Vision, Graphics and Interactive Systems

SEGMENTATION OF RGB-D INDOOR SCENES BY STACKING RANDOM FORESTS AND CONDITIONAL RANDOM FIELDS

May 28, 2015

Mikkel Thøgersen
Aalborg University
The Faculty of Engineering and Science
mthoge10@student.aau.dk

This master thesis is typeset using LaTeXand the template available from the department of electronic systems at Aalborg University.

AALBORG UNIVERSITY

Department of Electronic Systems
Frederik Bajers Vej 7B
9220 Aalborg Ø
Telephone (+45) 9940 8600
Fax (+45) 9940 9840
http://www.es.aau.dk

Title:
Segmentation of RGB-D Indoor Scenes by Stacking Random Forests and Conditional Random Fields

Theme:
Computer Vision

Project period:
Project start: 01-10-2014
Project end: 03-06-2015

Student:
Mikkel Thøgersen

Supervisors:
Sergio Escalera
Jordi Gonzàlez
Thomas B. Moeslund

No. printed Copies: 4

No. of Pages: 81

Total no. of pages: 89

Completed: 02-06-2015

Abstract:

This report proposes a new model using the Multi-class Multi-scale Stacked Sequential Learning framework as the solution to the problem of indoor semantic segmentation. Following recent trends in state-of-the-art, a base classifier uses an initial SLIC segmentation to obtain superpixels which provide a diminution of data while retaining object boundaries. A series of color and depth features are extracted from the superpixels and are used in a Conditional Random Field to predict superpixel labels. Furthermore, a Random Forest classifier using random offset features is also used as an input to the Conditional Random Field, acting as an initial prediction. As a stacked classifier, another Random Forest is used which acts on a spatial multi-scale decomposition of the confidence map to correct the erroneous labels assigned by the previous classifier. The model is tested and trained on the popular NYU-v2 dataset. The approach shows that simple features with the power of the MMSSL framework can achieve better performance than similar methods.

Preface

This thesis sums up the work done over the academic year 2014/2015. It began during the autumn semester in Barcelona at the Computer Vision Center at the Autonomous University of Barcelona and was continued in the spring semester at Aalborg University.

Upon finishing the work, a paper was submitted to the BMVC conference for review as the work shows promising results. The same paper is presented here as the main submission. Added are the worksheets, which details the implementations described in the presented paper.

Finally I would like to thank my three supervisors, Sergio, Jordi and Thomas, who all helped me and guided me through the work.

- Mikkel Thøgersen

Aalborg University, June 2, 2015

Mikkel Thøgersen
mthoge10@student.aau.dk

Contents

I Introduction

In the not so distant future, the statistics shows that there will be a majority of elderly people in a number of first-world countries. This is a result of the well-known fact that as societies become richer, the fertility rate falls and the life expectancy increases. The amount of social care needed to support these elderly people can become a difficulty for the social systems of said countries. One of the solutions, that is heavily investigated, in for example Japan, is to create social care robots to take on some of the tasks of the care takers [4].

Large companies, such as Honda and Toyota, already have prototype home service robots like the ASIMO and the Toyota Partner Robots [2, 1]. The main challenge is to control the actions of the robots, which is a combination of a range of different multimodal sensors and inputs where the vision system usually plays an important role.

Imagine a person asks a robot to get a cup from the kitchen. Although this seems as an easy task for humans, the robot has to be quite sophisticated to perform such tasks. Assume that the robot has understood the task. From here it needs to have an idea of the location of the room semantically labeled "kitchen" and navigate through a typical indoor environment, which can be cluttered and might be hard to navigate. When the robot finally finds the kitchen, it has to detect the cup, which again, might not be visible as cups are often in cupboards.

This simple example shows the complexity associated with a simple daily life task, and it displays the difficulties the robot have to be able to handle. And here all the controls of the robot, and things such as balance and grasping motions [32], are left out.

Hence, one of the challenges in home service robotics lies in computer vision and in particular robots perception of its surroundings.

i Problem Statement

In order to enable aware and adaptive robots that can react and interact with their surroundings, a visual perceptual sense have to be imitated through the use of cameras. In particular it is important to segment a scene and label each entity in the scene with some semantic label. By separating a scene into semantic groups, further

analysis and action can be applied and a range of features can be assigned to each group. In [28], Silberman *et al* defines four groups: *floor*, *structure*, *furniture* and *small objects*. Each of these groups have a number of properties that can be valuable for a robotic application. Consider the semantic group *structure*, it signifies that it is a structural entity in the scene such as walls or ceiling, hence it can be safely assumed that objects of this label cannot be passed and cannot be moved. Furniture and small objects on the other hand can for the most part be manipulated, moved or lifted.

The aim of the project is to create a vision based system, capable of semantically categorizing objects in indoor cluttered scenes using vision sensors.

Specifically, this report will focus on the machine learning techniques: Conditional Random Fields (CRF), Random Forests (RFs) and the lesser known Multi-class Multi-scale Stacked Sequential Learning (MMSSL) framework. As input, a collection of kinect sensor images will be used, specifically the NYU-v2 dataset[28] used for the current state-of-the-art approaches. The output will be a segmentation of the images, into the four semantic labels.

ii Reading Guide

This master thesis is structured in two main parts, first a summary of the work and a conclusion is presented for a fast overview. Following, a paper is presented which was submitted to the British Machine Vision Conference (BMVC) during this work. It holds the essence of the project and should be read as an appetizer. Note that the paper has its own bibliography list found at the end of it.

Following the paper are the worksheets where the methods used are clearly detailed and discussed as opposed to the paper that only summarizes the work. The worksheets are chronologically written, such that they can be read independently as well as sporadic.

The figures in the thesis and the paper are created by the author, except if there is a reference in the caption. In that case, the figure originates from the reference. Furthermore, the figures are either created using Matlab[1], CreativeDocs.NET[2], paint.NET[3] or Microsoft Office Word 2013[4]. Also, the reference style used is the IEEE format[5], a simple [number] in the text. When the references are important,

[1] http://se.mathworks.com/products/matlab/, accessed June 2015
[2] http://www.creativedocs.net/, accessed June 2015
[3] http://www.getpaint.net/index.html, accessed June 2015
[4] https://products.office.com/da-DK/word, accessed June 2015
[5] http://www.ieee.org/documents/ieeecitationref.pdf

the names of the authors are mentioned in text.

Products and software are referenced using a footnote, in which the website of the product is listed together with the date it was accessed. If it is important software the name of the creator is mentioned in text. Enclosed with the maser thesis is also a DVD, which contains:

- The thesis, named 'Master Thesis.pdf'
- A download of all websites referenced in the thesis, contained in the folder 'Websites'
- Some result images in the folder 'Images'
- Some results and intermediate results in .mat format, enclosed in the zip file 'Samples.zip'

iii Summary

In this master thesis the problem of indoor semantic segmentation is addressed. In particular, the NYU-v2 dataset is used as the basis for the project. The dataset consists of both color and depth images of a wide variety of scenes, from kitchens to public libraries. The problem is investigated using the four semantic super classes: floor, structure, furniture and small objects, originally proposed by the creators of the dataset [28]. Several works on this problem are considered, many of them rely on an initial over segmentation and a consecutive contextual classifier to label the segments. A similar approach is taken in this work.

As an over segmentation method, the SLIC segmentation is used for its accuracy and speed. It is a proven method for this problem as well as other problems. Each superpixel from the SLIC segmentation is processed and a series of cheap and simple features are extracted, utilizing both color and depth modalities. These features are merged as the input for a conditional random field (CRF), that will act as the contextual classifier. The CRF is used in several similar works and has an advantage from its ability to consider neighborhoods and the resulting smooth labeling.

This model alone achieves good results compared to literature. The three main classes, floor, structure and furniture are found with good accuracy, but as others have similarly noticed the performance is rather bad on the small objects class. This is a results of the fact that the class spans a wide variety of objects. To increase the performance, a random forest is added to give initial predictions as an input to the CRF. This gives an increase in the accuracy w.r.t the small objects, but unfortunately at the expense of accuracy of the three main classes.

Finally, to excel the model the multi-scale stacked sequential learning framework is applied. Following the previous work on this method, a new classifier is added on top of the existing system. The machine learning method used for this classifier is arbitrary, the essence of the framework is the use of a multi-scale decomposition of the confidence maps of the existing model. Essentially, it looks at the input *and* the output of the existing model and tries to improve the performance by learning when it makes mistakes. The multi-scale decomposition is a contextual feature and captures neighborhood dependencies of the query pixel and this can further make the classifications more coherent.

The system is tested on the NYU-v2 dataset and achieves state-of-the-art results.

iv Conclusion and Discussion

In this work a new model for indoor semantic segmentation has been presented. It is based on current state-of-the-art approaches and is evolved by adding a previously unconsidered model - multi-scale stacked sequential learning - to the problem. Furthermore, the work has shown that the method is useful for the problem and can enhance the accuracy of the method. The results show state-of-the-art results on the

main measures. Also, the addition of the framework is novel in that it has not, to my knowledge, been applied on RGB-D or 2.5D data before.

Many of the features used in the project are dependent on finding the main axis of the room. For this specific task, a structured way of finding them are presented using mean shift clustering and a specialized evaluation function. This same principle have been shown previously, here however, an exact method and evaluation is specified.

Overall, this project has shown a viable method for generating semantic labels for indoor scenes. In a future robotic application these methods will be a key component to achieve autonomous behavior.

The RGB-D sensors necessary for these applications become more common and cheap to acquire, and the field of this research is increasingly relevant. These types of sensors have yet to break completely through, but recent commercial interest makes them even more viable. In particular the recent announcement of projects like Google Tango, Microsoft HoloLens and the Intel RealSense will be catalysts for the technology and research.

The mainstreaming of these technologies could further help the development of home service robots, as the technology becomes familiar and more developers, as well as researches, become confident in the technology.

Future Work

Several interesting aspects of the project could be explored. For instance, there are several ways to extend the capabilities of the CRF: extending the neighborhood, adding higher-order potentials or making it densely connected.

One feature that could hold potential for more discriminative power is the room layout feature. The idea behind the feature is to flip the scene such that it is seen from above. This top view has the advantage that all objects become are aligned to the floor and thereby to a common plane. Many objects such as chairs, tables etc. are always aligned with the floor and this can be an advantage for segmentation and recognition purposes. It effectively dampens the inherit problem of camera view points. Consequently, instead of seeing objects from different view points, they could always be viewed from above, leaving only rotation about themselves as a problem. Following the same idea, well known scale and rotation-invariant feature descriptors, such as SURF, could be applied to find the objects in the top view.

II BMVC Paper

Segmentation of RGB-D Indoor scenes by Stacking Random Forests and Conditional Random Fields

Mikkel Thøgersen[1]
mthoge10@student.aau.dk
Sergio Escalera[2]
sergio@maia.ub.es
Jordi Gonzalez[3]
poal@cvc.uab.es
Thomas B. Moeslund[1]
tbm@create.aau.dk

[1] Aalborg University
Aalborg, DK
[2] University of Barcelona
Barcelona, Spain
[3] Computer Vision Center
Autonomous University of Barcelona
Barcelona, Spain

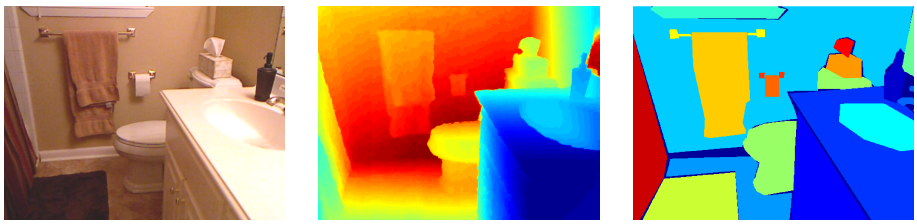
Abstract

This paper proposes a technique for RGB-D scene segmentation using Multi-class Multi-scale Stacked Sequential Learning (MMSSL) paradigm. Following recent trends in state-of-the-art, a base classifier uses an initial SLIC segmentation to obtain superpixels which provide a diminution of data while retaining object boundaries. A series of color and depth features are extracted from the superpixels, and are used in a Conditional Random Field (CRF) to predict superpixel labels. Furthermore, a Random Forest (RF) classifier using random offset features is also used as an input to the CRF, acting as an initial prediction. As a stacked classifier, another Random Forest is used acting on a spatial multi-scale decomposition of the CRF confidence map to correct the erroneous labels assigned by the previous classifier. The model is tested on the popular NYU-v2 dataset. The approach shows that simple multi-modal features with the power of the MMSSL paradigm can achieve better performance than state of the art results on the same dataset.

1 Introduction

Through cheap and readily available multimodal sensors, current and future robotics can gain a perception of their surroundings. To be useful, however, a meaning have to be extracted from the sensor data. As such, semantic segmentation of scenes is a foundation for robots and machines to navigate and act in the human-centric world.

A number of works have investigated the problem of indoor semantic segmentation previously. The majority of them are focused on a particular dataset, the NYUv2 dataset provided by Silberman et al.[14]. A sample image from the dataset is shown in figure 1. While the dataset has 894 classes in total, most works consider a semantic 4-class problem originally stated in [14]. Others increase the number of classes to 13 and 40. In this work, 4-class segmentation is tested. The dataset is a sequel to their first dataset, NYU-v1[13], where the authors proposed the initiating solution to this problem and provided the first densely labeled



(a) (b) (c)

Figure 1: (a) Sample data from the NYU-v2 dataset; (b) Normalized in-painted depth map; (c) ground truth with all classes.

RGB-D dataset for indoor semantic segmentation. Their approach was based on three main types of potentials in an energy function: a unary appearance potential incorporating a range of feature descriptors including location priors, SIFT features etc.; a class transition potential and lastly a spatial smoothness term. In a more recent work by Khan et al. [8] the dominant lines of the scene are used to align the scene to the major surfaces, e.g. the floors and walls. Using a combination of color and depth based edges the scene is superpixelated using a k-means clustering, the resulting regions are subsequently fitted with a plane and features are extracted and combined in a Conditional Random Field (CRF).

Extending on the previous ideas, Müller and Behnke[11] used a Random Forest approach to do an initial pixel-wise class prediction, as originally proposed in [16]. This initial segmentation is later aggregated in a SLIC segmentation on which a CRF infer the final class labels. Their approach showed state-of-the-art results. In a very recent paper by Hamedani and Harati[6], they propose a multi-scale CRF, where a downsampled version of the image is inferred and the predictions are propagated through a multi-scale pyramid. They also use a temporal pairwise potential to enforce the beliefs through video sequences.

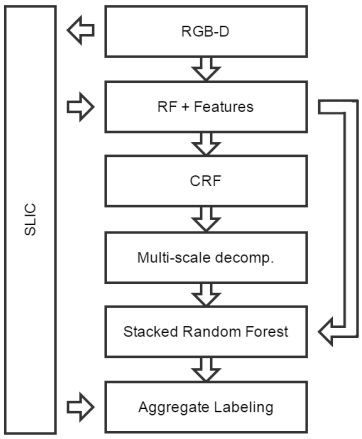
Puertas *et al*. in [12] present the Multi-class Multi-scale Stacked Sequential Learning (MMSSL) framework. Two classifiers are connected, where the second classifier acts on the spatial predictions of the first classifier combined with the original set of features. Results showed improved performance in 1D and 2D sequential problems. The framework however has not been previously tested on 3D data for indoor semantic segmentation.

In this work we explore and enhance current work on semantic segmentation in indoor cluttered scenes. Specifically, this paper focuses on introducing the MMSSL framework on the methods that have previously been proven effective for the problem[6, 8, 11]. Specifically, a CRF-based approach with an initial over segmentation acts as the base classifier. An RF is trained as the stacked learner and the dataset is extended using a spatial multi scale interpretation of the predictions from the CRF. The model is tested on the popular NYU-v2 dataset, outperforming state of the art results.

The rest of the paper is organized as follows. Section 2 presents the approach for semantic RGB-D scene segmentation, describing features and stacked classifiers. Section 3 shows the performed experiments. Finally, Section 4 concludes the paper.

2 RGB-D Scene Segmentation

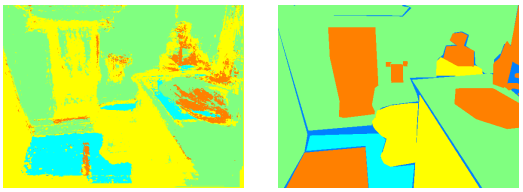
Our proposed technique for RGB-D Scene Segmentation trains an initial RF [11] to approximate class predictions. The confidence map of the RF together with an array of features are



(a)

Parameter	value
Num. of features	40
Num. of samples pr. image	1962
Num. of trees	20
Num. of sample at each decision split	13
Minimum num. of samples at leaf	10

(b)



(c)

Figure 2: (a) Model overview. Features are extracted from the RGB-D images, aggregated in superpixels and fed to the CRF. A multi-scale decomposition of the predictions from the CRF is created and used in an RF which predicts the final labels. (b) Parameters for the initial Random Forest. (c) Shows the Random Forest response together with the ground truth on the right.

accumulated into an over segmentation on which a CRF is trained. Applying the base model onto the training data gives a class confidence for each pixel. These are added back into the existing feature set in a multi-scale decomposition. Finally, an RF is trained on the extended feature set and lastly the results are aggregated back into the over segmentation to form the final predictions. The model pipeline is depicted in figure 2(a).

2.1 Base Classifier

Similar to other state-of-the-art approaches [11, 13, 14] the structure of the base classifier is centered around a CRF, that acts on an initial over segmentation of the input imagery. The outcome is a classification of the individual superpixels. The superpixel segmentation employed is the localized, k-means based SLIC algorithm by Zitnick and Kang[1] which has become popular for its speed and ability to create near uniform segments while preserving contours and homogeneity. It is an important part of the model, as it defines the boundaries of the segmentation and some features are derived from it. Also, as [5, 11], a Random Forest classifier predictions are embedded in the input for the CRF. The following subsections describes the features used in the base model.

2.2 Random Forest

An initial estimate of the classes is obtained through the use of a Random Forest with random offset features, that captures subtle details of either the depth channel or the Lab color space channels. This approach is similar to the one originally proposed in [15]. Random offset features are calculated as the difference between the sum of two patches at random offsets from the query pixel. Initially thousands of randomized offsets and patch sizes are trained

on the data. Following, the out-of-bag samples are tested with the trained model and thus the discriminative power of each of the random features can be evaluated. Based on this a subset of the random features can be selected and the model is retrained to obtain the final model. The RF is trained with the specifications shown in table 2(b) together with the the response from the trained RF is shown in figure 2(c).

2.3 Generic Features

A set of primitives is extracted from the data, as these are used as a basis for several of the features. These are the trivial Cartesian coordinates and the normals. The normals are calculated as the cross product estimates [9]. Several more elaborate methods are available, however in this context the normals will be averaged over regions which will act as a filtering.

From the obtained data, a set of features are readily available. For the individual superpixels, the features include: Color using the Lab color space, the normal and the standard deviation of the normal which captures curvature in either direction. Also, a blurred gradient of the image and the depth map are used, to provide clues on areas with high change of color. For the case of pairwise potentials, all of the above and 3D Cartesian coordinates are used. Most are calculated as the absolute difference with the exception of the color and normal features. The pairwise color feature is modulated to penalize in a non-linear fashion:

$$f_{\text{colDiff}}(R_1, R_2) = \exp(-\beta \ ||\mathbf{c}_1 - \mathbf{c}_2||^2), \tag{1}$$

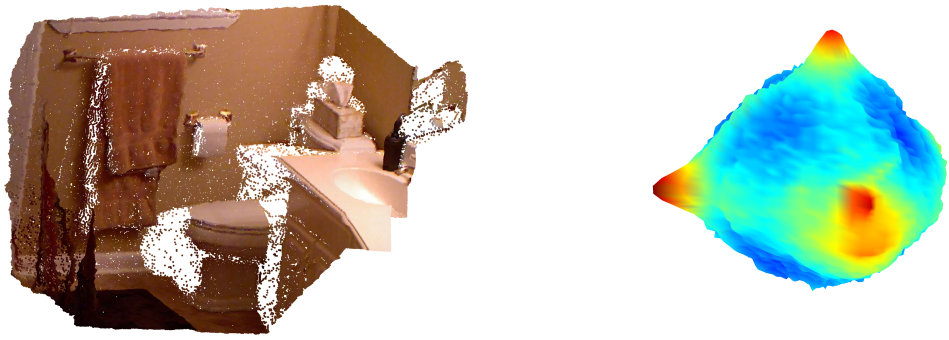
where R_1 and R_2 are two adjacent regions, $\mathbf{c}_1$ and $\mathbf{c}_2$ are the Lab colors of the superpixels and β controls the attenuation of the feature. The normals are compared using a normalization of the difference of angles and they are also compared w.r.t their inclination in spherical coordinates.

2.4 Dominant normal direction

An important concept of the model is estimating the layout of the room and finding the dominant axes of the room, e.g. the floor and wall normals. For indoor scenes the Manhattan principle is the assumption that lines and edges in indoor imagery are aligned with either the floor or the walls, first coined in [3]. Extending this to surfaces is a natural step, as the edges and lines only exist where surfaces meet. Based on this extended Manhattan principle - that major surfaces are aligned with the floor or the walls - the normals of the scene are investigated using a Mean Shift Clustering, providing candidate dominant directions. As several authors [5, 11, 14] similarly have noticed, the floor normal can be found by taking the candidate nearest to the upwards axis of the estimated Cartesian coordinate system. A visualization of the normals and their distribution for a given scene is shown in figure 3. In this model, the candidates, $\boldsymbol{P}$, from the mean shift clustering are evaluated using the following expression:

$$\mathbf{n}_{\text{floor}} = \underset{p \in \boldsymbol{P}}{\text{argmax}} \ \exp\left[-\left(\frac{|p_\theta - \theta_{\text{std}}|}{180} \right)^{\lambda_f} \quad \left(1 - \frac{p_\mu}{\sum_{p \in \boldsymbol{P}} p_\mu} \right) \right]. \tag{2}$$

In addition to choosing the candidate, p, with a normal direction, p_θ, close to some standard floor inclination, θ_{std}, the evaluation ensures that there is sufficient support - e.g. the number of normals p_μ pointing in the same direction as the candidate. Also the importance of inclination compared to support is controlled by adjusting the λ_f-parameter.



