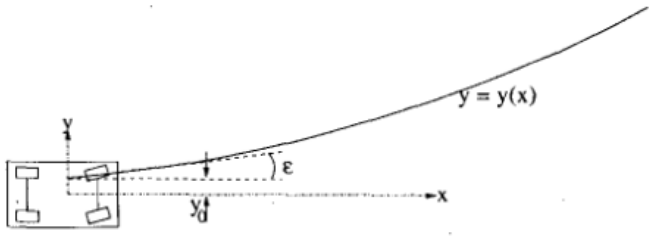


Figure 2.9: The camera, road and image co-ordinate systems in [21].

To extract the lane marks they run a Hough transform [22] algorithm whereas the road shape is estimated using a particle filter [23]. Bin and Jain[24] also use a Hough transform to extract the lane-markings.

Another quite different approach is presented by McCall and Trivedi [16], they propose to use steerable filters [25] for robust and accurate lane-marking detection. According to the experiments carried out by the authors, steerable filters provide an efficient method for detecting a wide variety of lane markings under varying lighting and road conditions. In this way, steerable filters help in providing robustness to complex shadowing, lighting changes from overpasses and tunnels, and road-surface variations. Moreover, steerable filters are computational simples, allowing a a fast implementation. Figure 2.10 depicts some examples of lane segmentations.

An alternative approach is based on the use of splines [26] to fit the limits of the roads. For example, Kaske *et al* [27] propose using Chi-Square fitting combined with random search to obtain the best set of parameters of a deformable template corresponding with the lane boundaries; an example of lane segmentation can be found in Figure 2.11. A similar approach is presented by Jung and Kelber [28] but using linear-parabolic splines. Finally, Wang et al. [15] proposed a lane detection and tracking algorithm able to describe a wide range of lane structures using B-snakes, an economical realization of *snakes* (also knows as *active contours*) by using far fewer state variables by cubic B-Splines as is depicted in Figure 2.12.

Although the cost of vehicle safety technology is dropping, most safety technologies are not available in economy vehicles and it will be a decade before the vast majority of cars on the

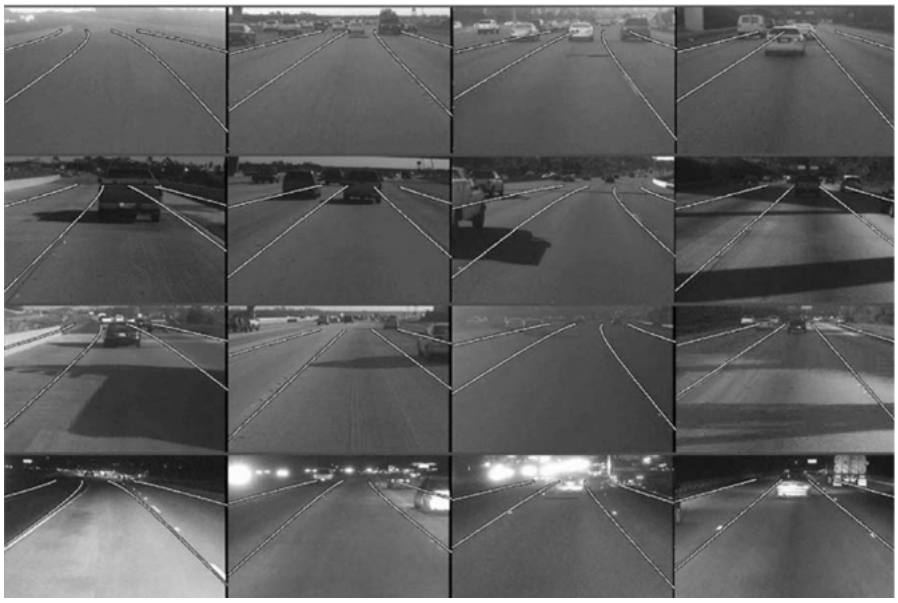


Figure 2.10: Results of lane segmentation based on steerable filters. Scenes from dawn (row 1), daytime (row 2), dusk (row 3), and nighttime (row 4). These scenes show the environmental variability caused by road markings and surfaces, weather, and lighting.

Figure 2.11: Image with bad contrast but well marked lane boundaries using splines

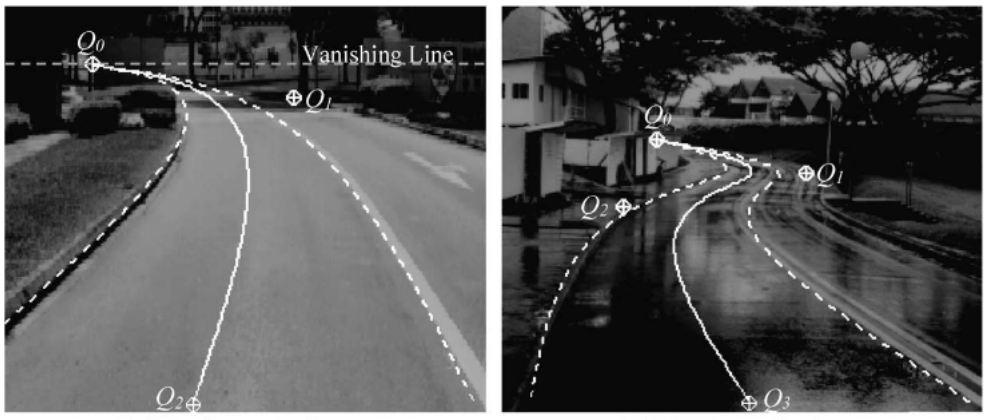


Figure 2.12: The model of left image uses 3 control points and the model of the right uses 4 control points.

road today have these safety features built-in. contrast, smartphone solutions can be used in all vehicles (new or old) and represent a cheap and disruptive technology. This is the reason why in the last years there has been an active work on using smartphones to assist drivers. One clear example that can be cited is *DriveSafe*, an safety app presented by Bergasa et al. [29] that detects inattentive driving behaviors and gives corresponding feedback to drivers, scoring their driving and alerting them in case their behaviors are unsafe. This works employs a modification of the Dickmans clothoidal road model-based method [30] to evaluate drowsiness. In brief, the app detects the lane following this steps:

1. Transformation of the image to gray scale.
2. Creation of two ROIs, one for the left markings and another for the right one
3. Detection of edges in the ROIs using an adaptive Canny algorithm which maximizes the edges in each ROI.
4. Elicitation of candidate lines for each of the two ROIs using the Hough transform together with some geometrical constraints.
5. Election of a representative line per ROI maximizing the length of the line, minimizing the angle difference between the candidate and the road model and the difference between the model vanishing point and the vanishing point obtained among the candidates for the left and the right side.

Figure 2.13 shows an example of these steps for a lane detection process at night.

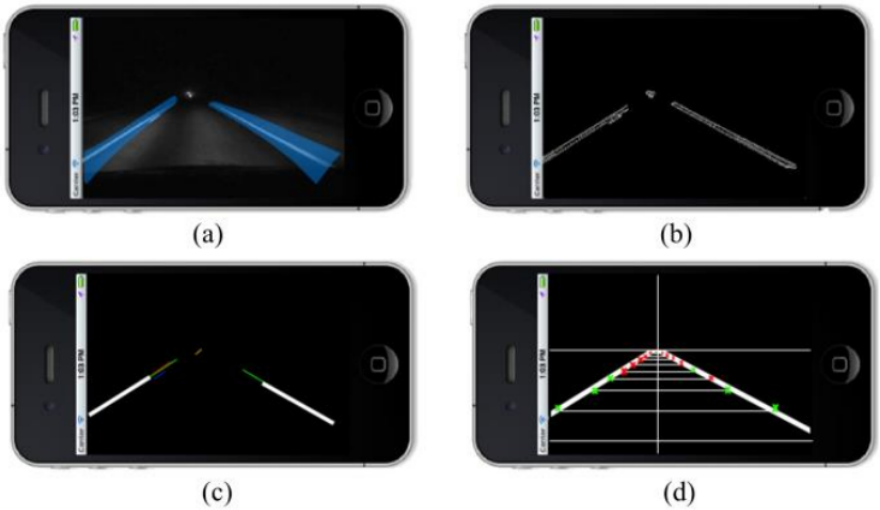


Figure 2.13: Example of lane detection process at night with *DriveSafe* a) ROIs in the gray scale image, b) Canny edges, c) Segmented and winner lines, d) Marker measures and lane model.

2.2. Unmarked Road Segmentation

For unstructured roads and structured roads without remarkable boundaries, road segmentation must be addressed from an alternative perspective to detection based on lane-markings.

Methods based upon segmenting the road using color cue have also proposed but they do not work well when the roads have little difference in colors between their surface and the environment, may failing with strong shadows and highlights.

Sotelo et al. [31] propose to use the color features of the HSI color space as the basis for performing the segmentation of nonstructured road. The HSI color space segments the image by using the cylindrical distribution of its color feature.

The approach presented by Tan et al. [32] uses color classification and learning to construct and use multiple road and background models. These color models are used to segment each color image into road and background by estimating the probability that a pixel belongs to a particular model. Since the color models are constructed on a frame by frame basis. The color model proposed by the authors uses normalized R and G because they are fairly robust to changes in illumination, while at the same time being fast to calculate. An example of detection is depicted in Figure 2.14.

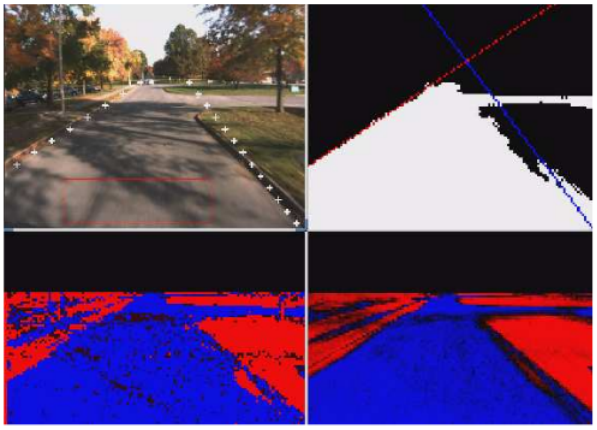


Figure 2.14: Nonhomogenous and complex road shape example. Upper-left shows the raw image of an intersection; bottom-left is the current road probability, bottom-right is the temporal fusion, and upper-right is the segmented road.

Álvarez and López [33] propose an approach to vision-based road detection robust to shadows. Their approach relies on using a shadow-invariant feature space combined with a model-based classifier. The model, a simple likelihood-based classifier, is built online to improve the adaptability of the algorithm to the current lighting and the presence of other vehicles in the scene. The Figure 2.15 depicts some road detection examples using this approach.

When there are little difference in color between the road and offroad areas, it is hard to find a intensity change to delimit state. The most plausible solution is using the texture. For example Nieto and Salgado [34] use a steerable filter bank to extract different edge images, then an enhanced edge image is composed with these images resulting in a clear identification of the lane markings. Finally, a fast Hough transform and minimum squares fitting find the best vanishing point of the image and the lane markings that delimit the road. The approaches of Rasmussen [35] and Kong et al. [36] (see 2.17) are pretty similar to this strategy but using Gabor filters to estimate the vanishing point. In all of these cases, the road segmentation is robust to variations in the illumination and the type of road. However, may fail with curved road, heavy traffic and strong shadow edges.

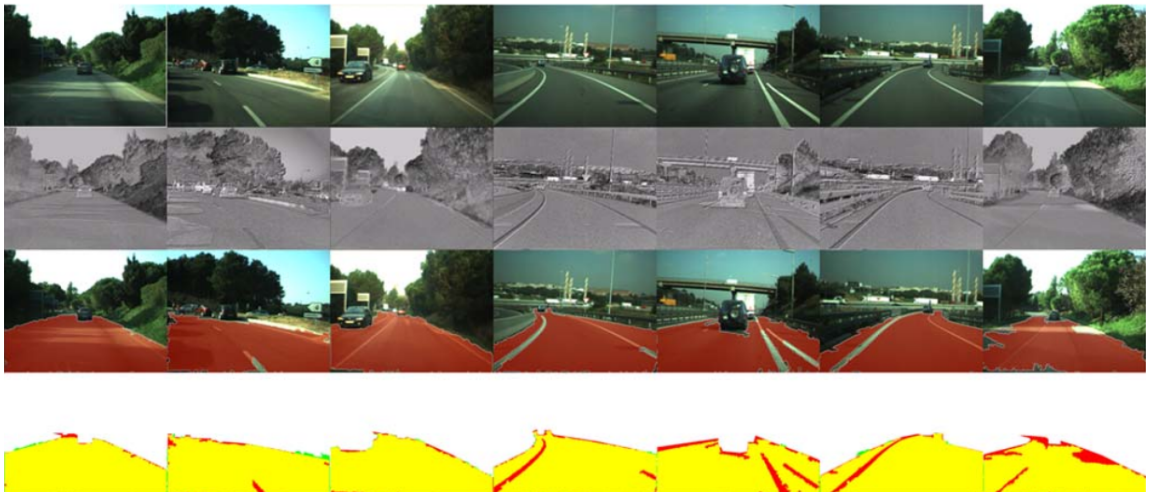


Figure 2.15: Road-detection examples. (Top row) Original image. (Second row) Illuminant-invariant image. (Third row) Detected road. (Bottom row) Comparison against hand-segmented result. (Yellow) Correctly classified pixels. (Green) Falsely detected road pixels. (Red) False background pixels.

Figure 2.16: Several examples of vanishing point estimation and road model extraction. Example (a) shows the most simple case where the road is almost empty, while cases (b) and (c) are quite more difficult as there are overtaking traffic and road traffic signals that difficult the correct detection. Case (d) shows a particular situation where the illumination conditions have abruptly changed due to the shadow casted by a bridge on the road.

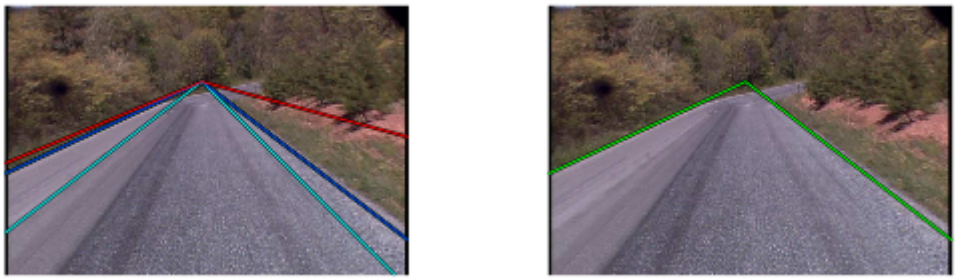


Figure 2.17: Road segmentation using the vanishing point.

The color and texture is used in [37] employing *Artificial Neural Networks (ANNs)*. This work employs a block-based classification method consists on dividing the image in blocks of pixels and evaluate them as a single unit. A value is generated to represent this group, this value can be the average of the RGB, entropy and others features from collection of pixels represented (see Figure 2.18. The main disadvantage of this system is that the roads can present aperiodic texture, which is hard to characterize.

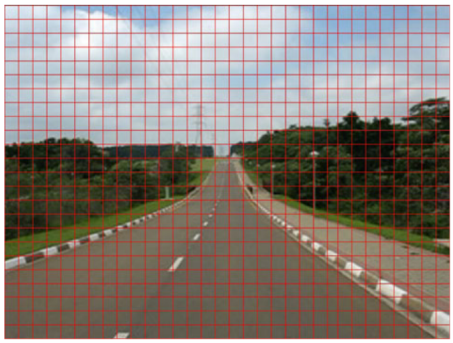


Figure 2.18: Blocks generated of frame of road scene. For each block, a feature value is calculated depending on the feature chosen. This strategy has been used to reduce the amount of image elements, allowing faster processing.

Lookingbill et al. [38] propose using reverse optical flow to provide an adaptive segmentation of the road scene finding examples of a ROI at previous time in the past. However, the method does not work well when the camera is unstable and the estimation of the optical flow is not robust enough. Some scene frames are depicted in Figure 2.19.

Figure 2.19: Segmentation unmarked road based on optical flow.

For unstructured roads and structure road without remarkable boundaries and markings, Alon et al. [39] propose a realtime system combining Adaboost [40] classification to form an initial segmentation with the texture boundaries constrained by geometric projection to find the road area for each frame.

The work of Kühnl et al. [41] is one of the approaches that achieves better results in the KITTI online evaluation website [8] (see 10) working in real time at the expense of use a powerful *Graphical Processor Unit (GPU)*.Their proposed system aims at detecting ego-lanes in cases of both explicit (lane-markings or curbstones) and implicit (unmarked road) delimiters. For that, the system represents visual properties of both, the road surface and delimiting elements in confidence maps based on analyzing local visual features. On such confidence maps, *Spatial Ray (SPRAY)* that incorporate properties of the global environment are calculated. The system, depicted in the workflow of Figure 2.20, consists of three parts:

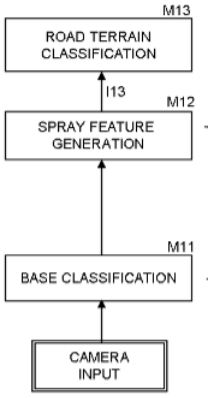


Figure 2.20: System block diagram showing the main processing steps in the SPRAY ego-lane extractor.

1. *Base classification*. A set of three base classifiers (boundary, road and marking), which work on preprocessed camera images create three confidence maps in a metric representation (using IPM). Different training strategies are used to specialize each classifier on its specif task.

2. *SPRAY feature generation*. Takes a confidence map from a base classifier and extracts a spatial feature vector for a defined number of points. Using radial vectors called *rays* the spatial layout with respect to the confidence map is captured at each individual point. A ray vector R_α includes all confidence values along a line with a certain angular orientation α. The SPRAY features corresponds with the locations where the ray value reaches a certain threshold. All the individual features computed from the different base classifiers are merged to obtain a spatial feature vector for each base point. The Figure 2.21 shows a confidence map of one base classifier in the metric space and the SPRAY feature generation process.

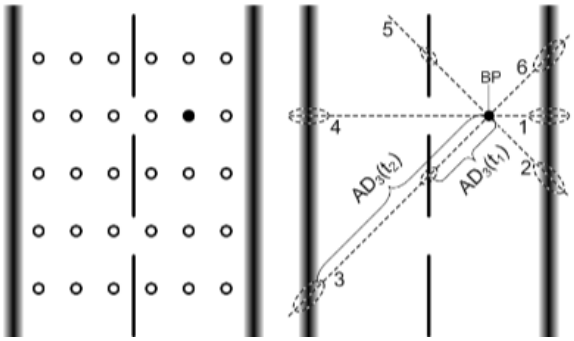


Figure 2.21: Distribution of base points over the metric space (left) and the SPRAY feature generation procedure illustrated for one base point (right).

3. *Road terrain classification* once the classifier is trained using GentleBoost [42], the system can process input images with the learned parameters.

All works that we have been mentioned rely on monocular vision. The stereo vision based approaches use *disparity map* acquired through stereo matching. Then the disparity map can

be analyzed to get the free space and the road. Labayrade et al. [43] present an approach based on the construction and investigation of the "v-disparity" image which provides a good representation of the geometric content of the road scene. Vitor et al. [44] create a set of probabilistic models using an adapted version of the Joint Boosting algorithm [45] with Texton (color and 2D texture) and Diston (3D information based on disparity map) feature maps. Essentially, the method consists on a set of weak classifiers analyzing the information according to Figure 2.22. Although this work achieves state-of-the-art results for the road segmentation, the classifier requires 2.5 minutes for each frame. In general, these methods need dense stereo matching which is time consuming and the error increases with the distance.

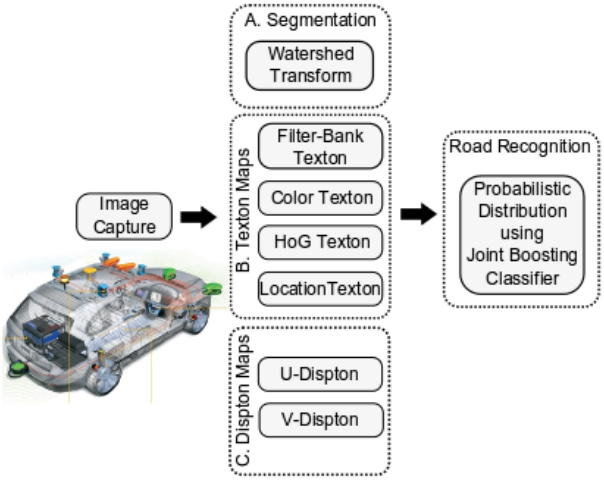
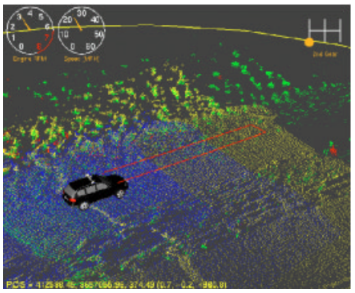


Figure 2.22: Block diagram of a probabilistic distribution approach to road segmentation.

Recently, several *LIght Detection And Ranging (LIDAR)* based road detection algorithms have been developed. This kind of methods use the accurate 3D location of the LIDAR points to analyze the scene and take the flat area as the road. For example, Thrun et al. incorporate LIDAR in the Stanley robot to detect nondrivable terrain a sufficient range to stop or take the appropriate evasive action [46]. Stanley is equipped with five single-scan laser range finders mounted on the roof, tilted downward to scan the road ahead. Figure 2.23a illustrates the scanning process. Each laser scan generates a vector of 181 range measurements spaced 0.5° apart. Projecting these scans into the global coordinate frame according, to the estimated pose of the vehicle, results in a 3D point cloud for each laser. Figure 2.23b shows an example of the point clouds acquired by the different sensors.

Moosmann et al. [47] present a graph-based approach to segment ground and objects from 3D LIDAR scans using a ngeneric criterion based on local convexity measures. Experiments show good results in urban environments including smoothly bended road surfaces. Chen et al. presents a algorithm for real-time segmenting three-dimensional scans of various terrains. An individual terrain scan is represented as a circular polar grid map that is divided into a number of segments. A one-dimensional Gaussian Process regression with a non-stationary covariance function is used to distinguish the ground points or obstacles in each segment. Thus, the proposed approach splits a large-scale ground segmentation problem into many simple Gaussian

(a) Illustration of a laser sensor: The sensor is angled downward to scan the terrain in front of the vehicle as it moves. Stanley possesses five such sensors, mounted at five different angles.



(b) Each laser acquires a three-dimensional 3D point cloud over time. The point cloud is analyzed for drivable terrain and potential obstacles.

Figure 2.23: Stanley, the robot who won the DARPA challenge.

Process regression problems with lower complexity, and can then get a real-time performance while yielding acceptable ground segmentation results.

Moreover, a recent work by R. Mohan [48] combines Deep Deconvolutional and Convolutional Neural Networks for the general task of scene parsing. Compared to different engineered features election methods, this is an alternative technique to automatically learn features directly from the images. This method is currently ranking first on KITTI ROAD benchmark. However, this approach is computationally intensive and it requires a GPU cluster to process the data.

CHAPTER 3

GRAPHICAL MODELS

The theory behind this work is essentially framed in the context of IU. The purpose of IU is, as its name suggests, to enable the machine to understand the world through performing complex reasoning on useful information extracted from processing of digital signals.

Mathematically, let $\mathbf{x}$ denote the *observed data*, typically the pixels or superpixels of an image, belonging to an *input domain* $\mathcal{X}$ and let $\mathbf{y}$ denote a vector of interest from an *output domain* $\mathcal{Y}$ that corresponds to a mathematical answer to the IU problem. Thus, IU can be formulated as finding a mapping from $\mathbf{y}$ to $\mathbf{x}$ [49]. For example, given an image, we would like to classify all objects in their class, which is essentially an *inverse problem* [50].

To this end, we often need to build a model of the real world that relates observed measurements to quantities of interest [51]. Unfortunately, several difficulties emerged in the modeling due to the fact that most of the vision problems are inverse, ill-posed and require a large number of latent and/or observed variables to express the expected variations of the answer [52]. Furthermore, the observed signals are usually noisy, incomplete and often only provide a partial view of the desired space in most cases.

Probabilistic Graphical Models (PGMs), usually referred to simply as *graphical models*, can help us to facing these situations. They combine harmoniously probability theory and graph theory towards a natural and powerful formalism for modeling and solving inference and estimation problems in various engineering fields [53].

A graphical model consists of a graph where each *node* (also called *vertices*) is associated with a random variable (or group of random variables) and an *edge* (also known as *link* or *arcs*) between a pair of nodes encodes probabilistic interaction between the corresponding variables. The absence of an edge between two variables represents *conditional independence* between those variables [52]. Conditional independence [54] means that two random variables a and b are independent given a third random variable c if and only if the conditional joint can be written as a product of conditional marginals, that is:

$$a, b \perp\!\!\!\perp c \Leftrightarrow p(a, b|c) = p(a|c)p(b|c) \tag{3.1}$$

where we are using the notation $a, b \perp\!\!\!\perp c$ to indicate that a and b are conditionally independent given c. It is noteworthy that in contrast two random variables a and b are *statistically* independent if and only if $p(a, b) = p(a)p(b)$.

Conditional independence is an important concept because it makes densities modular reducing the space required to represent densities, allowing us to to decompose complex probability distributions into a product of factors, each consisting of the subset of corresponding random variables. Thus, the complex computations required for inference can be carried out much more efficiently by using *message-passing* algorithms [51].

PGMs are powerful tools for visualizing the independence properties of complex probability models. They offers several useful properties, we can look at some of the more important ones identified by Bishop [53]:

- They provide a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new models.
- Insights into the properties of the model, including conditional independence properties, can be obtained by inspection of the graph.
- Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

Two types of graphical models are popular:

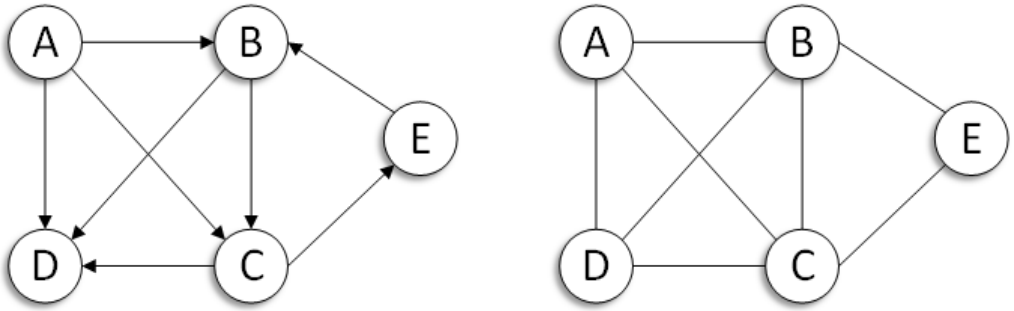
1. *Directed Graphical Models (DGMs)*, also known as *Bayesian Networks (BNs)*, in which the edges of the graphs have a source and a target provided by the particular directionality indicated by arrows. An example is depicted in Figure 3.1a.
2. *Undirected Graphical Models (UGMs)*, also known as *Markov Random Fields (MRFs)*, in which the links do not carry arrows and have no directional significance. Figure 3.1b shows an example.

