Master's Thesis

Casting Light on Object Pose Estimation via Object Coordinate Regression

Supervisor
Dipl.-Inf. Eric Brachmann

Responsible Professors
Prof. Ph.D. Carsten Rother
Prof. Dr. rer. nat. Stefan Gumhold

6. May 2015

Stephan Ihrke
Mat.-Nr.: 3617757

Statutory Declaration

I hereby confirm that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgements. This applies also to all graphics, schematics and tables included in the thesis.

Dresden, 6. May 2015

Stephan Ihrke

Contents

1 Introduction

6D object instance pose estimation is the task of inferring the position and orientation of a rigid object from a single image. It is an important problem in robotics e.g. for grabbing objects in a household assistance system. Current state of the art pose estimation algorithms are able to achieve impressive detection rates, however, some challenges such as occlusion, heavy clutter in the scene, the extension to a large scale of simultaneously detectable objects, and the robustness towards different lighting conditions are yet to overcome. The latter will be the main focus of this work.

The prevalent strategy to handle different lighting conditions in the literature is to make the system invariant towards them. That means that these algorithms try to compensate for the different appearances under different lighting conditions without the need for knowledge of the exact lighting parameters. This is appealing because it presents a catch-all approach which simplifies the task by a large margin. In order to achieve this lighting invariance, these approaches often rely on edge information recovered from the color image or shape information recovered from the depth image [12, 16]. These approaches are thus ignoring the benefits of pure color features for their inference process. The baseline for this work [6] by Brachmann et al. uses depth as well as color features to infer the final object pose. They present a test on different lighting conditions which shows that the system is capable of handling multiple lighting conditions well. In my investigation of their system, it became apparent that it relies heavily on the depth features to infer the correct pose. Future iterations of this system should be able to recognize poses from an RGB image only and therefore need to be able to exploit color features more effectively.

This work will explore means to explicitly model the lighting condition as a global latent variable in order to increase the performance of color features for the pose estimation procedure. It is the goal of this thesis to compare the proposed light modeling approaches to the baseline light invariance approach and therefore provide cues for

further research in this area. Now, I want to give a brief overview over the baseline method for this thesis and how I plan on extending it for the purpose of modeling lighting conditions. The system operates in a two stage pipeline:

(1) **Object Coordinate Regression:** A random forest densely predicts the object coordinate and class (which object is present) for each pixel in the image.
(2) **Geometric Verification via RANSAC:** An energy function which describes the quality of a hypothesis is optimized with a RANSAC-like scheme.

The random forest is an ensemble training method that maps the appearance of the patch around each pixel to the most likely object coordinate and class according to the training data. This discriminative prediction is then used as the input for the RANSAC optimization scheme. This optimization method samples point triplets from the prediction and recovers a full 6D pose hypothesis from the correspondence of the observation and the prediction. This results in many pose hypotheses for one image. The best matching hypothesis regarding the observation and the prediction is then chosen as the final pose. Figure 1.2 visualizes the baseline method. The problem with this method is, that the appearance of the same object parts will be different under varying lighting conditions. Figure 1.1 shows the Dragon object under different lighting conditions. If the training data consists of multiple different lighting conditions, the random forest will not distinguish between them and will have to compensate for a strong appearance variability of the patches which describe the same part of the object. My contribution will be the extension and evaluation of the random forest

Figure 1.1: Visualization of the appearance variability of one object under different lighting conditions.

to a conditional random forest which will incorporate lighting information into the object coordinate regression task. The conditional random forest requires another ground-truth labeling: the lighting condition present in the image. In contrast to

the random forest, the conditional random forest is able to differentiate between the lighting conditions to perform a more specialized prediction because the prediction will only be based on the training images of the corresponding lighting conditions. Thus, the distribution of the class and object coordinates are now conditional to the present lighting condition. The conditioning of the distributions requires new methods of inferring the correct pose and lighting condition. There are two basic approaches to use

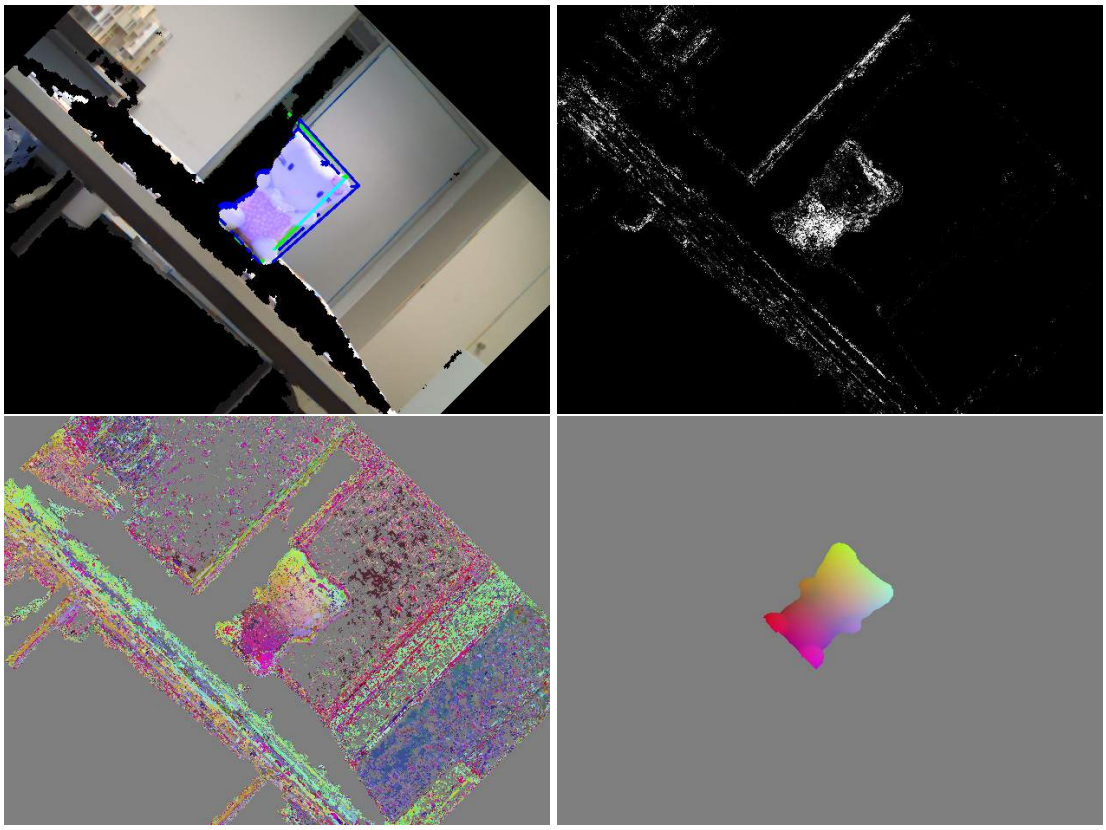

Figure 1.2: Example run of the baseline method for object pose estimation. **Top Left:** An example detection of the object pose (**Green:** ground truth pose, **Blue:** estimated pose). **Top Right:** Per-pixel prediction of the object probability. **Bottom Left:** Prediciton of the object coordinates. **Bottom Right:** Ground Truth of the object coordinates. The object coordinates are visualized using the color cube.

the conditional random forest in conjunction with the latent lighting condition: Either let the conditional random forest estimate the lighting condition itself or use a different algorithm to estimate it beforehand and feed the information into the conditional random forest. For the prior lighting estimation, I will investigate an approach based on an intrinsic image algorithm. Barron et al. [3] proposed an impressive intrinsic image system which jointly estimates the lighting and the albedo of a single RGB-D

image. The system estimates several lighting conditions for each image represented as spherical harmonics which are best described as a parametric, low dimensional representation of an environment lighting map. The output of this system is then used as input for the conditional random forest which can now make the prediction based on the estimated lighting. The standalone light estimation of the conditional random forest works more straightforward by evaluating it for each possible lighting condition to choose the best resulting energy as the solution.

Chapter 2 contains summaries of recent scientific work and puts them into the context of this thesis. Chapter 3 provides the necessary theoretical foundations by introducing the random forest framework, spherical harmonics, as well as the baseline method formally. Furthermore, the different methods for the construction and inference of the conditional random forest are presented. Chapter 4 will evaluate the system and draw conclusions about its performance in comparison to the baseline method. Chapter 5 will feature a discussion about practical relevance, problems, and improvement possibilities for the proposed approach.