(a) (b)

Figure 3: (a) Shows a scene plotted in 3D Cartesian coordinates, besides it, in (b) a plot of the concentration of normals is shown, it is created using a histogram of the normals, where the concentrations are plotted as elevations on a sphere, corresponding to the direction of the normals. Notice the three main bulges (red color), they correspond to the three dominant normal directions in the image.

Once the floor direction is determined, the wall and ceiling normals are found in a similar fashion from the remaining dominant normal candidates. When finding the wall normals, the assumption is that candidate directions are perpendicular to the floor normal and to other candidates - assuming that rooms are rectangular. This does not hold for rooms with more complex walls, but performs well in practice. Initially the vertical angle difference, $p_{\hat{\theta}}$, between the floor normal and the candidates is calculated and 90° is subtracted, so that candidates with an angle difference close to zero are more likely to be wall. Afterwards the process continues iteratively following these steps:

- (This step is skipped on first iteration). The candidates are compared with the found wall normals in the horizontal direction by finding the angle difference, $p_{\hat{\phi}}$, and subtracting whichever multiple of 90° is closest to the found value. This has the effect that candidates that are perpendicular or opposing to the already found wall normals are most likely to be wall.
- On every iteration the remaining candidates are evaluated using:

$$\mathbf{n}_{\text{wall}} = \underset{p \in \boldsymbol{P}}{\text{argmax}} \exp\left[-\left(\frac{|p_{\hat{\theta}} + p_{\hat{\phi}}|}{180} \right)^{\lambda_w} \left(1 - \frac{p_\mu}{\sum_{p \in \boldsymbol{P}} p_\mu} \right) \right], \tag{3}$$

  where λ_w controls the importance of either term. If the evaluation equation returns a value greater than some threshold, the candidate is accepted as a wall normal, otherwise the process is stopped.

Knowing the normals of each structure entity; walls and floor, the surfaces are found in the scene by fitting a plane to the furthest point from the viewpoint, in the opposite direction of the respective normal. This can be used for assigning a wall and floor probability.

Because the floor normal is known, it is possible to find all upwards pointing surfaces by comparing normals using the dot product and an exponentiation:

$$f_{\text{flat}}(\mathbf{n}) = (\mathbf{n} \cdot \mathbf{n}_{\text{floor}} - 1)^\alpha, \tag{4}$$

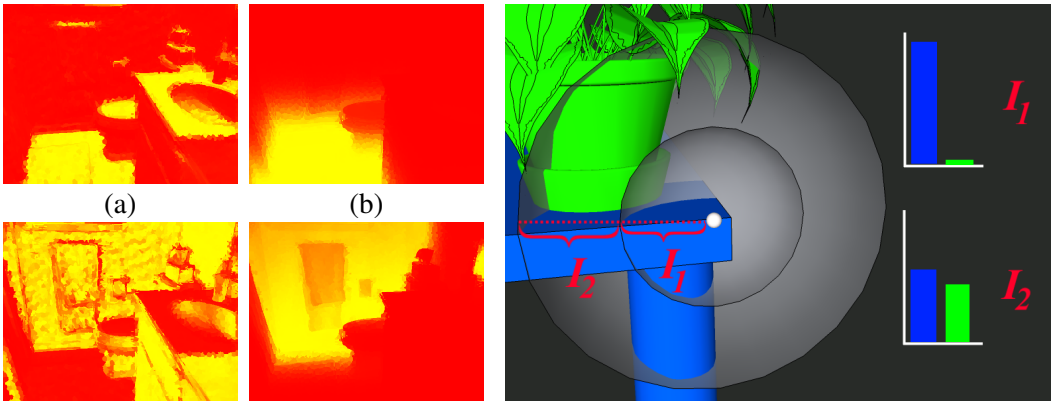


Figure 4: Normal features based on the dominant normal directions. The original image is shown in figure 1. (a) Intensity image of the vertical normal feature. (b) Intensity image of the floor probability. (c) Similar to (a), the wall normal feature compares the normals to the found wall normal directions. Lastly, (d) is similar to (b), using the normals of the walls. (e) Depiction of the multi-scale decomposition. From the query point in the center, there are two intervals, I_1 and I_2 that each generates a sphere. In each interval, the distribution of the confidence map is extracted, which gives the resulting feature vector.

where α attenuates the feature. This, amongst other similar features, is shown in figure 4.

The floor plane provides a means of finding the height of each individual superpixel. To make the feature more descriptive, it is divided into 20 bins, spanning 2.5 meters, assuming that most scenes do not exceed this height. In case they do, they are assigned to the last bin.

Another feature, derived from the dominant normal direction, is the room layout feature. The idea is simple: flipping the 3D scene, such that it is viewed from above. Once this is done, the walls can be found by following the camera rays out in every direction and picking the point furthest away. Following, a probability can be assigned to all points that are vertically aligned with this first point. This feature and assumptions are inspired by the work of Cadena and Košecka [2].

Another important derivation from the room layout, is the inverse of the feature. When the most distant pixels are found in each view direction, the opposite of this, must be everything *inside* the room.

The normal features described are also used in the pairwise potentials as absolute differences, except for the height feature where the absolute height difference between two superpixels is used instead of the binning described above.

2.5 Multi-scale Sequential Stacked Classifier

The MMSSL framework is a way to improve the results of a model by adding another discriminative classifier on top of the existing one. The stacked classifier is trained with a multi-scale decomposition of the confidence map of the base classifier as well as the original feature set. This effectively corrects errors of the base classifier.

The multi-scale decomposition is generated by analyzing the distribution of nearby pixels confidence maps in multiple distances from the point in question. For a point p with the pixels P_i situated in some half-closed distance intervals from p denoted by $i = \{1,2,...,n\}$,

Feature sets	Node	Edge
Generic features	G	
Lab color	•	•
3D position		•
Normal	•	•
Std. dev. of the normal	•	•
Blurred depth gradient magnitude	•	•
Blurred image gradient magnitude	•	•
Room Layout features	RL	
Room layout wall probability	•	•
Inverse room layout wall probability	•	•

Feature sets	Node	Edge
Normal features	N	
Discrete height	•	•
Vertical normal comparison	•	•
Wall normals comparison	•	•
Floor probability	•	•
Wall probability	•	•
Continuous height		•

Table 1: Feature reference table.

the multi-scale decomposition is the averaged distribution of the confidence map, C, of each class $c = \{1,2,...,k\}$ in each interval. It can be described as:

$$a_{n(c-1)+i} = \frac{\sum C_c(P_i)}{|P_i|}, \tag{5}$$

where C_c refers to the confidence map for the class c. The vector a is the resulting feature vector for p. In effect, this decomposition creates a feature vector that is linear in size with the number of classes, *e.g.* kn, as a consequence the framework is suited for problems with a relatively low number of classes, although it can be compressed when dealing with a large number of classes[12]. Notice that the points in P_i are found as the points with a euclidean distance from the query point p falling in the interval i. A depiction of the decomposition is shown in figure 4(e).

The multi-scale decomposition is added to the initial features set to form an extended feature set. An RF is trained on this extended dataset an finally the predictions are aggregated in the superpixel segmentation giving the final labeling.

3 Experiments

The proposed model is tested on the NYU-v2 dataset composed of 1449 densely labeled RGB-D images captured using a Kinect camera. Because of the reduced field of view of the Kinect, the usual 640×480 resolution of the cameras RGB channels are reduced to 561×427 and matched with the depth maps. Furthermore the dataset is provided the with in-painted depth maps, solving the common problem of filling holes in data from depth-sensing technologies. The manual colorization method of Levin *et al.* [10] is used for the in-painting. The dataset includes a standard data split, introduced with the first paper on the dataset [14]. This gives a total of 795 densely labeled training images and 654 test images spanning various different scenarios, from kitchens to public libraries. Following the available literature on the subject, the two main measures are per class accuracy and pixel accuracy.

3.1 Ablation Study

To verify the importance of features and methods, different settings are tested. The baseline is given as the generic features with the conditional random field. Each setting is shown in table 2, where the feature groups have been abbreviated according to table 1.

Methods	Features	Floor	Struct.	Furn.	Props	Pr. class acc.	Pix. acc.
CRF	G	85.3	77.6	60.5	1.8	56.3	60.0
CRF	G+N	92.8	77.3	**78.0**	14.5	65.7	68.9
CRF	G+RL	83.0	76.4	74.2	2.7	59.1	64.1
CRF	G+N+RL	94.3	78.8	81.1	13.8	67.0	70.6
CRF+RF	G+N+RL	93.3	79.4	74.0	33.4	70.0	71.5
CRF+RF+RFS	G+N+RL+S	**95.5**	**80.5**	77.1	**35.3**	**72.1**	**73.8**

Table 2: Table of feature sets used in the ablation study. The abbreviations used in the table refer to the sets shown in table 1.

Work	Floor	Structure	Furniture	Props	Per class acc.	Pix. acc.
Müller and Behnke[11]	94.9	78.9	71.1	42.7	71.9	72.3
Couprie *et al.* [4]	87.3	87.8	45.3	35.5	63.5	64.5
Khan *et al.* [8]	87.1	**88.2**	54.7	32.6	65.6	69.2
Gupta *et al.* [5]	82	73	64	37	65	64.9
Nico Höft *et al.* [7]	77.9	65.4	55.9	**49.9**	62.0	61.1
Ours	**95.5**	80.5	**77.1**	35.3	**72.1**	**73.8**

Table 3: State of the art comparison.

The CRF alone with the generic feature set is discriminative enough to get fair results on the three classes with least variance, whereas it fails to classify the props class, which spans a wide variety of objects. Adding the normal features with especially the vertical normal feature gives better results on the props class, as it can be used to separate objects on tables and on the floor. In general, it increases the performance across all classes.

To test whether the room layout feature has a positive effect, it is tested by itself with the CRF, showing that it adds some descriptive power. However, not as much as the normal features. This appeals to the intuitive sense, as it can only tell whether a given pixel is *inside* the bounding walls of the room or is in fact the walls.

As expected, combining the two feature sets shows an increase on all measures. Adding the RF initial predictions gives a large boost to the props class. It captures subtle details of data set and this pays off. Unfortunately it takes its toll on the furniture class. This is probably due to the fact that the props label is often assigned to areas with a high change in contrast, which often is on furniture. This effect is also visible in the results shown in figure 5. Finally adding the stacked RF and the multi-scale decomposition improves accuracy to all classes from the predictions of the base classifier. While the CRF is useful at pruning impurities in region classifications, it also has a tendency to merge across regions, where it is unwanted. The added contextual awareness of the multi-scale decomposition can prune these faulty labelled regions.

3.2 Results and Comparisons

In figure 5, sample results are displayed, along with the predictions at each step of the model. The initial segmentation using only the RF shows a very rough segmentation, with a high amount of impurities and scattered labellings. This is mended in the CRF with the SLIC segmentation, where the resulting labellings are coherent and respect most object boundaries. It does however have a tendency to merge across boundaries. This problem is solved using the stacked RF; notice how in the third image column, that the hole underneath the arm rest of the chair is labelled correctly as floor. This is one of the ways the multi-scale decomposition works with contextual awareness.

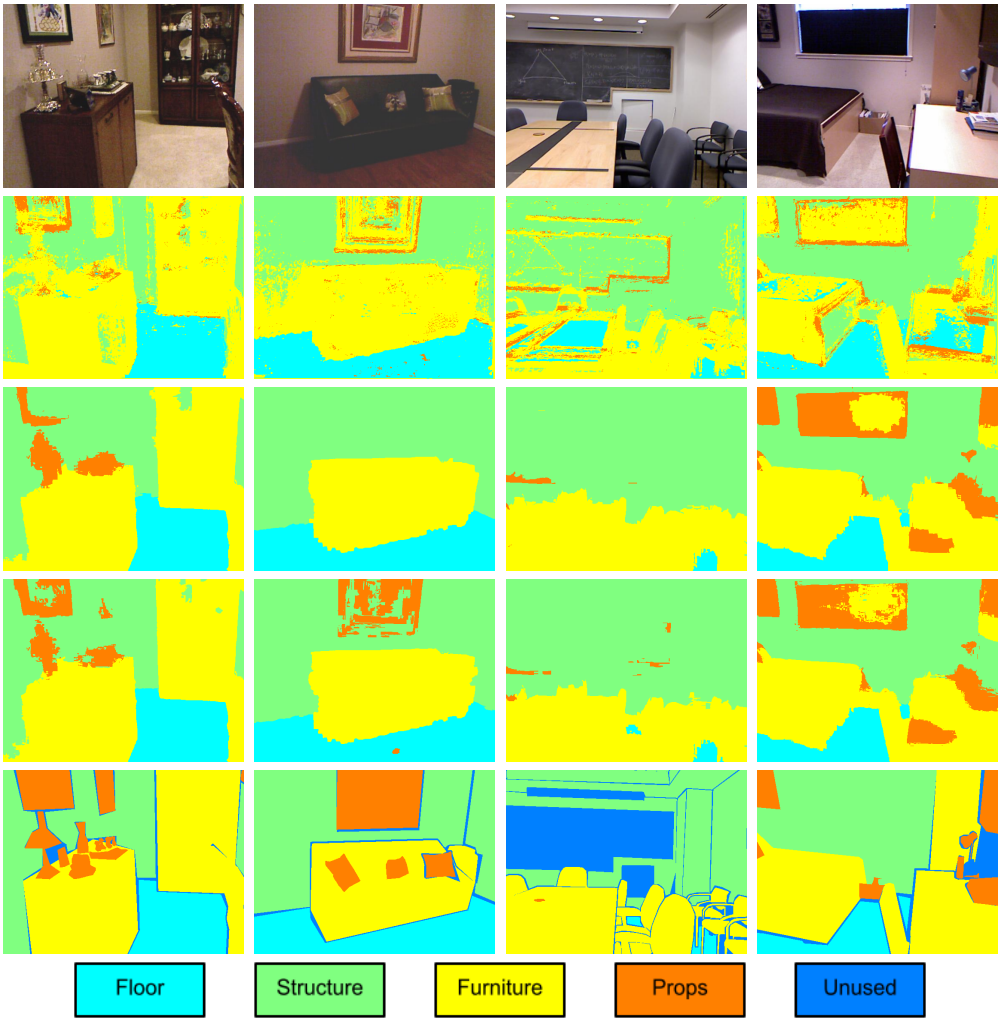


Figure 5: Sample results. The rows corresponds to: 1) Original RGB images, 2) initial RF label predictions, 3) CRF predictions, 4) full model predictions, 5) ground truth and finally the class labels.

To show the comparative performance, table 3 shows the results of current and previous state-of-the-art methods with the measures: average class accuracy, which is equivalent to the mean of the confusion matrix diagonal and the pixel accuracy. The unlabeled pixels in the ground truth are disregarded and are not considered for these measures. As shown in the table our approach excel on some measures and obtains state-of-the-art results.

3.3 Conclusion and Discussion

We presented a model for semantic segmentation of cluttered indoor scenes. The method bases on the stacking of Random Forest and Conditional Random Field results. The results show that the model excels the CRF classifier and adds a heightened contextual awareness to the model by including the multi-scale decomposition. The model is tested on a public data set, showing comparable and better performance than state of the art approaches.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, Nov 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.120.

[2] César Cadena and Jana Košecka. Semantic parsing for priming object detection in rgb-d scenes. *The International Journal of Robotics Research*, 34:582Ű597, April 2015.

[3] James M. Coughlan and Alan L. Yuille. Manhattan world: Orientation and outlier detection by bayesian inference. *Neural Computation*, 15:1063–1088, 2003.

[4] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *CoRR*, abs/1301.3572, 2013. URL http://arxiv.org/abs/1301.3572.

[5] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 564–571. IEEE, 2013.

[6] Taha Hamedani and Ahad Harati. Multi scale crf based rgb-d image segmentation using inter frames potentials. *RSI/ISM International Conference on Robotics and Mechatronics*, 2:920–925, 2014.

[7] Nico Höft, Hannes Schulz, and Sven Behnke. Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks. *Advances in Artificial Intelligence*, 8736:80–85, 2014.

[8] Salman Hameed Khan, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Geometry driven semantic labeling of indoor scenes. In *Computer Vision ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 679–694. Springer International Publishing, 2014. ISBN 978-3-319-10589-5.

[9] Klaas Klasing, Daniel Althoff, Dirk Wollherr, and Martin Buss. Comparison of surface normal estimation methods for range sensing applications. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3206–3211. IEEE, 2009.

[10] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 689–694, New York, NY, USA, 2004. ACM. doi: 10.1145/1186562.1015780. URL http://doi.acm.org/10.1145/1186562.1015780.

[11] Andreas Christian Müller and Sven Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6232–6237, May 2014.

[12] Eloi Puertas, Sergio Escalera, and Oriol Pujol. Generalized multi-scale stacked sequential learning for multi-class classification. *Pattern Analysis and Applications*, 18(2):247–261, 2015. ISSN 1433-7541. doi: 10.1007/s10044-013-0333-y. URL http://dx.doi.org/10.1007/s10044-013-0333-y.

[13] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 601–608, Nov 2011.

[14] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV'12 Proceedings of the 12th European conference on Computer Vision*, volume 5, pages 746–760, 2012.

[15] Jörg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense real-time mapping of object-class semantics from rgb-d video. *To appear in Journal of Real-Time Image Processing*, 2013.

[16] Jorg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense real-time mapping of object-class semantics from rgb-d video. *Journal of Real-Time Image Processing*, pages 1–11, 2013. ISSN 1861-8200. doi: 10.1007/s11554-013-0379-5. URL http://dx.doi.org/10.1007/s11554-013-0379-5.

III WORKSHEETS

Chapter 1

Problem Analysis

A number of works have investigated the problem of indoor semantic segmentation previously. The majority of them are focused on a particular dataset, the NYUv2 dataset provided by Silberman *et al* [28]. A sample image from the dataset is shown in figure 1.1. While the dataset has 894 classes in total, most works consider a semantic 4-class problem originally stated in [28], which is also the problem considered here. The dataset is a sequel to their first dataset, NYU-v1[27], where they proposed the initiating solution to this problem and provided the first densely labeled RGB-D dataset for indoor semantic segmentation. Their approach was based on three main types of potentials in an energy function: a unary appearance potential incorporating a range of feature descriptors including location priors, SIFT features etc.; a class transition potential and lastly a spatial smoothness term.

In a more recent work by Khan et al. [17] the dominant lines of the scene is used to align the scene to the major surfaces, e.g. the floors and walls. Using a combination of color and depth based potentials the scene is superpixelated using a k-means clustering, the resulting regions are subsequently fitted with a plane and features are extracted and combined in a CRF.

Extending on the previous ideas, Müller and Behnke[21] used a Random Forest (RF) approach to do an initial pixel-wise class prediction, as originally proposed in [29]. This initial segmentation is later aggregated in a SLIC segmentation on which a CRF infer the final class labels. Their approach showed state-of-the-art results. In a very recent paper by Hamedani and Harati[14] they proposed a multi-scale CRF, where the idea is to infer a downsampled version of the image and let the predictions propagate through a multi-scale pyramid. They also use a temporal pairwise potential to enforce the beliefs through video sequences. In [35], Zheng et al. uses a color and texture based approach, with TextonBoost and other features as input to a CRF to attempt to assign attributes such as 'shiny' or 'matte' to pixels. Furthermore, they create a fully connected CRF over the image and infer pixel-wise object classes.

Cohen and Carvalhos [8] shows how using a base classifier with a consecutive classifier acting on the predictions of the base classifier can improve segmentation

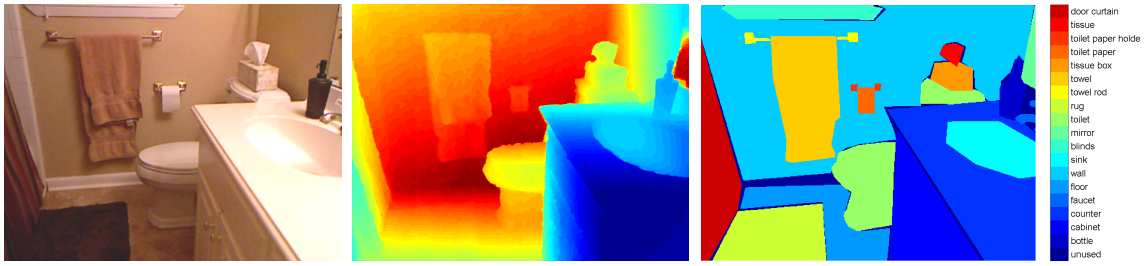


Figure 1.1: Sample data from the NYU-v2 dataset. On the left, enhanced brightness color image for easier viewing, normalized in-filled depth map and to the right the ground truth with all classes.

results considerably in 1D sequential problems. To obtain the improved performance, a contextual feature is created from the confidence map of the base classifier and this is where the method gains its discriminative power, especially when combined with non-contextual classifiers. They explore a number of different classifiers and evaluate stacked versions. They find impressive results on some datasets, especially they test a stacked CRF and find that it can improve the results of the standard CRF. Their work is further developed by Gatta *et al* [12] where they add a multi-scale decomposition. Instead of a single distance of contextual information, a multi-scale decomposition is used which outperforms previous methods. Also in [25], Sampedro *et al* shows an application of MMSSL framework, where they iteratively add stacked classifiers until some stopping criteria. They call it the Iterative Multi-class Multi-scale Stacked Sequential Learning or IMMSSL. The paradigm is used on 3D medical volume scans, but has not been previously tested on 3D data for indoor semantic segmentation which is not sequential in the same manner as for volume imaging.