In practice, directed graphs are better at expressing causal generative models whereas undirected graphs are better at representing soft constraints between variables. Sections 3.1 and 3.2 will provide a brief overview of both models.

For the purposes of solving inference problems, it is often convenient to convert both directed and undirected graphs into a different representation called *factor graph* [51]. Section 3.3 introduces this kind of representation.

3.1. Directed Models

Directed Graphical Models (DGMs) [55] can be described using a directed graph $G = (\mathcal{V}, \mathcal{E})$ which consists of a set $\mathcal{V} = \{X_1, X_2, \ldots, X_N\}$ of nodes and a set $\mathcal{E} = \{X_i, X_j\}$ of edges [53].



(a) Directed graphical model. (b) Undirected graphical model.

Figure 3.1: Examples of directed graphical model, also knows as Bayesian network, and undirected graphical model, also knows as Markov random field.

Each node represents a random variable X_i (for simplicity, we refer to a variable and its node interchangeably as X_i) whose realization we denote as x_i whereas edges indicate possible dependencies between these random variables.

A DGM describe a family of probability distributions according to the Equation (3.2):

$$p(\mathbf{x}) = \prod_{i=1}^{N} p(x_i|x_{\pi_i}) \tag{3.2}$$

where π_i indexes the *parent nodes* of X_i, that is to say, the sources of incoming edges to X_i (sometimes π_i may be the empty set). This key equation expresses the *factorization* properties of the joint distribution for a directed graphical model.

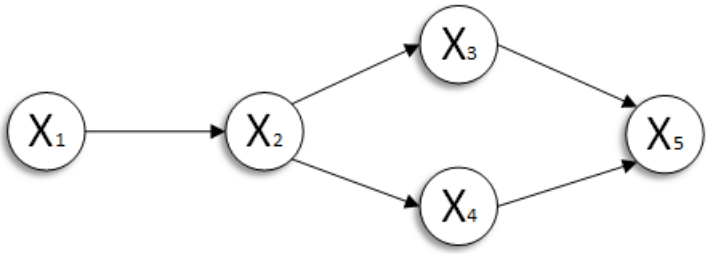


Figure 3.2: Example of a direct graphical model.

An example of directed model describing a set of five random variables is shown in Figure 3.2. This graphical structure implies the following parent relationships: $\pi_1 = \emptyset$, $\pi_2 = \{1\}$, $\pi_3 = \pi_4 = \{2\}$ and $\pi_5 = \{3, 4\}$. Using Equation (3.2) this yields the following factorization:

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_2)p(x_4|x_2)p(x_5|x_3, x_4) \tag{3.3}$$

To interpret the meaning of (3.3), our starting point is the factorization produced using the standard *chain rule* of probabilities. The chain rule allows us to factories a joint distribution into a product of distributions. One possible factorization according to chain rule of probabilities is:

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4) \tag{3.4}$$

This represents the joint distribution over X as the product of five distributions. When we compare (3.4) with (3.3), we notice that a number of conditioning variables are omitted in Equation (3.3). For example, the third term $p(x_3|x_2)$ has missed x_1 from its conditioning context. This omission represents a conditional independence relation, encoding our knowledge about the lack of inter-relatedness between the variables, and thus simplifies the joint probability distribution.

We can determine if a conditional independence $X \perp\!\!\!\perp Y|\{Z_1, \ldots, Z_k\}$ holds by appealing to a graph separation criterion called *d-separation*, which stands for *direction-dependent separation* [56]. X and Y are *d*-separated if there is no *active path* between them. Although the formal definition of active paths is somewhat involved, the set of active paths independence relations can be found using the Bayes' Ball algorithm [57], which analyzes paths connecting these sets of variables for possible sources of dependence.

Directed models are usually used to model causal relationships between random variables and have been applied in many fields such as computer vision, artificial intelligence, automatic control, etc. In computer vision, *Hidden Markov Models* [58] and *Kalman Filters* [59], two subsets of directed models, are popular in different tasks such as object tracking [60], denoising [61], motion analysis [62], sign language recognition [63], etc. due to its ability to model causal relationships. *Neural networks* [64] are other type of directed models that provide an important machine learning method to deal with vision problems [65].

3.2. Markov Random Fields

MRFs models [55,66] are useful in modeling a variety of phenomena where one cannot naturally to ascribe a directionality to the interaction between variables. For example modeling an image, it is plausible to suppose that the intensity values of neighboring pixels are correlated; however, being forced to choose a direction for the edges, as required by a DGM, is rather awkward. Furthermore, the undirected models also offer a different and often simpler perspective on directed models, in terms of both independence and structure and the inference task.

Undirected models represent a different factorization of the joint distribution to that of directed models, and with different conditional independence semantics. They are described by an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{X_1, X_2, \ldots, X_N\}$ are the nodes and $\mathcal{E} = \{X_i, X_j\}$ are the undirected edges. This graphical structure describe a family of probability distributions according to Hammersley-Clifford [67] theorem, which states that:

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(x_c) \text{ such that } \psi_c(x_c) > 0\,\forall c \in \mathcal{C} \tag{3.5}$$

where Z is the normalizing *partition function*; $\psi_c(x_c)$ denotes the *potential function* of a clique c,

which is a positive real-valued function on the possible configuration x_c of the clique c, and $\mathcal{C}$ denotes a set of cliques contained in the graph G. The kind of probability distribution specified in Equation 3.5 is usually called a *Gibbs* (or *Boltzmann*) distribution [52].

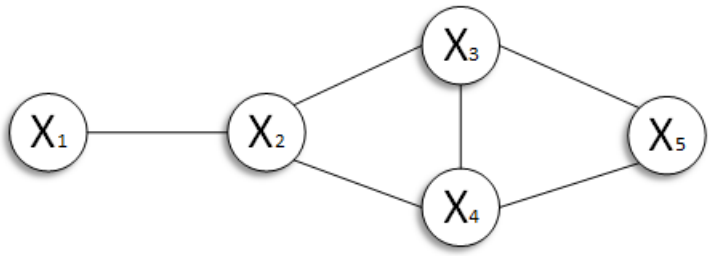


Figure 3.3: Example of a undirect graphical model.

The term *clique* describes a subset of nodes of an undirected graph such that its induced subgraph is complete; that is, every two distinct nodes in the clique are adjacent (i.e., connected by an edge). A *maximal-clique* is a clique that cannot be enlarged with additional nodes while still remaining fully connected. For example, the graph in Figure 3.3 contains the following cliques: $\{X_1\}$, $\{X_2\}$, $\{X_3\}$, $\{X_4\}$ $\{X_5\}$, $\{X_1, X_2\}$, $\{X_2, X_3\}$, $\{X_2, X_4\}$, $\{X_3, X_4\}$, $\{X_3, X_5\}$, $\{X_4, X_5\}$, $\{X_2, X_3, X_4\}$ and $\{X_3, X_4X_5\}$. Out of these, only are maximal cliques: $\{X_1, X_2\}$,$\{X_2, X_3, X_4\}$ and $\{X_3, X_4X_5\}$, in these three cliques we can subsume all the remaining cliques in the graph. If the potential functions, ψ_c, for each non-maximal clique is incorporated into the potential function of exactly one of its subsuming maximal cliques, the product of 3.5 can be limited to only maximal cliques without any loss of generality [68].

The partition function Z in (3.5) ensures that $p(\mathbf{x})$ is correctly normalized, i.e. $\sum_x p(\mathbf{x}) = 1$, being a valid probability distribution. This is achieved by summing out the numerator in (3.5) for every possible realization:

$$Z \triangleq \sum_{x_1} \sum_{x_2} \cdots \sum_{x_N} \prod_{c \in \mathcal{C}} \psi_c(x_c) \tag{3.6}$$

where each clique c indexes a subset of the variables in $\mathbf{x}$, as defined by its edges in the graph. This large summation arises as a direct consequence of the mostly unconstrained potential functions, ψ, in (3.5). Directed graphical models use conditional probability distributions as their potential functions and thus they need no normalization (i.e., $Z = 1$). However, this constant is not necessarily one for undirected models, and often proves intractable to calculate. This is because it requires summing over a number of realizations which is exponential in the number of random variables (if we have n discrete nodes in our graph having k possible states, then the evaluation of the normalization term involves summing over k^M states). For sparsely connected graphs, this summation can be made more efficient by dynamic programming [69].

The Hammersley-Clifford theorem applied to the graph $G = (\mathcal{V}, \mathcal{E})$ factored in accordance to the Equation (3.5) gives two further outcomes of interest:

1. *Local Markov property*: If $\mathcal{N}_i$ is the set of neighbors of X_i (i.e., those nodes which are connected to X_i by an edge) in G, then $p(x_i|\mathbf{x}\backslash x_i) = p(x_i|\mathcal{N}(X_i))$. Thus, we can easily see that

two nodes are conditionally independent given the rest if there is no direct edge between them.

2. *Global Markov property*: If we have three disjoints subsets of nodes, denoted $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$, we state that $\mathcal{A} \perp\!\!\!\perp \mathcal{B}|\mathcal{C}$ when the set $\mathcal{C}$ separates $\mathcal{A}$ and $\mathcal{B}$ in the graphical model. In others words, all paths connecting $\mathcal{A}$ to $\mathcal{B}$ must pass through the node $\mathcal{C}$. This is considerably simpler than the semantics for directed graphs, described in Section 3.1. Figure 3.4 illustrates an example of this property.

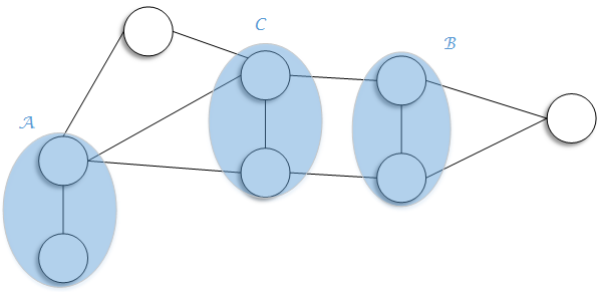


Figure 3.4: An example of an undirected graph in which every edge from any node in set $\mathcal{A}$ to any node in set $\mathcal{B}$ passes through at least one node in set $\mathcal{C}$. Consequently the conditional independence property $\mathcal{A} \perp\!\!\!\perp \mathcal{B}|\mathcal{C}$ holds for any probability distribution described by this graph.

An alternative representation avoids the positivity constraints in the potentials in Equation (3.5) by using the model known as the Gibbs distribution [70]. Thus, we can define an *energy function* expressing each potential function in Equation (3.5) as follows:

$$\psi_c(x_c) = \exp\left(-E(x_c)\right) \tag{3.7}$$

We can now rewrite Equation (3.5):

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{c\in\mathcal{C}} \exp\left(-E(x_c)\right) = \frac{1}{Z}\exp\left(-\sum_{c\in\mathcal{C}} E(x_c)\right) \tag{3.8}$$

In this way, finding the state $\mathbf{x}$ with the highest probability can now be seen as an energy minimization problem, with the benefit that the exponential can be moved outside the product.

The most used approach is to define the *log-potentials* as a linear function of the parameters of the model:

$$\log \psi_c(x_c) \triangleq \boldsymbol{\phi}_c(x_c)^T \mathbf{w}_c \tag{3.9}$$

where $\boldsymbol{\phi}_c(x_c)$ is a feature vector derived from the values of the variables x_c and w_c is a vector of model parameters. It is important to note that the potentials are not probabilities. Rather, they represent the relative "compatibility" between the different assignments to the potential.

This allows (3.5) to be re-parametrized over only the*log-potentials*:

$$p(\mathbf{x}) = \frac{1}{Z} \exp \sum_{c \in \mathcal{C}} \boldsymbol{\phi}_c(x_c)^T \mathbf{w}_c \tag{3.10}$$

The importance of Equation (3.10) is that allows characterize the global probability by the log-potential functions defined over maximal cliques, being completely unconstrained.

3.2.1. Conditional Random Fields

We often have access to measurements that correspond to variables that are part of the model. In that case we can directly model the conditional distribution [51] $p(\mathbf{y}|\mathbf{x})$ where $\mathbf{x}$ denote the observations that are always available and $\mathbf{y}$ is a tuple of latent variables (usually corresponds with the *labels*). This can be expressed compactly using CRFs [1].

A CRF is simply a conditional distribution $p(\mathbf{y}|\mathbf{x})$ with an associated *Undirected Graphical Model (UGM)*; it can be viewed as an MRF which is globally conditioned on the observed data $\mathbf{x}$. Accordingly, CRFs inherit global and local Markov properties. The major difference between these two is that MRFs are generative, i.e. model $p(\mathbf{y}, \mathbf{x})$, while CRFs are discriminative, i.e. model $p(\mathbf{y}|\mathbf{x})$.

Because the model is conditional, the dependencies among the input variables $\mathbf{x}$ do not need to be explicitly represented, affording the use of rich, global features of the input graphical structure [71]. In this way, we can draw our attention on modeling what we are really concern to us, namely the distribution of labels given the observations.

Generally, CRFs can be written as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \psi_c(y_c, \mathbf{x}) \text{ such that } \psi_c(y_c, \mathbf{x}) > 0 \, \forall c \in \mathcal{C} \tag{3.11}$$

We emphasize that the realization of random variables y_i are conditioned on the input $\mathbf{x}$ and that the potential function depends on the entire input and not only subsets of the input.

CRFs have been applied to various works in computer vision. For example, CRFs are used to model spatial dependencies in the image [72], to solve object class image segmentation [73, 74] and to jointly estimate the class category, location, and segmentation of objects/regions from 2D images [75].

Because in many computer vision problem the most common and fundamental type of interaction between pairs of variables is pairwise , the most used type of CRFs is the pairwise CRFs. In a pairwise CRF the Equation (3.11) is factorized into a sum of potential functions defined on cliques of order strictly less than three. In this way, we explicitly differentiate between types of potentials (unary and pairwise). The conditionally probability is then defined as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{i \in \mathcal{V}} \psi_i(y_i, \mathbf{x}) \prod_{(i,j) \in \mathcal{E}} \psi_{ij}(y_i, y_j, \mathbf{x}) \tag{3.12}$$

where ψ_i refers to the unary potentials whereas ψ_{ij} refers to the pairwise potentials.

3.2.1.1. Parameterization

PGMs define a family of distributions. By introducing parameters into the model we can identify members of this family with parameter values. The process of deciding what parameters should be used is known as *parameterization*. This idea is depicted in Figure 3.5.

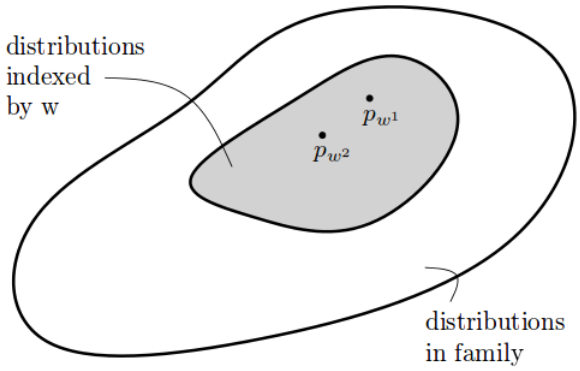


Figure 3.5: The full set of all distributions in the family defined by a PGM (shown in white) is restricted to a subset (shown in gray) by parameterization. This subset is indexed by $\boldsymbol{w}$ and any particular choice of $\boldsymbol{w}$ produces one probability distribution. Taken from [51].

Therefore, we can modify the Equation (3.12) to explicitly introduce the parameters:

$$p(\mathbf{y}|\mathbf{x};\boldsymbol{w}) = \frac{1}{Z(\mathbf{x};\boldsymbol{w})} \prod_{i\in\mathcal{V}} \psi_i(y_i,\mathbf{x};\boldsymbol{w}) \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(y_i,y_j,\mathbf{x};\boldsymbol{w}) \tag{3.13}$$

In this work we assume that each potential function is described by a log-linear combination of features extracted from the observed data $\mathbf{x}$. This ensures that the family of distributions over $\mathcal{V}$ parametrized by $\boldsymbol{w}$ is an exponential family [71]. Therefore, using (3.7), we have:

$$\begin{cases} \psi_i = \exp(\boldsymbol{w_i}^T \cdot \mathbf{f}_i(\mathbf{x})) & \text{(3.14a)} \\ \psi_{ij} = \exp(\boldsymbol{w_{ij}}^T \cdot \mathbf{g}_{ij}(\mathbf{x})) & \text{(3.14b)} \end{cases}$$

where $\mathbf{f_i}$ and $\mathbf{g_{ij}}$ are *feature functions* extracted from data.

3.2.1.2. The Three Basic Problems for Conditional Random Fields

Given a CRF model presented above, there are three basic problems of interest that must be solved for the model to be useful in practical applications. These problems are the following:

- *Marginal inference*: Given all the observations $\mathbf{x}$, the label configuration $\mathbf{y}$ and the model parameters $\boldsymbol{w}$, how do we efficiently compute $p(\mathbf{y}|\mathbf{x};\boldsymbol{w})$, i.e., the conditional probability of label configuration, given the model?

- *Label decoding*: Given all the observations $\mathbf{x}$ and the model parameters $\boldsymbol{w}$, how do we choose a corresponding label configuration $\mathbf{y}^* = y_1 y_2 \dots y_n$ which best "explain" the observations? Mathematically,

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}; \boldsymbol{w}) \tag{3.15}$$

- *Training* Given all the observations $\mathbf{x}$ and the label configuration $\mathbf{y} = y_1 y_2 \dots y_n$, how do we adjust the model parameters $\boldsymbol{w}$ to maximize $p(\mathbf{y}|\mathbf{x}; \boldsymbol{w})$?

3.3. Factor Graphs

Sometimes it is useful to factorize a global function of several variables as a product of simpler "local" functions over subsets of those variables in order to reduce the complexity of the problem. *Factor Graphs (FGs)* introduced by Kschischang *et al.* [76] permit us to visualize such factorization, using the distributive law to simplify the summations, and reusing intermediate values (partial sums).

Suppose that $g = (\mathbf{x})$ is a function that factors into a product of several "local" functions f_j, each one having some subset of $(x_1, \dots, x_n)$ as arguments. That is:

$$g(\mathbf{x}) = \prod_j f_j(\mathbf{x_j}) \tag{3.16}$$

where $\mathbf{x_j}$ denotes a subset of the variables and each factor f_j is a function of a corresponding set of variables $\mathbf{x_j}$.

A factor graph is a bipartite graph that expresses the structure of the factorization given by Equation (3.16). A factor graph has a variable node for each variable x_i, a factor node for each local function f_j, and an edge connecting variable node x_i to factor node f_j if and only x_i is the argument of f_j.

As an example, if we assume that the function of five variables $g(x_1, x_2x_3, x_4, x_5)$ can be expressed as a product:

$$g(x_1, x_2x_3, x_4, x_5) = f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5) \tag{3.17}$$

then, we have the factor graph of the Figure 3.6 where factor nodes are drawn as ■ and variable nodes as ◯.

DGMs, whose factorization is defined by Equation (3.2), represent special cases of (3.16) in which the factors $f_j(\mathbf{x_j})$ are local conditional distributions. To obtain one of the FGs associated with a DGM, the process is as follows:

1. For each node in the DGM we create a variable node in the FG.
2. For each conditional distribution in the DGM, we create a factor node in the FG.

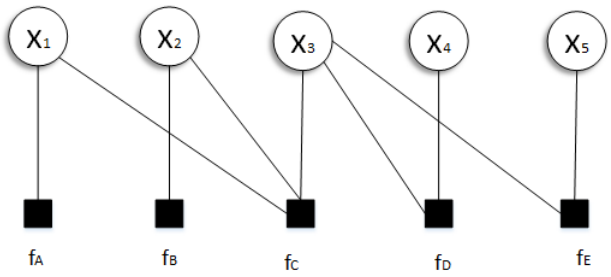


Figure 3.6: Example of a factor graph.

3. Add the corresponding edges.

It is important to note that here can be multiple FGs all of which correspond to the same DGM.

Similarly, UGMs, given by (3.5), are a special case in which the factors are potential functions over the maximal cliques (the partition function $\frac{1}{Z}$ can be viewed as a factor defined over the empty set of variables). To clarify this point, in the Figure 3.7 we represent the FG associated to the UGM of the Figure 3.3.

In the case of the CRFs the observations x_i are drawn as shaded variable nodes and the respective factors have access to the values of the observation variables they are adjacent to, as we depicted in the example of Figure 3.8.

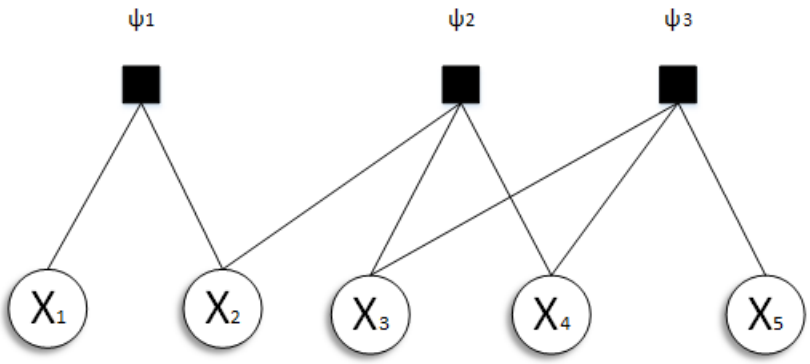


Figure 3.7: Factor graph associated to UGM of Figure 3.3

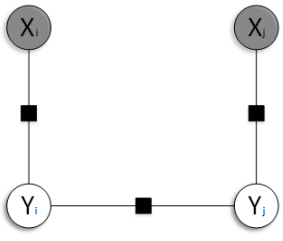


Figure 3.8: A factor graph specifying the conditional distribution $p(y_i, y_j) = \frac{1}{Z(x_i,x_j)}\psi_i(y_i, x_i)\psi_j(y_j, x_j)\psi_i j(y_i, y_j)$

3.3.1. Inference in Factor Graphs

Inference is the problem of computing the posterior distribution of hidden nodes given observed nodes in a factor graph. In particular, we are interested in the marginal distribution of each hidden node. The probabilistic inference problem in discrete factor graph models that do not contain cycles can be solved efficiently using a dynamic programming algorithm called the *sum-product algorithm* [76]. This inference is carried out by using an algorithm that involves *passing messages* on the factor graph. In brief, a message is a function that specifies how much it likes each of the possible values of a variable.

Let $\mathcal{N}_f(X_i)$ denote denote the set of factor nodes that are neighbors of the node X_i and let $\mathcal{N}_X(f_l)$ denote the set of variable nodes that are neighbors of factor function f_l. Then, we can compute probabilities in a factor graph by propagating messages from variable nodes to factor nodes and vice versa:

- Message from variable X_i to factor f_l,

$$m_{X_i \to f_l}(x_i) = \prod_{f \in \mathcal{N}_f(X_i)/f_l} m_{f \to X_i}(x_i) \tag{3.18}$$

- Message from factor f_l to variable X_i,

$$m_{f_l \to x_i}(x_i) = \sum_{x_1} \dots \sum_{x_M} f_l(x_1, x_2, \dots, x_M) \prod_{X \in \mathcal{N}_{X_i/X_i}} m_{X_i \to f_l}(x_i) \tag{3.19}$$

  both of the above message flow are depicted in Figures 3.9a and 3.9b.

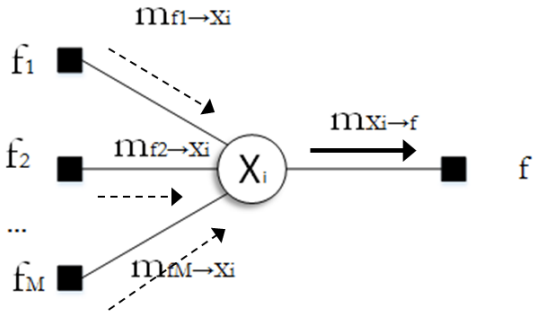
Once a variable has received all messages from its neighboring factor nodes, one can compute the probability of that variable by multiplying all the messages and renormalising:

$$\mu(x_i) \propto \prod_{f \in \mathcal{N}_f(X_i)} m_{f \to X_i}(x_i) \tag{3.20}$$

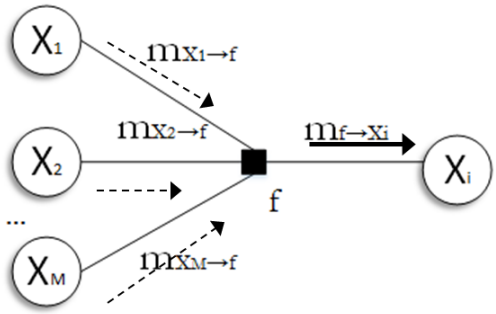
The equations 3.20 and 3.19 for computing the messages depend on previously computed messages. The only messages that do not depend on previous computation are the following:

- The variable to factor messages in which no other factor is adjacent to the variable; then the summation in (3.20) is empty and the message will be zero.
- The factor to variable messages in which no other variable is adjacent to the factor; then the inner summation in (3.19) will be empty.

For tree-structured factor graphs there always exist at least one such message that can be computed initially. The computed message in turn enables the computation of other messages. Moreover, we can order all message computations in such a way that we resolve all dependencies and eventually have computed all messages.



(a) Message from variable to node.



(b) Message from factor to variable.

Figure 3.9: Examples of message flows associated with the inference process

For tree-structured graphs this corresponds to the scheme shown in Figure 3.10 and 3.11 We first designate an arbitrary variable node (here we chose Y_m as the *tree root*. Then we compute all messages directed towards the root , starting with the leaf nodes of the factor graph because these are the only messages we can initially compute. We compute the remaining messages in an order that follows the leaf-to-root structure of the tree. From Figure 3.11 it is clear that for each message computation we will always have previously computed the information it depends upon. Once we have reached the root Y_m, we reverse the schedule as shown in Figure 3.11 and again we are sure to previously have computed the information the message depends on [51].