2 Related Work

This chapter will cover the relevant papers using conditional random forests which inspired the approach for this thesis, as well as the intrinsic image algorithm used to estimate the lighting. I will touch on different possibilities to perform pose estimation other than the object coordinate regression to give an overview on the state of the art in this area. Moreover, I want to examine different methods to estimate the lighting condition with different lighting representations.

2.1 Object Pose Estimation Papers

[12] proposes a template based method to infer the 6D pose from a single image. For this method, many views of the object need to be recorded to create the templates for each view. These templates are a set of image features located in a bounding box around the object. Each of these templates has to be compared to the input image. This is done by "sliding" the template over the image to compute a similarity score of the template and the image. The best scoring template with annotated 6D pose is then used as the output of the algorithm.

The approaches used in [11, 2] are based on hough forests. These methods infer a displacement vector for every pixel using a random forest. Every pixel votes for the position of the object. These votes are aggregated in a histogram of possible positions called hough space. The maximum of this hough space then represents the final estimation. [2] extends this method to the 6D pose estimation task. A two step hough voting scheme is introduced which infers the 3D orientation after the correct 3D position was found. [16] is based on a sparse interest point approach. They detect all the local maxima resulting from the SIFT interest point detector and compare them to a database of previously recorded interest points from different

views of the object. The found interest points are then used as an initialization of an optimization procedure which eliminates outliers via RANSAC and verifies the hypothesis geometrically.

2.2 Conditional Random Forest Papers

[22] deals with the task of inferring the complete human pose from a single RGB-D image. Traditionally, this is done by inferring the positions of the body joints independently from one image. Random forests haven proven to be fast and reliable estimators for this body joint regression task. They are used to infer the position of the joints per pixel. All these votes are aggregated in a hough space of 3D positions where the maximum is sought. Problems arise when the appearance of patches corresponding to the same body joints vary too much. This was the incentive for the authors to rework the random forest into a conditional model. They show that they gain a performance boost when they condition the random forest on the person height as well as the torso orientation which are represented as a global latent variable. They show and evaluate the different methods to model the conditional distributions. These models are described in detail in Subsection 3.4.1.

Also, different methods to perform inference on the conditional random forest are presented. The straight forward approach of performing complete inference for each conditional distribution and then using the best scoring outcome (described in detail in Subsection 3.4.2) produces the best results. The problem with this approach is the high computational cost which also increases linearly with the number of possible discrete states of the global variable. To overcome this, they estimated the latent variable without performing the complete pipeline by aggregating per pixel estimates of the latent variable. The variable with the best resulting score is then chosen. The complete inference with this prior information then only has to be performed with regard to its conditional distribution. This yielded slightly worse results but a far better computation time which is why I explored an adaption of this method in Subsubsection 3.4.2.3.

An additional insight from this paper is, that the different body joints share the same global latent variable. This fact is used in the inference step by choosing the best global variable and corresponding joint locations as the maximum score of all body joints

combined. This could be exploited in the object pose estimation task when multiple objects are present in the scene.

[9] solves the task of estimating the position of so called facial features from a single image of a human face. These features are shown in Figure 2.1. They also use a random forest for the regression of the position of the facial features per pixel as well as a hough voting scheme to infer the final estimation of the position. Because the appearance of the different facial features varies greatly between different head orientations, they choose to learn the correspondence between image patches and the corresponding position of the feature points conditional to the coarse head orientation. An individual forest is learned for each subset of the training data which is labeled with the same coarse head orientation. For inference, an additional random forest to predict the

Figure 2.1: Visualization of facial features. The boxes around the faces show the coarse head orientation.

global latent variable of the head orientation is constructed. This realizes a per pixel estimation of the latent variable. With this knowledge, only one conditional random forest needs to be evaluated. Since this inference of the coarse head pose is only an estimation of the global variable, the authors propose a mixing of the random forests for inference. It is suggested, to construct a new random forest from the trees of the existing conditional forests. From every conditional forest, trees are added to the new random forest proportional to the probability of the latent variable which is provided by the pre-estimation step. This new random forest is evaluated traditionally as described in Section 3.1.

2.3 Intrinsic Image Algorithm

Intrinsic image algorithms are of interest for this thesis because they provide an estimation of the lighting condition present in a single image without the need of a prior segmentation or pose estimation of the object. This light estimation can then be used as a prior estimation step to potentially improve the performance of the conditional random forest. The task of obtaining the intrinsic images from an input image, is to divide it into its material-dependent properties and light-dependent properties. Intrinsic image algorithms aim to find the solution to

$$I = RS, \tag{2.1}$$

where I is the original Image, R is the albedo, and S is the shading image. The albedo depicts the "true" color of every object in the image without any light highlights or shading involved and the shading image depicts all light-dependent properties of the image. In order to prevent trivial solutions, constraints need to be incorporated. A huge effort in this research area is put into finding the appropriate constraints to optimize Equation 2.1 with respect to these priors. [4] describes the shading image as a combination of the shape of an object and the underlying illumination and imposes constraints on the shape of the object and the albedo of the object.
Priors on the albedo:

(1) Smoothness: It is assumed that the albedo tends to be piecewise smooth.
(2) Minimal entropy: The possible number of albedos in the image is expected to be low.

Priors on shape:

(1) Flatness: This term prefers flat surfaces with a low slant from the view of the camera.
(2) Smoothness: It is assumed, that the depth variation is of low frequency.
(3) Occluding contours: The normals inferred from the contour of the image must be consistent with the estimated shape.

All these constraints are then incorporated into one objective function which is then minimized. As a byproduct of this procedure, an estimation of the lighting condition

represented as spherical harmonics is provided. A major limitation for this approach is the necessity of a prior segmentation of the object. [3] revisits the shortcomings of this method and proposes a new approach which can be applied to unsegmented images. They do this by introducting mixtures of lighting conditions and shapes. That means that every pixel in the image now is provided with a probabilistic correspondence to a fixed number of lights and shapes. The mixture for each pixel is established with a method similar to normalized cuts [20]. This normalized cut approach usually segments the image in multiple regions which are related regarding a certain similarity measure. In this paper, however, a soft segmentation approach is proposed, which leads to an influece of each mixture to each pixel. That means that every light can have an influence on every pixel theoretically. The albedo, lighting conditions, shapes and their corresponding soft segmentations are then optimized in a procedure similar to [4]. It must be noted that the optimization of the lighting condition uses only a discrete set of lighting conditions which were recorded beforehand. The relevant output of this procedure is the inferred lighting condition. The lighting condition is represented per pixel as a linear combination of eight spherical harmonics. That means that every pixel is associated with eight weigths for the base spherical harmonics which can be used to calculate the lighting condition. This output will be the basis for the prior light estimation in Subsubsection 3.4.2.2.

2.4 Papers on Light Estimation

[19] pioneered the use of spherical harmonics as light representation and showed that they are well suited for this task if some preconditions are met. The first assumption is the presence of distant light sources, which means that lighting is invariant to the position of the object in space. Furthermore, the object is assumed to be lambertian, which means that the color of the surface is the same regardless of the observer's angle of view. Since the diffuse reflection of the lambertian surface acts as a low-pass filter on the lighting, the smooth approximation by spherical harmonics with nine coefficients per color channel proves to be well suited. Lastly, the shadows present on the object are ignored by their approach. They derive a formula which directly relates the observed irradiance to the incoming lighting distribution. They estimate the lighting condition from a lambertian sphere with known position and radius. The idea of this light estimation algorithm is not directly applicable to the task of pose estimation because

the pose of the object has to be provided beforehand. The derivations in this paper lay the foundation for the rendering algorithm proposed in [10], which was also used by [3].

A lot of work in the field of light estimation has been done in the area of face detection and re-lighting. The work of Basri et al. [5] uses the foundation which was laid by the previously presented light estimation approach. They also use spherical harmonic lighting to recognize faces under arbitrary illumination. Assuming that the pose of the head is already known, they render "basis faces" from the albedo of each face and each spherical harmonic basis function. Provided with these basis faces for each face in the dataset, they can infer the best spherical harmonic coefficients which describe the most similar rendering to the observed image. Because the final rendering is a linear combination of the basis images, the optimal coefficients can be found by QR decomposition. The face in the database with the lowest difference to the input image is then chosen as the final recognition.