1.1 NYU-v2 dataset

The dataset used in this project consists of cluttered indoor scenes, from the NYU-v2 dataset [28]. These scenes a generally hard to segment, as the number of objects in such scenes are vast and the dataset itself contains 894 classes of objects. The dataset consists of RGB and depth data along with dense annotations of the scenes created using the Amazon Mechanical Turk service[1]. It spans from kitchens to public libraries and with several examples of each, ensuring a high diversity. It was captured using a custom built Kinect mount, with which they visited homes and public places around New York to gather test data. Examples of the images provided in the dataset are shown in figure 1.4. In general, the depth data obtained using Kinect sensors, which where used to create the dataset, is inconsistent and has missing data or 'holes' in it. The missing data occurs because of too great distances, occlusions, or matte black surfaces. If this problem was not overcome, it would be

[1]https://www.mturk.com/mturk/welcome, accessed May 2015.

(a) (b)

Figure 1.2: The figure (a) shows a sample image from the NYUv2 dataset, (b) shows a surface plot of the wall, marked with a red rectangle in (a). The purpose is to illustrate the amount of noise in the Kinect data on a flat surface.

difficult to implement the remainder of this project. Fortunately, the dataset has been pre-processed by Silberman et al. who provided the dataset. They used the colorization method by Levin et. al [20] to rule out the gaps in the depth maps. The method is based on energy minimization on relationships between neighboring pixels. Accompanying the dataset is a mapping from the 894 classes in the dataset down to only four, {'floor', 'structure', 'furniture', 'props'}. These are the base classes for evaluating the dataset and wil be used throughout this project. Furthermore, the dataset includes a standard test/train data split, which leaves 795 images for training and 654 for testing. All of the state-of-the-art methods use this exact data split. A closer look at the depth data is shown in figure 1.2, where a piece of wall has been plotted in 3D. The surface that supposedly should be flat is heavily distorted. It is obvious from these images, that the depth data has a severe amount of noise.

1.1.1 Kinect camera

Several datasets, including the NYU-v2, are captured using the Microsoft Xbox Kinect camera, which captures depth maps using an infrared camera and an infrared structured light projector. The sensor use an 11-bit representation of the depth-map, whereof one bit is used to signify an undefined depth. This allows for a quantization of the depth into 1024 values. The main sources of error in the depth measurements, assuming a good calibration, are related to the correlation algorithm, the quantization and the distance, as is the case with other setups relying on a two actuator depth sensing. The specifications and in particular the errors of the sensor are described in [18], where measurements show that the error varies from a few mil-

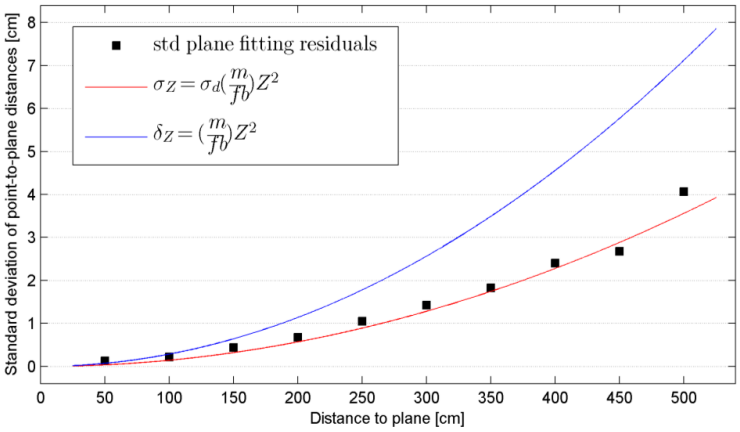


Figure 1.3: The graph shows the theoretical random error (red), the theoretical depth resolution (blue) and the measured standard deviation of the error as a function of distance [18]. Where f is the focal length, b is the base length (the distance between the IR-camera and the IR-projector). Z is the actual depth, m is a normalization parameter and $\sigma_{d'}$ and σ_Z being the standard deviation of the measured disparity and of the calculated depth. The final variable δ_Z is the depth resolution.

limeters at half a meters distance to around five centimeters at seven meters distance which also seems to be the case in figure 1.2. In figure 1.3, the theoretical random error is plotted in red and the theoretical depth resolution is plotted in blue. The measured standard deviation of the error is plotted in black.

In addition, the Kinect camera features an accelerometer and this data is also available for each image, albeit it is only a rough estimation of the orientation. Finally the ground truth data is provided as masks for the images, with a numbering of every pixel corresponding to an index of a list of 894 possible labels. Examples of the ground truth is displayed on the final rows of figure 1.4.

2.5D data challenges

The introduction of RGB-D data to a problem is a clear advantage as it has obvious uses, such as normal extraction and analysis. However, as the obtained point cloud only contains the visible surface, from the cameras viewpoint, it only detects 'half' of the objects. This is of course obvious, considering how the sensor works. This is also why it has been known in the scientific world as *2.5D data*, because of its incompleteness. As a consequence not many robust methods exists for processing this type of data as it is highly viewpoint dependent. Color data is also view dependent, but it is an old research field and is well documented with a range of robust methods.

Also, instead of analyzing neighborhoods in 2 dimensional structured image coordinates, the analyzes should now be conducted in 3 dimensional semi-structured

coordinates. In reality, for proper analyses every pixel should be compared to all other pixels, to determine its exact neighborhood. However in practice, the neighborhood in image coordinates is often used, because of processing time.

1.2 Assumptions and Observations

Indoor cluttered scenes have some common features. These are important, as they can be assumed to always be true and be used as basis for extracting other features. To confirm these assumptions, sample photos are shown in figure 1.4

- **The floor normal points upwards.** In all of the photos in the dataset, the view direction is approximately parallel to the floor plane. Therefore, it is assumed that the floor normal has an upwards pointing direction.
- **The room is concave.** It is assumed that each room is enclosing the scene in the sense of a floor surface with surrounding walls and furthermore these main structural surfaces are perpendicular to each other.
- **Objects are convex.** All physical objects are convex.

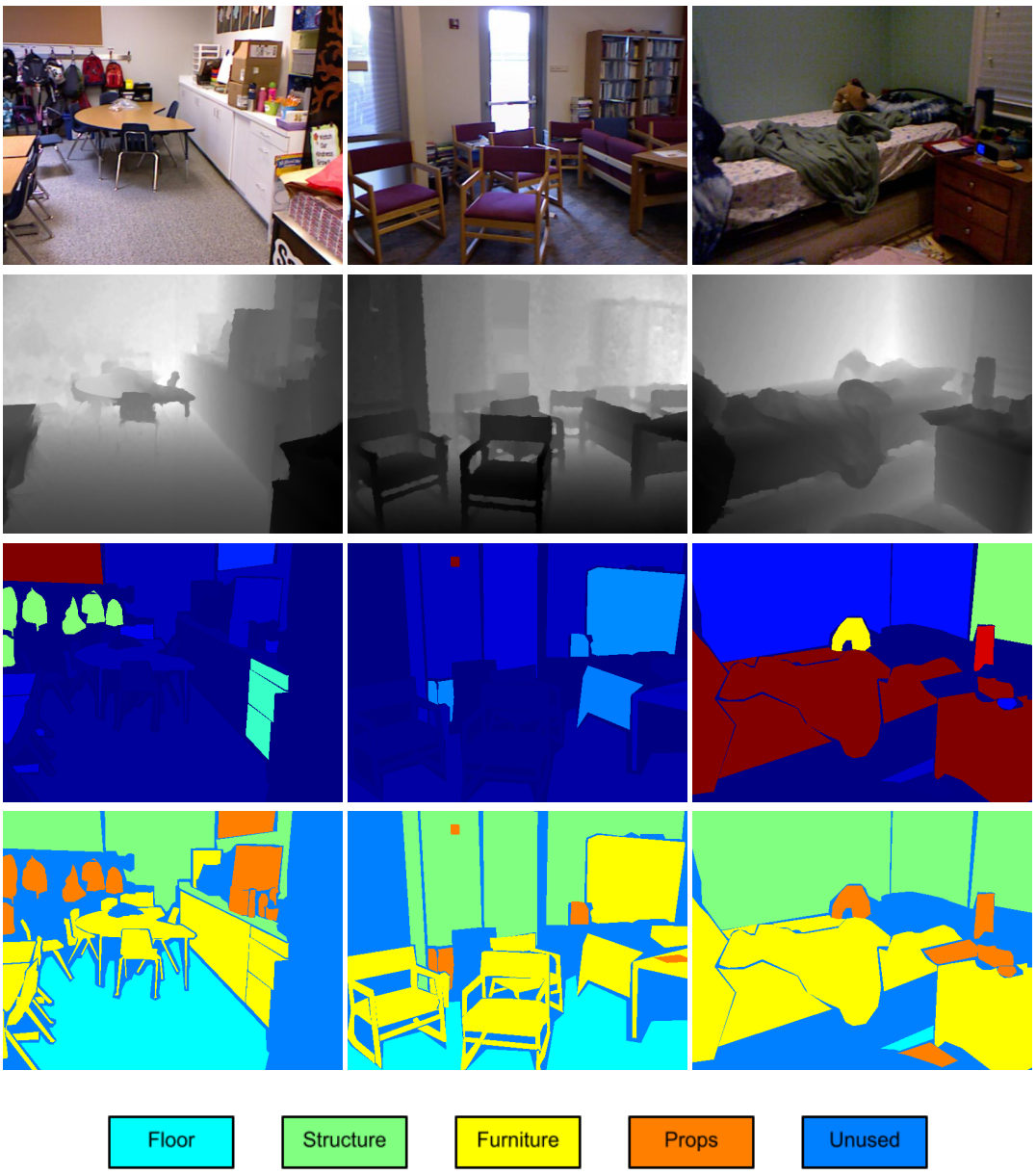


Figure 1.4: Sample images from the NYUv2 dataset, which contains a wide variety of objects and scenes from everyday indoor cluttered scenes. The top shows the original RGB image, second row is the depth information, which is truncated here, third is the ground truth silhouettes and lastly the 4 main classes which this project is focused on.

Chapter 2

Main Feature Design

The feature extraction is arranged as shown in figure 2.1. Initially the input data is preprocessed to find the Cartesian coordinates and a normal estimation. Also a random forest is shown in parallel, this random forest will be described in a later section, it is included here for completeness. Following the preprocessing and random forest, an over segmentation is performed to diminish the data and to find coherent patches in the images. A set of features are extracted from these patches to form the initial feature set. The conditional random field is a graphical model with undirected *edges*, this means that it looks at the context around a given patch as well as the features of the patch itself, the *node* features. It is therefore crucial to extract features or measures that can indicate whether adjacent image patches belong to the same class. As such, both node and edge potentials are created from the feature set and fed to the CRF. Note that edge and node features have multiple names as CRFs are used across a range of fields. Node features are also known as *node potentials* and edge features as *edge potentials* or *pairwise potentials*. These expressions will be used in certain sections as they fit better with graph theory. This chapter will describe

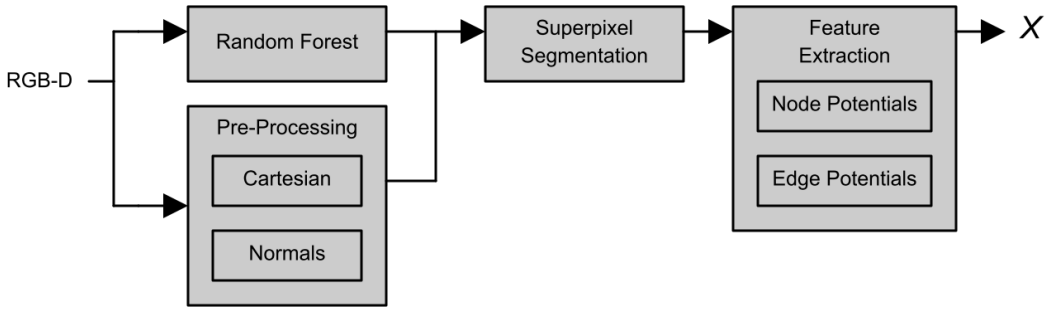


Figure 2.1: Pre-processing, superpixel segmentation and feature extraction pipeline.

the feature functions, $f_{\text{feature}}(x^n)$, used in the proposed CRF model. To the extent possible, the features will be explained and shown based on the sample image shown in figure 2.2.

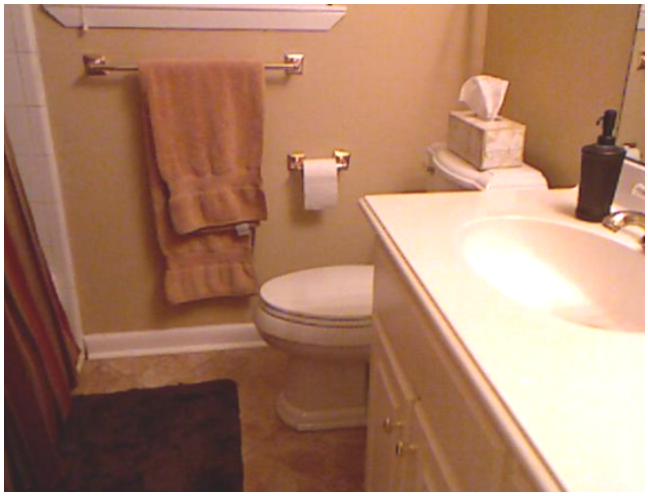


Figure 2.2: A sample image from the NYU-v2 dataset.

2.1 Pre-processing

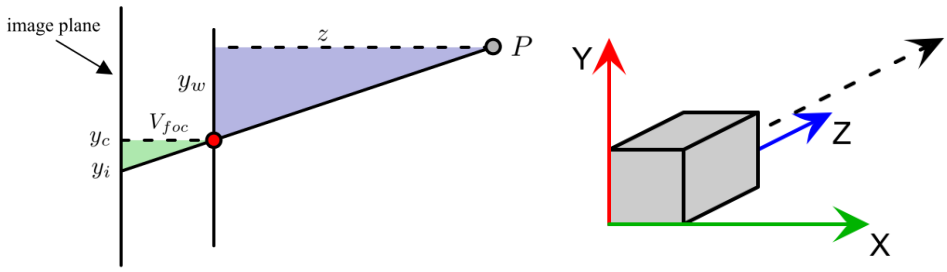
A number of pre-processing steps are required to further process the features of the images. This section will give a short explanation of these processes.

2.1.1 Depth to Cartesian coordinates

In order to analyze the data properly it will be converted from depth to Cartesian coordinates. This representation has obvious advantages, for instance for normal calculations and analyzing spatial layout of the scene. The Kinect pre-processes the data, to obtain a distance from the image plane of the Kinect camera to any point in the image, as opposed to the distance from the viewpoint of the camera. This essentially means that the distance obtained from the images is already converted to the Cartesian coordinate system. The other directions of the coordinate system are indicated in figure 2.3(b). Therefore the coordinates can be obtained using similar triangles, as shown in figure 2.3 and equation 2.1, which shows the calculation for the y-coordinate:

$$y_w = z\frac{y_i - y_c}{V_{foc}}, \tag{2.1}$$

where y_w is the Cartesian y-coordinate, z is the depth and the Cartesian z-coordinate, V_{foc} is the focal length in pixels (in the vertical direction), y_c is the principal point of the image (in image coordinates) and y_i is the image coordinate of the point. The same expression is calculated for the x-coordinate.



(a) (b)

Figure 2.3: (a) Geometric explanation of converting depth to Cartesian coordinates. The similar triangles have a similar relationship between the sides, this is exploited to find the x- and y-coordinates. P is the point, which is being converted to Cartesian coordinates, see equation 2.1. (b) Coordinate system of the resulting point cloud, the camera is the gray box with the view direction indicated with the dotted line.

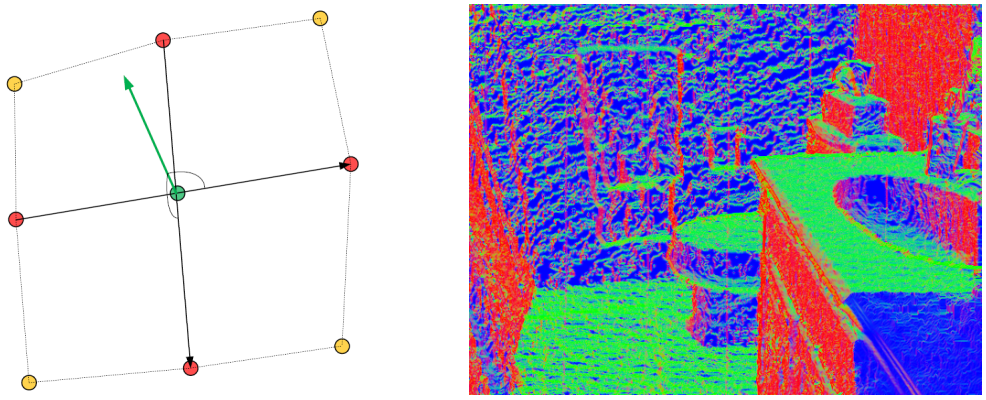
2.1.2 Normal extraction

Normals are a key component in the analysis of scene geometry, as they describe the orientation of the surfaces. With this information the surface shapes can be explored and semantic cues can be obtained. Various approaches can be used to extract the normals [16, 19]. A few of these will be discussed in the following.

A necessity when computing a normal, n_i, is to use the neighborhood, P of the point, p_i, for which the normal is being estimated. The definition of P is a very important factor for the estimation. Common definitions are: the points spatially close to p_i, defined by some euclidean range in 3D; the close neighbor points in image coordinates space; or a combination of the two, by some weighting scheme for instance.

Once P is defined, there are a few ways of extracting the normals. Most common is probably the linear least squares, which approximates the plane from the over-determined problem of fitting a plane to the neighborhood P of p_i. Other methods such as using the eigenvectors of the covariance matrix can also be used, however, computation of such a matrix is an expensive method as it involves many operations. Instead, a fast estimation is to simply use the cross product of the two tangential vectors to the surface defined by the four immediate neighborhood points as shown in figure 2.4(a). This method alone is very sensitive to noise, which is also visible in 2.4(b). The depth measurements, which are based on structured light, are prone to noise and therefore a smoothing can be applied after the extraction where the normals are averaged in a certain neighborhood [16]. The cross product together with an average filter is used in this case.

A crucial step before averaging the normals, is to make them point in the right direction. The cross product, as well as other methods, will produce normals turning either away or towards the camera, it therefore needs to be turned against the camera, as the normals should describe the surface of the convex objects in the scene.



(a) (b)

Figure 2.4: Normal estimation by vector cross product. (a) The normal (green) is calculated as the cross product of the two vectors between the red points [16]. (b) shows the resulting normals of a scene, plotted as the absolute value into the RGB channels of the image.

To turn the normals, every estimated normal is compared with the view direction corresponding to the point where it is estimated by doing a dot product. The dot product of two vectors give the cosine of the angle between them multiplied by their respective lengths. *I.e.* if the angle is greater than 90° the normal points away from the camera and the dot product is negative. Hence, if the dot product is negative the vector is turned in the opposite direction to point towards the camera viewpoint. To show the results of this method and its smoothened version, figure 2.5 displays a raw estimate using the cross product and besides it a smoothened version of the same set of normals.

After extracting the normals, they are further converted into the conventional spherical coordinate system, for further processing. The conversions are given as:

$$\begin{aligned} \theta &= \arccos(z) \\ \varphi &= \arctan\left(\frac{y}{x}\right) \end{aligned} \tag{2.2}$$

2.1.3 SLIC segmentation

To efficiently make inference on the data, several authors of current state-of-the-art utilize some sort of superpixel segmentation to reduce the amount of data to process. There are several approaches that rely on depth and color images in varying degree. Some authors rely solely on the depth data to do segmentation through local RANSAC plane fitting, though it is often in combination with extracting other features [28]. In the proposed model, the SLIC segmentation algorithm is used as it has shown to be good for generating regular superpixels at high framerates and with a good accuracy[3]. In [21] they have altered the SLIC algorithm to include depth,

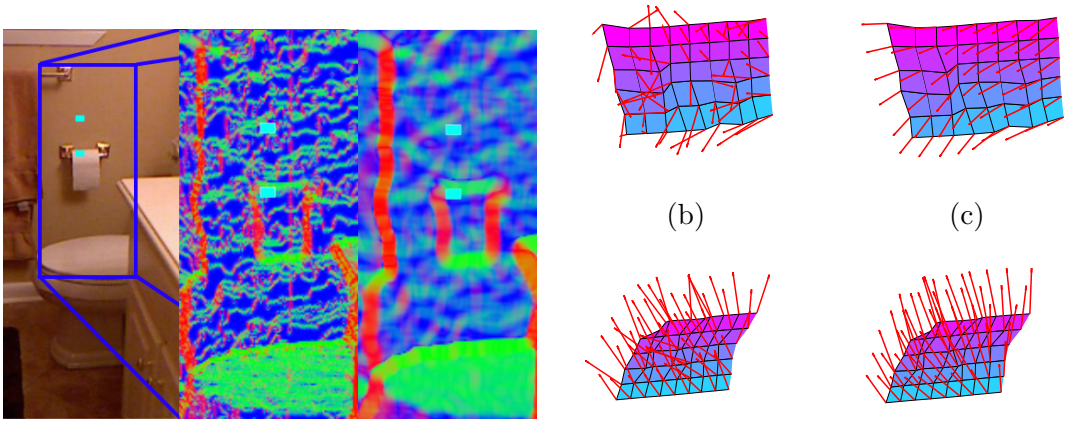


Figure 2.5: The figure (a) shows an example image, with two cyan colored rectangles and a marked blue area. The normals for this marked area is plotted beside the photo in a unfiltered and filtered version. Figures (b) and (c) shows a surface plot of a piece of the wall corresponding to the cyan rectangular area in figure (a). From the surfaces, red arrows are plotted to show the normals. Figures (d) and (e) are similar to (b) and (c) for the toilet paper roll, marked by the lower cyan rectangular area. Note, the plots are rotated to best display the normals.

however they report that the improvement is negligible.