When the factor graph is not tree-structured but contains one or more cycles, the previous propagation algorithm is not applicable as no leaf to root order can be defined. However, the message equations remain well-defined. Therefore, we can initializing all messages to a fixed value and perform the message updates iteratively in a fixed or random order to perform computations "similar" to the original exact algorithm on trees. The resulting algorithm is named *Loopy Belief Propagation* [51, 77]. The Loopy Belief Propagation algorithm made approximate inference possible in previously intractable models [78]. The empirical performance was consistently reported to be excellent across a wide range of problems and the algorithm is perhaps the most popular approximate inference algorithm for discrete graphical models.

In practice, the algorithm does not always converge. If it fails to converge then the beliefs are a poor approximations to the true marginals. Therefore, we consider an alternative approach based on the Tree-reweighted belief propagation algorithm [79], which will be detailed in the Chapter 6.

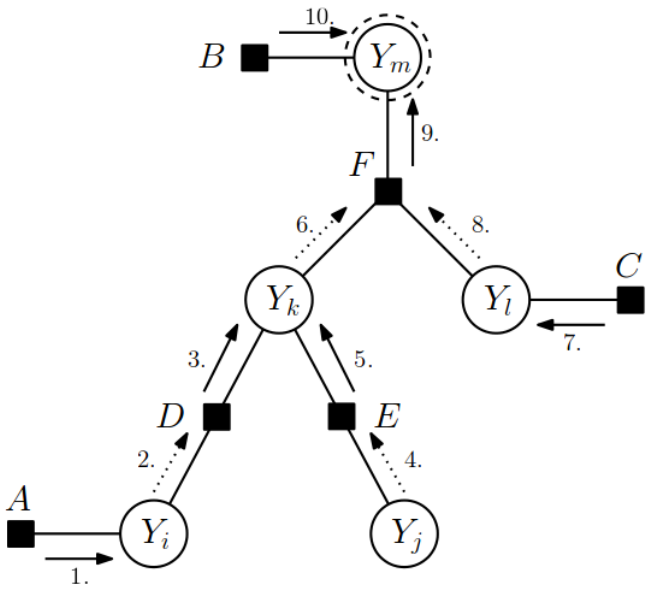


Figure 3.10: One possible leaf-to-root message schedule in the sum-product algorithm. Factor to variable messages are drawn as arrows, variable tofactor messages as dotted arrows. The tree is rooted in Y_m and the node is marked with a dashed circle. Taken from [51]

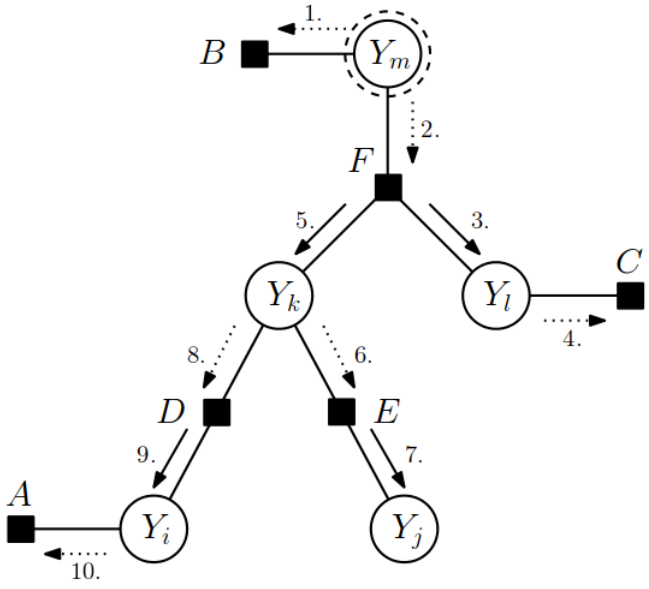


Figure 3.11: The root to leaf message schedule, the reverse of the schedule shown in Figure 3.10. Note that all edges now pass the message of type opposite of what they passed in the leaf toroot phase. Thus, for each edge both message types are available. Taken from [51]

CHAPTER 4

PREPROCESSING

The majority of the road scenes consist of vertical surfaces (i.e., buildings, vehicles, pedestrians, trees) positioned on a horizontal ground (i.e., road or sidewalk) with possible parts of the sky [11]. Due to physical and continuity constraints derived from vehicle motion and road design, the processing of the whole image can be replaced by the analysis of specific regions of interest only (the so-called *focus of attention*), in which the features of interest are more likely to be found [80]. Therefore, the road detection process can be bounded to a specific *Region of Interest (ROI)* where the road is more likely to be found on the images.

In our proposal, assuming a priori knowledge on the road environment, each road scene is preprocessed by passed through a filter to obtain the ROI removing non-relevant elements from the image (e.g., the sky, trees, buildings, etc.). This process will reduce considerably the computational cost of the rest of stages involved in the road segmentation algorithm because there are less pixels to process. This idea has been widely used in experiments similar to ours [31,81,82]. The Figure 4.1 depicts this idea, the rectangular mask removes the pixels that are out of the expected ROI, which contains the road.

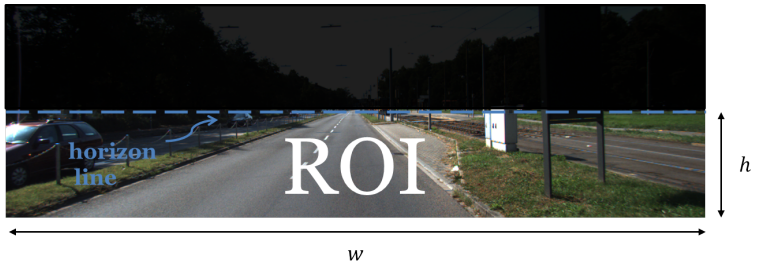


Figure 4.1: Rectangular mask filters out non-road expected pixels and the ROI contains the road. The horizon line is estimated from a set of training images.

Unlike the cases mentioned above, based on the use of an arbitrary mask, we propose obtain the ROI of the road scene employing a more appropriate mask. In our approach, the height h of the ROI depends on the estimation of the horizon line of the urban road scene, while its width w equals the image width (allowing us to work with various lanes).

To estimate the horizon line, a set of training images are employed to detect the vanishing points of the road scene. In particular, we use a locally adaptive soft-voting scheme proposed by Kong *et al.* [36].Then, a value for h is obtained, also leaving a certain margin about the mean height of the vanishing points.

4.1. Vanishing Point on the Horizon Detection

A set of parallel lines in the 3D space do not look like a parallel under the perspective projection caused by a pinhole camera, however these lines converge to some point on the image plane, the so-called *vanishing point* [83]. Consequently, a a straight road segment has a unique vanishing point associated with the dominant orientations of the pixels belonging to the roads whilst a curved road segments induces a set of vanishing points.

In ADAS and autonomous driving vehicles, many computer vision applications rely on knowing the location of the vanish point on a horizon. The horizontal vanish point's location provides important information about driving environments (e.g., instantaneous driving direction of roadway [36,83,84], the search direction of moving objects [16] and sampling regions of the drivable regions' image features [85], etc.). ADAS or autonomous driving cars can exploit such information to detect neighboring moving objects and decide where to drive [86].

Inspired in the work of Kong [36], the approach which we have adopted is based on the texture cue. This vanishing point detection algorithm has two main steps:

1. Confidence-weighted texture orientation estimation at each pixel of the image for which a confidence level is provided (Subsection 4.1.1).

2. Seek the vanishing point of the road by a voting scheme taking into account the confidence level and the distance from the voting pixel to the vanishing point candidate

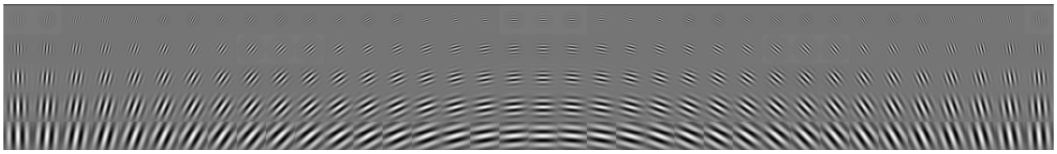
4.1.1. Confidence-Weighted Texture Orientation Estimation

The first step is the estimation of a *texture flow*. The notion of texture flow can be abstracted [87] as an orientation function $\theta(\mathbf{z})$ that defines the the strongest local parallel structure (called *dominant orientation*) at each pixel $\mathbf{z} = (x, y)$.

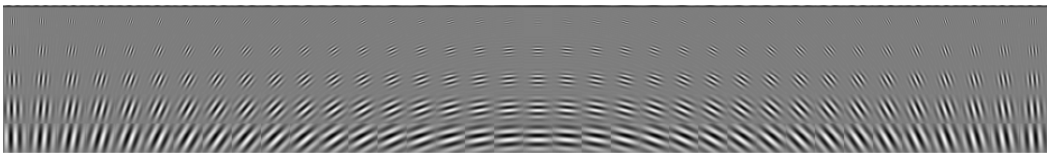
Among the different techniques which can be used for texture orientation estimation, the approach used by Kong relies on bank of 2D Gabor wavelet filters, since they are are known to be accurate [35]. For a wavelet orientation in radians ϕ, a radial frequency in radians per unit length ω (also called *scale*) and odd or even phase, the Gabor wavelet are defined by [88]:

$$\Psi(x, y, \theta, \omega) = \frac{\omega}{\sqrt{2\pi}\kappa} \exp\left\{-\frac{\omega}{8\kappa^2}(4a^2 + b^2)\right\} \left(\exp\{ja\omega\} - \exp\left\{\frac{-\kappa^2}{2}\right\}\right) \tag{4.1}$$

where:



(a) Real part wavelets.



(b) Imaginary part wavelets.

Figure 4.2: Gabor wavelet with 5 scales and 36 orientations. The rows and columns correspond with the different scales and orientations respectively.

$$a = x\cos\phi + y\sin\phi \tag{4.2}$$

$$b = -x\sin\phi + y\cos\phi \tag{4.3}$$

$$\kappa = 2.2 \text{ for a frequency bandwidth of } 1.7 \text{ octaves} \tag{4.4}$$

According with [36, 83], we consider 5 scales, $\omega = \omega_0 \times 2^s$ with $s = \{0, \ldots, 4\}$, on a geometric grid and $A = 36$ orientations for our bank of filters. The Figure 4.2 depicts the Gabor wavelets under the previous constraints.

Let $\mathcal{I}(\mathbf{z})$ a gray scale image, the response of the Gabor filter of scale ω and orientation ϕ is given by Equation (4.7):

$$\mathcal{G}_{\omega,\phi}(\mathbf{z}) = \mathcal{I}(\mathbf{z}) \star \Psi(\mathbf{z}, \theta, \omega) \tag{4.5}$$

The Figure 4.3 shows the response of a Gabor filter of scale ω and orientation ϕ for a gray scale image given.

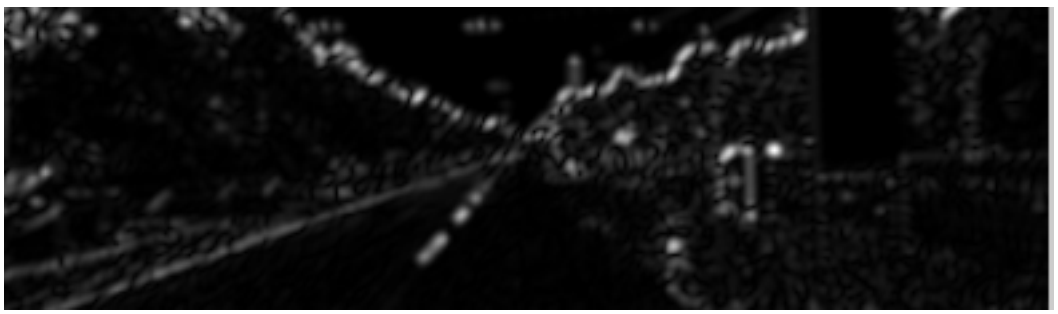


Figure 4.3: Response of a Gabor filter for an input gray image.

The convolution results at pixel $\mathbf{z}$ has two parts: a real and an imaginary component. In order to characterize the local texture properties, we compute the so-called *Gabor energy* for each orientation:

$$E_{\mathcal{G}_{\omega,\phi}} = \Re(\mathcal{G}_{\omega,\phi}(\mathbf{z}))^2 + \Im(\mathcal{G}_{\omega,\phi}(\mathbf{z}))^2 \tag{4.6}$$

Loss Functions

The *response image for an orientation* is defined as the average of the Gabor energies at the different scales:

$$R_\phi(\mathbf{z}) = \mathrm{Avg}_\omega E_{\mathcal{G}_{\omega,\phi}} \tag{4.7}$$

The *texture orientation* $\theta(\mathbf{z})$ of a texture flow at pixel $\mathbf{z}$ is the filter orientation which elicits the maximum response at that location (this average is taken over the 5 scales):

$$\theta(\mathbf{z}) = \arg\max_\phi R_\phi(\mathbf{z}) \tag{4.8}$$

The texture orientation can be efficiently computed by *Fast Fourier Transform (FFT)*. If we apply the well-known convolution theorem (i.e., $\mathcal{F}\{f \star g\} = \mathcal{F}\{f\}\mathcal{F}\{g\}$ to Equation (4.7), we have:

$$\mathcal{F}\{\mathcal{G}_{\omega,\phi}(\mathbf{z})\} = \mathcal{F}\{\mathcal{I}(\mathbf{z})\}\mathcal{F}\{\Psi(\mathbf{z},\theta,\omega)\} \tag{4.9}$$

Thus

$$\mathcal{G}_{\omega,\phi}(\mathbf{z}) = \mathcal{F}^{-1}\{\mathcal{F}\{\mathcal{F}\{\mathcal{I}(\mathbf{z})\}\mathcal{F}\{\Psi(\mathbf{z},\theta,\omega)\}\}\} \tag{4.10}$$

being the Fourier transform of (4.1):

$$\mathcal{F}\{\Psi(x,y,\theta,\omega)\} = \sqrt{8\pi}\frac{\kappa}{\omega}\left(\exp\left\{-\kappa^2\frac{(\alpha-\omega)^2+\beta^2}{2\omega^2}\right\} - \exp\left\{-\kappa^2\frac{\alpha^2+\omega^2+\beta^2}{2\omega^2}\right\}\right) \tag{4.11}$$

with:

$$\alpha = \xi\cos\phi + \nu\sin\phi \tag{4.12}$$

$$\beta = -\xi\sin\phi + \nu\cos\phi \tag{4.13}$$

being ξ and ν the transformed variables from spatial domain (i.e., x and y) to the spatial frequency domain.

Kong et al. observed that the estimated dominant orientation is not reliable at all pixels, especially at those, that are not related with road. Therefore, they propose to use a confident score, which measure how peaky the function $R_\phi(\mathbf{z}$ is near the optimum angle.

Let $r_1(\mathbf{z}), r_2(\mathbf{z}), \ldots, r_A(\mathbf{z})$ the values of the Gabor response for each orientation. Then, we pick up the strongest response of these values:

$$r_{max}(\mathbf{z}) = \max\{r_1(\mathbf{z}), \ldots, r_A(\mathbf{z})\} \tag{4.14}$$

Thus, the confidence for an orientation $\theta(\mathbf{z})$ is given by Equation (4.18):

$$\text{Conf}(\mathbf{z}) = 1 - \frac{\text{Avg}\vartheta}{r_{max}} \tag{4.15}$$

with:

$$\vartheta = \{r_{max-b}(\mathbf{z}), \ldots, r_{max+b}(\mathbf{z})\} \tag{4.16}$$

where b is the coefficient that determines how much weaker the other response are expected to be. In our implementation, $b = \frac{A}{4} - 1 = 8$.

The confidence level is normalized to the range $[0, 1]$, discarding the pixels having a confident level lower than an empirical threshold $T = 0.3$ while the remaining pixels are consider as "voting" pixels.

4.1.2. Locally Adaptive Soft-Voting

The possible vanishing points for an image pixel P located on $\mathbf{z}$ with dominant orientation $\theta(\mathbf{z})$ are all of the points along the line defined by $(P, \theta(\mathbf{z}))$. Let $\alpha(P, V)$ the angle of the line joining an image pixel V and a vanishing point candidate V, the original "hard-voting" strategy of Rasmussen [35] proposes the P votes for V if the difference between $\alpha(P, V)$ and $\theta_{max}(P)$ is within the dominant orientation estimator's angular resolution, which has a finite value of $\frac{2\pi}{A}$. This scheme defines a voting function as follows:

$$\text{Vote (P,V)} = \begin{cases} 1 & \text{if } |\alpha(P, V) - \theta_{max}(P)| \leq \frac{A}{2\pi}, \\ 0 & \text{otherwise} \end{cases} \tag{4.17}$$

Therefore, for a given vanishing point candidate V, we have the following objective function:

$$\text{Votes (V)} = \sum_{P \in R(V)} \text{Vote (P,V)} \tag{4.18}$$

where $R(V)$defines a voting region. Rasmussen sets $R(V)$ to be the entire image, minus edge pixels excluded from convolution by the kernel size, and minus pixels above the current candidate V.

This "hard-voting" strategy tends that the vanishing point candidates higher in the image to have more potential pixels, leading sometimes to large errors in the estimation of the vanishing point. The Figure 4.4 intents to explain this issue.

To address this issue, Kong et al. [36] propose a "soft-voting" scheme where the voting score received by a vanishing point V from a voter P takes into account the distance between P and V. To do this, Kong et al. reduce the region of voting pixels $R(V)$ to the intersection of the Gabor response image with a half-disk below the vanishing centered at V. The radius of the half-disk is $r = 0.35 \times imH$ where imH is the height of the image. Each pixel P inside $R(V)$, for which the texture orientation have been confidently estimated, will vote for the vanishing point

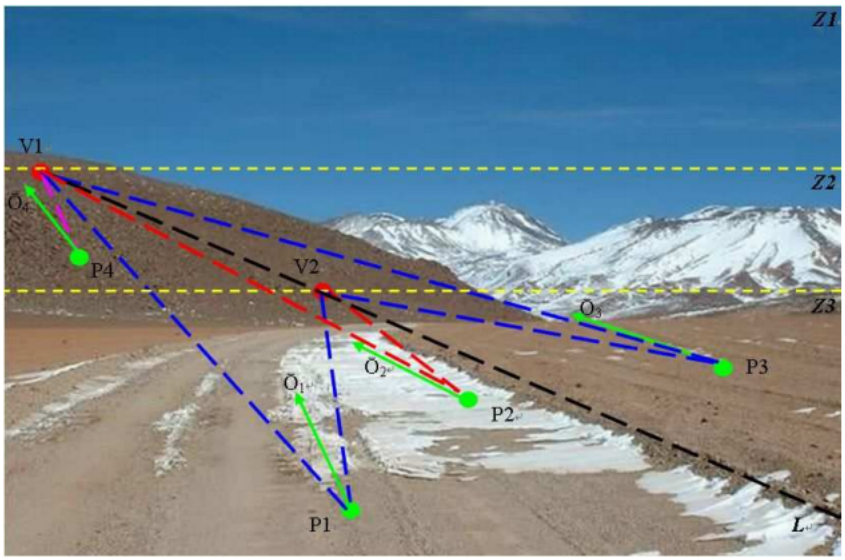


Figure 4.4: Illustration of the problem in vanishing point estimation by conventional voting strategy. $P1$, $P2$, $P3$ and $P4$ are four possible voters. $V1$ and $V2$ are two vanishing point candidates (assuming that $V2$ is the true vanishing point). O_1, O_2, O_3 and O_4 are respectively the texture orientation vectors of the four voters. The two vanishing point candidates divide the whole image region into three zones, denoted as $Z1$, $Z2$ and $Z3$. $Z1$ does not vote for both candidates. Both $Z2$ and $Z3$ potentially vote for $V1$ while $V2$ receives votes only from $Z3$. Therefore, the higher vanishing point candidates tend to receive more votes than the lower candidates. This example is taken from [36]

.

candidate V all the more as P is close to V and the orientation of its texture $\theta_{max}(P)$ coincide with the direction $\alpha(P,V)$. Specifically, if we define the following variables:

- $d(P,V)$ equal to the distance between P and V divided by the diagonal length of the image.
- γ equal to the angle in degrees between the direction and the texture orientation at P (that is, $|\alpha(P,V) - \theta_{max}(P))$

Thus,

$$\text{Vote (P,V)} = \begin{cases} \frac{1}{1+(\gamma d(P,V))^2} & \text{if } \gamma \leq \frac{5}{1+2d(P,V)} \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

As the threshold γ relies on the distance $d(P,V)$, the points of $R(V)$ who are further away to V are taken into account only if the angle γ is very small. In this way, the influence of the pixels at the bottom of the image is much less significant than in the original "hard-voting" scheme.

4.2. Determination of the ROI

Once the theoretical background is known, our approach to define the ROI is really easy. We described this process by using the Algorithm 1

Algorithm 1 Determination of the ROI

Input:
1: $\mathcal{X}$, a set of training images.
2: $\varepsilon > 0$, a certain margin.
Output:
3: h, the height of the ROI.
Algorithm:
4: $n := NumberImages(\mathcal{X})$;
5: **for** $i = 0 \rightarrow n$ **do**
6: $\quad [u_i, v_i] := VanishingPoint(\mathbf{x_i})$; ▷ Localization of the vanishing point in i image
7: **end for**
8: $h := mean(v_i) + \varepsilon$;

CHAPTER 5

CRF MODEL

When we face the resolution of a problem using CRFs, the first question that must be solved is the choice of the most appropriate model. This issue is completely dependent on the nature of the problem. For example, in natural language processing [89, 90] or shallow parsing [91] the most appropriate structure are the linear-chain CRFs. In vision, usually grid-like structure are the most used because are particularly suitable for the lattice of pixels, providing a natural and reasonable representation for images.

According to the order of interactions between variables, CRF models can be classified into:

- Pairwise models.
- Higher-order models.

Higher-order models have been frequently used to model image textures [92, 93], to restore images [94] and to segment texture [95]. However, the lack of efficient algorithms for performing inference in these models has limited their applicability. Traditional inference algorithms such as Belief Propagation are computationally expensive (recent work has been partly successful in improving their performance for certain classes of potential functions in higher order MRFs [92, 96]).

In our approach we made the decision to work with a pairwise CRFs, specifically with a pairwise CRF of grid-like structure. The reasons are twofold:

1. The interaction between pairs of variables is the most common and fundamental type of interactions required to model many vision problems.
2. There are more efficient inference algorithms available.

5.1. Model Description

In the chapter 3 we presented the CRFs as a discriminative model represented by an UGMs in which the nodes represent random variables and the edges represent conditional dependencies between variables. The general structure of our pairwise grid-like CRF is drawn in the Figure 5.1 using FGs, where we can distinguish two types of variables: observed variables and latent (unobserved variables) represented as shaded and unshaded variable nodes respectively. We have opted to use a 4-neighbor system because the increase in accuracy using systems with larger number of neighbors is so small that does not justify the computational efforts

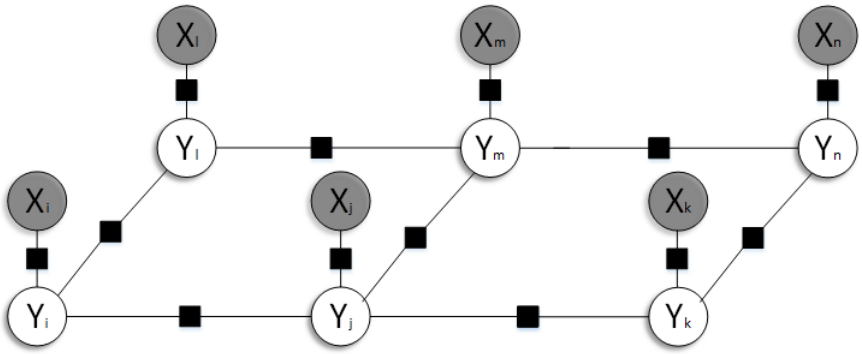


Figure 5.1: Portion of the a pairwise grid-like Conditional Random Field in a 4 neighbor system.

We use a CRF to encode spatial dependencies in the images easily. We align the graphical structure of the CRF with the image's lattice of superpixels in the ROI image of the image. Thus, for each superpixel i in the ROI, we have:

- An observable random variable X_i indicating the value of certain local features like colors, normalized positions or other descriptors that will be exposed in Chapter 8. Really, for a given image $\mathbf{x}$, there is no randomness in the value of the realization x_i.
- A latent random variable Y_i whose realization y_i takes a value from a set of labels $\mathcal{L} = \{0, 1\}$ corresponding to *off-road* and *road* respectively.

Accordingly we can represent the mentioned situation with the draw of Figure 5.2 where the background scene represents the entire set of observed variables $\mathbf{x}$.

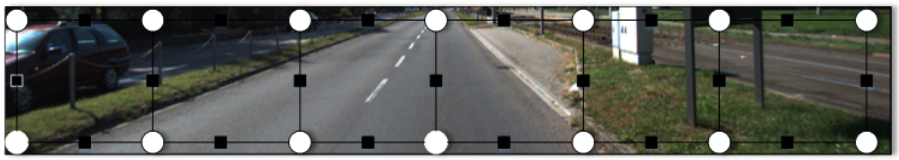


Figure 5.2: Graph of the CRF model aligned with the ROI.

Once the observation variables are known, our aim is to determinate the overall realization of the random variables Y_i. This requires the use of sophisticated inference methods, knows as *message-passing* algorithms, like *Loopy Belief Propagation (LBP)* [51] or *Tree-Reweighted Belief Propagation (TRW)* [79], to infer the optimal configuration for each of the nodes. This kind of

algorithms relies on the exchange of vectors, called *messages* [51], between factor and variables nodes.

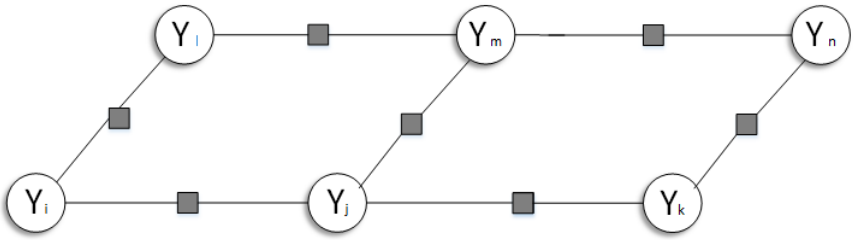


Figure 5.3: Simplified structure of the random field to implement message passing algorithms.