[15] also aim to estimate the lighting condition from a single image of a human face to render artificial objects with consistent lighting into the scene. They overcome the limitation of a known albedo by learning radiance transfer functions for each part of the face. These radiance transfer functions supersede the neccesity of face albedos. Each of these light estimation methods based on the human face take knowledge of the pose for granted which makes them unusable for the pose estimation task. Yet, a similar approach could be used to improve the RANSAC step of the object coordinate regression method. In each iteration step, the object could be rendered under the estimated illumination in this pose. Then, the difference of the rendered and the observed image could be incorporated into the energy function.

3 Method

This chapter will cover the details of the proposed method. At first, I will explain the general concept of the random forest estimator. After that, I present the baseline method [6] which applies the random forest framework to the pose estimation task. To establish the theoretical basis for the description of the lighting condition, I will introduce the formal definition of the spherical harmonics. Then, I will go into detail about the construction methods for the conditional random forest. The last subsection of this chapter will focus on the different possibilities of performing inference on the conditional random forest.

3.1 Random Forests

Random forests are a popular machine learning paradigm wich can be used to solve regression and classification tasks [8, 7]. This section will only cover the classification task since the random forest used throughout the thesis is constructed using only the training procedure of the classification task. Classification means, that every sample s is associated with a discrete class $c \in C$. Each sample represents a datapoint in an n-dimensional space. During the training of the forest, the corresponding class of all samples is provided. At test time, it is the task of the forest to infer the class of new unlabeled samples based on the information obtained from the training set. The random forest contains multiple trees $T \in \mathcal{T}$, which are autonomously able to solve the given task. Each tree consists of split and leaf nodes. Every split node represents a weak learner which must be designed according to the specific classification task. For a given datapoint, the tree applies a simple binary feature test at each node to decide whether it should be passed to the right or left child of the node. This test has the form

$$h(s, \phi, \tau) = [\, f_\phi(s) \geq \tau \,], \tag{3.1}$$

where $f_\phi(s)$ is the feature response function which has to be defined based on the specific task the forest has to solve, ϕ is a set of parameters chosen according to this task, and τ is a threshold. Both of these parameters are learned during the training procedure. $[.]$ denotes the 0-1 indicator function which represents the decision for the left or right path. This process is repeated until the point reaches a leaf l^T where the distribution $p(c|l^T)$ to perform classification is stored. The basic idea of the forest is to average the outputs of these trees. This process is called bootstrap aggregation and helps to avoid overfitting regarding the data. The maximum of this averaged output is then chosen as the final prediction of the random forest classifier.

3.1.1 Training

The training of the forest is done separately for each tree. Prerequisite for the training procedure is a set of training samples S_0, which are annotated with their corresponding class. During training, the set of samples will be split subsequently. The samples which arrived at node j are denoted by S_j. The training starts at the root node. It is the task to find the best parameters $\{\phi, \tau\}$ for the binary split. The best parameters of the split should separate the dataset as well as possible. Ideally, one or multiple classes are completely separated by one split. The measure for the quality of a split at node j is the information gain

$$I_j = H(\mathcal{S}_j) - \sum_{k \in \{1,2\}} \frac{|\mathcal{S}_j^k|}{|S_j|} H(\mathcal{S}_j^k), \tag{3.2}$$

where $H(\mathcal{S})$ denotes the entropy of a set of samples and is defined as $H(\mathcal{S}) = -\sum_{c \in C} p(c) \log(p(c))$. $p(c)$ in this context denotes the normalized histogram of classes in the set $\mathcal{S}$. $\mathcal{S}_j^k$ denotes the split of the samples which arrived at the node j into two disjunct subsets according to Equation 3.1. The higher the information gain of node j, the peakier the resulting histograms in the sets $\mathcal{S}_j^k$ and, thus, the better the split will be. In order to find the best split parameters, an optimization over I_j has to be performed. For practical purposes, a discrete set of parameters $\{\phi, \tau\}$ is sampled randomly. From this set, the parameters which maximize I_j are chosen. This procedure is then applied recursively to the two sets which result from the chosen split. The training procedure terminates once a predefined depth or minimal number of samples is reached. The distribution $p(c|l^T)$ is collected in the leaves. Figure 3.1 visualizes a simple example of the sampling of split parameters at the root node.

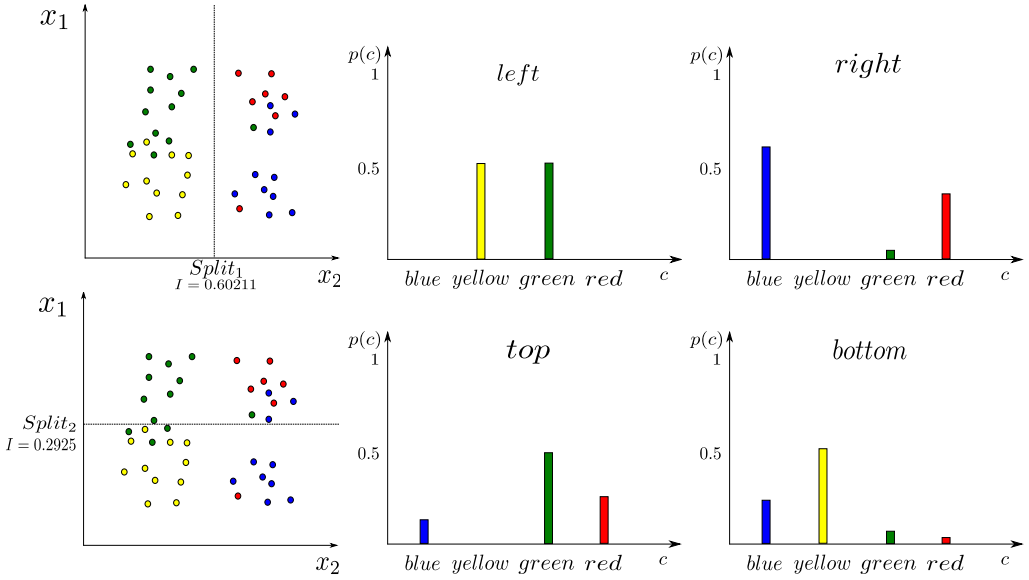

Figure 3.1: Example of a simple decision forest.
Visualization of the split selection: Depicted are two of many possible splits for the root node of a decision forest. The data points are labeled points $s = (x_1, x_2)$ with four possible classes, which are visualized with different colors. The feature response function is axis aligned and defined as $f_\phi(s) = x_1$ or $f_\phi(s) = x_2$ depending on the sampled parameter $p \in \{vert, horizontal\}, \phi = \{p\}$. So, the randomly sampled parameters for this example are just τ and p. The top row shows a possible vertical split and its corresponding histograms for the left and right sets. The quality of the split is reflected in the high resulting information gain and the peaky histograms. The bottom split reaches a lower information gain. The algorithm chooses $Split_1$ and will continue recursively on the left and right subset.

3.1.2 Testing

To obtain a prediction from the decision forest, it is necessary to combine the outputs of the trees. This may be done by simple averaging:

$$p(c|s) = \frac{1}{|\mathcal{T}|} \sum_{T \in \mathcal{T}} p(c|l_s^T), \tag{3.3}$$

where l_s^T denotes the corresponding leaf from tree T to the sample s. Alternatively, a multiplication may be used. However, it must be noted that the tree outputs are not statistically independent.

$$p(c|s) = \frac{1}{Z} \prod_{T \in \mathcal{T}} p(c|l_s^T) \tag{3.4}$$

$$Z = \sum_{c \in C} [\prod_{T \in \mathcal{T}} p(c|l_s^T)], \tag{3.5}$$

where Z is the partition function. The most probable class is then used as the prediction of the forest.

3.2 Random Forests for Object Coordinate Regression

I will now explain in detail how the forests are applied to the concrete task of object coordinate regression and object classification used by [6].