The SLIC algorithm was first proposed in [3] by Achanta *et al*, which was an improved version of the original idea by Zitnik and Kang proposed in [36], but for a slightly different purpose. Other authors have previously used the same segmentation method with good results [24, 21].

Overall, the SLIC algorithm creates a uniformly spaced grid of patches, that adopts to the intensity changes in the image. An example of the segmentation is given in figure 2.6. As one of two parameters for the algorithm, the number of superpixels, k, is defined. Based on this value, k equally spaced points are scattered in a grid throughout the image. These k points will act as the initial centers of a k-means clusterings. K-means clustering works by assigning all the points to whichever k-center is closest to them and thereby clustering the points. Then the k-centers are updated to be the mean of all the points assigned to the cluster. And the same process is iterated until some convergence criteria. Note that the 'points' in this case are measures of the pixels and will be explained shortly.

Before proceeding with the clustering, the immediate 8-connected neighborhood of each center is analyzed to find the pixel with the lowest gradient magnitude. This step is important as the center could be placed on a boundary, which would result in clustering the actual boundary instead of the objects. Following, a k-means clustering is performed, it is here the SLIC algorithm excels over the original:Instead

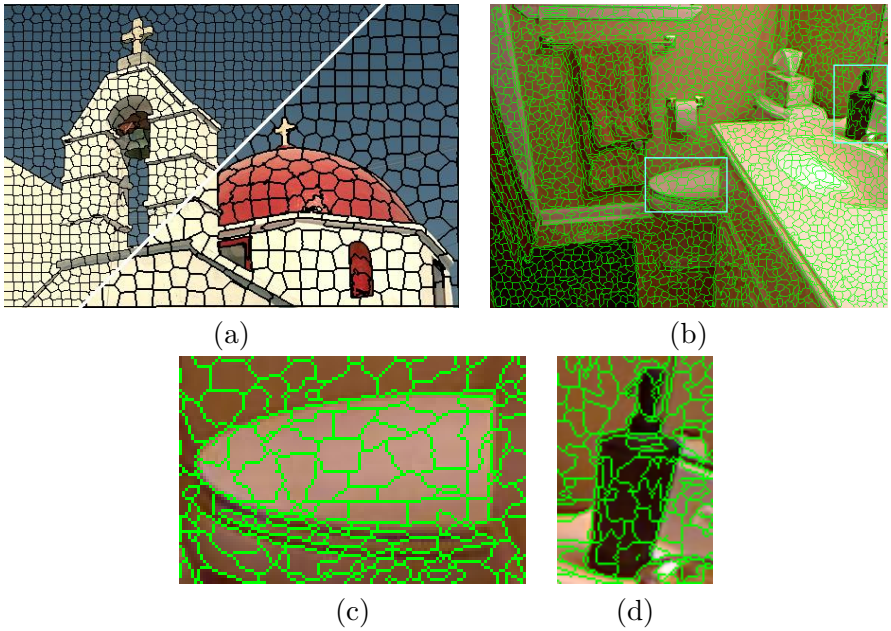


Figure 2.6: Image (a) shows an example of a SLIC segmentation with varying number of superpixels, provided by the authors of [3]. Image (b) shows a segmentation of an image from the NYUv2 dataset, the green lines indicate the SLIC segmentation. In (c) and (d) cutouts of image (b) are shown a zoomed version, to display how the segmentation conforms to the objects.

of exhaustively searching the whole image during the k-means clustering, the search space is refined to a square with the initial cluster origin as a center and the dimensions of $2S$, where S is a function of the k parameter and the number of pixels in the image, N, as $S = \sqrt{N/k}$.

The algorithm proceeds by calculating and adjusting the cluster centers. The measure, which is evaluated, is a mix of the semi-normalized CIELab color space and the normalized spatial difference. The exact expression is:

$$\begin{aligned} D &= \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2}, \\ d_c &= \sqrt{(L_j - L_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \\ d_s &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \end{aligned} \tag{2.3}$$

where subscript i indicates the current point being evaluated and subscript j indicates the cluster centers. D is the final measure, which is composed of the difference measure, d_c, and the normalized spatial distance, d_s. The normalization of d_s is w.r.t. to the maximum expected distance, S, in the square search space for a single cluster.

d_c corresponds to the difference measure in the CIELab color space, composed of L, a and b, the lightness and the a and b coordinates, respectively. The normalization of the CIELab color space is not obvious, and the authors have argued to use the parameter m instead, which essentially becomes a weighting term, defining the ratio of importance between color and spatial arrangement.
Once the clustering is done, there might be pixels which are not connected to a cluster as there is no enforcement of connectivity. To overcome this issue, the final step of the algorithm is to run a connected component method which merges the non-connected pixels to the nearest-center cluster.

As opposed to other authors using the CRF on this problem, the use of the SLIC segmentation is a key component. Most of the other proposed models use various segmentation techniques that rely on spatial consistency, but creates irregular patches. This irregularity might introduce error in the final inference in the CRF, as the features extracted are not based on equal area and are therefore not coherent throughout the model. The SLIC segmentation assures similar sized patches with fairly regular shapes. The implementation of SLIC used in this project is from the VLFeat library[1].

2.2 Primitive features

From the preprocessed data, a set of features are readily available, these might be more or less descriptive. Luckily, one of the benefits of the CRF is that it will determine, based on the parameter estimation, which features matter for which classes. Therefore, the features that might not be descriptive for all classes can be added to the complete model without dominating the model. As such, a number of features can be added directly from the primitives. One important fact about these features is that they have to signify some importance, meaning that a large number indicates a high probability. Together these features become a sufficient statistic to infer the probability distribution of an sample input. The features used in the model is shown in table 2.2.

For the individual superpixels, the features include: Color using the Lab color space, the normal and the standard deviation of the normal. This last one is used to captures curvature in either direction, a round surface will have a large standard deviation as opposed to a flat one where all the normals point approximately the same direction. Also, a blurred gradient of the image and the depth map are used, to provide clues on areas with high change of color. For the case of the features describing the context around a certain patch, the edge features, all of the above and 3D Cartesian coordinates are used. These features are mostly calculated as the absolute difference with the exception of the color and normal feature. The pairwise color feature is modulated to penalize in a non-linear fashion. This is to allow for small changes in color with low penalty and a high penalty for larger differences.

[1] http://www.vlfeat.org/, accessed June 2015

The normals are compared using a normalization of the difference of angles and also only w.r.t their inclination in spherical coordinates as described in table 2.2.

There are many reasons to include the primitive features. For instance, the standard deviation can capture flat areas as opposed to curvature which might be able to discriminate between shapes. Especially the edge potentials are intuitive. For example, an object often has the same color, and that is exactly expressed by the Lab color potential as it encourages adjacent regions to have the same color. Similarly, the mean Cartesian coordinates encourages the regions to be spatially adjacent.

2.3 Dominant normal directions

An important concept of the model is estimating the layout of the room and finding the dominant axes of the room, e.g. the floor and wall normals. For indoor scenes the Manhattan principle is the assumption that lines and edges in indoor imagery are aligned with either the floor or the walls, first coined in [9]. Extending this to surfaces is a natural step. Lines only occur when surfaces meet in a right angle, therefore the surfaces must exhibit the same principle. Based on this extended Manhattan principle - that major surfaces are aligned with the floor or the walls - the normals of the scene are investigated using a Mean Shift Clustering.

The mean shift clustering works in a somewhat similar fashion as the previously discussed k-means clustering method. The easiest way to interpret the difference is that each point is considered a center instead of the previous k-points. The algorithm works in steps similar to the k-means. First the density of the data around each point is measured in a fixed area. Then a gradient ascent like method is run to find the direction to get the most increase in density. The centers of each data point is then moved in the direction of highest density and the process is repeated until convergence. The result will be that the points gather in the same spots and hereby the clustering is obtained. These local spots are the most prominent normal directions and are the candidate dominant directions. The implementation used in this project is by Bryan Feldman[2] and is available on the Matlab forums.

The advantage of the mean shift clustering, as opposed to k-means clustering, is that it works without specifying a predefined number of clusters. The mean shift is significantly slower than the k-means, to relieve the algorithm only a random subset of the normals are used. A visualization of the normals and their distribution for a given scene is shown in figure 2.7. The spherical coordinates of the normals are used here, as the third coordinate, the radius, can be omitted, because the normals are unit vectors. This gives a simplification of the algorithm as well as a boost to processing time.

[2] http://www.mathworks.com/matlabcentral/fileexchange/10161-mean-shift-clustering/content/MeanShiftCluster.m, accessed June 2015.

Node features	Mean Lab color	$f_{\text{Lab}}(P) = \frac{1}{\lvert P\rvert} \sum_{c \in P} c$
	Mean normal	$f_{\text{N}}(P) = \frac{1}{\lvert P\rvert} \sum_{n \in P} n$
	Standard deviation of the normal	$f_{\text{NStd}}(P) = \sqrt{\frac{1}{\lvert P\rvert} \sum_{n \in P_i} (n - f_{\text{N}}(P))^T (n - f_{\text{N}}(P))}$
	RGB blurred gradient magnitude	$f_{\text{RGB}\nabla}(P) = \frac{1}{\lvert P\rvert} \sum_{p \in P} \nabla p$
	Depth blurred gradient magnitude	$f_{\text{D}\nabla}(P) = \frac{1}{\lvert P\rvert} \sum_{d \in P} \nabla d$
Edge features	Lab color difference	$f_{\text{Lab}}(P_i, P_j) = \exp\left[-\beta \ \lvert f_{\text{Lab}}(P_i) - f_{\text{Lab}}(P_j)\rvert^2\right]$
	Spatial difference	$f_{\text{distance}}(P_i, P_j) = \left\lvert \frac{1}{\lvert P_i\rvert} \sum_{v \in P_i} v - \frac{1}{\lvert P_j\rvert} \sum_{v \in P_j} v \right\rvert$
	Normal normalized angle difference	$f_{\text{angDiff}}(P_i, P_j) = \arccos\left(f_{\text{N}}(P_i) \bullet f_{\text{N}}(P_j)\right)/360$
	Inclination angle difference	$f_{\text{angInc}}(P_i, P_j) = \lvert\lvert \frac{1}{\lvert P_i\rvert} \sum_{\theta \in P_i} \theta - \frac{1}{\lvert P_j\rvert} \sum_{\theta \in P_j} \theta \rvert\rvert/180$
	Standard deviation of the normal	$f_{\text{NStdDiff}}(P_i, P_j) = \lvert\ f_{\text{NStd}}(P_i) - f_{\text{NStd}}(P_j)\ \rvert$
	RGB blurred gradient magnitude difference	$f_{\nabla\text{RGBDiff}}(P_i, P_j) = \lvert f_{\text{RGB}\nabla}(P_i) - f_{\text{RGB}\nabla}(P_j)\rvert$
	Depth blurred gradient magnitude difference	$f_{\nabla\text{DDiff}}(P_i, P_j) = \lvert f_{\text{D}\nabla}(P_i) - f_{\text{D}\nabla}(P_j)\rvert$

Table 2.1: Table of feature functions. The notation is as follows: P corresponds to the information from one region, e.g. a set of points; R_i and R_j are adjacent regions; c corresponds to the Lab color values; n is the normal; p are the RGB values; v are the vectors with Cartesian coordinates; d are depth values and finally θ are the inclinations of the normals in spherical coordinates.

Figure 2.7: Figure (a) shows a scene plotted in 3D Cartesian coordinates, besides it, in (b) a plot of the concentration of normals is shown. It is created using a histogram of the normals, where the concentrations are plotted on a sphere, corresponding to the direction of the normals. Notice the three main bulges (red colors), they correspond to the three dominant normal directions in the image. (In the plotting of (b), the concentration is dampened to make it easier to visualize).

2.3.1 Floor normal

As several authors [21, 28, 13] similarly have noticed, the floor normal can be found by taking the candidate nearest to the upwards axis of the estimated Cartesian coordinate system. Consequently it is assumed, that the floor normal vector, n_{floor}, is pointing upwards. Using this assumption and the mean shift modes, the floor normal is found as the argmax from the following equation:

$$n_{\text{floor}} = \underset{p \in \boldsymbol{P}}{\text{argmax}} \ \exp\left[-\left(\frac{|p_\theta - \theta_{\text{std}}|}{180} \right)^{\lambda} \quad \left(1 - \frac{p_\mu}{\sum\limits_{p \in \boldsymbol{P}} p_\mu} \right) \right] \tag{2.4}$$

where n_{floor} is the inferred floor normal, $\boldsymbol{P}$ is the set of candidates found from the mean shift clustering and p is a normal candidate. Each candidate, p, has a polar angle, p_θ, and a support value, p_μ, which are both found from the mean shift clustering. The θ_{std} is a standard inclination of the floor normals in the scenes. It is found by hand tuning the parameter. λ is a factor which controls the influence of the two terms. The first term in the equation weights the inclination w.r.t to a standard angle, this is normalized by dividing with the maximum angle difference, 180°. The term is also depicted in figure 2.8. The second term is used to weight the candidates based on the evidence which translates to how many of the normals have the same direction. The support is therefore the size of the cluster corresponding to the candidate, p, found from the mean shift clustering. The first term is furthermore regulated by a term, λ, which attenuates the influence between the two terms. Lastly the terms are mixed in a negative log, to give an output between zero and one.

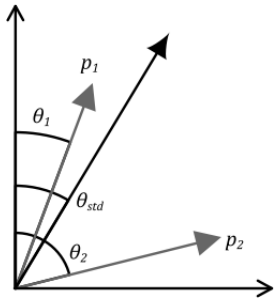


Figure 2.8: This figure depicts the first term in the floor normal equation 2.4. Notice that in the formula the azimuthal angle is not included. Therefore this can be illustrated in two dimensions. The gray arrows represent two normal candidates, p_1 and p_2. The black arrow represent the standard inclination angle θ_{std}. Hence, the first term is just the difference between the polar angle of the normal candidate and the standard inclination, normalized by the maximum difference.

Once the floor direction is determined, the wall normals are found in a similar fashion from the remaining dominant normal candidates. When finding the wall normals, the assumption is that candidate directions are perpendicular to the floor normal and to other candidates - assuming that rooms are rectangular. This does not hold for rooms with more complex walls, but performs well in practice. The evaluation of the wall candidates is initiated and afterwards runs iteratively until there are no more valid candidates. Initially the vertical angle difference, $p_{\hat{\theta}}$, between the floor normal and the candidates is calculated and 90° is subtracted so that candidates with an angle difference close to zero are more likely to be wall. Afterwards, the process continues iteratively following these steps:

- (This step is skipped on first iteration). The candidates are compared with the found wall normals in the horizontal direction by finding the angle difference, $p_{\hat{\varphi}}$, and subtracting whichever multiple of 90° is closest to the found value. This has the effect that candidates that are perpendicular or opposing to the already found wall normals are most likely to be wall.
- On every iteration the remaining candidates are evaluated using:

$$\mathbf{n}_{\text{wall}} = \underset{p \in \boldsymbol{P}}{\text{argmax}} \ \exp\left[-\left(\frac{|p_{\hat{\theta}} + p_{\hat{\varphi}}|}{180} \right)^{\lambda_{\text{w}}} \quad \left(1 - \frac{p_\mu}{\sum\limits_{p \in \boldsymbol{P}} p_\mu} \right) \right], \tag{2.5}$$

  where λ_{w} controls the importance of either term. If the evaluation equation returns a value greater than some threshold, the candidate is accepted as a wall normal, otherwise the process is stopped.

The evaluation equation updates the scores of the candidates and takes the one with the best score iteratively to allow for multiple wall normals. Until a certain threshold of the evaluation equation, the candidates are accepted as walls.

(a) (b) (c)

Figure 2.9: The original image is shown in (a). (b) shows the floor probability evaluated using equation 2.6, which has a high value when the surfaces are near parallel with the floor normal direction and low values otherwise. (c) shows a similar feature as (b) for the walls.

2.3.2 Horizontally aligned surfaces

The floor normal is used to find all flat horizontal surfaces, such as tables, desks and so on will have the same normal. In the feature function, the normal is measured up against the current regions mean normal by a simple dot product in a negative exponential function:

$$f_{\text{flat}}(n) = \exp\left(-4 + 4(n \cdot n_{\text{floor}})\right) \tag{2.6}$$

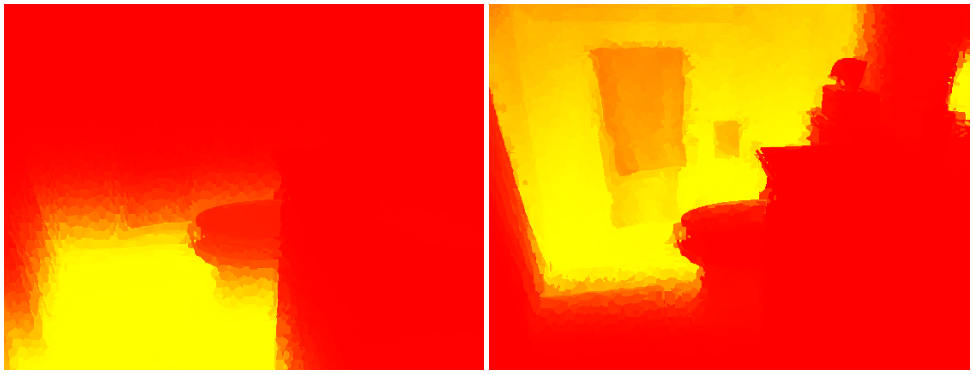
where n is the mean normal of the current region. This provides a rapidly decreasing expression giving only high values to the normals that are in the same direction as the floor normal. A similar concept is used for the wall normals. A depiction of both the floor and wall feature is given in figure 2.9 (b) and (c) with the original image in (a).

2.3.3 Floor and Structure Probabilities

Once the normal of the floor is computed, the floor can be detected in the image, by finding the plane that is furthest away from the camera, in the direction of the floor, meaning the opposite of the found floor normal. From this plane, the distance to all points can be found and by modulating this distance, a measure can be created which expresses the probability of a certain point being floor. A similar method is used in [21].

Assuming that the furthest point in the opposite direction of the normal is p_{floor}, then a plane can be expressed by using this fix point and the normal of the floor, n_{floor}, found previously. A feature is created by applying a modulation to the distance from this plane to all other points in the scene. Once the distance is extracted it is modulated using a negative exponential, similar to a Gaussian:

$$f_{\text{floor}}(p) = \exp\left(-\frac{|n_{\text{floor}} \cdot (p - p_{\text{floor}})|^2}{0.02}\right), \tag{2.7}$$



(a) (b)

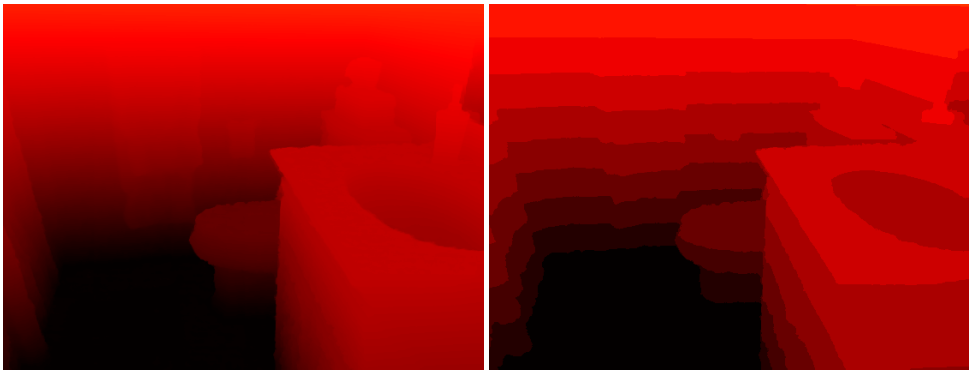
Figure 2.10: Probability based on the dominant normal directions. The original image is shown in (a). Intensity image of the floor probability in (b) and the wall probability based on fitting a plane to the normals of the walls in (c).

where p is the point in which the floor measure is taken. Knowing the normals of the walls they are found in the scene by fitting a plane to the furthest point from the viewpoint using the same logic. Just as a floor probability can be assigned, a similar wall probability is assigned. A depiction of these features are shown in figure 2.10 (a) and (b).

There is one obvious caveat in this feature; if there is no visible floor in the image and depth measure, this will be wrong. Often, the images have visible floor, but not always. As a consequence, if the point, p_{floor}, is less than 1.2m beneath the camera, floor plane is simply set to 1.2m beneath the camera point, the reason for doing this is an obvious advantage in the next section.

2.3.4 Height above the floor

Based on the floor measurement and the floor plane, each region in the scene can be assigned a height above the floor. Intuitively, this feature makes sense, as it discriminates according to the spatial position of objects w.r.t a fixed surface. For instance, cups, plates and props are usually placed on tables or similar and not on the floor. There is a disadvantage to this feature, however. The measure has to be quantized into several input features for the CRF. A high value of an input usually signifies some importance, but in the case of this feature it is simply the measurement and not an expression of importance. The height above the floor is therefore quantized into 20 binary inputs for the node features, corresponding to 12.5cm intervals. The height measurement is shown in figure 2.11 along with a quantified version. This problem of quantization only arises for the node potential, as the corresponding edge potential simply uses the difference in height between the adjacent regions.