To perform the required message flow, we we need to know certain parameters of the simplified structure of the graph depicted in the Figure 5.3. Note that in this graph we conducted the absorption of the observation nodes by the factor graphs [97], as shown in Figure 5.4.

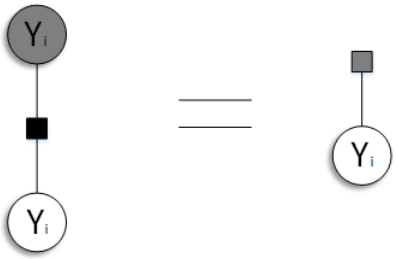


Figure 5.4: Any observed variable node in a factor graph can be absorbed as parameters in the factor nodes that they are connect to, leaving only "hidden" variable nodes and factor nodes that depend on the observations.

5.2. Software Structures Associated with the CRF Model

Our aim is to develop the required software structures that underpin the PGM because they are needed to support the flow of messages in the algorithms. To do this, our starting points are the dimension of the grid (height h and wide w) and the number of states that a node can take ($n_{vals} = 2$). We note that the size of the grid associated with the CRF differs from the original size of the image due to the employ of miniaturized scenes using superpixels.

The first software structure that we need is the grid of the CRF. The grid can be constructed using a a matrix N which elements are associated to nodes in the grid graph. This trivial step is shown in the Algorithm 2. We should like to point out that the notation of algorithms is inspired in the *MatLab* standard notation.

Once the nodes are calculated, we linked them by cliques. Also, we compute the type of each clique using the Algorithm 3 because it will be useful in the Section 8.3.3 to ensure an interdependent parameterization of vertical and horizontal edges.

Finally, we create two structures N_1 and N_2 to do efficiently message passing. In the i row of N_1 we store the number of different cliques where the node i is the first of the link (source),

Algorithm 2 *ComputeGrid*: Construction of the graph.

Input:
w, wide of the grid.
h, height of the grid.
Output:
N, matrix representing the grid of nodes over the lattice of superpixels.
Algorithm:

```
index := 0;
for i = 1 → w do                          ▷ Iterate through the number of rows in the grid
    for j = 1 → h do                      ▷ Iterate through the number of columns in the grid
        N[j][i] := index;
        index := index + 1;
    end for
end for
```

Algorithm 3 *ComputePairs*: Computation of nodes connected by the cliques and the type of each clique.

Input:
N, matrix consisting of grid nodes over the lattice of pixels.
w, wide of the grid.
h, height of the grid.
Output:
Pairs, a matrix consisting of the nodes connected by the cliques.
Pairtype, a list indicating the type of each clique (vertical or horizontal).
Algorithm:

```
n_C = NumberCliques(w, h);                ▷ Compute the number of cliques in the grid
index := 0;
Pairs := Ω_{n_C × 2};
for i = 0 → w − 1 do          ▷ Iterate through the number of rows and columns in the grid.
    for j = 0 → h − 1 do                                          ▷ Firts vertical edges.
        Pairs[index][0] := N[j, i];
        Pairs[index][1] := N[j + 1, i];
        Pairtype[index] := VERTICAL;
        index := index + 1;
        if i < w then                                                  ▷ Horizontal edges
            Pairs[index][0] := N[j, i];
            Pairs[index][1] := N[j, i + 1];
            Pairtype[index] := HORIZONTAL;
            index := index + 1;
        end if
    end for
end for
```

similarly for N_2 (second place, sink). In both cases, a negative value indicates empty value. The reason for creating the N_1 and N_2 structures is that when we are doing message passing, we will often want to iterate through all the pairs that touch a given variable, and it would be expensive to go hunting through all of *Pairs* variables to find them. This is developed in the Algorithm 4.

Algorithm 4 *FindNode2PairArray*: Creates two structures to do message passing in the factor graph.

Input:

N, matrix consisting of grid nodes over the lattice of pixels.
w, wide of the grid.
h, height of the grid.

Output:

N_1, a matrix indicating in that cliques the node i is the first of the link.
N_2, a matrix indicating in that cliques the node i is the last of the link.

Algorithm:

$N_{Max} := MaximumDegree(N)$; ▷ Number of neighbors in the node with more neighbors
$n_n := NumberNodes(N)$;
$N_1 := (-1) \times J_{n_n \times N_{Max}}$;
$N_2 := (-1) \times J_{n_n \times N_{Max}}$;
$index1 := (-1) \times J_{n_n \times 1}$; ▷ Auxiliary list
$index2 := (-1) \times J_{n_n \times 1}$;
$n_C = NumberCliques(w, h)$; ▷ Compute the number of cliques in the grid
for $c = 0 \rightarrow n_C$ **do**;
    $i := N[c][0]$;
    $j := N[c][1]$;
    $index1[i] := index1[i] + 1$;
    $index2[j] := index2[j] + 1$;
end for

All algorithms described in this chapter are fully programmed in language C++ with some code developed using the algebra libraries *eigenweb*.

CHAPTER 6

INFERENCE

Once a factor graph model has been fully specified and parameterized, there are two tasks left to do [51, 98]:

- *Model learning*: Using a supervised learning approach we will adjust the parameters $\boldsymbol{w}$ of the CRF model. Our starting point is a set of training data generated by an unknown distribution, so that the correct choice of the parameters, allows us to better represent the true distribution The meaning of"better", of course, must be made precise.
- Use the model for solving *inference* task on future data instances that we could describe as is dedicated to the operations which we perform using the graphical model.

This two basic stages they can be also referred to as training and testing stages from a classification point of view.In both stages, because we are using a supervised training, a inference process is necessary:

- Utilization of the inference in the training process, choosing the model parameters best suited to the true distribution of data.
- Utilization of inference process to predict a maximum a posteriori realization.

In this chapter we provide a framework to tackle the inference task, leaving the training task to the next chapter. However, we should not forget that both processes were dependents as illustrates the Figure 6.1. This chapter is based mainly on the work of Wainwright *et al.* developed in [99] and [100].

6.1. Maximum Posterior Marginal Inference

Suppose we have some distribution $p(\mathbf{y}|\mathbf{x};\mathbf{w})$, we are given the observation $\mathbf{x}$, and we need to guess the single output vector $\mathbf{y}^*$. The graphical model will be used in a problem-dependent way to make the choice with the best expected outcome.

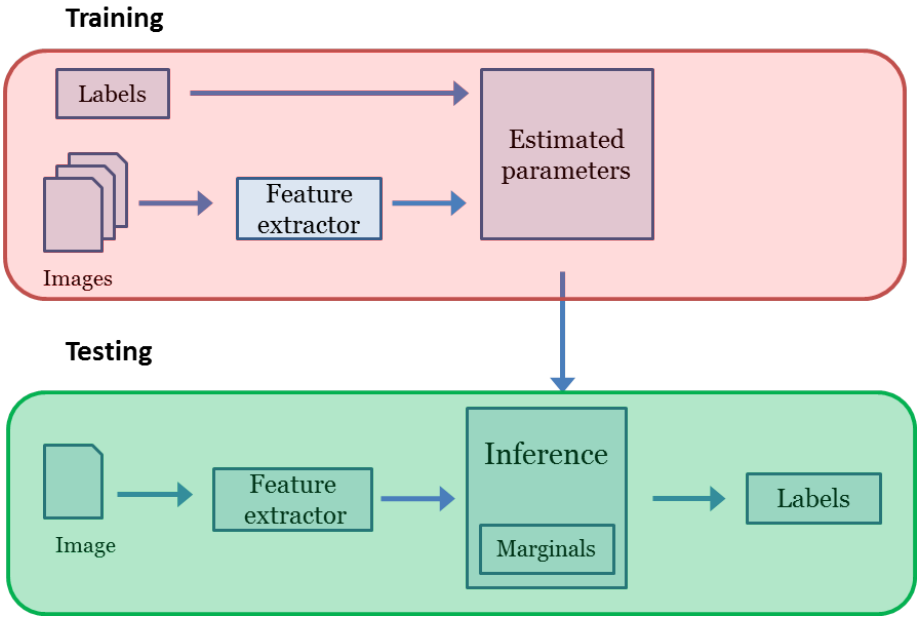


Figure 6.1: Pipelines corresponding to training and inference.

However, the answer depends on the meaning of "best". For this reason, is expedient to define some utility function $\mathcal{U}(\mathbf{y}, \mathbf{y}')$ [98, 101] that measure the satisfaction to have predicted $\mathbf{y}$ if the real output is $\mathbf{y}'$. Then, we look for the best solution to the next problem:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{\mathbf{y}'} p(\mathbf{y}'|\mathbf{x}; \mathbf{w}) \mathcal{U}(\mathbf{y}, \mathbf{y}') \tag{6.1}$$

Particularizing the utility function to the indicator function $\mathcal{U}(\mathbf{y}', \mathbf{y}) = [[\mathbf{y} = \mathbf{y}']]$, where the evaluation of the function $[[\cdot]]$ is 1 if its argument is true (otherwise is 0), we transform the Equation (6.1) into the Equation (6.2):

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}; \mathbf{w}) \tag{6.2}$$

The expression obtained in (6.2) for the estimated output is knows as *Maximum A Posteriori (MAP)* estimation, a very popular technique [70, 102]. However, is not well adapted to pixel-wise image segmentation because the utility function assigns zero to every labeling that is not pixel-by-pixel identical to the intended one, irrespective of the number of components of the output vector that are corrected.

An alternative utility function is based on the Hamming distance [103] between $\mathbf{y}$ and $\mathbf{y}'$, being proportional to the number of well classified pixels. Let $U(\mathbf{y}', \mathbf{y}) = \sum_i [[y_i = y_i']]$, then Equation (6.1) gives:

$$y_i^* = \arg\max_{y_i} p(y_i|\mathbf{x}; \mathbf{w}) \tag{6.3}$$

The estimation gives by Equation (6.3) is knows as *Maximum Posterior Marginals (MPM)* [104]. The main difference between MAP and MPM is that MAP inference pursues

that the joint vector $\mathbf{y}^*$ has maximum probability whereas MPM does this separately for each variable. In other words, if one cares about the number of variables in error, as opposed to if all variables are simultaneously correct or not, MPM is to be preferred to MAP inference [98].

The MPM inference consists of two stages:

1. Computing the marginals:

$$p(y_i^*|\mathbf{x};\mathbf{w}) = \sum_{\mathbf{y}:y_i=y_i^*} p(\mathbf{y}|\mathbf{x};\mathbf{w}) \tag{6.4}$$

2. Choosing the value with maximum marginal probability for each index i.

In computational terms, the first stage is more complex and problematic because requires a sum over all vectors $\mathbf{y} : y_i = y_i^*$ while the maximization of the second stage is trivial.

It is therefore clear that our objective must be the computation of marginals, both for its own sake, and for enabling MPM inference.

For general factor graphs inference is known to be *Non-deterministic Polynomial-time (NP)* hard [105], while for graphs that do not contain cycles the problem can be solved efficiently. In the case of graphs with cycles various alternatives have been proposed to obtain approximate answers. These approximated inference methods can be divided into two groups:

1. Deterministic approximations, which are solved exactly.
2. *Monte Carlo* based approximations.

6.2. Exponential Family

In this section, we describe the *exponential family*, a broad class of distributions that have been extensively studied in the statistics literature [106].

A probability distribution in the exponential family can be defined by:

$$p(\mathbf{x};\boldsymbol{w}) = \exp(\boldsymbol{w}^{\boldsymbol{T}} \cdot \mathbf{f}(\mathbf{x}) - A(\boldsymbol{w})) \tag{6.5}$$

where $\mathbf{f}(\mathbf{x})$ is some vector of "sufficient statistics" (that is that essentially, the elements of $\mathbf{f}$ can be arbitrary features of $\mathbf{x}$), $\boldsymbol{w}$ is a vector of parameters and the log-partition function gives for (6.6) ensures normalization:

$$A(\boldsymbol{w}) = \log \sum_{\mathbf{x}} \exp\left(\boldsymbol{w}^T \cdot f(\mathbf{x})\right) \tag{6.6}$$

It can be shown that the first and second order derivatives of $A(\boldsymbol{w})$ correspond to the expected value, and covariance matrix of f.

An important consequence of Hammersley-Clifford theorem [67] provides that the entire collection of UGMs associated with a given graph constitutes an exponential family [9,99,100], with sufficient statistics $\mathbf{f}(\mathbf{x})$ consisting of indicator functions for each possible configuration of each clique c[1], namely $[[\mathbf{X_c} = \mathbf{x_c}\ \forall c \in \mathcal{C}]]$.

Any member of the family is specified by an exponential parameter, the elements of which are weights for potential functions defined on the graph cliques. In this way, each vector $\boldsymbol{w}$ indexes a particular UGM in this exponential family.

Now, we will demonstrate that the first derivative of $A(\boldsymbol{w})$ correspond to the expected value of $\mathbf{f}(\mathbf{x})$:

$$\frac{dA(\boldsymbol{w})}{d\boldsymbol{w}} = \frac{\sum_{\mathbf{x}} \exp\left(\boldsymbol{w}^T \cdot \mathbf{f}(\mathbf{x})\right)\mathbf{f}(\mathbf{x}))}{\exp\left(\boldsymbol{w}^T \cdot f(\mathbf{x})\right)} = \sum_{\mathbf{x}} p(\mathbf{x};\boldsymbol{w})\mathbf{f}(\mathbf{x}) = \boldsymbol{\mu}(\boldsymbol{w}) \tag{6.7}$$

For an exponential family corresponding to a UGM, computing μ is equivalent to computing all marginals probabilities. We can demonstrate this sentence substituting indicator functions in the Equation(6.7). Thus:

$$\boldsymbol{\mu}(\mathbf{x_c};\boldsymbol{w}) = \sum_{\mathbf{x}} p(X_c;\boldsymbol{w})[[X_c = \mathbf{x_C}]] = p(\mathbf{x_c};\boldsymbol{w}) \tag{6.8}$$

As the size increases, the Equation (6.7) becomes more intractable. That is why we are seeking approximate methods for computing marginals.

6.3. Variational Inference

The variational methods deterministic approximations, assuming the posterior probability (given the evidence) factorizes in a particular way. We will begin computing the second derivative of $A(\boldsymbol{w})$ that will be correspond to the covariance matrix of $\mathbf{f}(\mathbf{x})$:

$$\begin{aligned}\frac{d^2A(\boldsymbol{w})}{d\boldsymbol{w}d\boldsymbol{w}^T} = \sum_{\mathbf{x}} \mathbf{f}(\mathbf{x})\frac{d}{d\boldsymbol{w}^T}p(\mathbf{x},\mathbf{w}) = \sum_{\mathbf{x}} \mathbf{f}(\mathbf{x})p(\mathbf{x};\boldsymbol{w})(\mathbf{f}(\mathbf{x})^T - \boldsymbol{\mu}^T) = \\ = \sum_{\mathbf{x}} p(\mathbf{x};\boldsymbol{w})\mathbf{f}(\mathbf{x}) \cdot \mathbf{f^T}(\mathbf{x}) - \boldsymbol{\mu} \cdot \boldsymbol{\mu}^T = COV_w[\mathbf{f}(\mathbf{x})]\end{aligned} \tag{6.9}$$

We can note that Equation (6.9) implies that the log-partition is convex in $\boldsymbol{w}$ since the covariance matrix must be positive semidefinite. Hence, it is possible to write $A(\boldsymbol{w})$ in terms of its conjugate (or Legendre) dual [107] function as:

$$A(\boldsymbol{w}) = \sup_{\boldsymbol{\mu}}\{\boldsymbol{w}^T\boldsymbol{\mu} - A^*(\boldsymbol{\mu})\} \tag{6.10}$$

[1]For clarity, we consider nodes as cliques of order 1.

The connection to approximate inference and entropy approximations stems from the dual function. It can be shown that A^* is the negative entropy of p, when the mean parameters $\boldsymbol{\mu}$ are *achievable*. The set of achievable mean parameters is so called *marginal polytope* [100] and is given by the expression:

$$\mathfrak{M} = \{\boldsymbol{\mu} : \exists\, \boldsymbol{w}, \boldsymbol{\mu} = \sum_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{w}) \mathbf{f}(\mathbf{x})\} \tag{6.11}$$

The entropy is defined by:

$$H(\boldsymbol{\mu}) = -\sum_{\mathbf{x}} p(\mathbf{x}; \boldsymbol{\mu}) \log p(\mathbf{x}; \boldsymbol{\mu}) \tag{6.12}$$

So finally,

$$A^*(\boldsymbol{\mu}) = \begin{cases} -H(\boldsymbol{\mu}) & \text{if } \boldsymbol{\mu} \in \mathfrak{M} \\ \infty & \text{if } \boldsymbol{\mu} \notin \mathfrak{M} \end{cases} \tag{6.13}$$

Since $A^*(\boldsymbol{\mu})$ is convex, $H(\boldsymbol{\mu})$ must be concave. By substituting Equation (6.13) into Equation (6.12), we can write the partition function in terms of a constrained optimization:

$$A(\boldsymbol{w}) = \sup_{\boldsymbol{\mu} \in \mathfrak{M}} \{\boldsymbol{w}^T \boldsymbol{\mu} + H(\boldsymbol{\mu})\} \tag{6.14}$$

Returning to the issue of the Equation (6.7), we can obtain the derivative of $A(\boldsymbol{w})$ using 6.14.

$$\frac{dA(\boldsymbol{w})}{d\boldsymbol{w}} = \frac{d}{d\boldsymbol{w}} \sup_{\boldsymbol{\mu} \in \mathfrak{M}} \{\boldsymbol{w}^T \boldsymbol{\mu} + H(\boldsymbol{\mu})\} \tag{6.15}$$

Applying Danskin's theorem [108] to (6.15) yields that:

$$\frac{dA(\boldsymbol{w})}{d\boldsymbol{w}} = \arg\max_{\boldsymbol{\mu} \in \mathfrak{M}} \{\boldsymbol{w}^T \boldsymbol{\mu} + H(\boldsymbol{\mu})\} A \tag{6.16}$$

We pose the inference problem in UGM in terms of the exponential families. TIn treelike graphs this optimization can be solved efficiently. However, in general graphs, the computational problems still continue to exist:

- $\mathfrak{M}$ will be difficult to characterize.
- The entropy is in general intractable (and not even defined for $\boldsymbol{\mu} \notin \mathfrak{M}$).

To derive a tractable algorithm we can take some approximate entropy $\widetilde{H}$, and an approximate marginal polytope $\widetilde{\mathfrak{M}}$. In this way, is natural to define an approximate partition function, according to Equation (6.17).

$$\widetilde{A}(\boldsymbol{w}) = \sup_{\boldsymbol{\mu} \in \widetilde{\mathfrak{M}}} \{\boldsymbol{w}^T \boldsymbol{\mu} + \widetilde{H}(\boldsymbol{\mu})\} \tag{6.17}$$

The approximate marginals yields:

$$\widetilde{\boldsymbol{\mu}} = \frac{d\widetilde{A}}{d\boldsymbol{w}} = \arg\max_{\boldsymbol{\mu}\in\widetilde{\mathfrak{M}}}\{\boldsymbol{w}^T\boldsymbol{\mu} + \widetilde{H}(\boldsymbol{\mu})\} \tag{6.18}$$

This close relationship between the log-partition function and marginals is heavily used in the derivation of approximate marginalization algorithms. To compute approximate marginals, we follow the next steps:

1. Derive an approximate version of the optimization.
2. Take the exact gradient of this approximate partition function.

6.4. Tree-Reweighted Belief Propagation

6.4.1. Theoretical Basis

A variety of algorithms exist that help to calculate approximate marginals such as *mean field* [66], *Loopy Belief Propagation* [51] or TRW [99]. We have chosen the TRW due to its good results in some works similar to ours [109].

The key ideas behind TRW to approximate the variational principle are [9]:

- To replace the marginal polytope $\mathfrak{M}$ with a superset $\mathfrak{L} \supset \mathfrak{M}$ that only ensures local constraints of the marginals. As a logical consequence, the value of the approximate log-partition is increased.
- Approximate the entropy with a tractable upper bound $\widetilde{H}(\boldsymbol{\mu})$.

These two approximations, taken together, lead to an upper bound of $A(\boldsymbol{w})$. In this way, the optimization problem is raised by the following optimization problem:

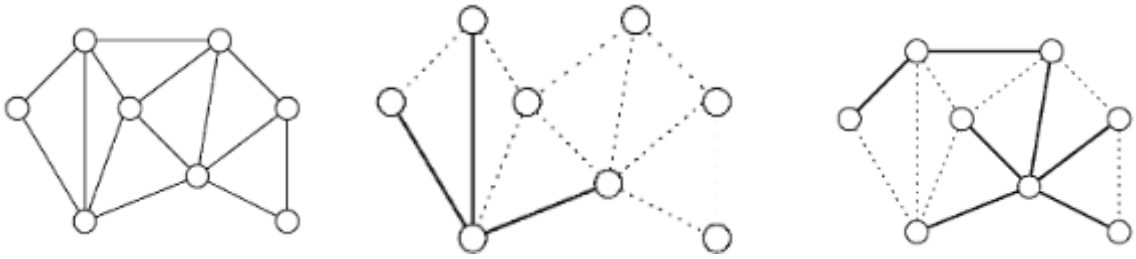
$$\widetilde{A}(\boldsymbol{w}) = \sup_{\boldsymbol{\mu}\in\mathfrak{L}}\{\boldsymbol{w}^T\boldsymbol{\mu} + \widetilde{H}(\boldsymbol{\mu})\} \tag{6.19}$$

Thus, the approximate marginals yields:

$$\widetilde{\boldsymbol{\mu}} = \arg\max_{\boldsymbol{\mu}\in\widetilde{\mathfrak{L}}}\{\boldsymbol{w}^T\boldsymbol{\mu} + \widetilde{H}(\boldsymbol{\mu})\} \tag{6.20}$$

In [99] we find the following restrictions to be verified by the polytope $\mathfrak{L}$:

$$\mathfrak{L} = \{\boldsymbol{\mu} : \boldsymbol{\mu}(\mathbf{x}_c) \geq 0, \sum_{\mathbf{x}_{c\setminus i}} \boldsymbol{\mu}(\mathbf{x}_c) = \boldsymbol{\mu}(x_i)\,, \sum_{x_i} \boldsymbol{\mu}(x_i) = 1\} \tag{6.21}$$



(a) Graph with cycles. (b) A tree is a cycle-free subgraph (c) A spanning tree reaches every vertex of the graph

Figure 6.2: Differences between graph, tree and spanning tree.

Its is obvious that any valid marginal vector can meet the constraints in (6.21). Therefore, $\mathfrak{L}$ may contains unrealizable vector. Thus, the marginal vector returned by TRW could be inconsistent in the sense that no joint distribution yields into marginals.

Furthermore, the expression of the entropy approximation used by TRW is given by Equation (6.22):

$$\widetilde{H} = \sum_i H(\mu_i) - \sum_{c \in \mathcal{C}} \rho_c I(\mu_c) \tag{6.22}$$

where $H(\mu_i)$ is the *univariate entropy* associated with variable i and $I(\mu_c)$ is the mutual information corresponding to the variables in the clique c given by Equation (6.23) and Equation (6.24) respectively.

$$H(\mu_i) = -\sum_{x_i} \mu(x_i) \log(\mu(x_i)) \tag{6.23}$$

$$I(\mu_c) = -\sum_{\mathbf{x_c}} \mu(\mathbf{x}_c) \log \frac{\mu(\mathbf{x}_c)}{\prod_i \mu(x_i)} \tag{6.24}$$

In the Equation (6.22) the terms ρ_c are called *edge appearance probabilities*. It is a probability distribution over the set of *spanning trees* of the graph. Before explaining the meaning of $\boldsymbol{\rho}$, we need to clarify the meaning of *tree* and *spanning tree*. In brief, given a graph $G = (\mathcal{V}, \mathcal{E})$ we define:

- *Tree*: Is a cycle-free graph consisting of a single connected component (see Figure 6.2b).
- *Spanning tree*: Is a tree that reaches every vertice in $\mathcal{V}$ (see Figure 6.2c).

Now, we define $\boldsymbol{\rho}$ as a probability distribution over the set of spanning trees. Let the edge $(s,t) \in \mathcal{E}$, the element ρ_{s-t} of the vector $\boldsymbol{\rho}$ can be interpreted as the probability that edge $(s,t) \in \mathcal{E}$ appears in a spanning tree chosen randomly. The Figure 6.3 illustrate this concept.

In [99] is demonstrated that, under appropriate circumstances, defining $\rho_c = p(c \in G)$, the Equation (6.22) gives an upper bound on the true entropy:

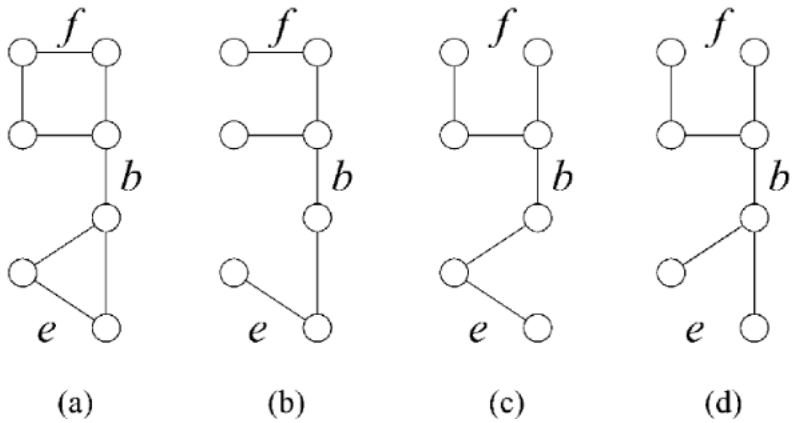


Figure 6.3: Original graph is shown in (a). Probability $\frac{1}{3}$ is assigned to each of the three spanning trees in (b), (c) and (d). Edge b is a so-called *bridge* in G, meaning that it must appear in any spanning tree. Therefore, it has edge appearance probability $\rho_b = 1$. Edges e and f appear in two and one of the spanning trees, respectively, which gives rise to edge appearance probabilities $\rho_e = \frac{2}{3}$ and $\rho_f = \frac{1}{3}$ respectively. Taken from [99].

$$\widetilde{H}(\boldsymbol{\mu}) \geq H(\boldsymbol{\mu}) \tag{6.25}$$

As a direct result of Equation (6.25), we have:

$$\widetilde{A}(\boldsymbol{w}) \geq A(\boldsymbol{w}) \tag{6.26}$$

6.4.2. Computation of Approximate Marginals

The results of the previous subsection allows us to obtain an algorithm to compute approximate marginals. The way the algorithm works is by computing vectors termed *messages* for each edge in the FG with one element for each state as we described previously in Section 3.3.1.