3.2.1 Training

Prior to the training, a dataset containing RGB-D images of the target objects has to be recorded. Every single image has to contain the information of the pose of the object relative to the camera. The guide on how to create such a dataset is given in Section 4.1. The training begins with a sampling of all training images. Pixel samples are denoted by $s = (\vec{p}_s, I_s) \in \mathcal{S}_0$, where $\vec{p}_s$ is the pixels 2D position, and I_s is the RGB-D image from which the sample was drawn. Note that a pixel sample is completely defined by the appearance of the patch. Thus, a sample is a datapoint in the space of possible appearances of patches. The given notation is a more convenient method to describe a sample. Each sample is annotated with the corresponding ground truth object coordinate y and class c. Samples are drawn from each image according to the segmentation mask of the objects which is calculated from the object pose. The feature response functions are inspired by [21] and operate on each appearance channel (R,G,B or D)

$$f_\phi^d(s) = d(\vec{p}_s + \frac{\vec{\omega}_1}{d(\vec{p}_s)}) - d(\vec{p}_s + \frac{\vec{\omega}_2}{d(\vec{p}_s)}) \tag{3.6}$$

$$f_\phi^{rgb}(s) = I(\vec{p}_s + \frac{\vec{\omega}_1}{d(\vec{p}_s)}, \gamma_1) - I(\vec{p}_s + \frac{\vec{\omega}_2}{d(\vec{p}_s)}, \gamma_2), \tag{3.7}$$

where $I(\vec{p}, \gamma)$ returns one of the appearance channels according to γ, and $d(\vec{p})$ is used to represent the depth at pixel position $\vec{p}$. All these values are evaluated on the image I_s corresponding to the sample. $\vec{\omega}$ denotes a small 2D offset inside a patch region. These

feature tests can be seen as a comparison of the similarity of the pixels at the chosen offsets in the chosen appearance channel. The division by $d(\vec{p}_s)$ serves the purpose of making the features depth invariant. This approach was proposed by [23]. Each node in the forest stores the parameters $\{\phi, \tau\}$, where $\phi = \{\vec{\omega}_1, \vec{\omega}_2, \gamma_1, \gamma_2, z\}$ and $z \in \{d, rgb\}$ decides whether to use the depth or rgb channel.

The training is now performed as explained in Section 3.1. It is important to note that this forest is built to perform a regression on the object coordinates y as well as a classification to determine which object class c is present. This is done by reducing both tasks to one single classification during training. To realize this, the object coordinates are discretized into 125 bins resulting from a 5x5x5 discretization of the object coordinates using the bounding box of the object. The resulting discrete object coordinates are denoted by $\hat{y}$. It is then possible to create a labeling which is unique for each discrete object coordinate of each object. There are $125|\mathcal{C}| + 1$ possible labels $\hat{c} \in \hat{\mathcal{C}}$ for each pixel. Each label describes one discrete object coordinate as well as the object class. The additional label describes the samples which belong to the background. The Information gain I_j at each node j to obtain the best split is calculated as described in Equation 3.2 using these class labels. Ultimately, the tree is learning the distribution $p(\hat{c}|l^T)$. Because the information of both the object coordinate and the class label is captured in the class labels $\hat{c}$, these two variables are learned jointly by the tree.

After the training of the tree structure, the training samples are pushed through the forest once more to gather all the continuous object coordinates y in the leaves. Per leaf, a mean shift operation is applied on the coordinates of each object to store only the top mode as the final prediction $y_c(l^T)$. This is a notable step because the tree structure which was learned by classification is used for a regression task on the object coordinates. The object class probability $p(c|l^T)$ is stored in the leaf as well to perform the classification.

3.2.2 Usage of the Random Forest

During recall, all pixels in the test image are pushed through each tree of the forest to establish the mapping from the sample s to the corresponding distribution $p(c|l_s^T)$ and the object coordinate prediction $y_c(l_s^T)$. Alternatively to the averaging of the

outputs described in Subsection 3.1.2, this intermediate result is then used as input for a RANSAC-like scheme which aims to optimize an energy function. This energy function is defined as:

$$E(H) = \lambda_1 E^{depth}(H) + \lambda_2 E^{coord}(H) + \lambda_3 E^{obj}(H), \tag{3.8}$$

where H is the 6D pose hypothesis and the λ are weighting parameters. $E^{depth}(H)$ is the part of the energy function which penalizes the deviation of the rendered pose from the observed depth values and does not depend on the forest output. $E^{coord}(H)$ does depend on the forest output $y_c(I_s^T)$ and penalizes the deviation of the forest prediction to the rendered object coordinates. $E^{obj}(H)$ describes the forests certainty about the location of the object according to the object probabilities $p(c|I_s^T)$.

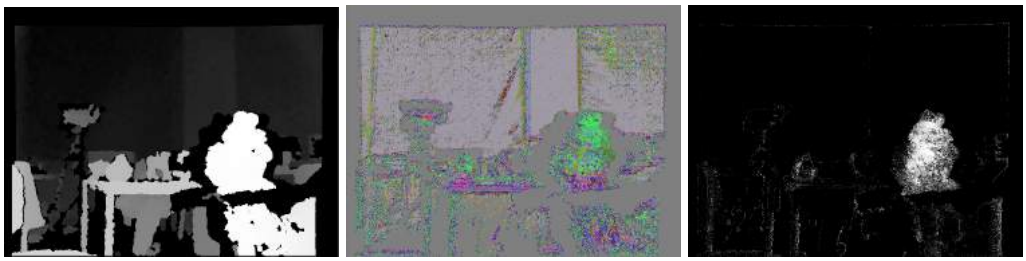

Figure 3.2: A visualization of the intermediate output of the random forest. **Left:** The depthmap of the scene. $E^{depth}(H)$ uses the depthmap to verify the pose hypothesis. **Middle:** The prediction image of the object coordinates. The object coordinates are mapped to the color cube. $E^{coord}(H)$ penalizes the deviation of the inferred object coordinates to rendered object coordinates of the pose H. **Right:** The prediction image of the object class. Bright pixels show a high probability that the pixel belongs to the object. $E^{obj}(H)$ uses this prediction to verify the pose hypothesis.

This function is then optimized using a RANSAC approach. As usual for this paradigm, many object coordinates are sampled and then verified with the observation. This is done by sampling a single pixel first. A second and third pixel are chosen randomly within a window which is determined by the diameter of the object and the observed depth value to reject pixels which can not belong to the object. The object coordinates of the forest and the world coordinates calculated from the depth values at these pixels now each form correspondences $(\vec{y}^1_{obj}, \vec{x}^1_{world})$, $(\vec{y}^2_{obj}, \vec{x}^2_{world})$, and $(\vec{y}^3_{obj}, \vec{x}^3_{world})$. Note that each of the forests predicts one object coordinate per pixel. For this process, one of the forests is selected at random to obtain the object coordinate per pixel. For a correspondence of at least three point pairs, the Kabsch algorithm can be used to calculate the best transformation to optimally align these points. This transformation then serves as

the pose hypothesis which is evaluated using Equation 3.8. This procedure is repeated many times and the best scoring pose hypotheses are stored. As a last step, these best hypotheses are refined. This refinement is done by incorporating every pixel of the calculated object mask into the transformation calculation step. The transformed object coordinates from each tree are then compared to the observed world coordinates. For each transformed object coordinate prediction, the transformation error to the corresponding observed world coordinate is computed. The object coordinate predictions which achieved a score below a certain threshold are used as the input for the Kabsch algorithm. This procedure is iterated until the score no longer decreases. The best scoring pose is chosen as the final output.

3.3 Spherical Harmonics

Spherical Harmonics are a way of representing a lighting condition. The premise of this representation is the assumption that all light sources are located infinitely far away from the scene. In this way, the light can be represented as a function $L(\vec{n})$ over the sphere

$$\vec{n} = \begin{pmatrix} \sin(u)\cos(v) \\ \sin(u)\sin(v) \\ \cos(u) \end{pmatrix}. \tag{3.9}$$