(a) (b)

Figure 2.11: (a) shows the height measured for all pixels, the floor is black and the value increases as the height above the floor increases. (b) shows the same plot, but quantized so the splits in height are visible.

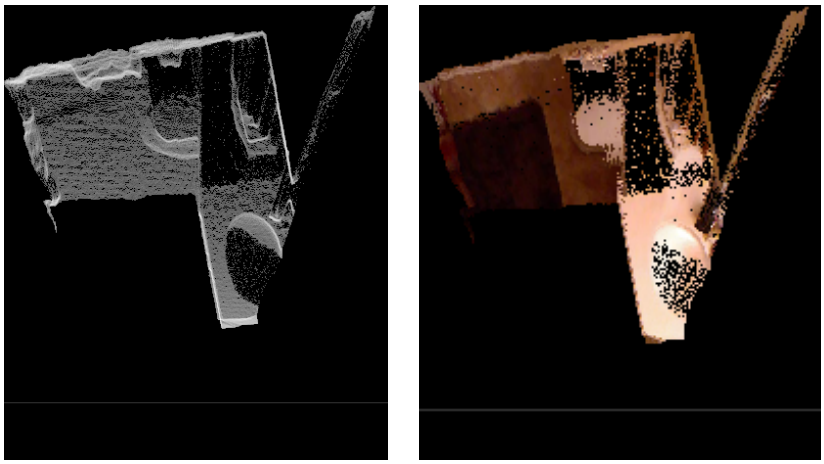


Figure 2.12: Room layout view of the sample image shown in figure 1.1. The 3D point cloud has been rotated, such that a 'top view' of the scenery is obtained. The left image is created by counting the number of 3D points corresponding to the pixel in the top view image. The right image shows the topmost 3D points' color.

2.3.5 Room Layout Feature

Another feature derived from the dominant normal direction, is the room layout feature. The idea is simple: flipping the 3D scene, such that it is viewed from above as shown in figure 2.12. As well as other objects in the scene, the walls are aligned and appear as a straight line in the top view. Utilizing this, the idea can be used as another means of finding the walls of the room. This is achieved by following each line of sight from the camera and registering the last point, which is then assumed to be wall. These assumptions are inspired by the work of Cadena and Košecka

[7].

Chapter 3

Learning Methods

This chapter explains the machine learning methods used in the model. The overall pipeline of the proposed algorithm follows the Stacked Sequential Learning framework which is depicted in figure 3.1. The previous chapter discussed some of the features that are used in the model. These features are used to do inference in the *base* classifier, which consists of a CRF model in conjunction with a random forest using random offset features.

The outcome of the base classifier is a confidence map, Z, which specifies how confident the model is that a given pixel should take on a certain value. The confidence map is modulated into a contextual confidence feature and concatenated with the original feature set to get a superset, X'. This superset is the basis for another classifier, referred to as the *stacked* classifier. By giving a classifier both the original features and this contextual confidence feature, it will be able to correct some of the errors of the base model, and an overall better model is obtained.

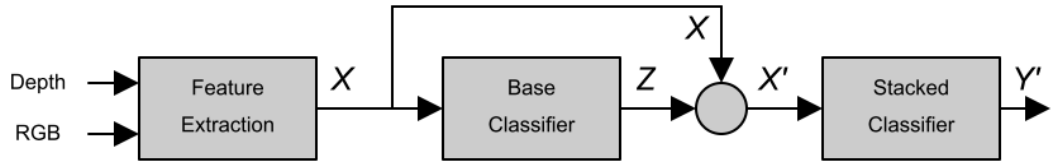


Figure 3.1: Simplified overview of the pipeline.

3.1 Model Overview

To specify the model completely, figure 3.2 shows the pipeline of the model with all parts and interactions included. Furthermore the figure is color coded to display the different sections and also how they are arranged in this chapter. The features, preprocessing and SLIC segmentation have been covered in the previous and are gray in the figure. In the following, section 3.2, the CRF model will be explained (blue in the figure). Afterwards, the random forest with random offset features (yellow)

is explained in section 3.3. And lastly the stacked sequential learning with another random forest and the contextual confidence feature are detailed in the final section, section 3.4, of this chapter.

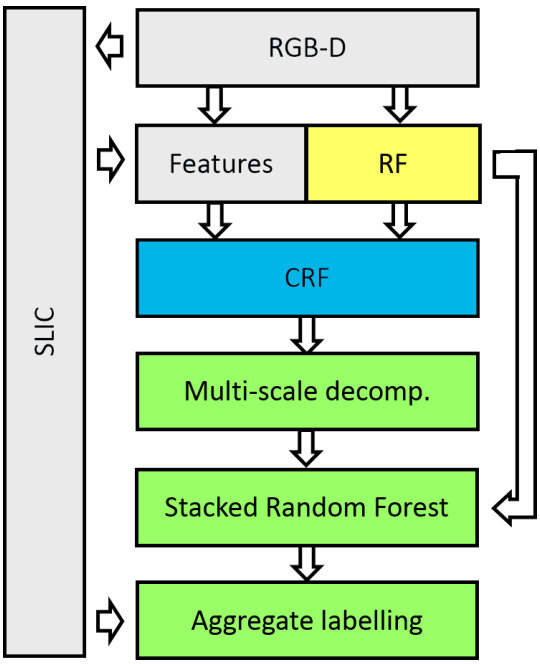


Figure 3.2: Pipeline including all parts.

3.2 Conditional Random Field framework

This concept is the core of the project since it enables the combination of features of the input data into a structured prediction model that encapsulates the contextual dependencies within the scenes. An image consist of a large grid of adjacent pixels, that in general are very correlated. Objects in images often span a large region of an image and are uniformly colored. Therefore, the probability of seeing a blue pixel besides another blue pixel in a photo, is generally greater than seeing it beside a red one, *i.e.* there is a relationship between adjacent pixels in the images. These relations is essentially what a CRF realizes. Taking the same intuition and applying to the problem at hand, the segmentation explained above in section 2.1.3 resulted in regular, uniformly spread regions, which also have strong connections to the adjoining regions. The CRF is a structured way of capturing these dependencies between adjacent regions. As the direct CRF formulation builds on parent concepts, the following will shortly discuss the origins of CRFs and revolve around some of the key aspects, used to define and infer CRFs.

The method of CRFs originate from Bayesian networks and Markov Random Fields (MRF). Bayesian networks are also known as directed graphical models, which

means it consists of conditional probabilities, and they are arranged as trees and therefore acyclic. A simple Bayesian network is shown in figure 3.3(a). The notion of these two properties are important as they implicitly invoke another property - *memory*. The notion of memory, of course, refers to the probabilities throughout such a graphical model and the fact that the previous probabilities have to be 'remembered' in order to calculate the following probabilities. The factorized form for calculating a probability in such a model is given in equation 3.1, where the joint probability distribution is given as a product of the previous or parenting conditional probabilities. The notation and theory is adopted from [22].

$$p(y) = \prod_{k \in V} p(y_k | y_{pa_G(k)}) \tag{3.1}$$

where y is a sequence of random variables in the graph, y_k is a single random variable, G is the directed acyclic graph, $y_{pa_G(k)}$ are the 'parent' random variables to y_k in the graph G and V are the set of random variables in the graph.

Moving to MRFs, they give up the limitation of being acyclic and MRFs are undirected graphs as opposed to the Bayesian networks. These notions gives rise to the Markov Property, which specifies that the model only depend on the current state, e.g. it is *memoryless*. More precisely, the variable, y_i, is conditionally independent of all other nodes given its neighbors. As a consequence, the interactions between the nodes can be factorized to form a factor graph. In this representation, the graph consists of nodes which are the random variables and factors that connect all nodes in a clique to each other. The graphical convention for writing these graphs are shown in figure 3.3(c). The factors do not need to be probabilities, in this case they are called a unnormalized measures.
The joint probability distribution of an MRF, where the value of the random variables is the product of the factors is given as:

$$p(y) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \psi_C(y_C) \tag{3.2}$$

where $\mathcal{C}(G)$ is the set of all cliques of G, G is the undirected, possibly cyclic graph, $\psi_C(y_C)$ is the factor, for the clique C, y_C are the set of random variables in the clique C, Z is the partition function, which is the sum of the product of all possible configurations of y, e.g. it is a normalization of the probabilities. Remark, the calculation of the *partition function* or the normalization constant, Z, is omitted here. In general, this measure assures a normalized output and is just the sum over the nominator.

There are a few limitations to the factorization in equation 3.2. The factors must be non-negative and the graph must be chordal for such a factorization to exist.

The combination of these two well-known graphical models, results in Conditional Random Fields. This graphical model is similar to an MRF, but it can be arbitrary in structure and it is conditional on the observations, or more commonly known as features in computer vision. The state of a single random variable in a CRF is

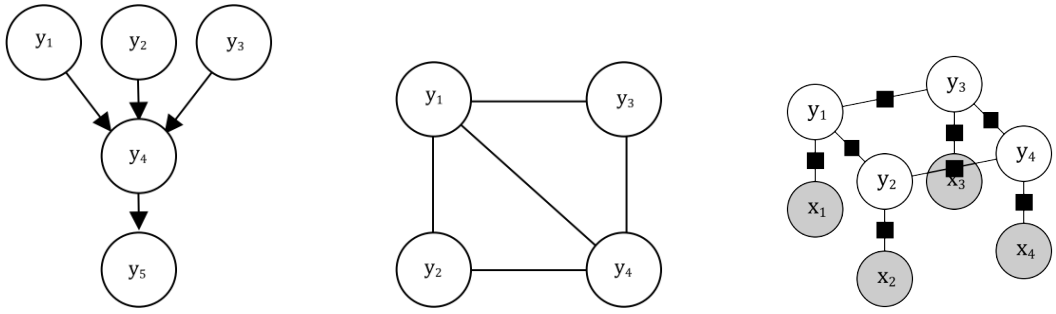


Figure 3.3: Figure (a) shows a graphical model of a directed acyclic Bayesian network, (b) an MRF graph, which is chordal, undirected and cyclic, and finally (c) a CRF factor graph, where the factors are expressed as black boxes and observations are marked by a gray coloring.

therefore a combination of potential functions with the scope of the features and the neighboring random variables. In figure 3.3, three graphical models are shown, one Bayesian network, an MRF and a CRF.

The general expression for a CRF expresses the probability of a certain configuration of the random variables, y, as a function of the input features through the factors, ψ_F:

$$p(y|x) = \frac{1}{Z(x)} \prod_{F \in \mathcal{F}} \psi_F(y_F; x_F) \tag{3.3}$$

where x is the observations or the features, $\mathcal{F}$ is the set of all factors, F is a single factor, $\psi_F(y_F; x_F)$ is the factor, with the random variables y_F and features x_F of y_F. $Z(x)$ is the partition function given in equation 3.4, which is the sum of the product of the configurations of y based on x, e.g. it is a normalization of the probabilities.

$$Z(x) = \sum_{y \in \mathcal{Y}} \prod_{F \in \mathcal{F}} \psi_F(y_F; x_F) \tag{3.4}$$

3.2.1 Feature functions in CRFs

Putting this into the context of computer vision, and the problem domain, each of the regions from the segmentation is seen as one random variable. The objective of the CRF model is to label each of these regions with one of the possible labels from the training data.

In the CRF, the random variables, y_k, corresponds to the probability of the region, k, belonging to a certain class, whereas y is all the possible labels of all the regions. The observations, x_k, corresponds to the features extracted from the image at the region k. Factors represent the relationships between the random variables in each clique, essentially they can consist of any function, but certain functions or families

of distributions can make simplifications to the model, and are typically used [22]. A very typical model is to use a exponential of a *feature function*, f, with scalar multiple, w, as a parameter as shown in equation 3.5. The feature function can be arbitrary, and could just be the feature.

$$\psi(y, x, w) = \exp\left(w^T f(y, x)\right) \tag{3.5}$$

This type of model is called a *log-linear model*, which will be more clear when it is introduced in the general equation:

$$p(y|x) = \frac{1}{Z(x, w)} \prod_{F \in \mathcal{F}} \exp\left(w_F^T f_F(y, x)\right) \tag{3.6}$$

The partition function is changed accordingly. Now the product of exponentials can be concatenated and the log-linear model is obtained:

$$p(y|x) = \frac{1}{Z(x, w)} \exp\left(\sum_{F \in \mathcal{F}} w_F^T f_F(y_F, x_F)\right) \tag{3.7}$$

The factors, often referred to as *potentials* in literature[22][31], are commonly separated in two, the node potentials and the edge potentials, which are also known as unary and binary potentials. This also provides a more intuitive way to order the CRF. The parameters, w, are often shared across all the potentials of the CRF. Finally, a very common CRF model, limited to node and edge potentials can be expressed as:

$$p(y|x, w) = \frac{1}{Z(x, w)} \exp\left(w_n^T \sum_{k=1}^{K} f_n(y_k)\right) \exp\left(w_e^T \sum_{F \in \mathcal{F}} f_e(y_F)\right) \tag{3.8}$$

Where f_n is a node potential, f_e is an edge potential, k is a node index, w_n and w_e are the parameters for each type of potential. Notice that here, instead of iterating over the factors for the node potentials, the iterations are over the node index k. The pairwise potentials still needs to be over the factors, however. This makes the idea (hopefully!) more intuitive, compared to the general expressions in the previous. Also, the input to a feature function is switched to y_k instead. By this, it should be understood as $f_n(y_k)$ is the node potential to the node y_k, which will be explained further in the following section with an example, similarly y_F describes the nodes involved in the factor F, corresponding to two adjacent nodes here, as f_e is an edge potential. What is not obvious from the model above, is that the parameters can be shared amongst classes as well. When parameters are shared across classes in edge potentials, it is referred to as Ising potentials, because it simulates Isings model for interactions between electron spins in physics. It essentially just means that the same parameter will be used if the potential is in between nodes of class a and b or a and c. It can be more costly to do inference without the Ising potentials, as they have to be optimized w.r.t each of the classes.

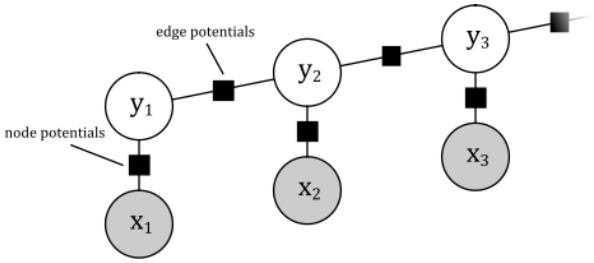


Figure 3.4: Shows an example of a linear chain CRF. White nodes corresponds to random variables, gray nodes to features and the potentials are the black squares, whereof two of them are marked to indicate the two types of potentials.

Once the model is defined, e.g. the graph and thereby the potentials are defined, the objective is to find the configuration which gives the maximum probabilities to all random variables. Solving the problem directly is generally intractable, as all possible combinations have to be evaluated to find the optimal one. The problem arises when the graph includes cycles or loops. Fortunately, one popular method for exact inference in tree structured graphical models, can be modified to approximate the optimal configuration, which gives rise to a whole set of algorithms for this purpose. This method is known as Belief propagation.

3.2.2 Inference for CRFs

There are a few different methods that can be used for inference, the possibly most used is the Belief propagation, which stems from acyclic graphs. It works by sending *messages* to and from edges and nodes, to realize the interactions between them, these messages are commonly known as *beliefs*, referring to the idea that an edge communicates a *belief* about which class neighboring nodes should be. The algorithm gives exact inference in acyclic graphs, but can only approximate it for the case of cyclic graphs.

The idea of beliefs stems from a dissection of the partition function. For small graphs, the partition function is tractable to solve, but the complexity is exponential, $\mathcal{O}(C^K)$, where C is the number of classes in the model and K is the number of nodes. To put this into context, a general SLIC segmentation could contain 500 to a few thousand nodes and in this problem there are four classes, clearly this is infeasible to calculate. Therefore, the partition function is generally intractable for larger graphs and problems like this are consequently known to be NP-hard. However, for linear chain and tree graphs, it is possible to translate the problem to a dynamic programming problem, which enables solving it sequentially and the belief propagation algorithm is obtained. Remark the belief propagation algorithm also goes by the names sum-product algorithm and forward/backwards algorithm.

For the purpose of understanding how the inference in loopy graphs works, the belief propagation for a linear chain graph will be explained, as it works in exactly the

same way. If a linear chain CRF is assumed, e.g. there are no loops, just a straight line of nodes with the corresponding gray, fixed feature nodes, as the example shown in figure 3.4. Then the expression for such a chain is:

$$p(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{k=1}^{K} f_n(y_k) + \sum_{k=1}^{K-1} f_e(y_k, y_{k+1})\right) \tag{3.9}$$

$$Z(x) = \sum_{y \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} f_n(y_k) + \sum_{k=1}^{K-1} f_e(y_k, y_{k+1})\right) \tag{3.10}$$

$$Z(x) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_K} \exp\left(\sum_{k=1}^{K} f_n(y_k) + \sum_{k=1}^{K-1} f_e(y_k, y_{k+1})\right)$$

Notice here that the parameters have been taken out, as they would only clutter the equations and it will not change any concurrent conclusions. The partition function, and its explicit form on the second line, is a sum over the *whole* output space, *i.e.* all possible combinations and it is this partition function which accounts for the complexity of the problem, as it is a series of nested sums. Using dynamic programming the complexity can be reduced to linear, $\mathcal{O}(KC^2)$. The steps taken to get to this result are first to make the exponential of the sums into products of exponentials, now because the sums are independent, except in the pairwise potentials, it can be rearranged and factored out, as follows:

$$Z(x) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_K} \prod_{k=1}^{K} \exp\left(f_n(y_k)\right) \prod_{k=1}^{K-1} \exp\left(f_e(y_k, y_{k+1})\right) \tag{3.11}$$

$$Z(x) = \sum_{y_K} \sum_{y_{K-1}} \cdots \sum_{y_1} \prod_{k=1}^{K} \exp\left(f_n(y_k)\right) \prod_{k=1}^{K-1} \exp\left(f_e(y_k, y_{k+1})\right)$$

$$Z(x) = \sum_{y_K} \exp\left(f_n(y_K)\right) \left(\sum_{y_{K-1}} \exp\left(f_n(y_{K-1})\right) \exp\left(f_e(y_{K-1}, y_K)\right) \right. \tag{3.12}$$

$$\cdots$$

$$\left. \left(\sum_{y_1} \exp\left(f_n(y_1)\right) \exp\left(f_e(y_1, y_2)\right) \right) ... \right)$$

From the result of this derivation, the problem becomes manageable, as the terms can be calculated from the inside out in a very simple fashion. Looking at the innermost term in equation 3.12, it can be thought of as a function of y_2, which will be referred to as $\alpha_1(y_2)$. If there are four possible classes for all y, then there will be four different iterations over y_2. If these are computed and stored in a vector, the values can simply be multiplied when the following nested sum iterates over y_2. And in a similar fashion, the iteration over y_3 can be computed once and stored in a vector and simply be multiplied in the next nested sum. Each of these computations

are stored for further calculations and these *forward* pass vectors are referred to as the α-table.
In a completely similar fashion a swapped version of the expression, where the innermost term is a sum over y_K, can be calculated and this is referred to as the *backward* pass and the vectors are stored in the β-table.
In praxis, the α and β values are calculated in log space, to get a numerically stable algorithm.

If these tables are kept, they can be used to infer the marginals of any y_k, this is possible because of the conditional independence in the model. The equation for evaluating the marginals for the node k is given here:

$$p(y_k|X) = \frac{\exp\left(f_n(y_k) + \log\alpha_{k-1}(y_k) + \log\beta_{k+1}(y_k)\right)}{\sum_{y_k}\exp\left(f_n(y_k) + \log\alpha_{k-1}(y_k) + \log\beta_{k+1}(y_k)\right)} \tag{3.13}$$

Notice that the α_{k-1} contains the summations of the terms up until $k-1$ from the left side and β_{k+1} from the right side. They exactly sum up the values until the neighbors of k, which is the only values needed for inferring the node in accordance with the conditional independence property.

The same principle of message passing, meaning the vector messages α and β, can be used in cyclic graphs. In this case, it is known as *loopy belief propagation* and in praxis it is either random message passing between nodes and factors, or ordered in some way. But the principle is the same. However, this inference method is only approximate, but it has shown to be effective in many cases and is probably, according to [22] the most used method for inference in discrete graphical models. The two general equations for messages, μ, are:

$$\mu_{y_k\to f}(i) = \prod_{f'_e\in \mathrm{Ne}(y_k)\setminus f_e} \mu_{f'\to y_k}(i) \tag{3.14}$$

$$\mu_{f\to y_k}(i) = \sum_{y'\in \mathrm{Ne}(f)\setminus y_k} \left(\exp(f_n(y_k=i) + f_e(y', y_k=i)) \prod_{y'_k\in \mathrm{Ne}(f)\setminus y_k} \mu_{y'_k\to f}(y_k=i)\right) \tag{3.15}$$

Where i is one of the possible labels of y_k and y' is the neighborhood of y_k (Hence, the notation $y'\in \mathrm{Ne}(f)\setminus y_k$). Equation 3.14 corresponds to the messages going from node k to a neighboring factor. And conversely, the second equation, 3.15, are the messages going from a factor to a node. By using these messages and propagating them in the graphical model, the messages should start to converge and the algorithm can be stopped, when the changes are sufficiently small. If these loopy belief messages are used for a graph, like the linear chain CRF above, the α and β tables will be recovered. Note that the notation is limited to pairwise potentials, however, it can be formulated for any connectivity.