In our approach we have a pairwise CRF of grid-like structure. As this graphical model is not a tree, we have to calculate its spanning tree. This idea is depicted in Figure 6.4.

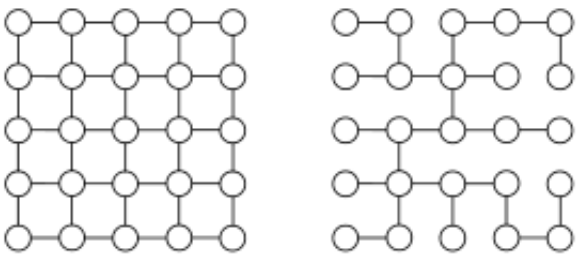


Figure 6.4: Pairwise grid like CRF and one example of spanning tree associated with it.

The main formulation of the message-passing approach [99] is reproduced here for clarity. At each iteration, every node i of the spanning tree sends a message $m_C(y_i)$ to its neighbor $\mathcal{N}_i$ in the clique. Then, the message passing update is as follows:

$$m_{i\to j}(y_j) \propto \sum_i \psi_i(y_i, \mathbf{x}) \cdot \psi_{ij}^{\frac{1}{\rho_{ij}}}(y_i, y_j, \mathbf{x}) \cdot \frac{1}{m_{i\to j}^{1-\rho_{ij}}(y_j)} \prod_{n\in\mathcal{N}_i\setminus j} m_{n\to i}^{\rho_{ni}}(y_i) \tag{6.27}$$

where $\propto$ means assigned after normalization.

After the messages have converged, each node can form an estimate of its local approximate marginal defined as:

$$\mu_i(y_i) \propto \psi_i(y_i) \prod_{n\in\mathcal{N}_i} m_{n\to i}^{\rho_{ni}}(y_i) \tag{6.28}$$

We have opted for a simplified approach based on TRW, *Uniformly Reweighted Belief Propagation* [110] assigning a constant value appearance probability to all edges, thus$\rho_{ij} = \rho \; \forall i, j$. It reduces the computational complexity being also an optimal choice for our graph structure. Besides, this simplified scheme turns out to outperform Loopy Belief Propagation in graphs with cycles. Also, note that in the special case of $\rho_{ij} = 1$, this message passing algorithm simplifies into Loopy Belief Propagation.

Additionally, it must be noted that according to [99], the optimal value of ρ for graphs, satisfying certain symmetry conditions as in our case, can be approximated with the number of vertices ($|\mathcal{V}|$) and edges ($|\mathcal{E}|$) using Equation (10.7).

$$\rho^* \approx \frac{|\mathcal{V}| - 1}{|\mathcal{E}|} \tag{6.29}$$

In general, it can be stated that this parameter tends to 0.5 for bigger grid graphs. However, this is not necessarily the optimal choice. In fact, when using a grid, 0.5 is the largest number that leads to a convex inference problem [111].

In order to facilitate the calculations, the Equation (6.27) can be broken down into *variable to factor message* and *variable to factor messages*. This idea is depicted in Figure 6.5.

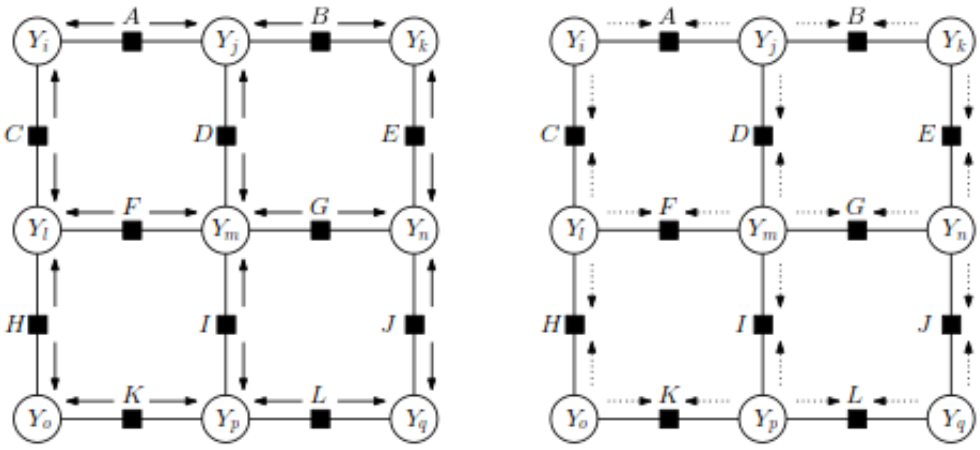


Figure 6.5: Flow of messages between variable nodes to factor nodes and vice versa (from [51].

Once we have the messages, the approximate marginals can be easily calculated using Equation (6.28). This yields the following vector for each node:

$$\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{n_{vals}}) \tag{6.30}$$

where the i component of the vector correspond to the approximate marginal obtained assuming that the correct label was i. Finally, to set the values of the labels in each node is as easy as take the i label which maximizes μ_i

With the aim of providing a fast and efficient approach for road detection, this inference algorithm has been implemented in C++. We have opted to use the library *Eigen 3.2* [112], a C++ template library for linear algebra (matrices, vectors, numerical solvers, and related algorithms) to ensure compatibility between different hardware platforms.

In order to carry out the algorithms is necessary two matrices previously obtained by training:

- p_F: determines the univariate factors. Specifically, the vector of log-potentials for node i is given by multiplying p_F with the features for node i.
- p_G: determines the pairwise factors. The vector of log-potentials for clique (i, j) is given by multiplying p_G with the features for edge (i, j).

The computation of inwards and outwards messages according to Equation (6.27) is developed in the Algorithm 5.

Algorithm 5 Messages update using TRW

Input:

p_F, matrix to compute node potential.
p_G, matrix to compute edge potentials.
F, node features.
G, edge features;
ρ edge appearance probability.
N, matrix consisting of grid nodes over the lattice of superpixels.
N_1, a matrix indicating in that cliques the node i is the first of the link
N_2, a matrix indicating in that cliques the node i is the first of the link w, wide of the grid.
h, height of the grid.
n_v, number of class in the dataset - in this case only two ("road", "off-road").
n_n, number of nodes in the graph.
N, matrix consisting of grid nodes over the lattice of pixels
N_1, a list of which variable appears first in each pair.
N_2, a list of which variable appears second in each pair.

Output:

y_i, the label associated with each node.

Algorithm:

$\phi_i := p_F \cdot F; \phi_{ij} = p_G \cdot G$ ▷ Log-linear potentials
$\psi_i := \exp(\phi_i); \psi_{ij} := \exp(\phi_{ij})$;
$n_n := NumberNodes(N)$;
$n_C := NumberCliques(w, h)$;
$n := J_{n_v \times n_n}$;
$m_1 := J_{n_v \times n_C}$; ▷ Messages to first variable in clique
$m_2 := J_{n_v \times n_C}$; ▷ Messages to second variable in clique
for $i = 0 \rightarrow MAX\_ITERATIONS$ **do**
    for $j = 0 \rightarrow n_C$ **do**
        $[m_1, m_2] := trw(\psi_i, \psi_{ij}, N1, N2, n_v, \rho)$; ▷ Implements Eq. 6.27
    end for
end for
for $i = 0 \rightarrow n_n$ **do**
    $\mu_i := approximate\_marginals(m_1, m_2, \psi_i, \psi_{ij})$; ▷ Implements Eq. 6.28
    $y_i := maximum\_posteriori(\mu_i)$; ▷ This function assigns the MPM label to node.
end for

CHAPTER 7

LEARNING

On top of inference, another task of great importance in our case is *learning*, which aims to select the optimal model for the task at hand [52]. The notion of learning in CRFs is slightly ambiguous because each part of a CRF (i.e., random variables, factors and parameters) can in principle be learned from data [51]. However, in this book, as in the most computer vision applications, the model structure and parametrization are manually specified and learning amounts to finding a vector of real-valued parameters $\boldsymbol{w}$.

In our approach we have considered *supervised learning* to select the optimal model from its feasible set based on *training data*. The alternative is to attempt to select the parameters $\boldsymbol{w}^*$ by hand or by experiment; this is known to be difficult, and quite impractical if the dimensionality of $\boldsymbol{w}$ is at all large [68]. It also proves to be a challenging problem.

Using supervised learning, the input is a set of P training samples $\{(\mathbf{x}_p, \mathbf{y}_p)\}_{p=1,\dots,P}$ where $\mathbf{x}_p$ and $\mathbf{y}_p$ represent the observed data and the ground truth configuration of labels of the p sample, respectively. We note that the $\mathbf{y}$ values are known for each $\mathbf{x}$ in the training dataset, become observed variables in the context of training. Moreover, we assumed that the node and edge potentials of the p training data instance can be expressed as a linear combination of feature vectors extracted from observed data according to Equation (8.5) that we repeat here for convenience:

$$\begin{cases} \psi_i(y_v, \mathbf{x}) = \exp(\boldsymbol{w_i}^T \cdot \mathbf{F}(y_v, \mathbf{x})) & \text{(7.1a)} \\ \psi_{\mathcal{C}}(y_u, y_v, \mathbf{x}) = \exp(\boldsymbol{w_{\mathcal{C}}}^T \cdot \mathbf{G}(y_u, y_v, \mathbf{x})) & \text{(7.1b)} \end{cases}$$

Therefore, the problem of learning boils down to estimating the vector $\boldsymbol{w}$ using as input the above training data. In this way, if $d(\mathbf{y}|\mathbf{x})$ is the true conditional distribution of labels in the training data, learning is the task of finding the vector $\boldsymbol{w}^*$ that makes $p(\mathbf{y}|\mathbf{x}; \boldsymbol{w})$ closest to $d(\mathbf{y}|\mathbf{x})$.

7.1. Learning as Minimization of Empirical Risk

Given the input domain $\mathcal{X}$ and the output domain $\mathcal{Y}$, we would to learn a function (called *hypothesis*) $h : \mathcal{X} \to \mathcal{Y}$ which outputs an object $\mathbf{y}$ assigning the correct class at each pixel given $\mathbf{x}$. Once we have the hypothesis, the predictions are extremely easy to obtain: $\mathbf{y}' = h(\mathbf{x})$.

The quality of the prediction is evaluated using a loss function [51] $\Delta(\mathbf{y}, \mathbf{y}'; \boldsymbol{w})$ measuring the similarity between the ground-truth training labels $\mathbf{y}$ and the estimated labels $\mathbf{y}$ obtained once the inference step has finished. Specifically, we take the goal of learning to be to minimize the empirical risk:

$$R(\boldsymbol{w}) = \frac{1}{N} \sum_{\mathbf{y}} \Delta(\mathbf{y}, \mathbf{y}'; \boldsymbol{w}) \tag{7.2}$$

where the summation is over the all the samples in the dataset.

Therefore, the learning formulation can be approached as follows:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i} \Delta(\mathbf{y}_i, \mathbf{y}'_i; \boldsymbol{w}) \tag{7.3}$$

In this formulation the choice of parameters $\boldsymbol{w}$ influences the loss function through $\mathbf{y}'$. As the parameters change, the estimate $\mathbf{y}'$ also changes. This approach adds flexibility by separating the learning criterion from the structure of the model.

The general strategy for implementing the minimization of Equation (7.3) is to compute the gradient $\frac{\partial \Delta}{\partial \boldsymbol{w}}$ for using with a gradient-based optimization algorithm such as gradient descent, Newton's method, etc. Using the chain rule, the gradient can be broken into two parts:

$$\frac{\partial \Delta}{\partial \boldsymbol{w}} = \frac{\partial \mathbf{y}'^T}{\partial \boldsymbol{w}} \frac{\partial \Delta}{\partial \mathbf{y}'} \tag{7.4}$$

where computing the gradient $\frac{\partial \mathbf{y}'}{\partial w}$ is the difficult step, since $\mathbf{y}'$ relates to w through an argument of the minimum operation.

7.1.1. Loss functions

The choice of the loss function is important because, on the one hand, it influences the accuracy, and on the other hand, the simpler the loss the easier the gradient is to compute. A typical choice for the loss function is the *likelihood*, with:

$$\Delta(\mathbf{x}, \boldsymbol{w}) = \log p(\mathbf{x}; \boldsymbol{w}) = \boldsymbol{w}^T \cdot f(\mathbf{x}) - A(\boldsymbol{w}) \tag{7.5}$$

Since we are working with CRF, we have the conditional likelihood loss:

$$\Delta(\mathbf{y}|\mathbf{x}; \boldsymbol{w}) = \log p(\mathbf{y}|\mathbf{x}; \boldsymbol{w}) \tag{7.6}$$

The gradient of (7.6) can be obtained using the Equation (6.7):

$$\frac{\partial \Delta}{\partial \boldsymbol{w}} = \mathbf{f}(\mathbf{y}|\mathbf{x}) - \boldsymbol{\mu}(\boldsymbol{w}) \tag{7.7}$$

However, the likelihood presents two issues [9]:

1. If the graphical model is not tree structured, exact inference is computationally intractable. As consequence, we have to use approximate algorithms.

2. Usually models are mis-specified, either because parametrization is extremely simple or because conditional independent asserted by the graph are not exactly true. These approximations might be made out of ignorance, due to a lack of knowledge about the domain being studied, or deliberately because the true model might have too many degrees of freedom to be fit with available data

For these reasons, various loss function have been proposed [9, 51, 113]. Given the difficulties listed above to calculate true marginals, these functions are defined with respect to the marginal predictions of some approximate algorithm (μ) instead of the true marginals (p). The remainder of the section reviews some loss functions (a more comprehensive review can be found at [98]).

7.1.1.1. Univariate Logistic Loss

The *univariate conditional logistic loss* was proposed by Kakade *et al.* [113] who also provide an algorithm for calculating the gradient for models with exact inference and linear features.

$$\Delta = -\sum_i \log \mu(y_i|\mathbf{x}; \boldsymbol{w}) \tag{7.8}$$

The idea behind univariate loss functions is to fit the model only to predict univariate marginals well [98]. This approach makes sense if one can "trade" joint accuracy for marginal accuracy.

7.1.1.2. Univariate Conditional Quadratic Loss

The *univariate conditional quadratic loss* [114] tries to minimize the expected squared difference of approximate marginal probabilities.

$$\Delta = -\sum_i (-2\mu_i(y_i|\mathbf{x}; \boldsymbol{w}) + \sum_{y_i} \mu_i(y_i|x)^2) \tag{7.9}$$

7.1.1.3. Clique Logistic Loss

Univariate loss functions are not consistent, simple examples show cases where a model predicts perfect univariate marginals, despite the joint distribution being very inaccurate.

A *clique loss* is similar to a univariate loss but is evaluated on the cliques. This ensures the consistency of clique loss because CRFs can be seen as a member of the exponential family with indicator functions on cliques as sufficient statistics, if all clique-wise marginals match, the distributions thus must be the same.

The *clique logistic loss* is given by:

$$\Delta = -\sum_{c} \log \mu(\mathbf{y_c}|\mathbf{x}; \boldsymbol{w}) \tag{7.10}$$

7.2. Back Tree-Reweighted Belief Propagation

In order to minimize (7.3) we must calculate loss gradients. This requires the prior computation of marginals. These calculations can be intractable, so some convergence threshold must be used. Domke [9] analyzes the effects of using different thresholds in the learning stage and at test time. He observed that too loose a threshold in the learning stage can lead to a bad estimated risk gradient, and learning terminating with a bad search direction. Meanwhile, a loose threshold can often be used at test time with few consequences.

Domke proposes a method, called *Back Tree-Reweighted Belief Propagation (Back TRW)* on the basis of TRW to reduce the expense of training. For this, redefines the training objective in terms of the approximate marginals obtained after message-passing is "truncated" to a fixed small number N of iterations.

The algorithm is quite complex with a strong philosophical similarity to *backpropagation* algorithm used in ANNs. The learning algorithm can be divided into the following phases:

1. Calculation of predicted marginals after a small number of iterations of TRW.
2. The predicted marginals are plugged into one of the marginal-based loss functions.
3. Calculation of the gradient of the loss function with respect to parameters $\boldsymbol{w}$.
4. Back-propagate predicted marginals, modifying the parameters slightly to make the loss lower.

these steps are repeated until the loss are lower than a threshold.

As time reduction is not a priority for the training task, we use the Toolbox of Justin Domke [115], mainly developed in *MatLab* with some Mex files to speed up certain algorithms, instead of developing our own solution based on *C++*. Besides, this toolbox calls *minFunc* [116] a *Matlab* function developed by Mark Schmidt for unconstrained optimization of differentiable real-valued multivariate functions using line-search methods. In particular, the *Broyden Fletcher*

Goldfarb Shanno algorithm (BFGS) [117] is used to compute the minimization of loss gradient. The BFGS method approximates Newton's method, a class of hill-climbing optimization techniques that seeks a stationary point of a function.

The final result to apply learning task are two matrices:

- p_F: determines the univariate factors. Specifically, the vector of log-potentials for node i is given by multiplying p_F with the features for node i.
- p_G: determines the pairwise factors. The vector of log-potentials for clique (i, j) is given by multiplying p_G with the features for edge (i, j).

In order to facilitate the exchange of data between the training stage and the rest of system we develop a simple API with two functions:

- `OpenCV2Matlab`: Converts a type *Mat* structure of *OpenCV* into a *cell* type of MatLab.
- `MatLab2OpenCV`: Converts a *cell* type of *MatLab* into a *YAML* file.

Moreover, inspired in the formulation of *Support Vector Machines (SVM)* that employs a misclassification penalty C to control the trade-off between minimizing training errors and controlling model complexity:

$$\arg\min_{\mathbf{w},\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \tag{7.11}$$

we introduce a ridge regularization penalty of λ relative to empirical risk (as measured per pixel) on all parameters to avoid over-fitting.

CHAPTER 8

CRF POTENTIALS

Perhaps functions to obtain image features are the most important components of our CRF model. In brief, the feature functions link the potential factors in the CRF formulation to the actual image take accounting the local neighborhood.

The choice of the functions that we incorporate into our model is not straightforward. On the one hand, we can achieve better accuracy results using specific features function oriented to a particular problem or dataset; on the other hand, we can employ more general features functions that allow us to face more problems. Furthermore, the main trade-off is the classic one [118]: using larger features sets can lead to better prediction accuracy, because the final decision boundary can be more flexible, but larger feature sets require more memory to store all the corresponding parameters, they are computationally more expensive and could have worse prediction accuracy due to over-fitting.

In our approach we try to keep the features relatively simple for two basic reasons:

1. A marginal increase in accuracy normally requires a large computational effort, which can turn into no real-time operation.
2. Simpler features functions allows us to gauge easily the performance varying the parameters of the model.

8.1. Introducing Features in the Model

The purpose of this section is to explain the process to incorporate some characteristic of the empirical distribution in the model distribution. Since our CRF model has a pairwise grid-like structures, two types of feature functions will be encoded in the model:

1. *Node features* related to singleton potentials ψ_i.
2. *Edge features* aassociated with pairwise potentials ψ_{ij}.

as is shown in Figure 8.1. In both instances, potentials are obtained from a linear combination of node and edge features functions.

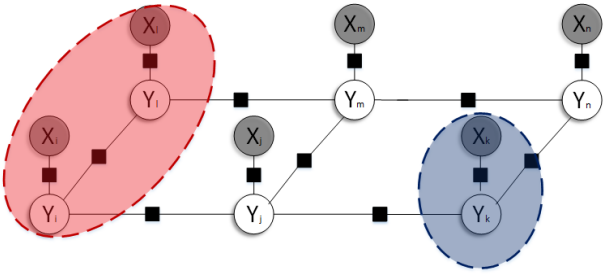


Figure 8.1: Portion of the proposed graphical model. The node and edge features are overlaid in blue and red, respectively.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ one image belongs to a set of images $\mathcal{X}$, we want to infer the class $y_i \in \mathcal{L}$ to which each pixel i belongs. Since training and inference tasks are computationally expensive, we need a practical approach to reduce the computation time. Inspired on recent works of visual place recognition with low resolution images [119,120], we have opted to compute the visual features on the original images first, and then reduce their resolutions by using superpixels, plug them into the CRF potentials.

For this purpose, we align the graph of our pairwise CRF on the image and extract some features from the vector $\mathbf{x}$ to generate the node and edge features as depicted in the Figure 8.2.

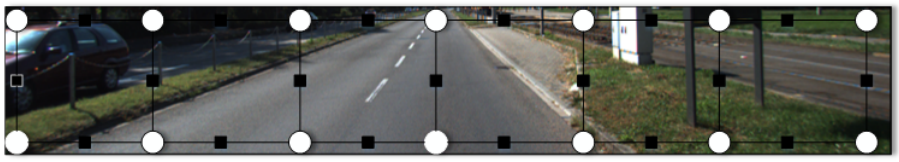


Figure 8.2: Example of a pairwise CRF aligned with the ROI of a road scene.

In computer vision we define a *image feature* as a function that provides some properties for each pixel i in an image. There is an enormous variety of image features that have been proposed in the computer vision literature, for example, gradients of the image, texton features [121,122] and SIFT features [123]. Importantly, these features do not depend on the pixel i alone; most interesting features depend on a region of pixels, or even the entire image a region of pixels, or even the entire image. For us, these functions pose as $\mathbf{i} : \mathcal{X} \rightarrow \mathbb{R}^n$, so that every pixel is transformed into a vector of n elements. Therefore, for each image $\mathbf{x}$ we obtain a matrix $\mathbf{I}$.

Once the image features have been calculated, we can build the *observation feature for nodes* [124] reducing the matrix $\mathbf{I}$ by using superpixels. In this way, each superpixel is associated to a unique node in the graph, assigning it a *feature function* $\mathbf{f}_k$. Concatenating all K observation features for nodes, we obtain the matrix of Equation (8.1) which size is $n \times K$. An scheme in the Figure 8.3 pretends clarify this operation.

$$\mathbf{F} = \mathbf{f_1}||\mathbf{f_2}||\ldots||\mathbf{f_K} \tag{8.1}$$

For a node i we transform an observation feature for nodes into a *node feature function* using the relationship of Equation (8.2) exploiting the replacement property of Kronecker's delta [125].

$$f_k(y_v, \mathbf{x}) = \delta(y, y_v)\mathbf{f_k}(\mathbf{x}) \tag{8.2}$$

In other words, each feature is nonzero only for a single output configuration y_v, but as long as that constraint is met, then the feature value depends only on the input observation. More detail in [118].

We define an *observation feature for edges* as a function that provides some properties for each edge between nodes in the graph. We place them into the model to exploit a basic characteristic of images: neighboring pixels tend to have the same label. Thus, an observation feature for edges poses $\mathbf{g}_k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{n_C}$ (where n_C is the number of cliques in the graph).

If we concatenate all edge observations, we have the matrix of Equation (8.3).

$$\mathbf{G} = \mathbf{g_1}||\mathbf{g_2}||\ldots||\mathbf{g_J} \tag{8.3}$$

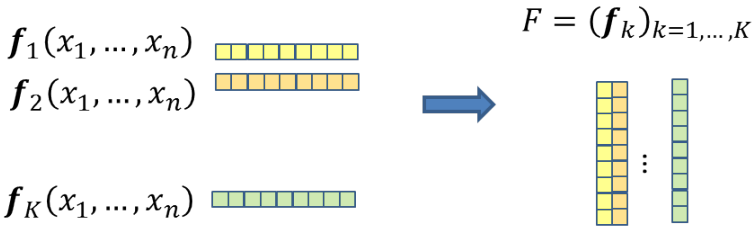


Figure 8.3: Concatenation of node feature functions to obtain the matrix F

In a similar way to the node features, we transform an observation feature for edges into a *edge feature function* using the relation described by Equation (8.4):

$$g_k(y_u, y_v, \mathbf{x}) = \delta(y, y_u)\delta(y, y_v)g_j(\mathbf{x}) \tag{8.4}$$

Since the potential functions are restricted to be positive, we can describe them by an exponential function. The most used approach is to define the log potentials as a linear function of the feature vector. Thus, we have the system of Equations (8.5):

$$\begin{cases} \psi_i(y_v, \mathbf{x}) = \exp(\boldsymbol{w_i}^T \cdot \mathbf{F}(y_v, \mathbf{x})) & \text{(8.5a)} \\ \psi_C(y_u, y_v, \mathbf{x}) = \exp(\boldsymbol{w_C}^T \cdot \mathbf{G}(y_u, y_v, \mathbf{x})) & \text{(8.5b)} \end{cases}$$

The code associated with the functions used in both potential factors occupies some hundreds of lines. As this code is invoked each time an image is processed, we developed an optimized code using various programming tricks:

- Use of *Look Up Tables (LUTs)* to compute the sinusoidal functions.

- Use of shift operations instead products.
- Using *inline* functions.
- Exchange of loops to allow memory access by *stride*.

The remainder of the chapter is dedicated to describe the features used in our model, distinguishing between node and edge features. In both cases, we carry out a *feature scaling* or whitening to speed up and favor the gradient-descent loops in the learning phase. Basically, the procedure consists in scaling the features, subtracting the mean, and dividing by the standard deviation.

8.2. Node Features

Several features are extracted to represent the visual appearance related to every node of the graph. They carry information about color, position, texture and shape that are concatenated into $\mathbf{f}_i$ to span a rich feature space.

Node features are associated with node factors that describe the interaction between the image content and the variables of interest. We use multiple image features representing appearance statistics based on color, position, texture and shape to span a rich feature space. In the Figure 8.4 we represent the location of the node features in the FG of a CRF.

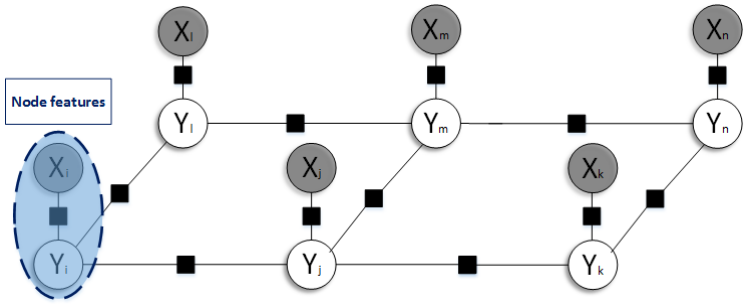


Figure 8.4: The node features in each node are calculated on the observation in the same node.