The normal $\vec{n}$ is parameterized by $0 \leq u \leq \pi$ and $0 \leq v \leq 2\pi$. Spherical harmonics can be used to approximate this function using only a small number of coefficients of frequency-space basis functions. Spherical harmonics can be described as the analogy of the fourier series for functions defined over the sphere. In the domain of the fourier series, a periodic function can be expressed in the form of a linear combination of sine and cosine basis functions. In analogy to this, the reconstruction of the function using spherical harmonics is defined as

$$L(\vec{n}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} c_l^m y_l^m(\vec{n}). \tag{3.10}$$

c_l^m are the coefficients of the basis functions $y_l^m(\vec{n})$. Spherical harmonics are defined by orders l. For each order, $(2l+1)$ basis functions are defined. The degree m is used to index the basis functions for each order. Provided with an infinite number of orders, the original function can be reconstructed perfectly. Using only a discrete number of orders, it is only possible to approximate the function. This work will use 3-order spherical harmonics for the representation of the lighting. [19] shows that 3-order spherical harmonics suffice to model the lighting condition for lambertian objects. Because every color channel is represented using spherical harmonics, the final representation of the lighting condition will have 27 degrees of freedom. For reasons of convenience, an index $k = l(l+2)+m$ is introduced to reduce the formula to

$$L(\vec{n}) = \sum_{k}^{9} c_k y_k(\vec{n}). \tag{3.11}$$

The basis functions are defined as

$$y_l^m(u,v) = \begin{cases} \sqrt{2}K_l^m cos(mv) P_l^m(cos(u)) & \text{if } m > 0, \\ K_l^0 P_l^0(cos(u)) & \text{if } m = 0, \\ \sqrt{2}K_l^m sin(-mv) P_l^{-m}(cos(u)) & \text{if } m < 0, \end{cases} \tag{3.12}$$

where K_m^l are normalization constants denoted by

$$K_l^m = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}, \tag{3.13}$$

and P_l^m are Legendre Polynomials:

$$P_0^0(z) = 1, \tag{3.14}$$

$$P_m^m(z) = (2m-1)!!(1-z^2)^{\frac{m}{2}}, \tag{3.15}$$

$$P_{m+1}^m(z) = z(2m+1)P_m^m(z), \tag{3.16}$$

$$P_l^m(z) = \frac{z(2l-1)}{l-m}P_{l-1}^m(z) - \frac{(l+m-1)}{l-m}P_{l-2}^m(z). \tag{3.17}$$

Note that $y_l^m(u,v) = y_l^m(\vec{n})$ given Equation 3.9. Figure 3.3 shows a visualization of the first three orders of the basis functions.

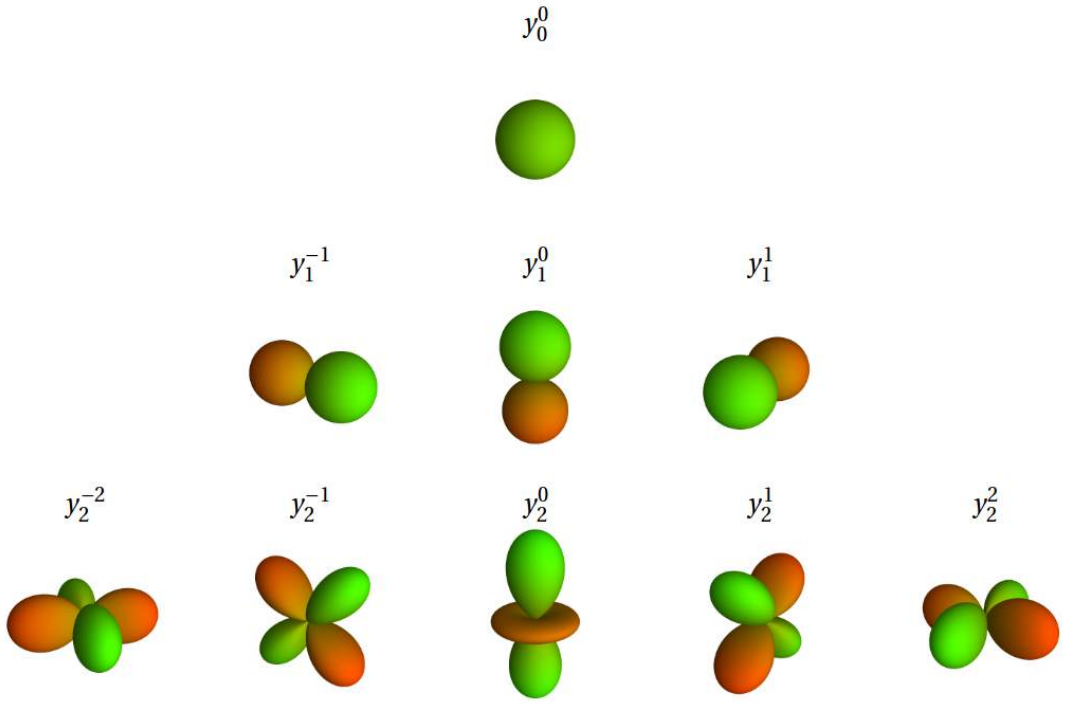

Figure 3.3: A visualization of the first three orders of spherical harmonic basis functions. Green indicates negative values and red indicates positive values. The value of $y_l^m(\vec{n})$ is proportional to the distance of the intersection point of the depicted figure and a line from the origin in direction of $\vec{n}$. ©by Wojciech Jarosz [14].

3.4 Conditional Random Forests

Conditional random forests are an extension of the random forest framework which was introduced by [9] and [22]. The main benefit of conditional random forests is that they are able to incorporate prior knowledge into the prediction process because they learn probability distributions conditioned on a discrete latent variable $a \in \mathcal{A}$. In the domain of this work, that means that the conditional random forest learns distributions of labels $\hat{c}$ conditioned on the current lighting condition $p(\hat{c}|l^T, a)$. This procedure only incorporates the training images of one distinct lighting scenario into the learning of one conditional probability distribution. A benefit of this method is that the learning

procedure of the conditional random forest has to deal with a much smaller deviation in appearance for each lighting condition. This procedure requires an additional ground truth labeling of the latent variable for the dataset. Each sample needs to be associated with one lighting condition a. The disjunct subsets of the samples $\mathcal{S}$ which are labeled with a are denoted by $\mathcal{S}_a$. The conditional random forest can be constructed using the Full Model or the Partial Model.

3.4.1 Training

3.4.1.1 Full Model

For each value of a, a full forest structure is trained. The training of each forest is done as before on a subset $\mathcal{S}_a$. This results in $|\mathcal{A}|$ unique forests $\mathcal{T}_a$. In order to compute the prediction images, each forest must be evaluated per pixel. The pipeline can then be performed for each of the forests as described before. This means that the RANSAC procedure has to be performed for each of the trained conditional forests.

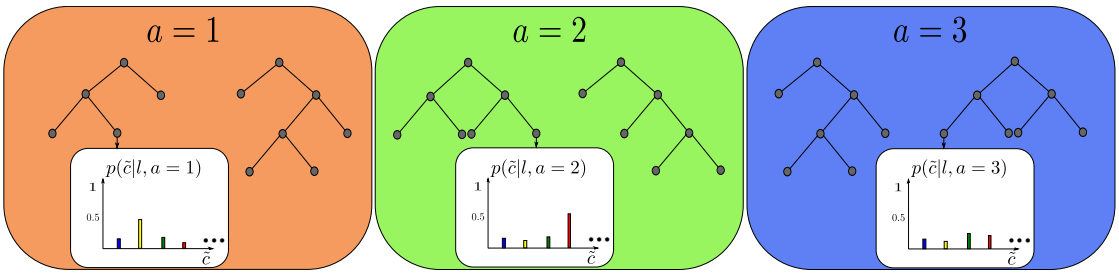

Figure 3.4: Visualization of a conditional random forest with three different latent variables. For one leaf of each forest, an example distribution is visualized. The object class distributions and object coordinates are reconstructed from $p(\hat{c}|l^i, a)$.

3.4.1.2 Partial Model

Only one forest structure is trained and the conditional distributions are split in the leaves of the forest. I adapted the training procedure by introducing a new unique labeling of each discrete object coordinate for each object and lighting condition. The total number of new labels $\tilde{c} \in \tilde{\mathcal{C}}$ will then add up to $125|\mathcal{C}||\mathcal{A}| + 1$. The training

procedure is performed as described in Subsection 3.2.1, even though the light condition is considered now because the introduced class labels $\tilde{c}$ capture the lighting condition in addition to the object class and object coordinate. As a final step of the learning procedure, the mode of the object coordinates $y_c(l^T, a)$ and the object class probability $p(c|l^T, a)$ is stored in the leaves. Note that every leaf now contains multiple conditional probability distributions.