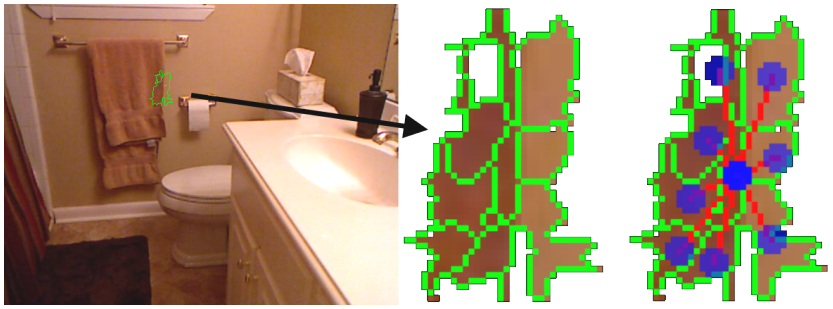


Figure 3.5: Shows 9 adjacent SLIC segmentation patches marked by the green borders. To the right the image is scaled up and to the rightmost edge are added in red and node potentials in blue

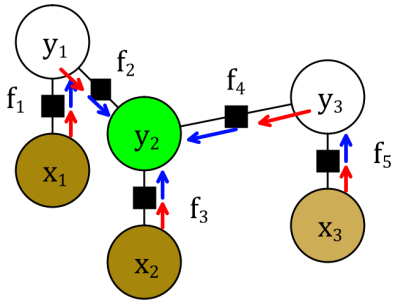


Figure 3.6: Shows a small CRF net. The colors of the observations, x_k, correspond to some of the top segmented areas in figure 3.5. The messages from each node to factor is colored in red and blue for the factor to node messages.

3.2.3 Applying the CRF

To simplify this sections theory, this will shortly discuss how the theory is applied in the context of this exact problem. In figure 3.5 a sample cutout of an image is shown. It corresponds to 7 segmented adjacent areas. The blue circles in the rightmost image corresponds to the node potentials. These include the mean color, normal standard deviation etc. The red lines shown indicate the adjacencies and thereby the edge potentials. Note that this image only shows the connections for the segmented area in the center. Hence, in reality all the surrounding blue dots should be connected by red lines, but they have been removed here for the sake of simplicity.

To get the best overview, this will be explained from the end point, *i.e.* the wanted outcome which is the marginal probability of a given node to have a certain class label. This is exactly expressed for linear CRFs in equation 3.13 with the α and β tables, the general expressions from equations 3.14 and 3.15 needs to be used. It turns out that doing the message passing is equivalent to calculating the α and β tables.

Figure 3.6 shows an even simpler graph, with arrows indicating the message pass-

ing. Imagine that the lower circles, x_k, correspond to the known features of each region.. The white circles, y_k, are the labels being inferred and in particular the marginal probabilities of the green circle, y_2 are desired. At the beginning of the message passing, the messages from observed variables (the x_k) are set to one. Hence, the initial message from x_1 to the above factor node is given in equation 3.14, which says that the message to the factor is the product of all the messages from the connected factors, except the factor for which the message is calculated. In this case there is only the one factor connected to x_1 and as it is set to one initially, the message is consequently one.

Following, the message from the factor f_1 to the node y_1 is given in equation 3.15, which means that the message is equal to the sums over the neighboring nodes' potentials, multiplied by the product of their respective incoming messages. For the message to y_1 from the factor below, this translates to one multiplied by the potentials of the nodes involved, which is only node y_1:

$$\mu_{f_1 \to y_1} = \exp(f_n(y_1)) \tag{3.16}$$

Note here that y_1 can take on different labels, which in the case of the current problem could be either of {*floor, structure, furniture, props*}. This means that the message is essentially a four element vector. Remember that f_n and f_e are just substitutes for the feature functions and their respective parameters, w, which is also a vector holding one value for each available class. Next is the message from node y_1 to the following factor, f_2. This message is just the product of all the factor messages to node y_1, which in this case is just equation 3.16. Finally the last message from the factor to node y_2 is going to be the sum of all potentials of the nodes included. In this case, both nodes y_1 and y_2 are included, which means that their edge potentials should also be used:

$$\mu_{f_2 \to y_2} = \sum_{y_1 = 1 \ldots C} \exp(f_n(y_1) + f_e(y_1, y_2)) \tag{3.17}$$

Notice here that the sum runs across all classes of y_1, but the overall message will still be four elements, as each value of y_2 has to be evaluated. If this message passing was to continue in the same direction, *i.e.* opposite the arrows going from f_4 to y_2, then more and more potentials should be included. This exact derivation is what corresponds to the aforementioned α and β tables, which are essentially just the messages being passed between the nodes. Returning to the problem of finding the marginal probabilities for y_2, the final messages going into y_2 are used in equation 3.13 to get the marginals and in this case it would be exactly:

$$p(y_2|X) = \frac{\exp\left(f_n(y_2) + \sum_{y_1=1\ldots C} \exp(f_n(y_1) + f_e(y_1, y_2)) + \sum_{y_3=1\ldots C} \exp(f_n(y_3) + f_e(y_3, y_2))\right)}{\sum_{y_k=1\ldots C}\left(\exp\left(f_n(y_2) + \sum_{y_1=1\ldots C} \exp(f_n(y_1) + f_e(y_1, y_2)) + \sum_{y_3=1\ldots C} \exp(f_n(y_3) + f_e(y_3, y_2))\right)\right)} \tag{3.18}$$

which is equivalent to equation 3.13. Similarly the same principles apply in the case of the more complex graphs in this project.

3.2.4 Parameter estimation for CRFs

The aim of CRFs and other machine learning problems is to fit a model to some known data. The model has been defined in the previous sections and now the parameters of the model have to be tuned to fit the data. The dataset will be denoted by $\mathcal{D}\{y_n; x_n\}^{n=1,2,...N}$, with y being the labels and x the features. The theory is adopted from [22].

Learning objective

The general objective for learning parameters, is to optimize the model w.r.t the training data. If we assume that the data is i.i.d., the likelihood of the model, w.r.t w, is the product over all the training data. And so, the simplest approach to parameter learning in the case of a conditional random field is finding the argmax of the likelihood, w.r.t w and the training data:

$$w* = \underset{w}{\operatorname{argmax}} \prod_{n=1}^{N} p(y^n|x^n, w) \tag{3.19}$$

It is now straight forward to insert the general expression, equation 3.7 from section 3.2.1. However it is common, and an obvious advantage if the log of the model is used instead, it has no implications on the argmax as it is monotonic. The equation now becomes:

$$\begin{aligned} w* &= \underset{w}{\operatorname{argmax}} \log \left[\prod_{n=1}^{N} \frac{1}{Z(x^n, w)} \exp \left(\sum_{F \in \mathcal{F}} w_F^T f_F(y_F^n, x_F^n) \right) \right] \\ &= \underset{w}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{F \in \mathcal{F}} w_F^T f_F(y_F^n, x_F^n) - \sum_{n=1}^{N} \log Z(x^n, w) \end{aligned} \tag{3.20}$$

This last expression, 3.19, is referred to as the maximum log-likelihood, because the parameters are chosen such that they maximize this log-likelihood. Consequently the log-likelihood as a function of the parameters, w, is given as:

$$\mathcal{L}(w) = \sum_{n=1}^{N} \sum_{F \in \mathcal{F}} w_F^T f_F(y_F^n, x_F^n) - \sum_{n=1}^{N} \log Z(x^n, w) \tag{3.21}$$

The objective is now clearly to maximize the log-likelihood, as it will yield the best parameters [31]. It can be shown that the Hessian matrix of the likelihood is positive semi-definite, which means that there is a global optimum. Computing the partition function part of the expression is problematic, because the term is dependent on w

in non-linear ways, there is no closed form solution, and the optimization will rely on gradient descent techniques.

In this work, a different learning method is used, as it yields similar results and is significantly faster to train. So, before descending to optimization techniques, the *Pseudo-Likelihood* and the regularization will be introduced.

Pseudo-likelihood training

The problem of the maximum conditional likelihood and the computation of the partition function term can be overcome, at the expense of the accuracy of the model estimation. Instead of maximizing over the whole conditional probability distribution, local optimization over each node is possible, if each node, y_k, is only dependent on known data. In praxis this mean that the maximization only happens in the neighborhood of y_k, which yields a more effective way to learn the parameters. If the conditional probability is set with the pseudo likelihood it follows that:

$$p(y|x,w) := \prod_{s=1}^{M} p_{\mathrm{PL}}(y_s|y_{-s},x,w) \tag{3.22}$$

$$= \prod_{s=1}^{M} \frac{1}{Z_s(x,y_{-s},w)} \exp(w^T f(x,y)) \tag{3.23}$$

with the partition function:

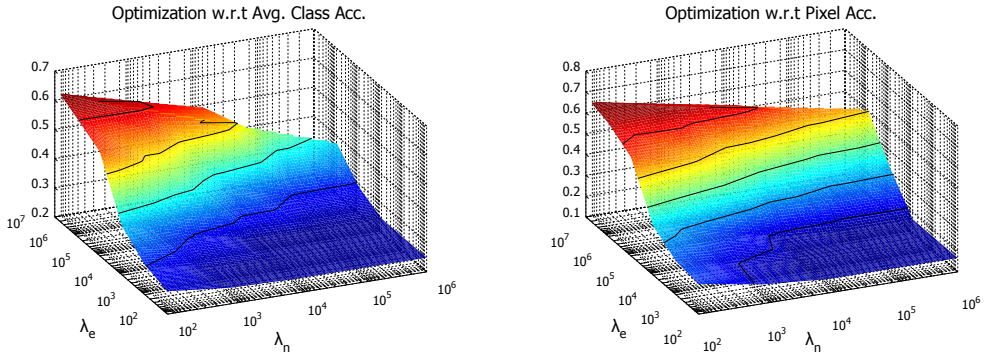
$$Z_s(x,y_{-s},w) \; = \sum_{y_s \in \mathcal{Y}_s} \exp(-w^T f(x,y_{-s},y_s)) \tag{3.24}$$

Also, to avoid noise in the parameters, an $L2$-norm regularization is added, $\lambda \|w\|^2$. The pseudo-likelihood then assumes the form of:

$$\mathcal{L}_{\mathrm{PL}}(w) = \; \lambda \|w\|^2 + \sum_{n=1}^{N}\sum_{s=1}^{M} w^T f(y^n,x^n) - \sum_{n=1}^{N} \log Z_s(x^n,y^n_{-s},w) \tag{3.25}$$

Where the notation of factors, F, have been taken out and it becomes implicit in f. The iteration over s corresponds to iterations over the single random variable in y^n and the subscript of y^n_{-s} signifies that it is all random variables in y^n except y^n_s. This is used because the aim is to optimize over y_s locally with all other variables, y_{-s}, observed. Notice, this idea stems from the local Markov property, which invokes conditional independence.
Finally this expression can be optimized efficiently, as the conditional independence make the number of variables included in the optimization limited to the neighborhood of the random variables. The gradient is obtained as:

$$\nabla_w \mathcal{L}_{\mathrm{PL}}(w) = 2\lambda w + \sum_{n=1}^{N}\sum_{s=1}^{M} f(y^n,x^n) - \mathbb{E}_{y_s \sim p_{\mathrm{PL}}(y_s|y_{-s},x,w)}\left[f(x^n,y^n_{-s},y_s)\right] \tag{3.26}$$



(a) (b)

Figure 3.7: Shows the initial grid search performed to identify optimal regularization parameters for the node and edge potentials. The axis of the graphs are; λ_n, the regularization parameter of the node potentials; λ_e, the regularization parameter for the edge potentials and finally the respective measures of the titles to each plot.

the first term in the sum expresses the value of the features, whereas the second term is the expectation of y_s based on the proposed model. If these two terms are equal, it means that the expectation of the features functions involved with y_s, was 'right' and the two terms cancels out [22]. By adding the regularization term, it is ensured that the parameters do not tend to large values and prevent them from over-fitting to the data.

3.2.5 Test and Optimization of CRF

As explained previously, the parameters of the model needs to be learned. The methods used are pseudo negative log likelihood for training the model and loopy belief propagation for doing inference. The model is evaluated against the test data. The dataset is split in train and test dataset according to a predefined split, originally used by [28] which gives 795 training images and 654 test images. An initial search for parameters is conducted. As there are both edge and node regularization parameters, a 2-dimensional search is needed. The first search is conducted by searching in the range $[10^{-1} : 10^{6}]$ for the node potential, and $[10^{0.1} : 10^{7}]$ for edge potentials, with increments of one in the exponent. For some of the values, the algorithm simply starts to oscillate due to the heavy regularization. The model is trained on a randomly selected subset of the training data, which amounts to 2/3 of the total training data. The remaining 1/3 is used for validation of the model. Note that the ground truth is provided with a thin boundary between each object, which is also visible in the sample images in figure 1.4. These boundaries are marked as zero which translates to unlabeled area. As a consequence these are not counted when evaluating the model and are not considered in the training.

The resulting accuracies are plotted in figure 3.7 and the corresponding feature

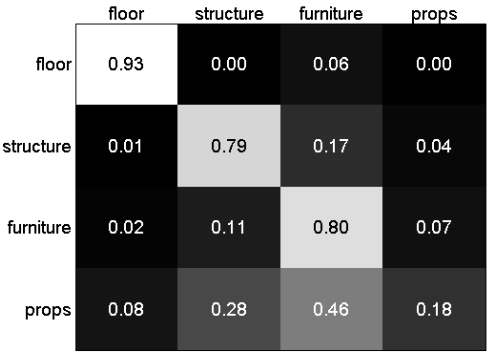


Figure 3.8: Shows the confusion matrix for 4-class classification with the regularization parameters $\lambda_e = 10^6$ and $\lambda_n = 10^1$.

set used is displayed in table 3.2. After consulting the results of this test, a more informed search is conducted. The second grid search optimization is done over the range $[10^1 : 10^3]$ for the node regularization and $[10^6 : 10^8]$ for the edge parameters. The test is not shown here, but the optimal parameters are found and shown in table 3.1 together with the resulting accuracies.

Furthermore in figure 3.8 the confusion matrix is shown. From these results a few things are clear, the model performs well on the three main classes but poorly on the small objects class, props. This does make sense as this class constitutes all possible small objects ranging from water bottles to art pictures. However it is clear that performance have to be improved on this class, in order to increase the overall performance.

The implementation of CRFs used in this project is by Mark Schmidt from his UGM library[1].

Opt. Param.	
λ_n	10^1
λ_e	10^6

Resulting accuracies	
Pix. acc.	70.9
avg. class acc.	67.7

Table 3.1: Optimal parameters found from the grid search on the range $[10^1 : 10^3]$ for the node regularization and $[10^6 : 10^8]$ for the edge parameters.

[1]http://www.cs.ubc.ca/ schmidtm/Software/UGM.html, accessed June 2015

Feature sets	Node	Edge
Generic features	G	
Lab color	•	•
3D position		•
Normal	•	•
Std. dev. of the normal	•	•
Blurred depth gradient magnitude	•	•
Blurred image gradient magnitude	•	•
Room Layout features	RL	
Room layout wall probability	•	•
Inverse room layout wall probability	•	•
Normal features	N	
Discrete height	•	•
Vertical normal comparison	•	•
Wall normals comparison	•	•
Floor probability	•	•
Wall probability	•	•
Continuous height		•

Table 3.2: Feature reference table.

3.3 Random Forest with Random Offset Features

To enhance the current models ability to detect the props class, another machine learning model is added. This is similar to [21, 30], where a Random Forest using random offset features is also added as an initial estimate of the labeling. It fits in the overall model as shown in figure 3.9. The estimated confidences from this method is aggregated into the superpixels and is further fed into to the CRF. In general random forests have shown to be effective for segmentation tasks before [26, 30, 11]. The main problem is that it usually results in scattered classifications. This however, can be resolved using it in conjunction with the CRF as well as the random offset features that each add a spatial and contextual link.

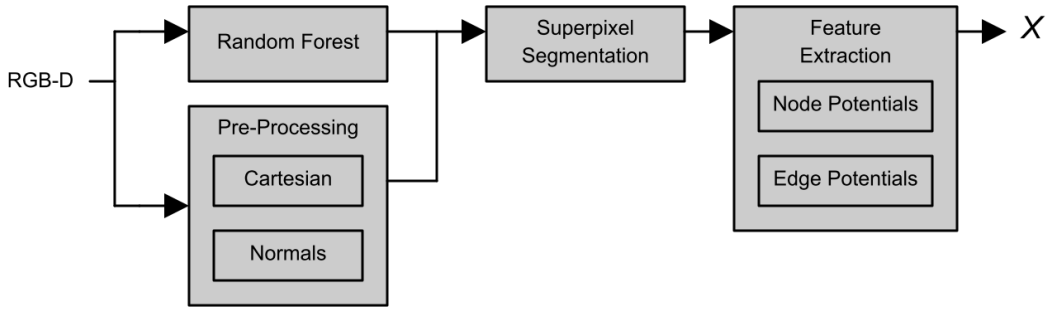


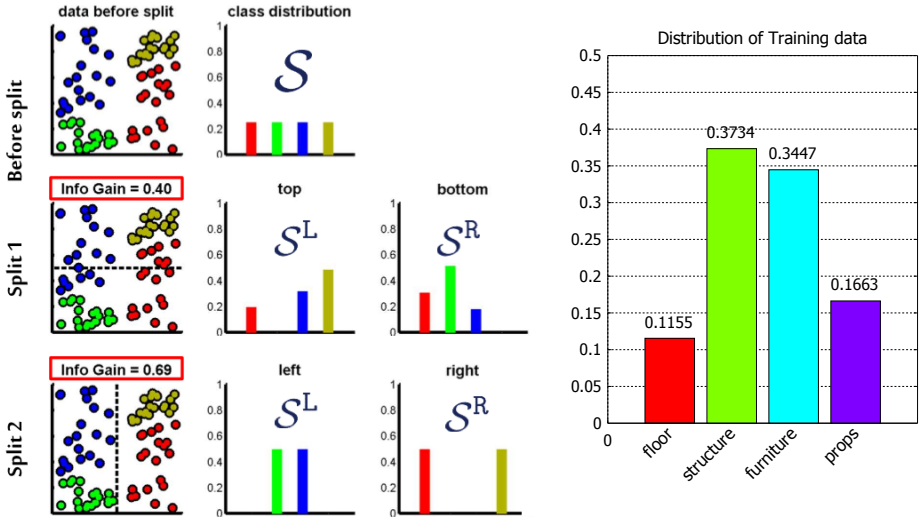
Figure 3.9: Overview of the pipeline into the CRF. The random forest is used in parallel with the preprocessing and aggregated in the SLIC segmentation.

3.3.1 Random Forests

The name random forest covers a generalized framework where a set of decision trees are grouped in an ensemble to form a stronger classifier. The model is however flexible and can be varied in a few ways.

A decision tree is made up of weak learners in a sequential tree structure and in each branching of the tree the data is separated into two branches. In this fashion an input feature vector, $\mathbf{v}$, can propagate through the tree and end in a single leaf, which gives some probability estimate of the class label of the sample. Three trees are depicted in figure 3.11.

The decision tree is trained using a sample set X, with ground truth Y. The sample set is evaluated according to some objective function. A popular method is to use the information gain with Shannon's entropy, which is a measure of how much information is gained from making a certain split. The split is the one that maximizes the information gain out of every possible split. This method is feasible to use for simple threshold split functions, however, other split functions such as a hyperplanes or conic sections can also be used where more elaborate objective functions have to



(a) (b)

Figure 3.10: Figure (a) shows examples data distributions and their corresponding splits and information gains. [11]. Figure (b) shows the distribution of the classes in the training data.

be used. The Information gain is defined as:

$$I(S,\theta) = -H(S) \sum_{i \in \{L,R\}} \frac{|S^i|}{|S|} H(S^i) \tag{3.27}$$

Where S is the complete set of data and θ is some split parameter, which could just be a threshold. $H(S)$ is an entropy and $\{L, R\}$ is the left and right part of the data after performing some split. The entropy function could be the Shannon entropy:

$$H(S) = -\sum_{c \in C} p(c) \log(p(c)) \tag{3.28}$$

where C are the possible classes and $p(c)$ is the distribution of the classes in S. Putting these two expressions together, the Information gain expresses the available information in the set, S, by the first term in equation 3.27. The second part evaluates the information in each partition, S^L and S^R, and this is subtracted to obtain the information gain. This concept is illustrated in figure 3.10 where some split thresholds are displayed on two dimensional data and to the left, the data distributions are shown.

After the data has been split at the initial node, the same process is carried out in the following nodes and at each step some information is gained. When there is only a certain number of samples left in a node, or a certain depth in the tree is reached, the branch is ended in a leaf. The distribution of the samples left in the tree is then

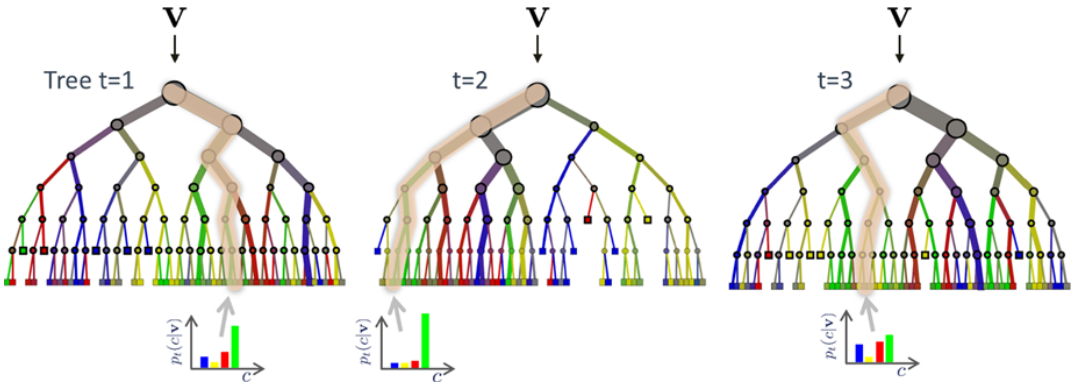


Figure 3.11: A set of three decision trees. A vector **v** is propagated through each tree and the resulting probability distributions are displayed at the leaf nodes [11].

used as the probability at the leaf. For the single decision tree a sample will end up in a single leaf and obtain a probability distribution of that leaf.