8.2.1. Color Patches

Since certain objects such as roads, trees, the sky and many others have a definite color pattern, we decided to incorporate to the model some information describing the color distribution in the scene. However, the processing of road scenes from KITTI dataset using the RGB color schemes does not provide a successful outcome. The Figure 8.5 shows that a possible cause: the information that a channel provides about road is very similar to that provided by others channels.

Then, after testing different color spaces as normalized R and G [32], HSV [126], LAB [127] and grayscale [128], we empirically found that HSV yielded the lowest overall errors. Then, we construct a set of feature functions from the components *hue* and *saturation* from the HSV

(a) Original image.

(b) R.

(c) G.

(d) B.

Figure 8.5: There is no sufficient discrimination in the RGB color model to extract the road. To appreciate the differences bewteen images, look at the cars.

model described by Figure 8.6, quite immune to scene illumination changes of each pixel i. In Figure 8.7 we show the same scene from the Figure 8.5a transformed to the HSV color space and their associated channels.

In fact, testing the approach in [9], which uses a $(2k+1)$ squared patch surrounding each pixel i, for the values $k = 0, 1, 2$, we found that bigger patches did not improve performance, but increased computation time considerably. In particular, a 2% accuracy difference at the cost of increasing the dimensionality from 2 ($k = 0$) values to 50 ($k = 2$).

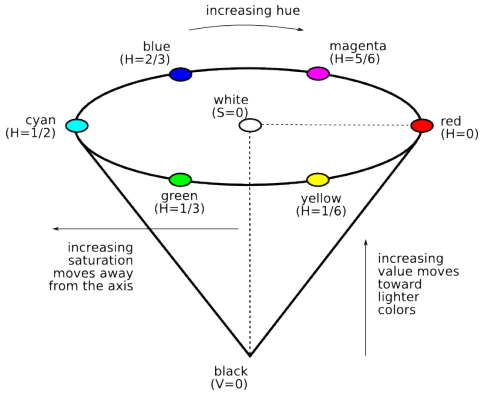


Figure 8.6: Cone model of HSV space.

8.2.2. Position

Usually object position in an image is not arbitrary but that maintain a certain order. For example in the majority of driver assistance system road occupies the center and bottom of the image; the sky is at the top of the image, etc.

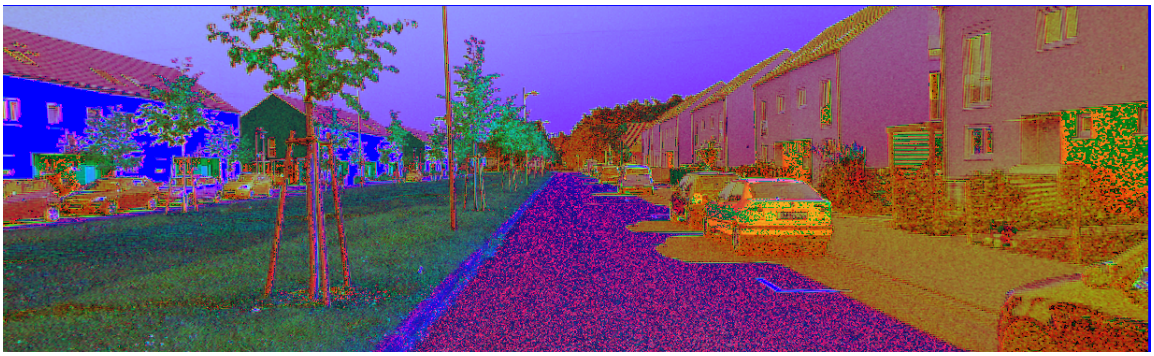
We can incorporate some global consistency in our classifier exploiting this idea. To do this, we made two features $\mathbf{f_U}$ and $\mathbf{f_V}$ account for the normalized position along the horizontal and vertical axis respectively. In the Figure 8.8 we depicted this two features for a toy example of an 3×4 image.

8.2.3. Histogram of Oriented Gradients

To capture the local object appearance and shape we compute the descriptor *Histogram of Oriented Gradients (HOG)* proposed by Dalal and Trigg [129]. HOG is reminiscent of edge orientation histogram, *Scale Invariant Feature Transform (SIFT)* [123] descriptor and shape context [130]. Essentially the algorithm counts occurrences of gradient orientation in localized portions of an image.

HOG have become very popular and successful technique in the computer vision community that has been used to describe the appearance of image regions. Its popularity lies in its robustness to:

- Small geometric distortions.



(a) Original image transformed to the HSV color model.

(b) H.

(c) S.

(d) V.

Figure 8.7: There is more discrimination in the HSV color model.

[0.33, 0.25]	[0.33, 0.50	[0.33, 0.75]	[0.33, 1.00]
[0.66, 0.25]	[0.66, 0.50]	[0.66, 0.75]	[0.66, 1.00]
[1.00, 0.25]	[1.00, 0.50]	[1.00, 0.75]	[1.00, 1.00]

Figure 8.8: Normalized position of each pixel. In each cell, the first component is the position along the horizontal and the second is the position along the vertical axis.

- Different lighting conditions.
- Little changes in the contour of the image.
- Differences images backgrounds.
- Different scales.

The essential thought behind the HOG descriptors is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The implementation of these descriptors can be achieved by dividing the image into small connected regions, called *cells*, and for each cell compiling a histogram of gradient directions or edge orientations for the pixels within the cell. The combination of these histograms then represent the descriptor.

For improved performance, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a *block*, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination or shadowing.

The Figure 8.9 presents the flow chart of the algorithm with the steps required to obtain the HOG descriptors. The main steps are:

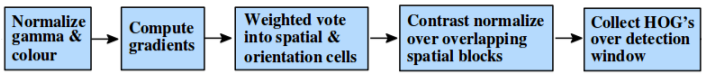


Figure 8.9: An overview of the HOG features extraction chain.

- *Gamma/Color normalization*. In practice, these normalization have only a modest effect on performance, so we have overlook it.
- *Gradient computation*. The image gradients are computed by convolving the image I with some mask D:

$$\begin{cases} I_u = I \star D_u & \text{(8.6a)} \\ I_v = I \star D_v & \text{(8.6b)} \end{cases}$$

The magnitude of the gradient is given by Equation (8.7) and the orientation of the gradient is given by Equation (8.8):

$$\|G\| = \sqrt{{I_u}^2 + {I_v}^2} \tag{8.7}$$

$$\theta = \arctan\left(\frac{I_v}{I_u}\right) \tag{8.8}$$

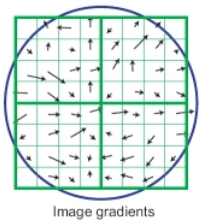


Figure 8.10: Calculation of the gradients in the HOG algorithm. Taken from [129].

- *Spatial/orientation binning*. This step is the fundamental nonlinearity of the descriptor. The image is divided into small connected regions, called *cells* (in our case with size 8×8) and each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. Depending on whether the gradient is unsigned or signed, the histogram channels are spread over $0° - 180°$ or $0° - 360°$

  In our work, we have achieved slightly better results using the unsigned version of the algorithm, which histogram is depicted in Figure 8.11.

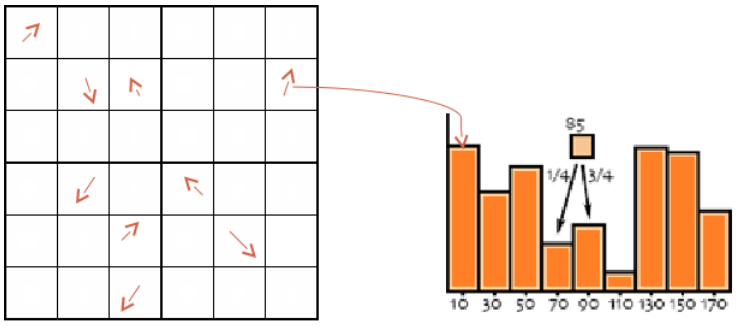


Figure 8.11: Orientation bins are evenly spaced. Taken from [129].

- *Descriptor blocks*: To compensate the illumination, histogram counts are normalized by accumulating a measure of local histogram energy over *blocks* formed by grouping the cells together into spatially connected regions. The HOG descriptor is then the vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor. Two main block geometries exist: rectangular *R-HOG* blocks and circular *C-HOG* blocks, in our case we choice the first option.

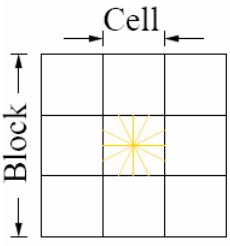


Figure 8.12: The blocks are obtained by grouping cells (there may be overlap).Taken from [129].

- *Normalization*: Let $\mathbf{v}$ the unnormalized descriptor vector obtained in the last step. We apply the L-2 normalization of the Equation (8.9) followed by clipping (limiting the maximum values of $\mathbf{v}_n$ to 0.2) where ϵ is a small constant:

$$\mathbf{v}_n = \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\| + \epsilon}} \tag{8.9}$$

After orientation and normalization processes, we have a 36-dimensional feature vector $\mathbf{f}_{HOG}$. As an example, for the road scene in the Figure 8.5a we have the HOG descriptors represented in the Figure 8.13.

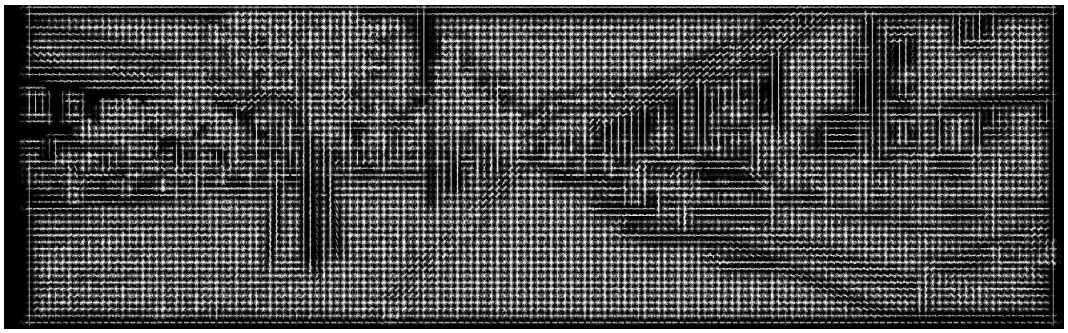


Figure 8.13: The result of applying the HOG algorithm to a road scene

The HOG descriptor used in this work is an implementation in *C++* based in the code provides by Piotr Dollár in his toolbox [131].

8.2.4. Local Binary Patterns

In the Figures 8.7b and 8.7c is easy to note that texture is well-defined. Therefore, it would be desirable to capture this information. We have opted for the use of *Local Binary Patterns (LBPs)* [132], an efficient texture operator to help us in the road extraction. Our implementation ports the code of Guoying Zhao [133] from *MatLab* to *C++*. In few words, this descriptor looks at points surrounding a central pixel i and tests whether the surrounding points are greater luminance than or less than the central point giving a binary result (which is usually converted to decimal for convenience). The histogram of these $2^8 = 256$ different labels can then be used as a texture descriptor. The described algorithm is exemplified in Figure 8.14, where the complete process iterated for each pixel is shown.

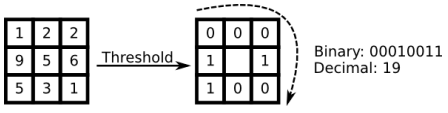


Figure 8.14: An example of local binary pattern computation.

Ojala *et al.* [134] extended the descriptor to use neighborhoods of different sizes. using a circular neighborhood and bilinear interpolating values at non-integer pixel coordinate. For neighborhoods we will use the notation $(P; R)$ which means P sampling points on a circle of radius of R. Figure 8.15 shows examples of different combinations of P and R.

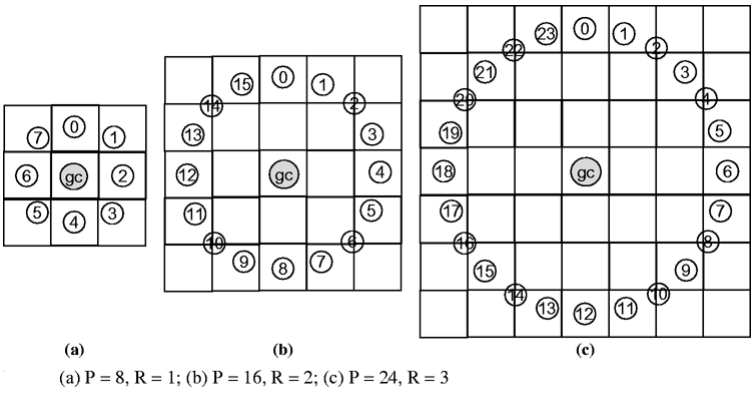


Figure 8.15: Examples of different combinations of P and R in LBP.

Another extension presented by Ojala is the definition of *uniform patterns*. A LBP is defined as *uniform* if it contains at most two $0 \rightarrow 1$ or $1 \rightarrow 0$ transitions when viewed as a circular bit string. Thus the strings 00000000 or 00110000 are uniform while 00110010 and 01010010 are not. Empirically using 8 sampling points, uniform patterns accounted for most of the patterns (90% in Ojala's dataset). Therefore, little information is lost by assigning all non uniform patterns to a single arbitrary number. Since only 58 of the 256 possible 8 bit patterns are uniform, this enables significant space savings when building LBP histograms.

As an example, for the road scene in the Figure 8.5a we have the LBP descriptors represented in the Figure 8.16. In particular, we employ $P = 4$ sampling points and a radius $R = 1$ pixel, obtaining a feature function $\mathbf{f_{LBP}}$ of $2^P = 16$ elements. According to our experiments $\mathbf{f_{HOG}}$ and $\mathbf{f_{LBP}}$ are the most discriminative features in terms of the overall accuracy.

Figure 8.16: The result of applying the LBP algorithm to a road scene

8.2.5. Summary

As a result of the concatenation of previous node features, the observation variables X_i associated to each node in the presented graphical model correspond to 56-dimensional feature vectors ($D = 2 + 2 + 16 + 36 = 56$), which populate the potentials of the CRF formulation.

The Algorithm 6 shows the steps to obtain the node features. All code was developed using *C++* and *OpenCV*.

Algorithm 6 Computation of the node features.

Input:

x, an image.
N, matrix consisting of grid nodes over the lattice of pixels.
rez, resolution reduction percentage by using superpixels.

Output:

F, a matrix of node descriptors.

Algorithm:

$n_n := NumberNodes(N);$
for $i = 1 \rightarrow n_n$ **do**
    $[ColorFeatures(i)] := FeaturesHSV(\mathbf{x}(i), N);$ ▷ Discard the V channel.
    $[V(i), U(i)] := NormalizatedPositions(N);$ ▷ One matrix for normalizated position
                                                ▷ along vertical and other along horizontal axis.
    $[HogFeats(i)] :== HOG(\mathbf{x}(i));$
    $[LBPFeats(i)] :== LBP(\mathbf{x}(i));$
end for
                                                ▷ Concatenation of previous features.
$Feats := ColorFeatures||V||U||HogFeats||LBPFeats;$
$F := DownSampling(Feats, rez);$ ▷ To speed things up.

8.3. Edge Features

Now, we describe the edge features functions incorporated to our model to encode the relations between the adjacet nodes in the undirected graph. In Figure 8.17 we depicted the location of edge features in the FG of a CRF. It should be pointed out that the inclusion of edge features increase the overall accuracy but they are less crucial that node features.

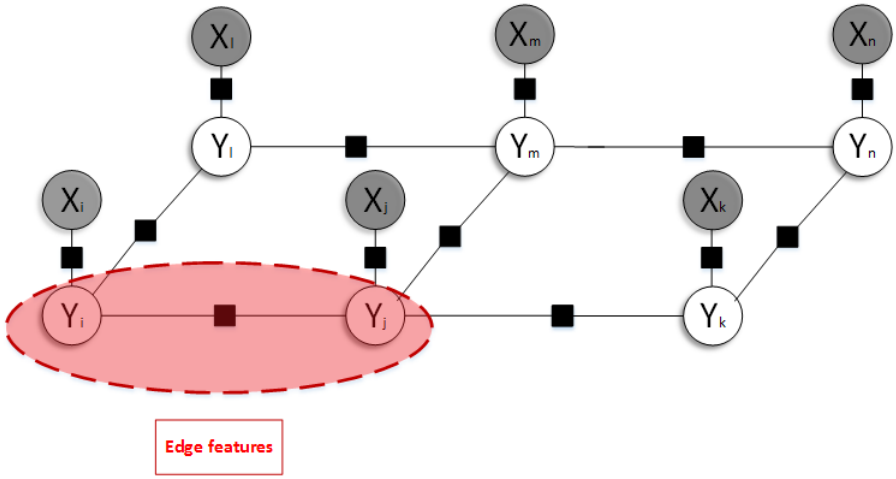


Figure 8.17: The edge features are calculated between nodes associated with the latent variables.

In our approach, edge features are discretized to increase representational power [135]. If we just use the original edge features, then we can only have a linear function. However, by discretizing, we can have a function which is piecewise constant. Strictly speaking, neither of these is more powerful, but experimentally, a piecewise constant function seems to work better.

Again, edge features consist of a concatenation of several descriptors. We shall now proceed to describe them in the remainder of the section.

8.3.1. Bias Feature

The first edge feature function we are considering is a "bias" feature $\mathbf{g_B}$ set to constant value 1 in the connected nodes. Bias features allow us to capture any effects on the states of the random variables that are independent on the other features. Since labels are not equally-likely, with bias features we incorporate this information to the model.

8.3.2. Difference Intensities Discretized

Next, we compute the L-2 norm of the difference of intensities for the nodes u and v belonging to the same clique $\mathcal{C}$ according to Equation (8.10).

$$g_D(y_u, y_v) = \sqrt{(H(x_u) - H(x_v))^2 + (S(x_u) - S(x_v))^2} \tag{8.10}$$

Now, we generate a new set of ten features by the discretization of the value obtained in 8.10 using a vector of 10 elements, so if the value exceeds $\frac{i}{10}$ with $i \in \{0, \ldots, 9\}$ then the first i elements are set to 1 and the remainder to 0. This strategy increase the complexity of the model, which implies greater accuracy according with the classic bias-variance trade-off [136] - as is depicted in Figure 8.18. However, this can lead to over-fitting. For that reason, we use only ten thresholds.

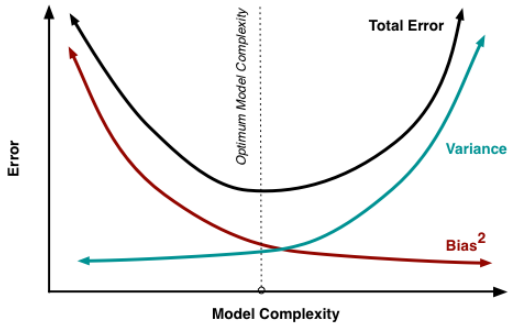


Figure 8.18: Bias and variance contributing to total error. As the model complexity is increased, the variance tends to increase and the squared bias tends to decrease in a bias-variance tradeoff

8.3.3. Different Parametrization of Vertical and Horizontal Links

The concatenation of previous features gives a vector of 11 feats for each clique in the graph. Now, we want to parametrize separately vertical and horizontal cliques. To generate the final feature vector for each clique, that corresponds with a column of the matrix of Figure 8.3, we doubled in size the number of features and arranged differently depending on whether the edges or cliques are vertical or horizontal. In the first case, the 11 features are put on the first

half, while the second half of the vector is set to zeros. For horizontal links, we apply the opposite way, first half are zeros.

8.3.4. Summary

The Algorithm 7 shows the steps to obtain the edge features using *C++* and *OpenCV*.

Algorithm 7 Computation of the edge features.

Input:

x, an image.
N, matrix consist of grid nodes over the lattice of superpixels.
w, wide of the grid.
h, height of the grid.

Output:

G, a matrix of edge feature descriptors.

Algorithm:

$n_e := NUMBER\_EDGE\_FEATURES$;
$n_{\mathcal{C}} := NumberCliques(w, h)$;
$EFeats := \Omega_{n_{\mathcal{C}} \times n_e}$; ▷ Temporal variable.
$G := \Omega_{n_{\mathcal{C}} \times 2n_e}$;
$EFeats[:][1] := Constant(EFeats)$; ▷ A column set to one
for $i = 1 \rightarrow n_n$ **do** ▷ Split the imagen into HS channels.
    $[H(i), S(i)] := FeaturesHSV(\mathbf{x}(i), N)$;
end for
$EFeats[:][2 : NUMBER\_EDGE\_FEATURES] := Difference(N, Pairs, H, S)$;
for $i = 1 \rightarrow n_{\mathcal{C}}$ **do** ▷ Different parametrization vertical and horizontal edges.
    if $Pairs(i) = HORIZONTAL$ **then**
        $G[i][:] := EFeats[i][:]||(0, \ldots, 0)$;
    else
        $G[i][:] := (0, \ldots, 0)||EFeats[i][:]$;
    end if
end for

CHAPTER 9

POST-PROCESSING

Once the approximate marginals are calculated in the miniaturized road scene, we upsample them to the original resolution and assign the most likely label to each pixel. Thus, for every input image, the output from the presented road segmentation approach is a binary image which pixels take the following values: 255 (white), if they are likely to belong to a road in the real world scene; 0 (blank), otherwise. However, the segmentation process is not perfect, some of the pixels may be misclassified.

In this chapter we are seeking to overcome an easily identifiable classification error, the presence of small specks classified as "road" in a large area corresponding to "off-road" and vice versa. The physical and continuity constraints derived from vehicle motion and road design [80] imply those situations are not feasible. Figure 9.1 show this issue.

(a) Input image.

(b) Predicted pixel labels.

Figure 9.1: Labeling of a road scene with some pixel misclassified

One of the possible origins of such errors are artifacts, which may be due to:

1. Scaling artifacts due to interpolation [137].

2. The CRFs formulation, mainly the scene label forward propagation [138].

The second circumstance can be sorted using stronger CRF formulation and training methods. Therefore the CRF model should be able to close small holes increasing the computational cost. Consequently, we focused on the first issue because its solution is relatively simple.

To deal with these specific misclassification, we pose the problem as a *salt-and-pepper noise* [137] because of its appearance as white and blacks dots superimposed on an image. The density function of salt-and-pepper noise is given by:

$$p(x) = \begin{cases} P_a & \text{for } x = a \\ P_b & \text{for } x = b \\ 0 & \text{otherwise} \end{cases} \tag{9.1}$$

In our problem, a and b are saturated values, in the sense that they are equal to the minimum and maximum allowed values in the image respectively. Therefore, this means that $a = 0$ and $b = 255$.

Although there are filters for reducing or virtually eliminating the effects of salt-and-pepper noise [137] (e.g., the *contraharmonic mean* filter, the *alpha-trimmed mean* filter, the *adaptive mean* filter, etc.), we have opted to eliminate both types of noise concatenating morphological operations because these operations are implemented very efficiently (there are algorithms of order $\mathcal{O}(n)$ for a $n \times n$ image) in OpenCV [139].

9.1. Elimination of small specks misclassified as "road"

The Figure 9.1b depicts a situation in which some specks of pixels have been misclassified as road. Given the small size of these white areas in the image, we propose that corresponds to a salt type noise. This type of noise can be deleted using a *morphological opening* [140] because it removes small objects from the foreground of an image, placing them in the background. In order to justifying the use of opening to deal with salt noise, we need to introduce two basis morphological operations: *erosion* and *dilation*.

The erosion operator takes two pieces of data as inputs. The first A is the binary image which is to be eroded. The second B is a (usually small) set of coordinate points known as a *structuring element* (also known as a *kernel*) which determines the precise effect of the erosion on the input image. The erosion of the binary image A by the structuring element B is defined by:

$$A \ominus B = \{\mathbf{z} | (B)_{\mathbf{z}} \subseteq A\} \tag{9.2}$$

where $(B)_{\mathbf{z}}$ is the is the translation of B by the vector $\mathbf{z}$, i.e., $B_{\mathbf{z}} = \{\mathbf{b} + \mathbf{z} | \mathbf{b} \in B\}$.

In words, the Equation (9.2) indicates that the erosion of A by B is the set of all points $\mathbf{z}$ such that B, translated by $\mathbf{z}$, is contained in A. To compute the erosion of a binary input image by the structuring element, we consider each of the foreground pixels in the input image in turn. For each foreground pixel (which we will call the *input pixel*) we superimpose the structuring element on top of the input image so that the origin of the structuring element coincides with the input pixel coordinates. If for every pixel in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is. However, if any of the corresponding pixels in the image are background, the input pixel is also set to background value.

Erosion can be used to remove small spurious bright spots ("salt noise") in images. This idea is illustrated in Figure 9.2 using a popular image. In this case, a square structuring element have been used. The size and shape of the structuring element conditions the process of erosion. In principle, the erosion seems to be working well for the elimination of salt noise. However, looking at the detail, the Figures 9.2a and 9.2c are not exactly the same. The erosion causes that the objects to shrink in size. The amount and the way that they grow depend upon the choice of the structuring element. One approach to this problem would be to apply the dual operation of erosion: dilation [140]. The dilation has the opposite effect, causes objects to grow in size.

The dilation of the binary image A by the structuring element B is defined by:

$$A \oplus B = \{\mathbf{z} | (B^s)_{\mathbf{z}} \cap A \neq \varnothing \} \tag{9.3}$$

where $(B^s)_{\mathbf{z}}$ denotes the symmetric of B, that is, $(B^s)_{\mathbf{z}} = \{\mathbf{z} | -\mathbf{z} \in B\}$.

Under the same conditions that we previously provided for erosion: if at least one pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value; if all the corresponding pixels in the image are background, however, the input pixel is left at the background value. The Figure 9.3 depicts a dilation applied to the Figure 9.2c using the same structuring element pretending recover the original image.

The linking of the morphological operations erosion and dilation in the order described is referred to as a *opening*. Then, a opening is just the dilation of the erosion of a set A by a structuring element B:

$$A \circ B = (A \ominus B) \oplus B \tag{9.4}$$

The Figure 9.4 shows the result of applying an opening to the image of the Figure 9.1b using a square structure of 15×15 pixels. This operation provides a better road detection because removes the false positives without modifying the shape of the other objects.

(a) Black and white original image.

(b) Noisy image affected by salt noise (20 %).