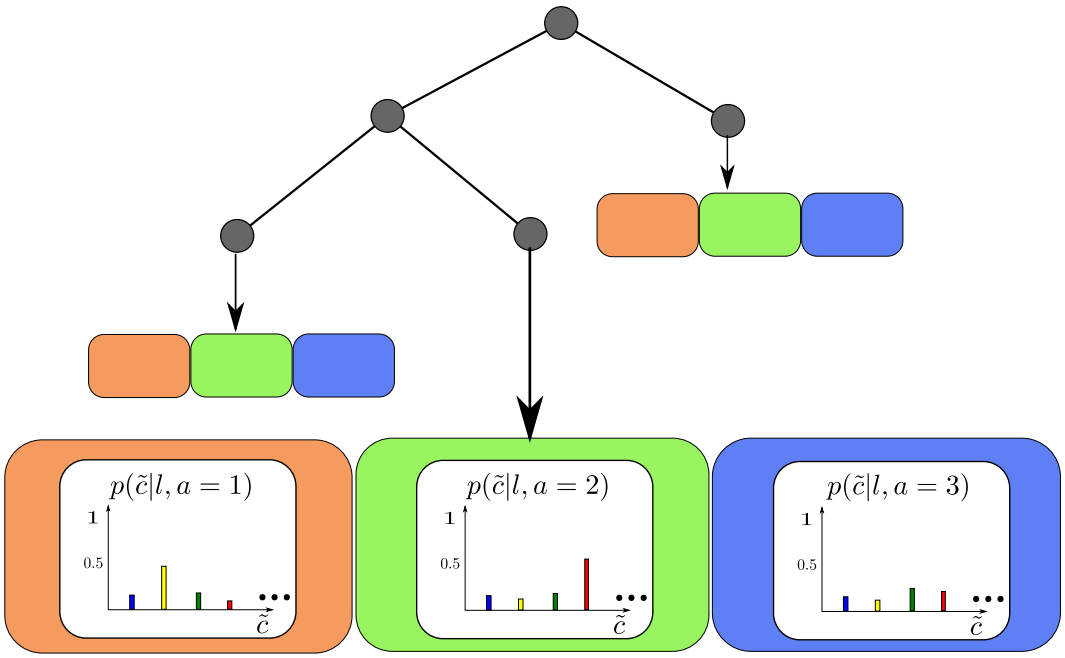

Figure 3.5: Visualization of a single tree of a conditional random forest with a single forest structure. For one leaf of the forest, an example distribution is visualized.

3.4.2 Inference on Conditional Random Forests

3.4.2.1 Joint Inference of a and H

This way of inferring the final estimate was proposed by [22]. The complete estimation pipeline is performed for each variable a. This results in multiple estimates of the pose H_a with corresponding scores $E(H_a)$ from the RANSAC scheme. The best estimation for the pose is then

$$H^* = \arg\min_{H_a} E(H_a). \tag{3.18}$$

The pose with the highest score does not always correspond to the correct lighting condition. That leaves some room for improvement of the lighting estimation. This fact was the original justification for the following prior light estimation.

3.4.2.2 Prior Estimation of a Using Spherical Harmonics

If a previous estimation of a is provided, the correct distribution for inference can be chosen beforehand. This saves a lot of computation time because the computational cost to run the complete pipeline for each forest is very high. Additionally, the estimation of the light condition provided by the joint inference method is not guaranteed to be perfect. So, by improving the lighting estimation step, the overall performance can be improved. To perform an estimation of the lighting condition, the intrinsic image algorithm by Barron et al. [3] is applied to the test image. The output of this procedure is the estimated log-albedo of the image as well as the log-shading image. They use the logarithm representation for both the shading image and the albedo to simplify the optimization task. As stated in Section 2.3, the shading image is rendered from the depthmap and the inferred lighting condition. The lighting condition is represented per pixel as log-spherical harmonics

$$\vec{c}_{log} = \sum_{i}^{8} w_i \vec{c}_{log,i}, \tag{3.19}$$

where w_i is the per-pixel-weight, $\vec{c_{log,i}}$ are the eight base log-spherical harmonic coefficients and $\vec{c}_{log}$ are the per-pixel output log-spherical harmonic coefficients. For each test image, I use a coarse grid sampling on the pixels and calculate the corresponding lighting conditions from the given weights and base spherical harmonic coefficients. This results in a discrete set of spherical harmonic coefficients for each test image. $\vec{c}_{log}$ can be used to reconstruct the log-lighting function. The next step is the comparison of the lighting conditions of the test image to the pre-recorded ground truth lighting conditions. To compare both lighting conditions, the quadratic differences of the reconstructed functions need to be integrated.

$$e_{light} = \int_{\vec{n}\in\Omega} (L(\vec{n}) - e^{L_{log}(\vec{n})})^2 d\vec{n} \tag{3.20}$$

Ω denotes the whole sphere, $L(\vec{n})$ is the distant lighting function reconstructed from the ground-truth spherical harmonics and $L_{log}(\vec{n})$ is the log-lighting function reconstructed

from the log-spherical harmonics obtained by the intrinsic image algorithm. To compute this integral, I precompute a regular sampling of the sphere using [17]. Then I replace the integral by a sum over these samples. The final lighting prediction is now obtained by choosing the ground truth which achieved the smallest score e_{light} with respect to any of the log-spherical harmonics in the test image.

3.4.2.3 Inference maxA

This method of inference is inspired by [22]. It is only applicable to the conditional random forest with one forest structure because the distribution $p(a|s)$ is sought for each pixel of the image. This distribution describes the certainty about the current lighting condition per pixel. This ignores the affiliation to a certain object class. In order to calculate this distribution per pixel, all samples belonging to a distinct lighting condition are aggregated in the leaf corresponding to this pixel and then normalized by the toal number of pixels which arrived in the leaf. This results in the probability $p(a|l_s^T)$ of each single tree. They are combined according to Equation 3.3 to obtain $p(a|s)$ per pixel. For each lighting condition a, all these values are aggregated into a single score. The condition a^* with the maximum score is then chosen as the best prediction

$$a^* = \arg\max_a \sum_{s \in \mathcal{S}_{img}} \frac{\sum_{T \in \mathcal{T}} p(a|l_s^T)}{|\mathcal{T}|}. \tag{3.21}$$

$\mathcal{S}_{img}$ denotes the samples corresponding to every pixel of the test image. This process is more efficient than the full joint estimation pipeline because the regression and RANSAC step need to be performed only once as opposed to $|\mathcal{A}|$ times. [22] achieved slightly worse estimation rates using this approach but accomplished a significant runtime boost.

4 Experiments and Results

4.1 Dataset

The dataset is the foundation for the whole method because it serves as the input for the training of the complete conditional random forest as well as for the testing of the algorithm. The dataset needs to contain a multitude of images containing objects under different lighting conditions. As a broad overview, these images must be annotated with the ground truth pose and the ground truth lighting condition. For this annotation, a 3D mesh of the object is required. The ground truth pose is then used to create segmented images of the object. Additionally, the ground truth object coordinates are rendered using a 3D model of the object. Figure 4.1 visualizes one segmented training image of the dataset. Lastly, a spherical harmonic ground truth must be calculated for each lighting condition in order to perform the intrinsic image inference method as described in Subsubsection 3.4.2.2. The dataset contains three different objects under

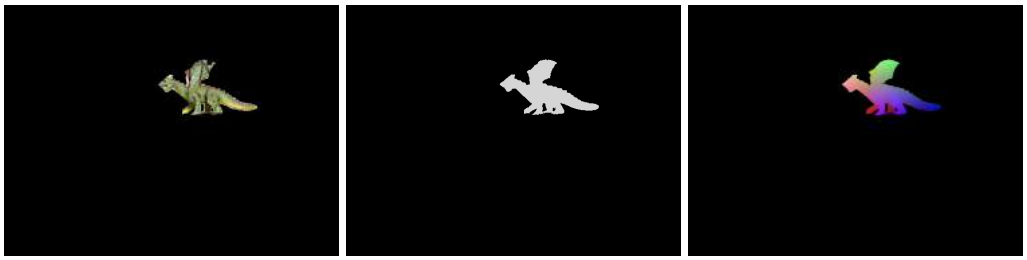

Figure 4.1: Left: The segmented color image. **Middle:** The segmented depth image. **Right:** The object coordinate ground truth.

eight different lighting conditions. The positioning of the light sources is shown in Figure 4.4. The light sources have different power levels, so the lighting intensities vary between them. The light sources are depicted in Figure 4.3. During all sequences, the camera is placed at the same position in the room and the object is moved around in front of the camera. This results in a static camera-light configuration. This static

Figure 4.2: The three different objects. **Left:** Watering Pot **Middle:** Frog **Right:** Dragon

Figure 4.3: The different floodlights used in the acquisition process.