Random forests are simply ensembles of decision trees with randomness injected to the model which has shown to improve performance [5]. The first randomness in the model is to use bagging. Bagging is a widely used method to make a model generalize better. It creates random subsets of the dataset to which some samples are left out. In this project, approximately 2/3 of the available data is used for each decision tree in the random forest. Because this process is randomized and with replacement, some of the samples are not going to be used at all during the training. These are called the *out-of-bag* samples and will be important in the coming.

The effect of doing bagging is to create a set of unique classifiers that are similar and instead of using the ultimate judgment of one decision tree, the decision becomes a combination of multiple opinions, which increases the generalization [5]. Apart from the bagging a second randomness is injection by only using a random subset of features for each split in the decision trees [6]. If there for instance was 100 features for each sample, then maybe only 10 features would be selected at random to determine each split. Each of these randomness injections increases the generalization of the model better.

When inference is done on a sample, it is given to each tree in the random forest and propagated through the trees. Once the leaf nodes are reached the probabilities assigned for each leaf will then be averaged and the final probability distribution obtained. Other ways of assigning probabilities can be used, such as taking the product of the distributions and normalizing. In the presented work, the Matlab implementation of random forests, TreeBagger[2], is used. This furthermore allows to add a prior probability to each class to accommodate for skewed data distributions. In the case of this problem the data is skewed with the distribution shown in figure 3.10.

[2]http://se.mathworks.com/help/stats/treebagger.html, accessed June 2015.

This works in the model by modifying the weight of each class in the information gain calculation. This re-balances the data so it is trained more accurately [11].

3.3.2 Random Offset Features

Based on the assumption that there exist non-obvious links and relationships between each pixel and its contexts, a special type of feature called *random offset features* are used in the random forest. The aim of the feature is to capture the surroundings of a query pixel and use this to infer the label of the pixel. The way the features are realized, is by randomly generating a set rectangles around the query pixel and subtracting the sums of these. Each feature is one subtraction between two of these randomly generated rectangles. Intuitively this does not lead to anything useful, but by evaluating a large number of these randomly generated features, some are more saying than others and eventually a subset of these random features can be discriminative.

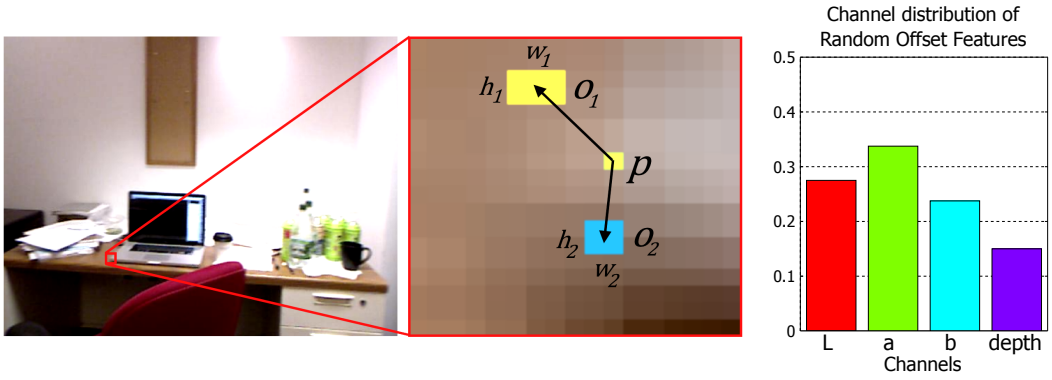
In praxis the random features are generated by making random offsets within some range of image coordinates, hence the name random offset features. Around these random offsets, randomly sized rectangles within some range are now defined. The idea is now to sum the values in the rectangle, and subtract them from one of the other offset which then constitutes one random offset feature. The idea is depicted in figure 3.12(a).

To find discriminative features, several thousands features are created randomly. All of these features are then extracted from a randomly chosen subset of the training data and this will act as a training set. Based on the training set, the features can be evaluated to find important ones. Here, the features are evaluated based on the *out-of-bag feature error rate* of a trained random forest. This error is generated by running the out-of-bag samples through the random forest when it is done training and then evaluating the results. Based on this the best features are chosen and the rest discarded. After finding the important features, the random forest is retrained on the training samples and the result is model which is a fast and cheap method to give an initial pixel label estimate. Remark, since the features are based on rectangular areas, this method is optimized using integral images.

In this project, the data that the offset features work on are the CIELab color space and the depth channel. The features are arranged such that the subtraction between two offsets with rectangles can happen either between the CIELab color channels or the depth channel, *i.e.* only depth can be subtracted from depth, but the color channels can be subtracted from one another to form a feature. The same principle is used in [21, 30], which is originally inspired by the work of Viola and Jones [33] using Haar features.

3.3.3 Test and optimization of model

Initially, 3000 features are trained and the samples used are chosen with a uniform distribution. The uniform sampling across the classes is to assure a similar number of



(a) (b)

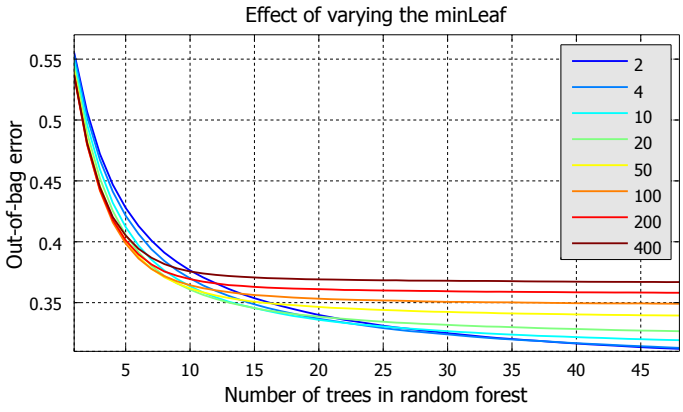
Figure 3.12: Figure (a) shows the concept of random offset features. The green pixel, p, is the query pixel, o_1 refers to offset one, w_1 refers to the randomly chosen width, h_1 to the randomly chosen height and likewise with subscript 2. (b) shows the distribution of the channels used by the 40 best features.

out-of-bag samples to evaluate the performance. Based on the performance of each feature, the 40 best are selected and these are used in the following random forest model.

To optimize the model a series of experiments are run. Initially, all the data is processed with the 40 samples found previously, this results in a dataset of 55.7 GB. It is unfeasible to process the complete dataset in a reasonable amount of time, therefore a random subset of data is chosen for optimization of the model.

The parameters of the random forest are codependent and therefore it is not clear how to best optimize them. Here the minimum number of samples at each leaf is optimized first. The test is run with 0.5 GB of data, corresponding to $1.563 \cdot 10^6$ samples and the number of features used at each split is set to 7, close to the square-root of 40 which is a rule-of-thumb. The exact parameters are shown in figure 3.13 together with a graph of the out-of-bag errors, which is just the accuracy based on the out-of-bag samples. Breiman [5] and Wolpert and Macready [34] shows that the out-of-bag error is a strong indicator of the generalization of the model and argues that a cross-validation is redundant.

The results in figure 3.13 clearly shows that a small number of samples at the leafs will give better performance. However, as Breiman notes in [6], the random forest may generalize better if the trees in it are not grown completely to one sample at each leaf. Also, a consequence of a smaller number of samples at each leaf the trees will grow bigger and the bigger the trees, the longer the inference time. The other variable in the figure is the number of trees. Here a number of trees where the accuracy starts to converge is sought. Also, a point to keep in mind is that the processing time increases linearly with the number of trees. By compromising on the number of trees and the performance a preferred operating point can be

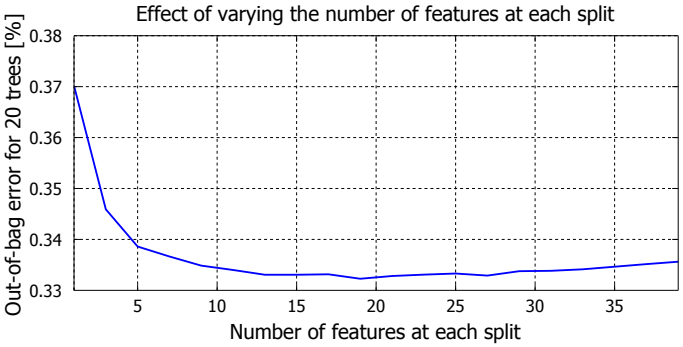


(a)

Parameter	value
nFeatures	40
nSamples	$1.56 \cdot 10^6$
nTrees	48
nSplitFeatures	7
minLeaf	-

(b)

Figure 3.13: Graph (a) shows the out-of-bag error for different values of the minimum number of samples at the leafs. Table (b) shows the parameters used for the test. The parameters are: nFeatures = number of features to use, nSamples = number of samples to train on, nTrees = number of trees, nSplitFeatures = number of random chosen features to use in split functions, minLeaf = minimum number of samples at each leaf



(a)

Parameter	value
nFeatures	40
nSamples	$1.56 \cdot 10^6$
nTrees	20
nSplitFeatures	-
minLeaf	10

(b)

Figure 3.14: Graph (a) shows the out-of-bag error for the different number of features to sample at each node. Table (b) shows the final parameters used for the model.

chosen. Here an operating point of 20 trees with a minLeaf of 10 is chosen as a good compromise to continue with. Next, this operating point is evaluated w.r.t the number of features to choose at random in each split. This evaluation is carried out in a similar fashion and the results are shown in figure 3.14 (a). Similar to the number of trees, the number of features at each split has a somewhat linear cost in terms of computational performance.

The number of split features is set to 13, as it seems to converge at that point. The final parameters used are shown in table 3.14 (b). Using the model, inference is

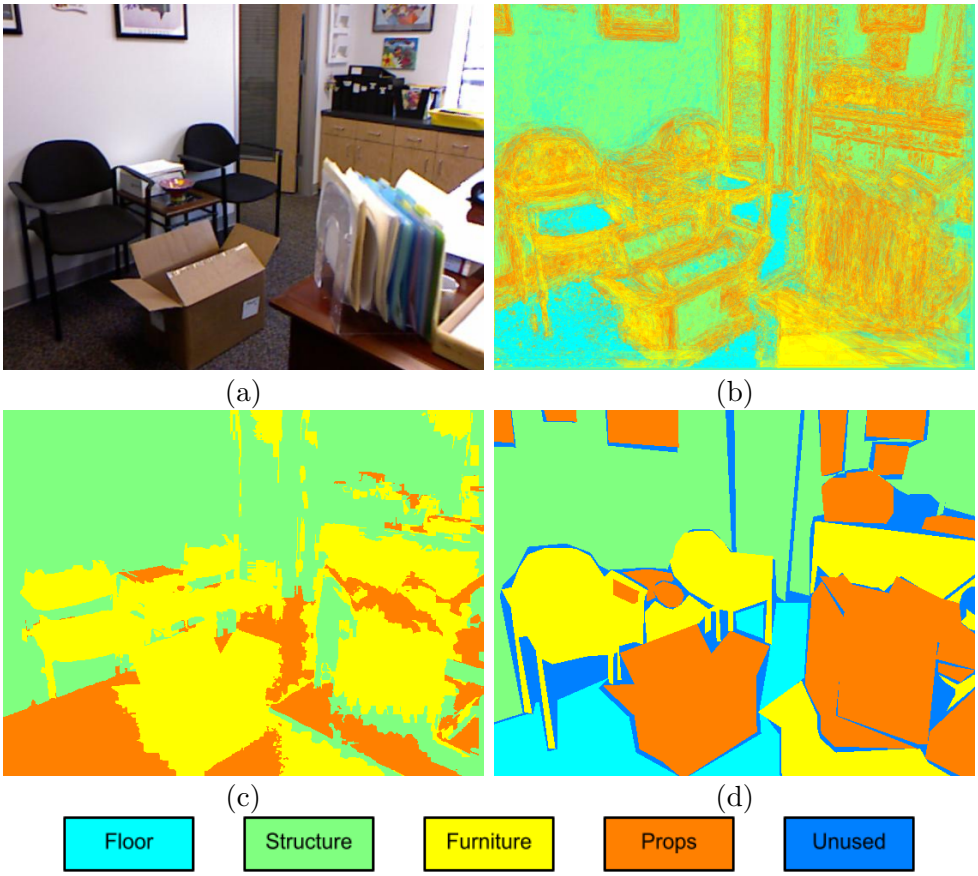


Figure 3.15: (a) shows an original image and its corresponding random forest response in (b). In figure (c) the corresponding CRF response (without the RF!) is shown and lastly in (d) the ground truth.

Parameter	value
nFeatures	40
nSamples	$1.56 \cdot 10^6$
nTrees	20
nSplitFeatures	13
minLeaf	10
Accuracy	66.6 %

Table 3.3: Optimal parameters found for the random forest. Note, the accuracy is the out-of-bag error and might not generalize to dataset.

done on a sample image, which is shown in figure 3.15 (a) and (b).

3.4 Multi-scale Stacked Sequential Learning

To enhance the model, the Stacked Sequential Learning (SSL) framework by Cohen and Carvalho [8] is used on the system. In simple words it stacks another classifier on top of the existing. The way it enhances the results is by considering both the original data *and* the output of the previous model. The overall pipeline with the SSL framework is shown in figure 3.16. While it is usually used in conjunction with a weak and noisy learner it has shown to improve the performance of conditional random fields. It considers the confidence map from the previous classifier and looks at the neighborhood of a query point. This neighborhood, in conjunction with the original features, are fed to a new stacked classifier and from that infer a label.

In [23] the framework is generalized to a multi-class case, where the ECOC framework generalizes the model. In this project, a mix of these will be used as a multi-scale decomposition similar to the one of [23] is used while the model is the original one from [8]. It is stated in [23] that the framework is most suitable for 1D or 2D applications, however it has not previously been applied to 3D indoor semantic segmentation and might show useful results.

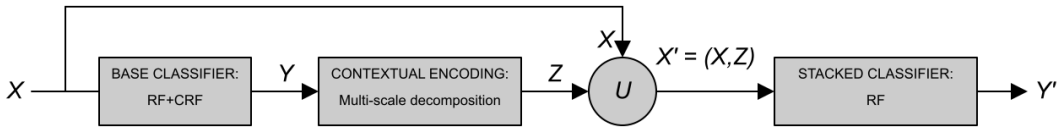


Figure 3.16: Graph showing the overall pipeline with the SSL framework added.

The initial feature set, X, is provided as input for the CRF classifier. The predictions, Y, of the classifier are then decomposed in a multi-scale decomposition to obtain a spatial context feature, Z. The initial features, X, and the newfound features, Z, are merged in an extended feature set, X'. The extended dataset is evaluated by another trained classifier to give the final predictions, Y'.

3.4.1 Multi-scale Decomposition

Following the framework, a contextual feature is extracted from the confidence map provided by the base classifier. The aim of this feature is to utilize the contextual information in the data to correct the labellings.

To encode the contextual information, a multi-scale decomposition is created from the spatial relations in the scene the depth map. The feature consists of the distributions of nearby pixels confidence maps. In this manner a confidence for each label in an area will be obtained, *e.g* if all nearby pixels are floor, then the current pixel is probably also floor. These distributions are created for multiple distances from the query point. For a point p with the pixels P_i situated in some half-closed distance intervals from p denoted by $i = \{1, 2, ..., n\}$, the multi-scale decomposition is the averaged distribution of the confidence map, C, of each class $c = \{1, 2, ..., k\}$ in each

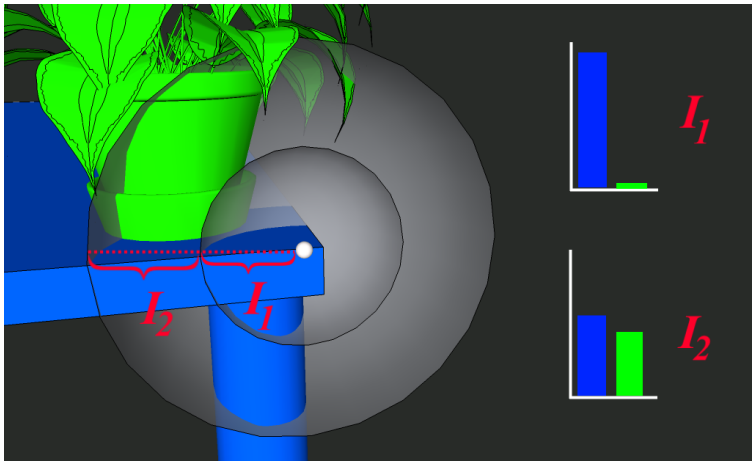


Figure 3.17: Depiction of the multi-scale decomposition. The figure shows a potted plant and a table, representing the green and blue class respectively. From the query point in the center, there are two intervals, I_1 and I_2 that generates concentric spheres. In each interval, the distribution of the confidence map is extracted, which gives the resulting feature vector.

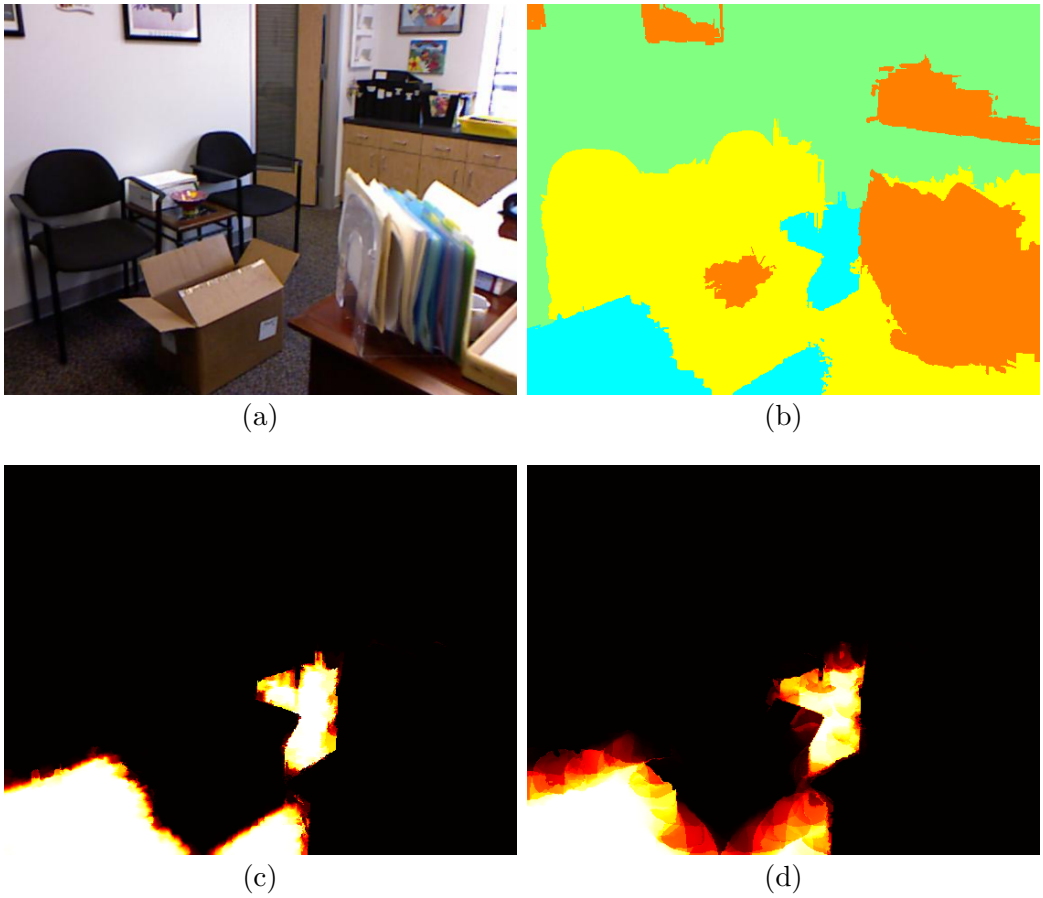
interval. It can be described as:

$$a_{n(c-1)+i} = \frac{\sum C_c(P_i)}{|P_i|}, \tag{3.29}$$

where the notion C_c refers to the confidence map for the class c. The vector $\mathbf{a}$ is the resulting feature vector for p. A graphical depiction of the multi-scale decomposition is shown in figure 3.17. In effect, this decomposition creates a feature vector that is linear in size with the number of classes, *e.g* kn, as a consequence the feature is only suited for problems with a relatively low number of classes, although it can be compressed when dealing with a large number of classes[23].

In [25], Sampedro *et al* describes the potential enhancement of problems with 3D sequential data, however RGB-D data is not sequential in the same way as for example medical volume imaging for which their method was originally intended. Instead of analyzing a fixed area surrounding a query pixel, the RGB-D data have to be analyzed w.r.t the spatial layout of a given scene. As a consequence, each and every pixel in the image have to be compared to all other pixels in the data, to find those points that lie within the given intervals. This has obvious implications on the speed of the algorithm. To mend the problem, each scene is split into a grid and initially the query pixel is compared to this grid to disqualify a large amount of the scene efficiently.