(c) Erosion of (b) with a square structuring element of 255's, 7 pixels on the side.

Figure 9.2: Use of morphological erosion for removing salt noise.

Figure 9.3: A morphological dilation applied to the Figure 9.2c. This Figure has a thiner stroker.

Figure 9.4: Removal of false positives using a morphological opening.

9.2. Elimination of small specks misclassified as "off-road"

Once the small specks misclassified as "road" issue has been resolved, the other problem that we need to resolve is pretty similar: some specks of pixels have been misclassified as "off-road" in a large area corresponding to "road". These scaling artifact can be seen as a series of small holes in the foreground produced by a pepper noise. This type of noise can be deleted using a *morphological closing* [140] because closing tends to narrow smooth sections of contours, fusing narrow breaks and long thin gulfs, eliminating small holes, and filling gaps in contour.

The basic effect of the operator dilation on a binary image is to gradually enlarge the boundaries of regions of foreground pixels. Thus areas of foreground pixels grow in size while holes within those regions become smaller [137]. In the Figure 9.5b is depicted a dilation using a a small disk structuring element of radius 4 pixels on the famous "Don Quijote y Sancho" of Pablo Picasso (Figure 9.5a).

However, the dilation also causes objects to grow. This issue can be remedied applying a morphological erosion with the same structuring element. The Figure 9.5c shows how an erosion applied after a dilation returns the objects to its original size.

The operation of dilation followed by the operation of erosion is known as a morphological closing. Therefore, a closing may be described as:

$$A \bullet B = (A \oplus B) \ominus B \tag{9.5}$$

The Figure 9.6 depicts the result of applying a closing to the image of the Figure 9.4 using a square structure of 15×15 pixels. This operation removes the false negatives located in large areas classified as "road".

(a) Black and white original image with holes.

(b) Image without holes after performing a dilation

(c) Image after performing a dilation followed by an erosion

Figure 9.5: Use of morphological dilation for eliminating holes.

Figure 9.6: Removal of false negatives using a morphological closing.

CHAPTER 10

EXPERIMENTAL RESULTS

In this chapter we will show that our system is able to perform road detection in real time. Our proposal is validated using the challenging KITTI-ROAD dataset [8], comparing our proposal against the state-of-art results. Also, this section provides different experiments, whose task it is to show the influence of different CRF components on the overall accuracy.

10.1. KITTI Road Dataset

Due to the lack of annotated datasets, the early road scene segmentation approaches focused on online scenarios [31, 141]. As time passed, trend changed to use different datasets to measure the quality of the road segmentation approaches. Unfortunately, most of the existing approaches have been evaluated on different datasets, making it difficult to performance a fair comparison. Furthermore, it is rare that a dataset distinguish ego-lane and opposite lane.

Besides, results from state-of-the-art algorithms reveal that methods ranking high on established datasets perform below average when being moved outside the laboratory to the real world. To reduce this bias, the Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago have developed the *KITTI Vision Benchmark Suite* [142], it is a project that provides challenging datasets, with novel difficulties to the computer vision community, collected by an autonomous driving platform *AnnieWAY* [143].

In our case, we will focus on the KITTI-ROAD dataset [8], the goals to be reach with this project are:

- Introduction of a benchmark for the evaluation of road detection algorithms using the same dataset. The KITTI on-line evaluation website seeks to serve as a common benchmark for road terrain detection algorithms. The process for evaluating an algorithm is described in Figure 10.1.

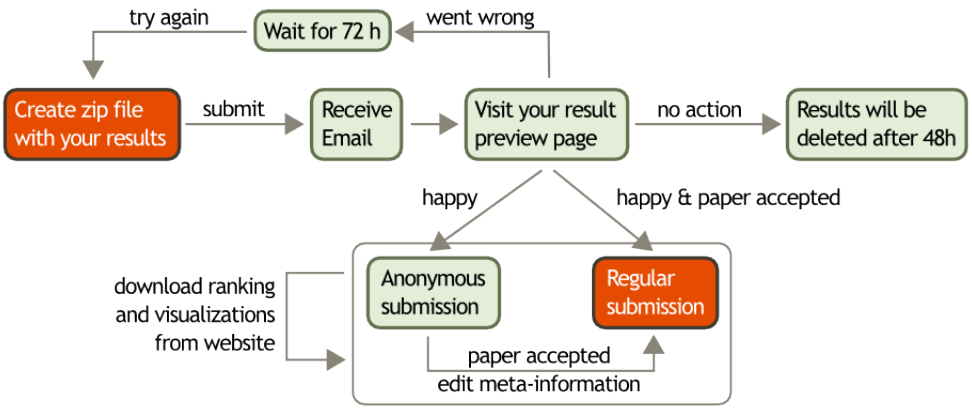


Figure 10.1: KITTI Vision Benchmark Suite submission process

- Replacing the pixel-based evaluation measures with a spatial representation in the 2D space metric on the road using the *Bird's Eye View (BEV)* more oriented to driving maneuverer. Figure 10.2 illustrates this concept.

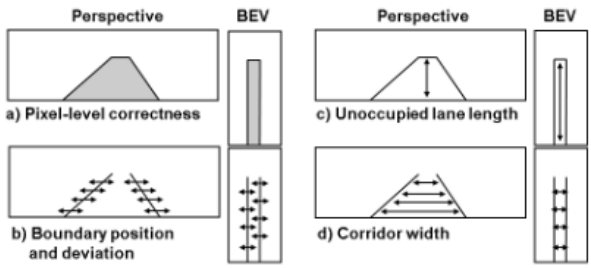


Figure 10.2: Visualization of state-of-the-art evaluation metrics.

This dataset consists of 600 frames ($\approx 375 \times 1242$ pixels) extracted from several video sequences at a minimum spatial distance of 20 meters. Besides, it is split in three subsets, each representing a typical road scene category in inner cities:

- *Urban Unmarked (UU)*.
- *Urban Marked Two-way Road (UM)*.
- *Urban Marked Multi-Lane Road (UMM)*.

besides, URBAN-ROAD quantifies the previous three categories in a single set of measurements.

For each category there are two set of images, *training* and *testing*. In the first case, is available a ground truth, generated by manual annotation of the images, with the road area (i.e., the composition of all lanes); also, for the UM dataset is available a ground truth with the ego-lane (i.e., the lane the vehicle is currently driving on).

In order to quantify the quality of road detection algorithms, the evaluations are carried out in the BEV. The used metrics are *precision*, *recall*, *F1-measure*, *accurancy* and *averange precision* (AP) computed for different recall values corresponding with Equations (10.1) to (10.6).

The expression of each metric are: *True Positive (TP)* the number of pixels correctly identified, *True Negative (TN)* the number of pixels correctly rejected, *False Positive (FP)* the number of pixels incorrectly identified and *False Negative (FN)* incorrectly rejected.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10.2}$$

$$\text{F1-measure} = 2\frac{\text{PrecisionRecall}}{\text{Precision} + \text{Recall}} \tag{10.3}$$

$$\text{Accurancy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{10.4}$$

The precision relates to the system's ability to identify positive samples, whereas the recall relates to the system's ability to identify negative samples. The F1-measure is defined as the mean of precision and recall. Accuracy is the pixelwise comparative between ground truth pixels and the output of the algorithms.

For methods that output confidence maps (in contrast to binary road classification), the classification threshold τ is chosen to maximize the F-measure, yielding F_{max}:

$$F_{max} = \arg\max_{\tau} F1 \tag{10.5}$$

In order to provide insights into the performance over the full recall range, the average precision [144] summaries the shape of the precision-recall curve and is defined as the mean precision at a set of eleven equally spaced recall levels $r \in [0, 0.1, \ldots, 1]$.

$$\text{AP} = \frac{1}{11} \sum_{r \in 0, 0.1, \ldots, 1} \max_{\tilde{r}: \tilde{r} > r} \text{Precision}\,(\tilde{r}) \tag{10.6}$$

We have employed the KITTI ROAD dataset with two purposes:

- Learning the parameters of the model using the provided training data.
- Reporting of results on the test data to compare with other state-of-art approaches.

10.2. Set Up the Classification Model

In this section the main goal is obtain the model that best suits our needs, optimizing the trade-off between accuracy and computation time. To achieve this aim, we have studied the influence of features and model parameters selected. The efficiency of each proposal can be checked using the Python code provides for local assessment.

For each experiment we perform a *k-fold cross-validation* in order to prevent over-fitting and to assess the results with different configurations. In k-fold cross validation the dataset $\mathcal{D}$ is randomly split into k mutually exclusive subset (the *folds*) $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ of approximate equal size. The classifier is trained and tested k times; each time $t \in \{1, 2, \ldots, k\}$, it is trained on $\mathcal{D} \backslash \mathcal{D}_t$ and tested on $\mathcal{D}_t$. Figures 10.3 and 10.4 shows this process.

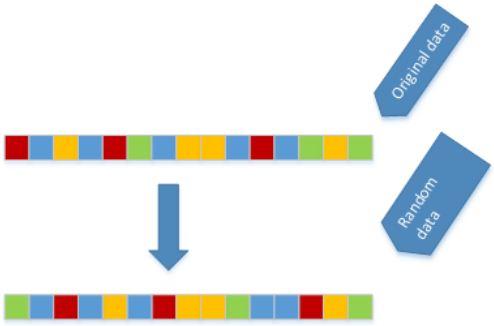


Figure 10.3: Randomization of the location of the samples in k-fold cross validation.

With a large number of folds, the bias of the true error rate classifier will be small (the estimator will be very accurate) but the variance of the true error rate classifier will be large and the computational time will be very large as well. Rather, with a small number of folds computation time are reduced and the variance will be smaller but the bias of the classifier will be large. In our experiments, we have opt for a compromise solution raising a 5-fold cross validation.

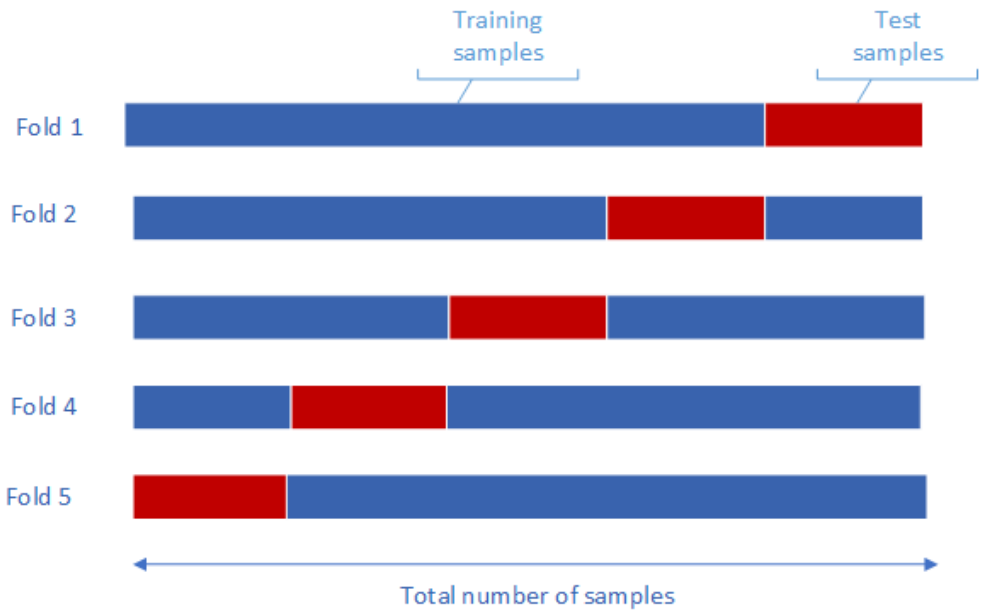


Figure 10.4: 5-fold cross validation

Benchmarks have been carried out on a system equipped with 16 GB DDR2-1600 RAM and an Intel Core i7 4700 MQ running at 2.4 GHz. The compiler used is GCC version 4.6.3, with optimization flags:

-O3

```
-fforce-addr -march=native -ftracer
-floop-interchange -floop-strip-mine
-floop-block -ftree-loop-distribution
```

10.2.1. Semantic Labeling in Miniaturized Scenes

Since training and inference tasks are computationally expensive, we seek a practical approach to reduce the computation time. Besides, inspired on recent works of visual place recognition with low resolution images [119, 120] we also opt to study and test lower resolutions in the KITTI ROAD dataset.

To do that, once the features have been computed on the original images, we reduce their resolution by using superpixels and we subsample the ground-truth labels accordingly to a percentage of the original size. Once the approximate marginals are calculated, we upsample them to the original resolution. With this approach, we shrink the space of hypotheses and achieve a speed-up both in training and testing stages, but more notable during inference, which is the application-oriented part of the algorithm.

To do that, once the features have been computed on the original images, we reduce their resolution by using superpixels and we subsample the ground-truth labels accordingly to a percentage of the original size. Once the approximate marginals are calculated, we upsample them to the original resolution. With this approach, we shrink the space of hypotheses and achieve a speed-up both in training and testing stages, but more notable during inference, which is the application-oriented part of the algorithm.

Table 10.1 shows the effectiveness of our approach, abbreviated as PGM-ARS, to extract the road while varying the reduction percentage of the image resolution evaluated with 5-fold cross validation on the training images. All evaluations are performed in the called "bird eye view" due to is best suited for vehicle control [8] using the standard metrics precision and recall. In all cases we have used the *clique logistic loss* (Section 7.1.1.3) for learning and 5 iterations inside TRW inference (Section 6.4).

Table 10.1: Road estimation results obtained for different sizes of the validation images. All results are in %.

	UU		UM		UMM	
Image Resolution	**Precision**	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
5 %	72.49	76.80	71.13	81.69	83.33	90.08
10 %	79.90	80.92	75.11	86.63	89.83	93.02
15 %	79.50	81.34	75.65	86.88	89.80	93.66
20 %	**82.75**	83.96	**84.44**	87.53	**90.07**	94.26
25 %	82.52	84.13	84.34	88.04	89.84	94.58
30 %	82.22	84.53	83.10	88.31	89.76	94.67
40 %	82.05	84.79	82.69	88.54	89.55	94.88
50 %	81.88	**85.24**	82.08	**88.72**	89.15	**94.92**

The highest precision values are obtained for 20% image resolution, whereas the recall values are slightly increased with the resolution. In fact, the gain between 20% and 50% rows is lower than 1.5% for all categories.

Therefore, there is not much improvement for increasing image size. Our explanation for this result is twofold. In first place, bigger images have more granular detail, partly reducing the intra-region similarity. Secondly, in lower-resolution models, there are fewer intermediate variables, facilitating the spread of messages and the CRF model convergence. These results validate our superpixels hypothesis as the optimal way for segmenting complex images with a fair time processing. Besides, due to memory limitations during training, we have not been able to test percentages over 50%.

10.2.2. Selection of Optimal Edge Appearance Probability Parameter

An important issue to resolve using TRW inference is the determination of the elements which comprise the edge appearance probabilities $\boldsymbol{\rho}$. In our case, we are using a simplified approach, *Uniform TRW* [110], assigning the same value ρ to all elements of the vector $\boldsymbol{\rho}$.

In this way, it is easy to obtain the optimum value empirically by performing a sweep of the ρ parameter in the interval $[0, 1]$. Thus, Table 10.2 summarizes our findings in this aspect for the reduced resolution at 20%.

Table 10.2: Sweep of values for the ρ parameter in the TRW inference. KITTI ROAD images at 20% resolution.

ρ	$\mathbf{F_{max}}$
0.20	86.11 %
0.40	86.45 %
0.50	87.32 %
0.60	87.27 %
0.80	87.12 %
1.00	86.94 %

Additionally, it must be noted that according to [99], the optimal value of ρ for graphs, satisfying certain symmetry conditions as in our case, can be approximated with the number of vertices ($|\mathcal{V}|$) and edges ($|\mathcal{E}|$) using Equation (10.7).

$$\rho^* \approx \frac{|\mathcal{V}| - 1}{|\mathcal{E}|} \tag{10.7}$$

Applying it, we obtain a theoretical value of $\rho = 0.51$. Observing the table, the highest accuracy is obtained for $\rho = 0.5$. In general, it can be stated that this parameter tends to 0.5 for bigger grid graphs like ours. Although this is not necessarily the optimum, it is the largest number that leads to a convex inference problem [111].

(a) Original road scene

(b) Road detection by using clique loss function.

(c) Road detection by using univariate loss function.

(d) Road detection by using quad loss function.

Figure 10.5: Example of road detection in the same road scene by using different loss functions.

10.2.3. Influence of Loss Functions

In order to compare the different loss functions presented in 7.1.1 we carry out a road segmentation on the training images with reduced resolution to 20% of the original size varying the loss function. At this small resolution, inference is computationally efficient requiring less than 50 ms per image without a big loss in accuracy, as it is shown in Table 10.3.

Table 10.3: Comparison of loss functions on KITTI ROAD.

Loss function	$\mathbf{F_{max}}$
Clique	87.32 %
Univariate	84.77 %
Quad	84.13 %

As might be expected, better results are achieved using the clique loss. However, the difference between obtained results is very small. Figures 10.5a, 10.5b, 10.5c and 10.5d illustrate some predicted samples.

10.2.4. Influence of Pre- and Post-processing stages

Our approach considers two stages which aim is to achieve a better overall accuracy. Naturally, these stages are totally dependent on the road detection problem. This is relevant because these stages are based on geometric restrictions imposed by the continuity of the road [**?**], therefore they should be modified or removed if we are interested in other semantic classes.

The Table 10.4 shows the improvement achieved in the overall accuracy by using this two stages versus the accuracy achieved without kind of processing. As these algorithms are not computationally intensive tasks, its introduction is a significant improvement, especially in the categories UM and UU.

Table 10.4: Influence of pre- and post-processing stages.

	Without additional stages			With pre- and post-processing		
Metric	**UU**	**UM**	**UMM**	**UU**	**UM**	**UMM**
$\mathbf{F_{max}}$	71.33 %	71.97 %	89.56 %	79.94 %	80.97 %	91.76%

As might be expected, better results are achieved using the pre and postproccesing process. The improvement is is particularly high in the most difficult categories, i.e., UU and UM.

10.3. Comparative with State of Art

To demonstrate the good performance of the road detection system proposed, we have validated our method, which we have called *PGM-ARS*, on the KITTI-ROAD dataset using the KITTI Vision Benchmark Suite. Nowadays, it is the most important vision benchmark suite, with constant submissions, so all data may not be fully updated.

According to the previous cross-validation experiments, we opt to reduce the images at 20% and then evaluate the road segmentation performance on the testing set. At this small resolution, inference is computationally efficient requiring less than 50 ms per image in a i7-4700MQ processor and without a big loss in accuracy. The results are shown in Table 10.5 using standard Kitti metrics.

Next, Table 10.6 depicts the most representative entries in the public KITTI ROAD benchmark. Our PGM-ARS proposal is placed among the state-of-the-art. We obtain similar values at much lower time costs, but ranking second in roads with multiple marked lanes (UMM_ROAD) and fourth in the general UM_ROAD category among monocular approaches to the date of April 2015. Besides, the proposals achieving higher accuracies require longer computation times and more hardware resources compared to ours. It must be also noted that our approach uses monocular images and does not require stereo vision nor 3D points.

Table 10.5: Road estimation results on the test set images

Benchmark	MaxF %	AP %	PRE %	REC %
UM_ROAD	81.20	69.82	78.32	84.30
UMM_ROAD	90.95	85.68	88.86	93.14
UU_ROAD	79.82	68.33	77.97	81.76
URBAN_ROAD	85.52	74.75	83.24	87.92

Table 10.6: Comparison of KITTI URBAN-ROAD state-of-the-art

Method	Setting	MaxF	Runtime	Environment
DDN	Mono	92.55%	2 s	GPU @ 2.5 Ghz (Python + C/C++)
ProBoost	Stereo	87.21%	2.5 min	>8 cores @ 3.0 Ghz (C/C++))
SPRAY	Mono	86.33%	45 ms	NVIDIA GTX 580 (Python + OpenCL)
PGM-ARS	Mono	85.52%	50 ms	4 cores @ 2.1 GHz
RES3D-Velo	Laser	85.49%	0.36 s	1 core @ 2.5 Ghz (C/C++)

10.3.1. Urban Marked Lanes

We first show results in the perspective image (Figure 10.6) where red denotes false negatives, blue areas correspond to false positives and green represents true positives.

In Figure 10.6a detection is very accurate; however, in Figure 10.6b sidewalks are being confused with road. It can be observed that the precision values are degraded for UM and UU categories. This is explained by an increase of false positives in complex scenes in which parking lots, garage entrances and crossroads have a road appearance. Our algorithm classifies them as road, but they are "off-road" in the ground-truth.

Figure 10.7 shows the evaluation in bird's eye view. We can appreciate the accuracy of the detection specialy in 10.7a.

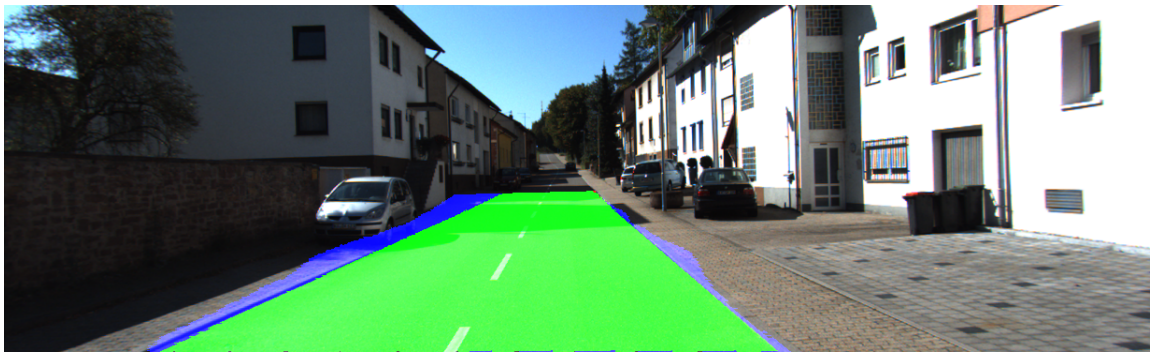
10.3.2. Urban Unmarked Lanes

Some results in the perspective view are depicted in the Figures 10.8a and 10.8b. In both cases detection is very accurate. In fact, detection fails only in a bunch of pixels belonging to sideways.

The BEV perspective serves to illustrate the high precision of this road detection process as we can see in Figures 10.9a and 10.9b. In both scenes, the segmentation on the image corresponds perfectly with the road lanes.

10.3.3. Urban Multiple Marked Lanes

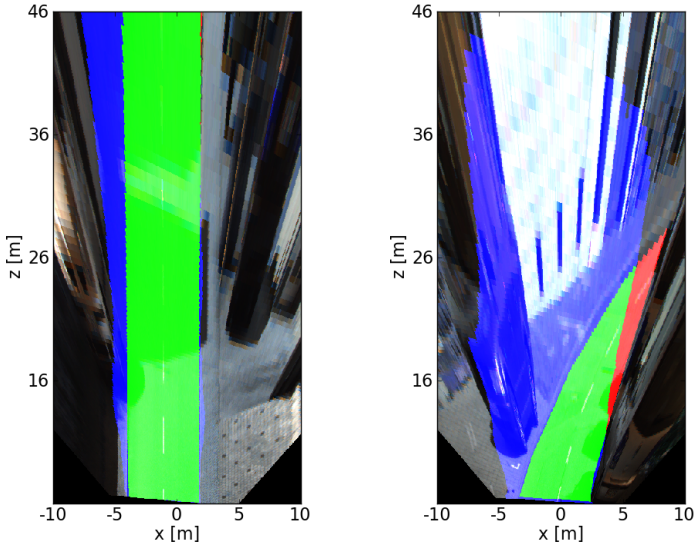
Figures 10.10a and 10.10b depict examples of road segmentation. In both cases detection is very accurate. In fact, detection fails only in a bunch of pixels belonging to sideways.



(a) Frame 77

(b) Frame 95

Figure 10.6: Examples of images illustrating the performance of the method in the category UM Road.



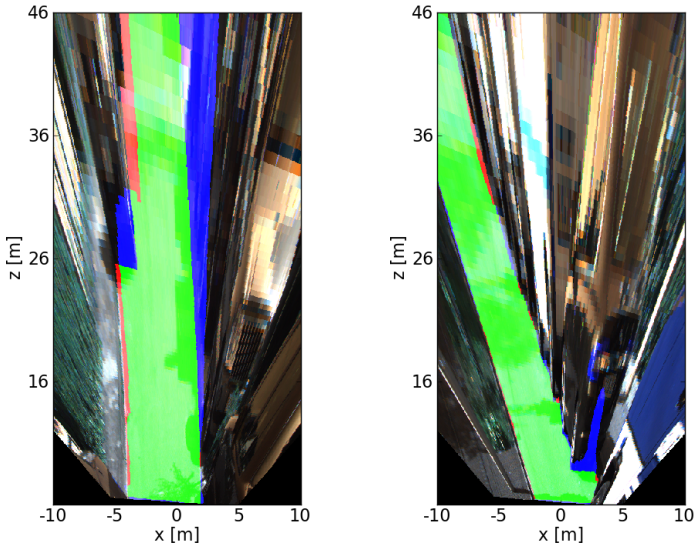
(a) Frame 77 (b) Frame 95

Figure 10.7: Example of images in BEV illustrating the performance of the method in the category UM Road.

(a) Frame 20

(b) Frame 82

Figure 10.8: Example of images in perspective view illustrating the performance of the method in the category UU Road.



(a) Frame 20 (b) Frame 82

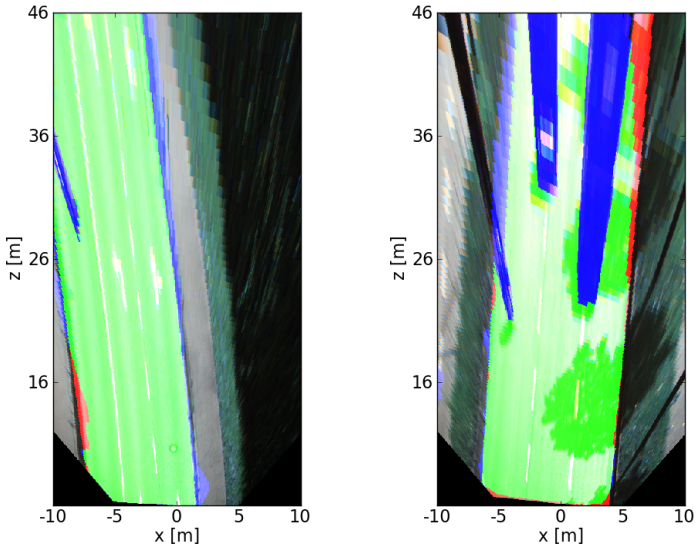
Figure 10.9: Example of images in BEV illustrating the performance of the method in the category UU Road.