configuration guarantees that the light orientation is not changing between the frames. This is an important design decision for the whole method. The alternative for the static camera-light configuration would be a static object-light configuration. The light orientation would be conditioned based on the object coordinate system. Both versions of conditioning are viable for the joint inference method. The pre-estimation inference method based on the intrinsic image algorithm requires the conditioning of the light orientation on the camera coordinate system because this prior estimation method estimates the lighting condition relative to the camera. To guarantee a meaningful mapping between the recorded ground truth lighting conditions and the estimated lighting conditions from the intrinsic image algorithm, the conditioning on the camera coordinate system is necessary. For the capturing of the training sequences, each object must be rotated in front of the camera until the complete upper hemisphere of the object is covered. The object is only covered without inplane rotation. The rotation around the viewing axis of the camera is not considered. [6] modeled the inplane

rotation by rotating the images synthetically around the origin of the image. In this light sensitive dataset this method is not applicable because synthetically rotated objects contain the wrong light orientation. For the test sequences I moved around the room freely with the object in hand. After the acquisition of the raw images, the 3D meshes of the objects need to be created. These 3D meshes are obtained by using the KinectFusion system [18, 13]. The next step is the annotation of the poses for every image of the

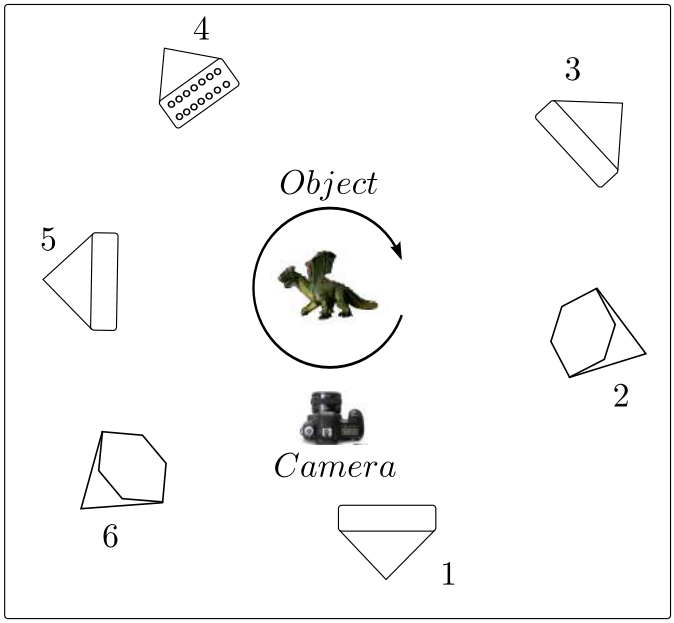

Figure 4.4: A schematic of the positioning of the lights relative to the camera.

dataset. This is done using a pose annotation tool based on the Iterative Closest Points algorithm (ICP). The input for this tool is one complete sequence of one object under one lighting condition and the 3D mesh of this object. The tool shows the depth map of each frame as a 3D pointcloud in camera coordinates. The object can be moved around this pointcloud to align to the real position of the object in this frame. This manual alignment does not need to be perfect because an ICP is then performed to obtain the optimal pose. After the initialization of the first pose, the other poses of the sequence are annotated automatically by performing iterative ICP. This works, because the spatial offsets between the frames are very small.

The next step is the creation of the segmentation images and the rendering of the object coordinate image. This is done by rendering the model with the annotated pose to an image with the same size as the training images. This rendering results in the object coordinate image and a segmentation mask which can be used to segment the training

images. As the last step of the acquisition procedure, the lighting condition has to be captured.

4.1.1 Capturing of the Lighting Condition

The lighting condition is described in terms of spherical harmonics. The spherical harmonic representation is a low dimensional approximation of the distant lighting function. The distant lighting function $L(\vec{n})$ is described by nine coefficients c_l^m:

$$L(\vec{n}) \approx \sum_{l=0}^{3} \sum_{m=-l}^{l} c_l^m y_l^m(\vec{n}). \tag{4.1}$$

These coefficients can be estimated using a linear least-squares approach if enough samples of the distant lighting functions are provided. The next part will deal with the acquisition of this environment map (the set of all samples). The least-squares estimation will be described at the end of this subsection.
The samples are acquired from one single image of a light probe. This light probe is a shiny sphere with near perfect reflection of the incident light. The color values on every pixel of the sphere are the values of the distant light function in direction of the reflected ray. [10] invented this method to capture the environment map. They recorded the sphere from 2 views 90° apart from each other to eliminate the image of the photographer in the sphere and to compensate for the poor sampling of the environment map on the border of the sphere. Since the goal of this capturing is the reconstruction of a low-dimensional approximation of the lighting function, I used only one image of the sphere. To gather a broad range of lighting intensities, the light probe must be recorded in high dynamic range (HDR). Regular color images only have 8 bits per color channel to capture only a small range of lighting intensities based on the exposure time of the camera. HDR images use a floating point representation for a higher range of possible intensities per color channel. They are computed using multiple images of the same scene captured with different exposure times. I used the open source software "Luminance HDR" [1] to acquire the HDR images of the sphere. A linear tone mapping is applied to obtain a discretized version of the HDR image. This discretization represents the final values which are used to compute the environment map. To calculate the environment map from one image, I use a raytracing approach to map the normals of the sphere to the colors seen in the image (Figure 4.6). For this to work, the position and size of the sphere as well as the intrinsic camera parameters

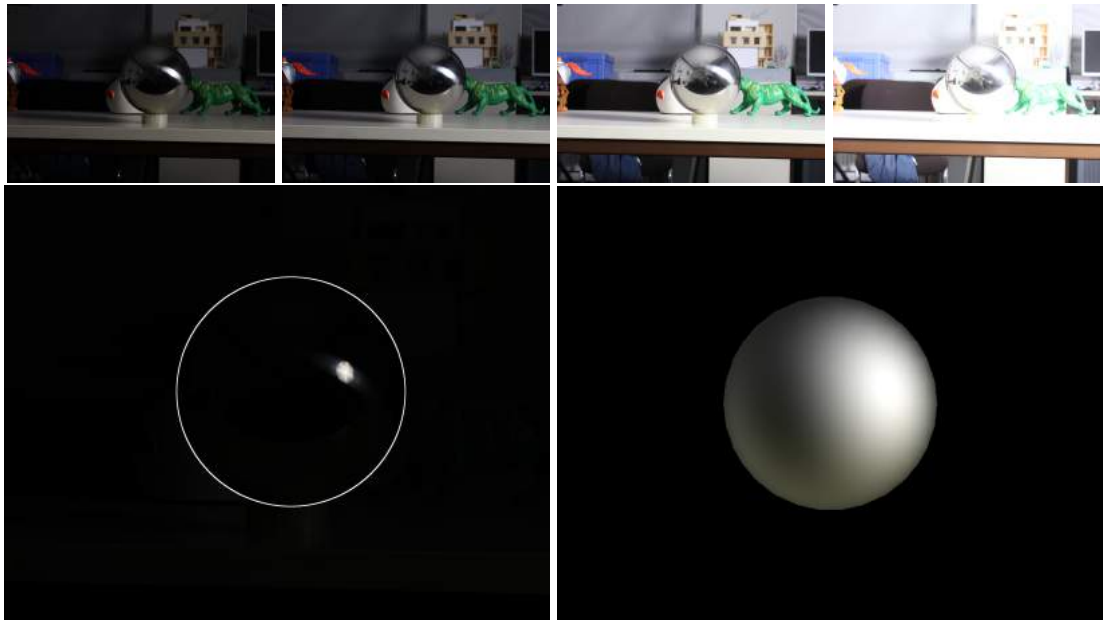

Figure 4.5: HDR recording of the sphere. **Top:** 4 of the 6 recorded images with different exposure times are depicted. **Bottom Left:** The final tone mapped image on a linear scale (white circle only for visualization). **Bottom Right:** The estimated spherical harmonic. Note that the small spotlight results in a big smoothed bright region on the right. This is the result of the low dimensionality of the spherical harmonics. The rendering of the spherical harmonic is done with the inverse "ray tracing" approach from Equation 4.3 to make the rendering and the original image visually comparable.