As the stacked classifier, another random forest is used for its good balance of speed, performance and its ease of implementation, see section 3.3 for details on this method. This classifier now has a feature vector consisting of the initial features and the multi-scale decomposition. To give an idea of how this last feature looks, figure



(a) (b)

(c) (d)

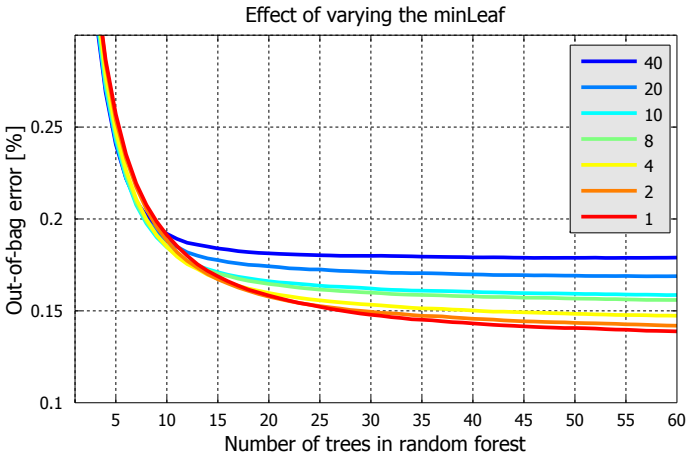
Figure 3.18: Multi-scale decomposition feature. Figure (a) shows the original image and (b) shows the CRF labeling. Figures (c) and (d) shows two different intervals for the floor class, the intervals are 2cm and 11cm respectively with the highest intensity being white.

3.18 shows an original image together with the CRF response and a decomposition of the *floor* class of this response. In this project the distance intervals have been set to $2, 5, 11, 30$, all measured in centimeters.

The intuitive understanding of the decomposition images is that for a given point, the intensity is equal to the distribution of the plotted class in the given interval.

3.4.2 Test and Optimization

To ensure the good performance, the stacked random forest is optimized using a similar strategy as in subsection 3.3.3. As the parameters are correlated, the parameters are optimized in turn. First, the minimum number of leafs is tested to see how the model performs for each setting. The resulting graphs are shown in figure



(a)

Parameter	value
nFeatures	65
nSamples	$0.96 \cdot 10^6$
nTrees	60
nSplitFeatures	8
minLeaf	-

(b)

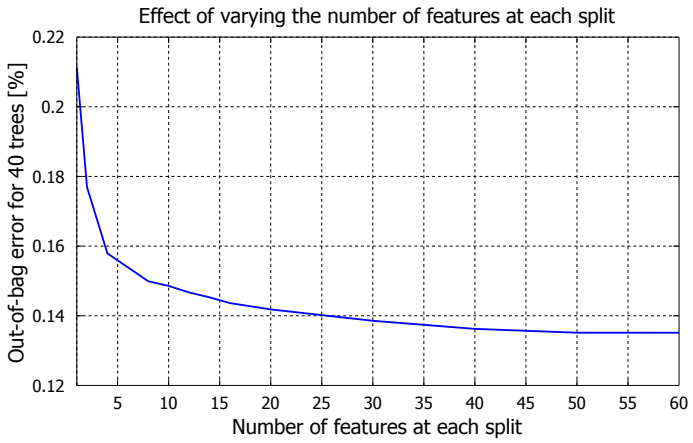
Figure 3.19: Graph (a) shows the out-of-bag error for different values of the minimum number of samples at the leafs. Table (b) shows the parameters used for the test. The parameters are: nFeatures = number of features to use, nSamples = number of samples to train on, nTrees = number of trees, nSplitFeatures = number of randomly chosen features to use in the split function, minLeaf = minimum number of samples at each leaf

Parameter	value
nFeatures	65
nSamples	$0.96 \cdot 10^6$
nTrees	40
nSplitFeatures	30
minLeaf	4
Accuracy	86.1 %

Table 3.4: Optimal parameters found for the stacked random forest. Note, the accuracy is the out-of-bag error and might not generalize to the test dataset.

3.19. Using the same considerations as before - that the model will generalize better if the tree depth is not complete - the minimum number of samples at each leaf are chosen to be 4. To optimize further, the number of samples to choose at random for each decision split is chosen by plotting the out-of-bag error for different number of features. The results is shown in figure 3.20. Based on the plot an operating point of 30 split features are chosen.

The optimized performance together with the specified parameters are shown in table 3.4.



(a)

Parameter	value
nFeatures	65
nSamples	$0.96 \cdot 10^6$
nTrees	40
nSplitFeatures	-
minLeaf	4

(b)

Figure 3.20: Graph (a) shows the out-of-bag error for the different number of features to sample at each node. Table (b) shows the final parameters used for the model.

Chapter 4

Experiments and Results

In this chapter the final optimizations of the combined model will be explained briefly, followed by an ablation study which shows the performance of the individual system parts. Finally the combined test results are shown. The model is mostly evaluated based on the simple measures of average class accuracy and pixel accuracy as this is the trend in literature on this exact problem. Accuracy for multi-class problems is confuse and multiple definitions exist. To clarify, the measure used here is the average of the confusion matrix, though more elaborate measures exist. The results are based on pixel wise comparison.

4.1 Re-optimization of Conditional Random Field

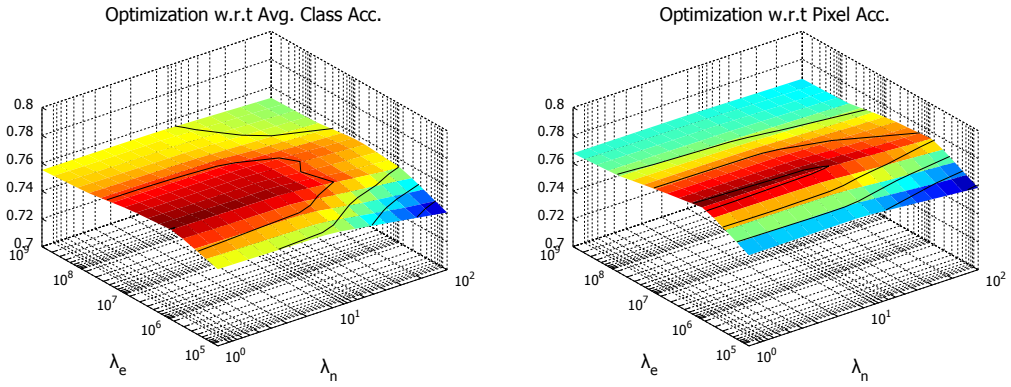
As the base model comprises two different methods, they have to be combined and optimized to get the best performance. Initially the predictions from the random forest with random offset features are provided as an input to the CRF. As this is a strong feature, the CRF have to be re-optimized. The same procedure as described in section 3.2.5 is used to optimize this first part of the model. The parameters are likely to change, however within the same vicinity of the previous parameters, therefore the grid search is conducted in the ranges $[10^0 : 10^2]$ for the node regularization and $[10^5 : 10^9]$ for the edge parameters. The resulting optimizations are shown in figure 4.1.

The optimal parameters and measures are the ones given in table 4.1.

Opt. Param.	
λ_n	10^0
λ_e	10^6

Resulting accureacies	
Pix. acc.	78.1
avg. class acc.	76.5

Table 4.1: Optimal parameters found from the grid search on the range $[10^0 : 10^2]$ for the node regularization and $[10^5 : 10^9]$ for the edge parameters. Note that the accuracies are



(a) (b)

Figure 4.1: Shows the grid search performed to identify optimal regularization parameters for the node and edge potentials. The axis of the graphs are; λ_n, the regularization parameter of the node potentials; λ_e, the regularization parameter for the edge potentials and finally the respective measures of the titles to each plot.

4.2 Ablation Study

In order to asses the role of each part in the model, an ablation study is conducted. The results of this test are shown in table **??**. Each row shows the results from one test. The first column shows the method used in the given case and the second shows the features used for the test, the exact abbreviation are G for generic features, N for normal features, RL for room layout features, ROF for random offset features and finally S for the multi-scale decomposition. This assessment is carried out on the entire test data set, which consists of 654 unseen images.

Methods	Features	Floor	Struct.	Furn.	Props	Pr. class acc.	Pix. acc.
CRF	G	85.3	77.6	60.5	1.8	56.3	60.0
CRF	G+N	92.8	77.3	78.0	14.5	65.7	68.9
CRF	G+RL	83.0	76.4	74.2	2.7	59.1	64.1
CRF	G+N+RL	**94.3**	78.8	**81.1**	13.8	67.0	70.6
RF	ROF	84.8	74.7	68.4	14.0	60.4	63.6
RF+CRF	ROF+G+N+RL	93.3	**79.4**	74.0	**33.4**	**70.0**	**71.5**

Table 4.2: Table of feature sets used in the ablation study. The abbreviations used in the table refer to the sets shown in table 3.2 in subsection 3.2.5.

4.3 Combined Results and State-of-the-Art comparison

Finally, the whole system is tested. With all models optimized, tested on all test data, the results are given in table together with the current state-of-the-art methods.

Work	Floor	Structure	Furniture	Props	Per class acc.	Pix. acc.
Müller and Behnke[21]	94.9	78.9	71.1	42.7	71.9	72.3
Couprie *et al*[10]	87.3	87.8	45.3	35.5	63.5	64.5
Khan *et al*[17]	87.1	**88.2**	54.7	32.6	65.6	69.2
Gupta *et al*[13]	82	73	64	37	65	64.9
Nico Höft *et al*[15]	77.9	65.4	55.9	**49.9**	62.0	61.1
This work	**95.5**	80.5	**77.1**	35.3	**72.1**	**73.8**

Table 4.3: State of the art comparison.

Qualitative results

Some example segmentations can be found in figure 4.2 and 4.3, where the different steps of the algorithms predictions are also shown.

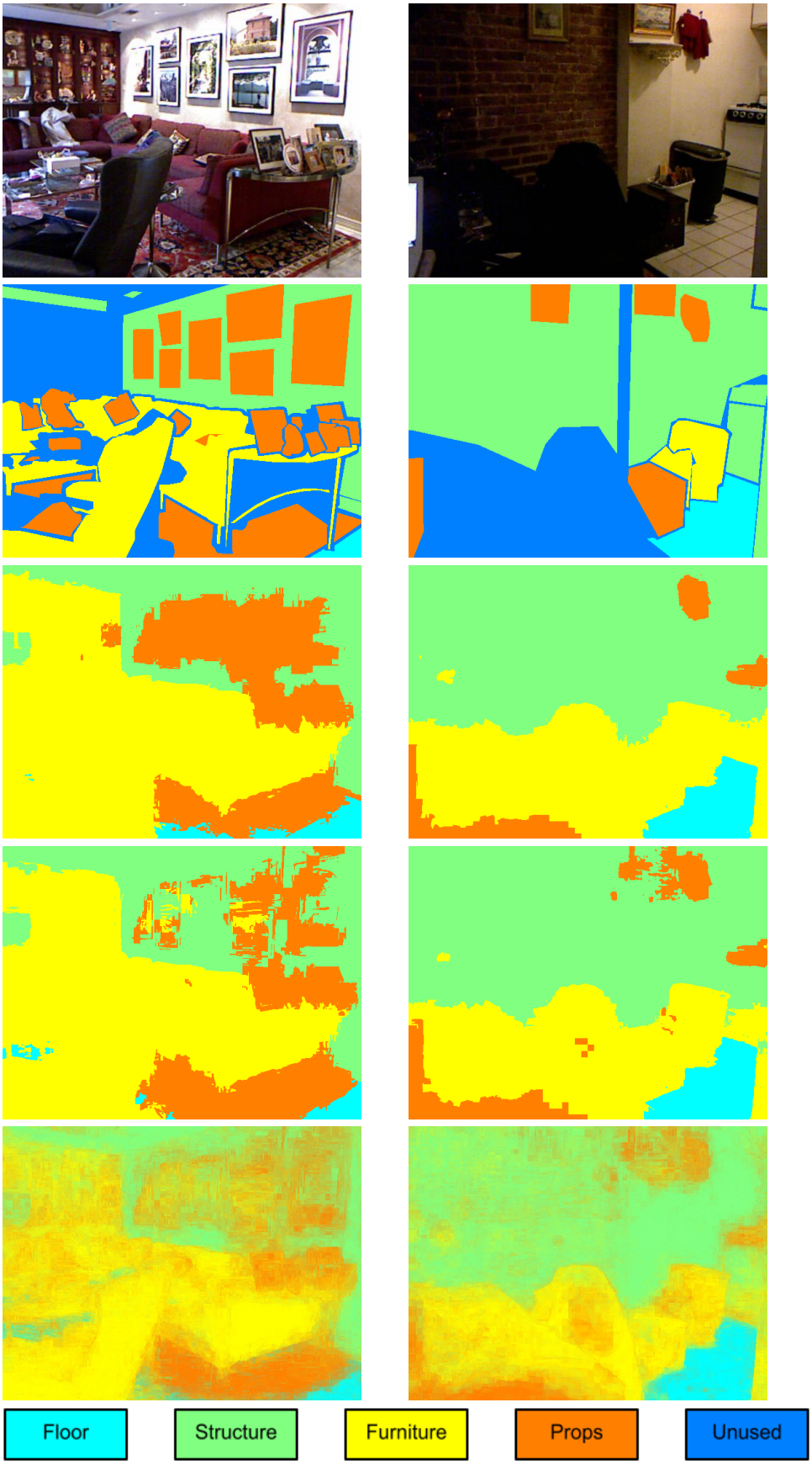


Figure 4.2: First two rows shows the original image and the ground truth, following are the CRF+RF, then the CRF+RF+RFS (which is the final labeling), and the same as confidences and finally the ground truth.

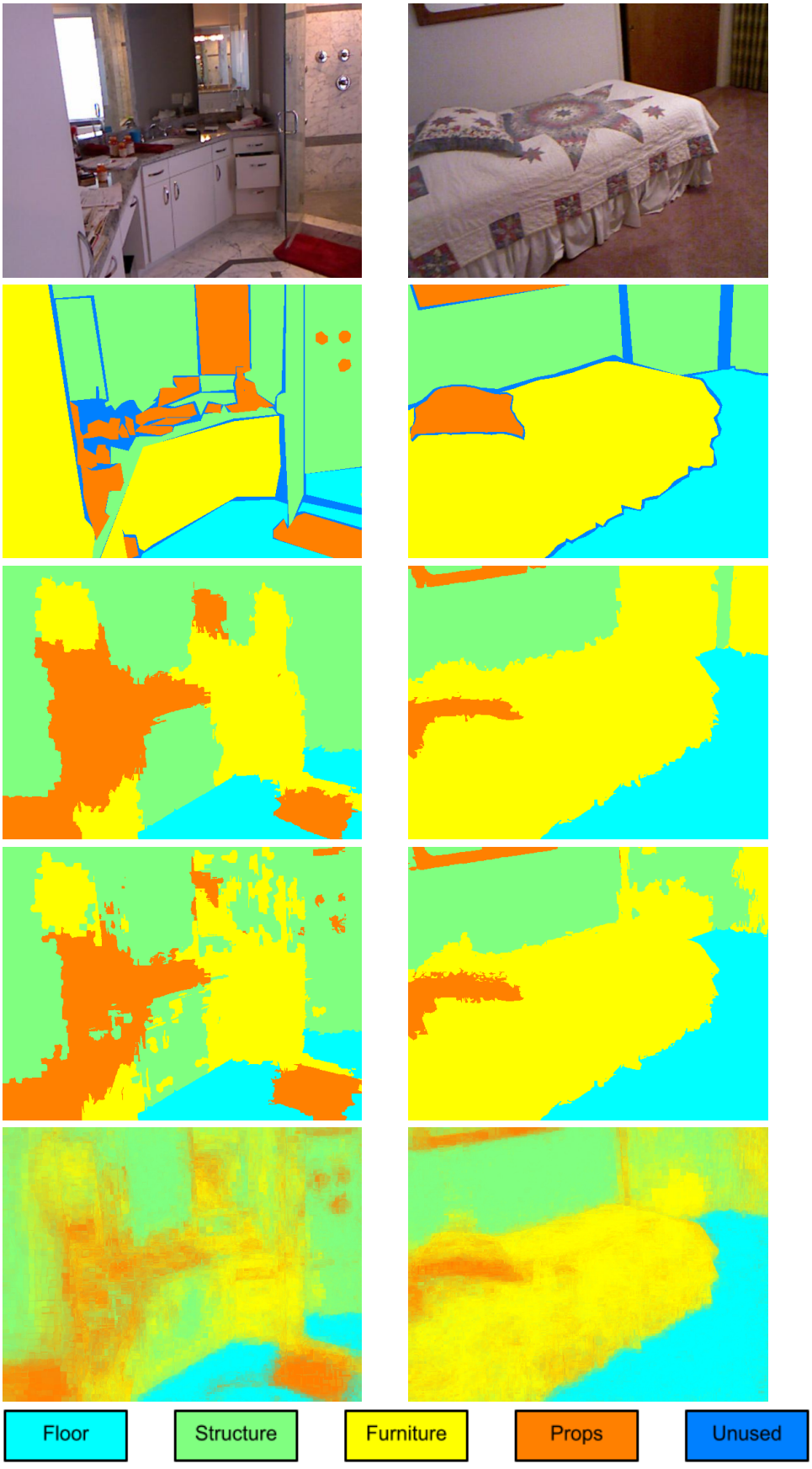


Figure 4.3: First two rows shows the original image and the ground truth, following are the CRF+RF, then the CRF+RF+RFS (which is the final labeling), and the same as confidences and finally the ground truth.

Bibliography

[1] Asimo - the honda humanoid robot asimo. website, January 2015.

[2] Toyota partner robot family. website, January 2015.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, Nov 2012.

[4] Gonzalez Alonso, I. Fernandez, M. Maestre, J.M., Garcia Fuente, and M.d.P.A. *Service Robotics within the Digital Home : Applications and Future Prospects*. Dordrecht : Springer Science+Business Media B.V., 2011.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] César Cadena and Jana Košecka. Semantic parsing for priming object detection in rgb-d scenes. *The International Journal of Robotics Research*, 2013.

[8] William W. Cohen and Vitor R. Carvalho. Stacked sequential learning. *Proceedings of IJCAI*, pages 671–676, 2005.

[9] James M. Coughlan and Alan L. Yuille. Manhattan world: Orientation and outlier detection by bayesian inference. *Neural Computation*, 2003.

[10] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *CoRR*, abs/1301.3572, 2013.

[11] Antonio Criminisi and Jamie Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

[12] Carlo Gatta, Eloi Puertas, and Oriol Pujol. Multi-scale stacked sequential learning. *Pattern Recognition*, 44:2414–2426, 2011.

[13] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 564–571. IEEE, 2013.

[14] Taha Hamedani and Ahad Harati. Multi scale crf based rgb-d image segmentation using inter frames potentials. *RSI/ISM International Conference on Robotics and Mechatronics*, 2:920–925, 2014.

[15] Nico Höft, Hannes Schulz, and Sven Behnke. Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks. *Advances in Artificial Intelligence*, 8736:80–85, 2014.

[16] Dirk Holz, Stefan Holzer, RaduBogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 306–317. Springer Berlin Heidelberg, 2012.

[17] Salman Hameed Khan, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Geometry driven semantic labeling of indoor scenes. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 679–694. Springer International Publishing, 2014.

[18] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors (Basel, Switzerland)*, 12(2):1437–1454, 2012.

[19] Klaas Klasing, Daniel Althoff, Dirk Wollherr, and Martin Buss. Comparison of surface normal estimation methods for range sensing applications. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3206–3211. IEEE, 2009.

[20] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 689–694, New York, NY, USA, 2004. ACM.

[21] Andreas Christian Müller and Sven Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of rgb-d images. *ICRA*, 2014.

[22] Sebastian Nowozin and Christoph Lampert. *Structured Prediction and Learning in Computer Vision*. NOW - the essence of knowledge, 2011.

[23] Eloi Puertas, Sergio Escalera, and Oriol Pujol. Generalized multi-scale stacked sequential learning for multi-class classification. *Pattern Analysis and Applications*, 18(2):247–261, 2015.

[24] Md. Alimoor Reza and Jana Kosecka. Object recognition and segmentation in indoor scenes from rgb-d images. 2013.

[25] Frederic Sampedro, Sergio Escalera, and Anna Puig. Iterative multi-class multi-scale stacked sequential learning: Definition and application to medical volume imaging. *Pattern Recognition Letters*, 46:1–10, 2014.

[26] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1297–1304, 2011.

[27] Nathan Silberman and Rob Fergus. Indoor scene segmentation using a structured light sensor. *ICCV*, 2011.

[28] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[29] Jorg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense real-time mapping of object-class semantics from rgb-d video. *Journal of Real-Time Image Processing*, pages 1–11, 2013.

[30] Jörg Stückler, Benedikt Waldvogel, Hannes Schulz, and Sven Behnke. Dense

real-time mapping of object-class semantics from rgb-d video. *To appear in Journal of Real-Time Image Processing*, 2013.

[31] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. Department of Computer Science University of Massachusetts.

[32] André Ückermann, Robert Haschke, and Helge Ritter. Real-time 3d segmentation of cluttered scenes for robot grasping. 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), pages 198–203, 2012.

[33] Paul Viola and MichaelJ. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[34] DavidH. Wolpert and WilliamG. Macready. An efficient method to estimate bagging's generalization error. *Machine Learning*, 35(1):41–55, 1999.

[35] Shuai Zheng, Ming-Ming Cheng, Jonathan Warrell, Paul Sturgess, Vibhav Vineet, Carsten Rother, and Philip H. S. Torr. Dense semantic image segmentation with objects and attributes. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3214–3221, June 2014.

[36] C. Lawrence Zitnick and Sing Bing Kang. Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision*, 75(1):49–65, 2007.