(a) Frame 40

(b) Frame 25

Figure 10.10: Example of images in perspective view illustrating the performance of the method in the category UMM Road.

The BEV perspective serves to illustrate the high precision of this road detection process as we can see in Figures 10.11a and 10.9a. In both scenes, the segmentation on the image corresponds perfectly with the road lanes.



(a) Frame 40 (b) Frame 25

Figure 10.11: Example of images in BEV illustrating the performance of the method in the category UMM Road

CHAPTER 11

CONCLUSIONS AND FUTURE WORKS

11.1. Conclusions

This master thesis has presented an efficient method for one of the most important current scientific challenge in intelligent vehicles as is road detection. Our proposal relies on PGMs, namely CRFs. Nowadays, CRFs are playing an increasingly important role in IU. The main reasons are threefold:

1. CRFs allow rich description and possibility overlapping feature representations can be included in the model without needing to worry about of the independence assumptions among the feature descriptors.
2. They are discriminative, i.e. model $p(\mathbf{y}|\mathbf{x})$, that is the natural distribution for classifying a given observation $\mathbf{x}$ into a class $\mathbf{y}$.
3. The graphical structure of the CRF allow us to encode spatial dependencies in the images easily.

Therefore, CRFs are one of the best alternatives in order to effectively address the image labeling task because of their ability to exploit context in images and to capture dependencies between the model variables.

However, CRFs present some issues such as:

- Exact inference is a NP-hard problem. Propagation.
- Learning in graphs with cycles is intractable and can also generate poor predictions when the model is misspecified.

The objective of our proposal is primarily to solve the issues presented above. The inference problem is especially relevant because it is is the application-oriented part of the model. Also, we have to confront how to achieve real time integration.

According to the results obtained in the previous chapters and considering the current state of the art context, the main contributions procured in this work are the following:

- Use of advanced techniques of machine learning, as is a robust Probabilistic Graphical Model, with the model learning based upon CRFs and the prediction of the semantic classes based on Uniformly Reweighted Belief Propagation, for real time automatic scene labeling.
- Parameter learning based upon approximate marginal inference instead the usual approach based on approximations of the likelihood.
- Providing a fast and efficient prediction employing miniaturized images based on superpixels.
- Different visual features have been selected to efficiently exploit the context and pixel dependencies in the road scene.
- The C++ implementation of the inference stage have contributed to achieve real-time performance, which may ease its integration into ADAS and autonomous driving systems.
- To validate the methodology with one standard and open dataset, KITTI ROAD, yielding state-of-the-art results with the lowest runtime per image using a standard PC. Our proposal was pioneer in the use of CRFs in this challenging dataset.

In conclusion, we have provide a probabilistic framework that exploit context in road detection applications with superior performance compared to other state-of-art approaches. Our model is flexible enough to support the addition of a large variety of feature

11.2. Future works

Although we have high accuracies for the automatic road semantic labeling task, given the flexibility of CRFs to incorporate new feature functions, the performance should improve using more specific appearance and geometric features. We propose the following improvements:

- To further study the difficult cases of UU and UM categories.
- Use the Walsh-Hadamard transform, an approximation of the cosine transform, to convert the image in the frequency domain and use the frequencies as features due to the road is likely to have some fine structure which will translate into high values for the high frequencies.
- Fuse visual feature with LIDAR and GPS information.
- Vary the geometric of the grid, to apply a finer mesh in the bottom of the scene, where it is more likely the road.

- Use of an adaptive Canny and Hough transform to extract the lane markings, which can be helpful in marked road detection.
- Use higher order CRFs and dense CRFs to capture more complex spatial dependencies.
- Use of *Graph Cut*, an efficient methods for solving inference problem for pairwise UGMs when state variables y_i are binary, because road segmentation is clearly a binary problem (road, no road).
- It would be desirable to try other machine learning techniques using the same features on the same datasets. It is allows us to verify how much of the accuracy is due to the model and how much is due to the features.
- Extend this work for multi-class road scene segmentation.

BIBLIOGRAPHY

[1] J. D. Lafferty, A. McCallum, and F. C. N.Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," in *International Conference on Machine Learning (ICML)*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

[2] European Commission, "Road Transport: a Change of Gear," [web page] http://ec.europa.eu/transport/modes/road/doc/broch-road-transport_en.pdf, 2012.

[3] European Transport Safety Council (ETSC), "Road Safety Target Outcome: 100,000 fewer deaths since 2001. 5th Road Safety PIN Report," [web page] http://ec.europa.eu/transport/modes/road/doc/broch-road-transport_en.pdf, 2011.

[4] C. B. et al., "Annual Statistical Report,Deliverable D3.9 of the EC FP7 project DaCoTA," [web page] http://ec.europa.eu/transport/road_safety/pdf/statistics/dacota/dacota-3.5-asr-2012.pdf, 2012.

[5] Wikipedia, "Advanced driver assistance systems," [web page] http://en.wikipedia.org/wiki/Advanced_driver_assistance_systems, 2014.

[6] O. Gietelink, J. Ploeg, B. D. Schutter, and M. Verhaegen, "Development of Advanced Driver Assistance Systems with Vehicle Hardware-in-the-Loop Simulations," *Vehicle System Dynamics*, vol. 44, no. 7, pp. 569–590, 2006.

[7] A. Guan, S. H. Bayless, and R. Neelakantanon, "Connected vehicle insights: An overview of vision-based data acquisition and processinge chnology and its potential for the transportation sector," in *The Intelligent Transportation Society of America (ITS America). Technology Scan Series 2011 -2012*, 2012, pp. 1–8.

[8] J. Fritsch, T. Kuehnl, and A. Geiger, "A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms," in *IEEE Intelligent Transportations Systems Conference (ITSC)*, Oct 2013, pp. 1693–1700.

[9] J. Domke, "Learning Graphical Model Parameters with Approximate Marginal Inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 10, pp. 2454–2467, 2013.

[10] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine Vision and Applications*, vol. 25, no. 3, pp. 727–745, 2014.

[11] J. M. Álvarez, T. Gevers, Y. LeCun, and A. M. López, "Road Scene Segmentation from a Single Image," in *European Conference on Computer Vision (ECCV)*, A. W. Fitzgibbon, Svetlana, P. Perona, Y. Sato, and C. Schmid, Eds., 2012, pp. 376–389.

[12] L. Xiao, B. Dai, D. Liu, T. Hu, and T. Wu, "CRF based Road Detection with Multi-Sensor Fusion," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 192–198.

[13] M. Felisa and P. Zani, "Robust Monocular Lane Detection in Urban Environments," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2010, pp. 591–596.

[14] T.-Y. Sun, S.-J. Tsai, and V. Chan, "HSI Color Model Based Lane-Marking Detection," in *IEEE Intelligent Transportations Systems Conference (ITSC)*, Sept 2006, pp. 1168–1172.

[15] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using B-Snake," *Image and Vision Computing*, vol. 22, no. 4, pp. 69–280, 2004.

[16] J. McCall and M. Trivedi, "Video-based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 20–37, March 2006.

[17] S. E. Umbaug, *Computer Vision and Image Processing: A Practical Approach Using Cviptools with Cdrom*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1997.

[18] H. A., H. Bülthoff, J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological Cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.

[19] C. Kuo-Yu and L. Sheng-Fuu, "Lane Detection Using Color-Based Segmentation," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2005, pp. 706–711.

[20] Y. He, H. Wang, and B. Zhang, "Color-Based Road Detection in Urban Traffic Scenes," *IEEE Intelligent Transportations Systems Conference (ITSC)*, vol. 5, no. 4, pp. 309–318, Dec 2004.

[21] B. Southall and C. Taylor, "Stochastic Road Shape Estimation," in *International Conference on Computer Vision (ICCV)*, vol. 1, 2001, pp. 205–212 vol.1.

[22] Wikipedia, "Hough transform," [web page] http://en.wikipedia.org/wiki/Hough_transform, 2014.

[23] ——, "Particle filter," [web page] http://en.wikipedia.org/wiki/Particle_filter, 2014.

[24] Y. Bin and A. Jain, "Lane Boundary Detection Using a Multiresolution Hough Transform," in *International Conference on Image Processing (ICIP)*, vol. 2, Oct 1997, pp. 748–751.

[25] W. Freeman and E. Adelson, "The Design and Use of Steerable Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 13, no. 9, pp. 891–906, Sep 1991.

[26] Wikipedia, "Spline (mathematics)," [web page] http://en.wikipedia.org/wiki/Spline_(mathematics), 2015.

[27] A. Kaske, R. Husson, and D. Wolf, "Chi-Square Fitting Of Deformable Templates For Lane Boundary Detection," 1995, pp. 66–78.

[28] C. Jung and C. C.R. Kelber, "A Robust Linear-Parabolic Model for Lane Following," in *Computer Graphics and Image Processing, 2004. Proceedings. 17th Brazilian Symposium on*, Oct 2004, pp. 72–79.

[29] L. M. Bergasa, D. Almería, J. Almazán, J. J. Yebes, and R. Arroyo, "DriveSafe: an App for Alerting Inattentive Drivers and Scoring Driving Behaviors," in *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, July 2014, pp. 240–245.

[30] E. Dickmanns and B. Mysliwetz, "Recursive 3-d road and relative ego-state recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, no. 2, pp. 199–213, Feb 1992.

[31] M. Sotelo, F. Rodriguez, and L. Magdalena, "VIRTUOUS: Vision-Based Road Transportation for Unmanned Operation on Urban-like Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 2, pp. 69–83, June 2004.

[32] C. Tan, T. Hong, T. Chang, and M. Shneier, "Color model-based real-time learning for road following," in *IEEE Intelligent Transportations Systems Conference (ITSC)*, Sept 2006, pp. 939–944.

[33] J. M. Álvarez and A. M. López, "Road Detection Based on Illuminant Invariance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 184–193, March 2011.

[34] M. Nieto and L. Salgado, "Real-Time Vanishing Point Estimation in Road Sequences Using Adaptive Steerable Filter Banks," in *Advanced Concepts for Intelligent Vision Systems*, ser. Lecture Notes in Computer Science, J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, Eds. Springer Berlin Heidelberg, 2007, vol. 4678, pp. 840–848.

[35] C. Rasmussen, "Grouping dominant orientations for ill-structured road following," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, June 2004, pp. 470–477.

[36] H. Kong, J.-Y. Audibert, and J. Ponce, "Vanishing point detection for road detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2009, pp. 96–103.

[37] P. Y. Shinzato and D. F. Wolf, "A Road Following Approach Using Artificial Neural Networks Combinations," *Journal of Intelligent and Robotic Systems*, vol. 62, no. 3-4, pp. 527–546, 2011.

[38] A. Lookingbill, J. Rogers, D. D. Lieb, J. Curry, and S. Thrun, "Reverse Optical Flow for Self-Supervised Adaptive Autonomous Robot Navigation," *International Journal of Computer Vision (IJCV)*, vol. 74, no. 3, pp. 287–302, 2007.

[39] Y. Alon, A. Ferencz, and A. Shashua, "Off-road Path Following using Region Classification and Geometric Projection Constraints," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, June 2006, pp. 689–696.

[40] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," in *International Conference on Machine Learning (ICML)*, L. Saitta, Ed. Morgan Kaufmann, 1996, pp. 148–156.

[41] T. T. Kühnl, F. Kummert, and J. Fritsch, "Spatial Ray Features for Real-Time Ego-Lane Extraction," in *IEEE Intelligent Transportations Systems Conference (ITSC)*, Sept 2012, pp. 288–293.

[42] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: a Statistical View of Boosting," *Annals of Statistics*, vol. 28, p. 2000, 1998.

[43] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real Time Obstacle Detection in Stereovision on Non Flat Road Geometry Through "v-disparity" representation," in *IEEE Intelligent Vehicles Symposium (IV)*, vol. 2, June 2002, pp. 646–651 vol.2.

[44] G. B. Vitor, A. C. Victorino, and J. V. Ferreira, "A Probabilistic Distribution Approach for the Classification of Urban Roads in Complex Environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[45] A. Torralba, K. Murphy, and W. Freeman, "Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, June 2004, pp. II–762–II–769 Vol.2.

[46] S. Thrun, M. Montemerlo, H. D. Hendrik, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Marke, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[47] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3D LIDAR Data in Non-Flat Urban Environments Using a Local Convexity Criterion," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2009, pp. 215–220.

[48] R. Mohan, "Deep Deconvolutional Networks for Scene Parsing," *arXiv preprint arXiv:1411.4101*, 2014.

[49] R. Szeliski, *Computer Vision: Algorithms and Applications*, 1st ed. New York, NY, USA: Springer-Verlag New York, Inc., 2010.

[50] Wikipedia, "Inverse problem," [web page] http://en.wikipedia.org/wiki/Inverse_problem, 2015.

[51] S. Nowozin and C. H. Lampert, "Structured Learning and Prediction in Computer Vision," *Foundations and Trends in Computer Graphics and Vision*, vol. 6, no. 3-4, pp. 185–365, 2011.

[52] C. Wang and N. Paragios, "Markov Random Fields in Vision Perception: A Survey," Institut National de Recherche en Informatique et en Automatique, Tech. Rep. RR-7945, Sep 2012.

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

[54] A. P. Dawid, "Conditional independence in statistical theory," *Journal of the Royal Statistical Society, Series B*, vol. 41, pp. 1–31, 1970.

[55] T. A. Cohn, "Scaling Conditional Random Fields for Natural Language Processing," 2007.

[56] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2014.

[57] R. Shachter, "Bayes-Ball:The Rational Pastime (for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams)," in *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, G. Cooper and S. Moral, Eds. Morgan Kaufmann, San Francisco, CA, 1988, pp. 480–487.

[58] L. R.Rabiner, "Readings in speech recognition," A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296.

[59] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," 1960.

[60] D. Terzopoulos and R. Szeliski, "Active vision," A. Blake and A. Yuille, Eds. Cambridge, MA, USA: MIT Press, 1993, ch. Tracking with Kalman Snakes, pp. 3–20.

[61] J. Kim and J. W. Woods, "Spatiotemporal Adaptive 3-D Kalman Filter for Video," *IEEE Transactions on Image Processing*, vol. 6, pp. 414–424, 1997.

[62] A. Hervieu, P. Bouthemy, and J.-P. L. Cadre, "A HMM-Based Method for Recognizing Dynamic Video Contents from Trajectories." in *IEEE Transactions on Image Processing*. IEEE, 2007, pp. 533–536.

[63] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 20, no. 12, pp. 1371–1375, 1998.

[64] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995.

[65] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image Processing with Neural Networks - A Review." vol. 35, no. 10, pp. 2279–2301, 2002.

[66] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[67] J. M. Hammersley and P. E. Clifford, "Markov random fields on finite graphs and lattices," Unpublished manuscript, 1971.

[68] A. Blake, P. Kohli, and C. Rother, *Introduction to Markov Random Fields*, A. Blake, P. Kohli, and C. Rother, Eds. The MIT Press, 2011.

[69] P. Felzenszwalb and R. Zabih, "Dynamic Programming and Graph Algorithms in Computer Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, no. 4, pp. 721–740, April 2011.

[70] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, no. 6, pp. 721–741, Nov 1984.

[71] C. Sutton and A. McCallum, "An Introduction to Conditional Random Fields for Relational Learning," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2007.

[72] S. Kumar and M. Hebert, "Discriminative Fields for Modeling Spatial Dependencies in Natural Images." in *Advances in Neural Information Processing Systems (NIPS)*, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, 2003.

[73] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñan., "Multiscale Conditional Random Fields for Image Labeling." in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 695–702.

[74] L. Ladický, C. Russell, P. Kohli, and P. H. STorr, "Associative Hierarchical CRFs for Object Class Image Segmentation," in *International Conference on Computer Vision (ICCV)*, Sept 2009, pp. 739–746.

[75] L. Ladický, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr, "What, Where and How Many? Combining Object Detectors and CRFs," in *European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 424–437.

[76] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 1998.

[77] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy Belief Propagation for Approximate Inference: An Empirical Study," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[78] B. J. Frey and D. J. MacKay, "A Revolution: Belief Propagation in Graphs With Cycles," *Advances in neural information processing systems*, pp. 479–485, 1998.

[79] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Tree-Reweighted Belief Propagation Algorithms and Approximate ML Estimation by Pseudo-Moment Matching," in *AISTATS*, 2003.

[80] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous Systems*, vol. 32, no. 1, pp. 1–16, 2000.

[81] M. A. Sotelo, F. J. Rodriguez, L. Magdalena, L. M. Bergasa, and L. Boquete, "A Color Vision-Based Lane Tracking System for Autonomous Driving on Unmarked Roads," *Autonomous Robots*, vol. 16, no. 1, pp. 95–116, 2004.

[82] P. Chandran, M. John, S. Kumar, and M. N.S.R., "Road tracking using particle filters for Advanced Driver Assistance Systems," in *IEEE Intelligent Transportations Systems Conference (ITSC)*, Oct 2014, pp. 1408–1414.

[83] O. Miksik, "Rapid vanishing point estimation for general road detection," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 4844–4849.

[84] Y.-W. Seo and R. Rajkumar, "Utilizing instantaneous driving direction for enhancing lane-marking detection," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2014, pp. 170–175.

[85] P. Moghadam and J. F. Dong, "Road direction detection based on vanishing-point tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2012, pp. 1553–1560.

[86] Y.-W. Seo, "Detection and Tracking the Vanishing Point on the Horizon," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-14-07, May 2014.

[87] O. Ben-Shahar and S. W. Zucker, "The Perceptual Organization of Texture Flow: A Contextual Inference Approach," vol. 25, no. 4, April 2003.

[88] T. S. Lee, "Image representation using 2D Gabor wavelets," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 18, no. 10, pp. 959–971, Oct 1996.

[89] A. McCallum, "Efficiently Inducing Features of Conditional Random Fields," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[90] A. McCallum and W. Li, "Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, ser. CONLL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 188–191.

[91] F. Sha and F. Pereira, "Shallow Parsing with Conditional Random Fields," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational*

Linguistics on Human Language Technology - Volume 1, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 134–141.

[92] X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black, "Efficient Belief Propagation with Learned Higher-Order Markov Random Fields." in *European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3952. Springer, 2006, pp. 269–282.

[93] R. Paget and I. D. Longstaff, "Texture Synthesis Via a Noncausal Nonparametric Multiscale Markov Random Field," *IEEE Transactions on Image Processing*, vol. 7, no. 6, pp. 925–931, Jun 1998.

[94] S. Roth and M. Black, "Fields of Experts: a Framework for Learning Image Priors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, June 2005, pp. 860–867 vol. 2.

[95] P. Kohli, M. Kumar, and P. H. S. Torr, "P3 Beyond: Solving Energies with Higher Order Cliques," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007, pp. 1–8.

[96] B. B. Potetz, "Efficient Belief Propagation for Vision Using Linear Constraint Nodes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007, pp. 1–8.

[97] J. S. Yedidia, "Message-Passing Algorithms for Inference and Optimization," *Journal of Statistical Physics*, vol. 145, no. 4, pp. 860–890, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10955-011-0384-7

[98] J. Domke, "Tractable Learning and Inference in High-Treewidth Graphical Models," 2009.

[99] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "A New Class of Upper Bounds on the Log Partition Function," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, San Francisco, CA, USA, 2002, pp. 536–543.

[100] M. Wainwright and M. Jordan, "Graphical Models, Exponential Families, and Variational Inference," Tech. Rep., sep 2003, published: Technical Report 649.

[101] M. Nikolova, "Model distorsion in bayesianmap reconstruction," *Inverse Problems and Imaging*, vol. 1, no. 2, pp. 399–422, May 2007.

[102] H. Elliott, H. Derin, R. Cristi, and D. Geman, "Application of the Gibbs distribution to image segmentation," in *Internacional. Conference Acoustics Speech, and Signal Processing (ICASSP)*, vol. 9, Mar 1984, pp. 678–681.

[103] Wikipedia, "Hamming distance," [web page] http://en.wikipedia.org/wiki/Hamming/_distance, 2014.

[104] J. Marroquin, S. Mitter, and T. Poggio, "Probabilistic solution of ill-posed problems in computational vision," *Journal of the American Statistical Association*, vol. 82, pp. 76–89, 1987.

[105] Wikipedia, "Np-hard," [web page] http://en.wikipedia.org/wiki/NP-hard, 2014.

[106] S. Amari and H. Nagaoka, *Methods of information geometry*. Oxford University Press, 2000, vol. 191, translations of Mathematical Monographs.

[107] Wikipedia, "Convex conjugate," [web page] http://en.wikipedia.org/wiki/Convex_conjugate, 2015.

[108] T. Başar and P. Tamer, "Appendix b: Danskin's theorem," in *H-infinity Optimal Control and Related Minimax Design Problems*, ser. Modern Birkhäuser Classics. Birkhäuser Boston, 2008, pp. 383–389.

[109] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A Comparative Study of Energy Minimization Methods for Markov Random Fields with Smoothness-Based Priors," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 6, pp. 1068–1080, june 2008.

[110] H. Wymeersch, F. Penna, and V. Savic, "Uniformly Reweighted Belief Propagation for Estimation and Detection in Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 11, no. 4, pp. 1587–1595, April 2012.

[111] O. Meshi, A. Jaimovich, A. Globerson, and NirFriedman, "Convexifying the Bethe Free Energy." in *Conference on Uncertainty in Artificial Intelligence (UAI)*, J. Bilmes and A. Y. Ng, Eds. AUAI Press, 2009, pp. 402–410.

[112] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," [web page] http://eigen.tuxfamily.org, 2010.

[113] S. Kakade, Y. W. Teh, and S. T. Roweis, "An Alternate Objective Function for Markovian Fields," San Francisco, CA, USA, pp. 275–282, 2002.

[114] J. Domke, "Learning Convex Inference of Marginals." in *Conference on Uncertainty in Artificial Intelligence (UAI)*, D. A. McAllester and P. Petri Myllymäki, Eds. AUAI Press, 2008, pp. 137–144.

[115] ——, "Graphical Models/CRF toolbox," [web page] http://users.cecs.anu.edu.au/~jdomke/JGMT/, 2013.

[116] M. Schmidt, "minFunc," [web page] http://people.cs.ubc.ca/~schmidtm/Software/minFunc.html, 2014.

[117] Wikipedia, "Broyden-fletcher-goldfarb-shanno algorithm," [web page] http://en.wikipedia.org/wiki/Broyden-Fletcher-Goldfarb-Shanno_algorithm, 2014.

[118] C. Sutton and A. McCallum, "An Introduction to Conditional Random Fields," *Foundations and Trends in Machine Learning*, vol. 4, no. 4, p. 267?373, 2012.

[119] M. Milford, "Visual route recognition with a handful of bits," in *Robotics: Science and Systems (RSS)*, 2012.

[120] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, J. J. Yebes, and S. Gámez, "Bidirectional Loop Closure Detection on Panoramas for Visual Navigation," in *IEEE Intelligent Vehicles Symposium (IV)*, Dearborn, USA, June 2014, pp. 1378–1383.

[121] J. Shotton, J. M. Winn, R. Carsten, and A. Criminisi, "TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation." in *European Conference on Computer Vision (ECCV)*, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951. Springer, 2006, pp. 1–15.

[122] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *International Journal of Computer Vision (IJCV)*, vol. 81, no. 1, pp. 2–23, Jan. 2009.

[123] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *International Conference on Computer Vision (ICCV)*, 1999.

[124] J. Weinman, A. Hanson, and A. McCallum, "Sign Detection in Natural Images with Conditional Random Fields," in *Machine Learning for Signal Processing, 2004. Proceedings of the 2004 14th IEEE Signal Processing Society Workshop*. IEEE, 2004, pp. 549–558.

[125] E. W. V. Chaves, *Mecánica del Medio Continuo. Conceptos Básicos*, 2nd ed. Centro Internacional de Métodos Numéricos en Ingeniería (CIMNE), 2012.

[126] Wikipedia, "Hsl and hsv," [web page] http://en.wikipedia.org/wiki/HSL_and_HSV, 2014.

[127] ——, "Lab color space," [web page] http://en.wikipedia.org/wiki/Lab_and_HSV, 2014.

[128] ——, "Grayscale," [web page] http://en.wikipedia.org/wiki/Grayscale, 2014.

[129] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, June 2005, pp. 886–893.

[130] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 4, pp. 509–522, 2002.

[131] P. Dollár, "Piotr's Computer Vision Matlab Toolbox (PMT)," http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.

[132] T. Ojala and M. Pietikainen, "Unsupervised texture segmentation using feature distributions Pattern Recognition," in *Pattern Recognition*, 1996, pp. 477–486.

[133] G. Zhao, "Local Binary Pattern (LBP) implementation," [web page] http://www.cse.oulu.fi/CMV/, 2014.

[134] T. Ojala and M. P. M. . T. Mäenpäää, "A generalized Local Binary Pattern Operator for Multiresolution Gray Scale and Rotation Invariant Texture Classification." in *ICAPR 2001 Proceedings*, 2001, pp. 397 – 406.

[135] C. X. Ling and H. Zhang, "The Representational Power of Discrete Bayesian Networks," vol. 3, pp. 709–721, Mar. 2003.

[136] S. Fortmann-Roe, "Understanding the Bias-Variance tradeoff," [web page] http://scott.fortmann-roe.com/docs/BiasVariance.html/, 2012.

[137] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson Education, 2009.

[138] C. Wojek and B. Schiele, "A Dynamic Conditional Random Field Model for Joint Labeling of Object and Scene Classes." in *European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, D. A. Forsyth, P. H. S. Torr, and A. Zisserman, Eds., vol. 5305. Springer, 2008, pp. 733–747.

[139] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 recipes to master this library of programming functions for real-time computer vision*. Packt Publishing Ltd, 2011.

[140] P. Soille, *Morphological Image Analysis: Principles and Applications*, 2nd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[141] J. M. Alvarez, T. Gevers, and A. M. López, "3D Scene Priors for Road Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 57–64.

[142] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, USA, 2012, pp. 3354–3361.

[143] K. I. of Technology, "Annieway," [web page] http://www.mrt.kit.edu/annieway/, 2014.

[144] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2010.