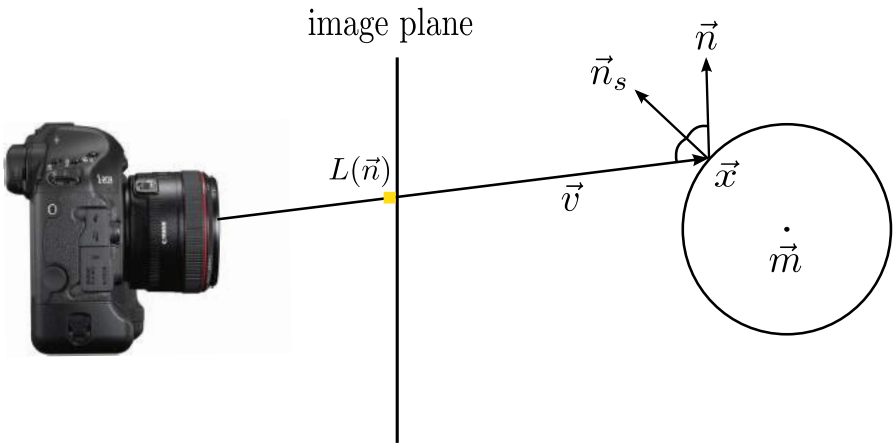

Figure 4.6: Figure of the capturing process: Every ray $\vec{v}$ through a pixel in the image is traced. If it intersects the sphere, the normal $\vec{n}$ into the distant lighting function is calculated. The value of the lighting funtion $L(\vec{n})$ is observed at every pixel of the sphere.

need to be provided. I wrote a tool to segment the sphere manually in an image in order to acquire the 2D position $\vec{m}_{px}$ and the radius of the sphere in pixels r_{px}. Given the focal length f of the camera (assuming it is the same for both dimensions) and the real radius r of the sphere in mm, the distance to the sphere can be computed as $d = \frac{rf}{r_{px}}$, assuming that the sphere is near the middle of the image. The final position of the sphere $\vec{m}$ is then computed as

$$\vec{m} = \begin{pmatrix} \frac{(m_{px,x} - p_x)d}{f} \\ \frac{((h - m_{px,y}) - p_y)d}{f} \\ -d \end{pmatrix}. \tag{4.2}$$

p is the pricipal point of the camera and h is the size of the image in y-direction. Given the 3D position of the sphere, the environment map as well as its SH representation can be computed. To compute the environment map, the observed color on the sphere must be assigned to the corresponding normal of the environment map at every pixel of the sphere. For each pixel, the viewing vector from the camera $\vec{v}$ is calculated as in Equation 4.2 with an arbitrary depth. To calculate the normal of the sphere per pixel, the line $g(\lambda) = \lambda\vec{v}$ in direction of the viewing vector is intersected with the sphere. In order to find the intersection point of the line and the sphere, $g(\lambda)$ is inserted into the implicit representation of the sphere. This equation is rearranged in order to be solvable by the standard method for quadratic equations

$$\lambda_s^2 + \lambda_s\left(\frac{-2 < \vec{v}, \vec{m} >}{< \vec{v}, \vec{v} >}\right) + \left(\frac{< \vec{m}, \vec{m} > -r}{< \vec{v}, \vec{v} >}\right) = 0. \tag{4.3}$$

The intersection point $\vec{x}$ can be found by inserting the smallest λ_s into the straight line

$$\vec{x} = g(\lambda_s). \tag{4.4}$$

The normal of the sphere $\vec{n}_s$ can then be obtained by

$$\vec{n}_s = \vec{x} - \vec{m}. \tag{4.5}$$

Finally, the vector which describes the origin of the light can be calculated by reflecting the viewing vector on the normal of the sphere.

$$\vec{n} = \vec{v} - 2(< \vec{v}, \vec{n}_s >) \tag{4.6}$$

Provided with M samples from the lighting functions with the corresponding normal $(b_m, \vec{n}_m)$, it is possible to solve for the optimal coefficients in the least squares sense. Let b_m denote the observed function value at pixel index m and $\vec{n}_m$ the corresponding normal, then the formulation of the least squares problem is as follows:

$$\vec{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_M \end{pmatrix}, \vec{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_9 \end{pmatrix} \tag{4.7}$$

$$A = \begin{pmatrix} y_1(\vec{n}_1) & \dots & y_9(\vec{n}_1) \\ \vdots & \vdots & \vdots \\ y_1(\vec{n}_M) & \dots & y_9(\vec{n}_M) \end{pmatrix} \tag{4.8}$$

$$\arg\min_{\vec{c}} |A\vec{c} - \vec{b}|^2. \tag{4.9}$$

This is the matrix formulation of Equation 4.1. This optimization problem can be solved by singular value decomposition. Note that the estimated coefficients need to be scaled in order to fit the range of the spherical harmonics which were estimated by the intrinsic image algorithm. The final results for the ground truth of the eight lighting conditions are depicted in Figure 4.7

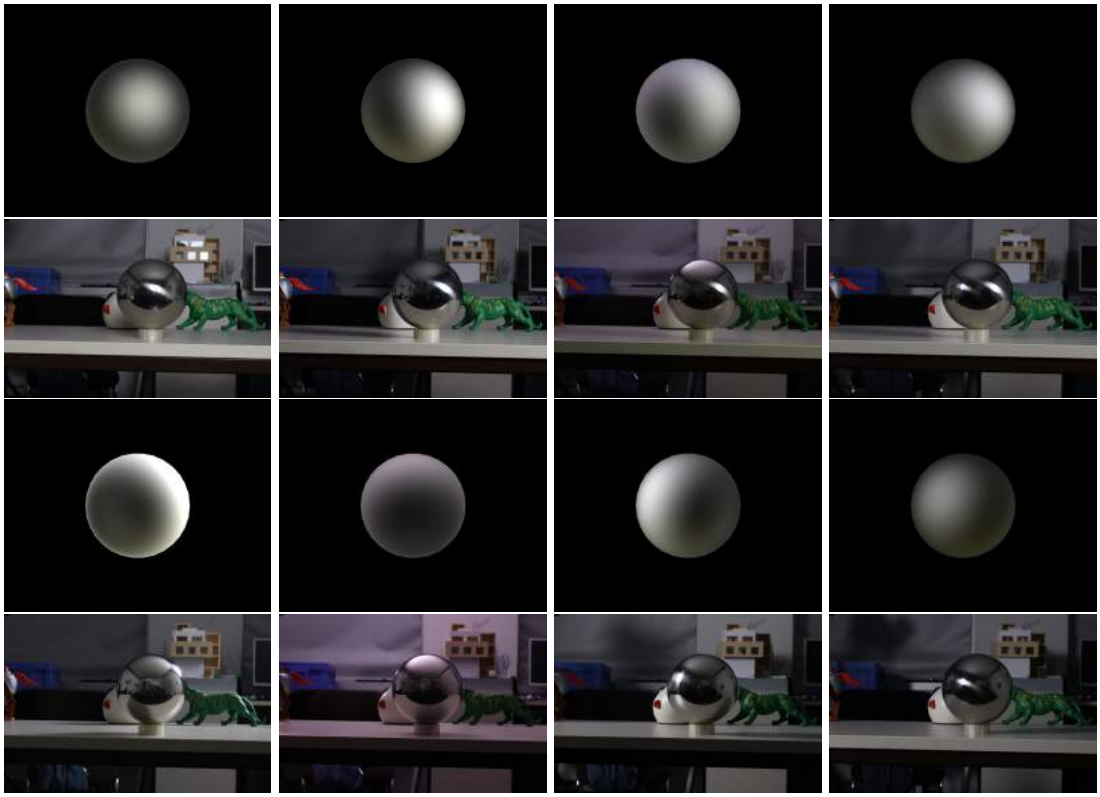

Figure 4.7: The eight light ground truth lighting conditions. The renderings in the black boxes show the reconstructed ground truth lighting functions. The image below each of them shows one of the images which were used to create the HDR image.

4.2 Investigation of the Intrinsic Image Algorithm

This section will investigate the viability of the intrinsic image algorithm as a pre-estimation step for the lighting. I compare the output of the algorithm for images of the same object under the same lighting condition. Figure 4.8 shows the original images, the estimated albedos, and the estimated lighting condition. The intrinsic image algorithm outputs one estimation of the spherical harmonic coefficients per pixel. The depicted lighting estimations show the reconstruction of the lighting function of one pixel on the object. The light source is positioned on the right side of the camera for this test. The ground truth lighting condition in Figure 4.9 correctly shows the highlight on the right side of the sphere.

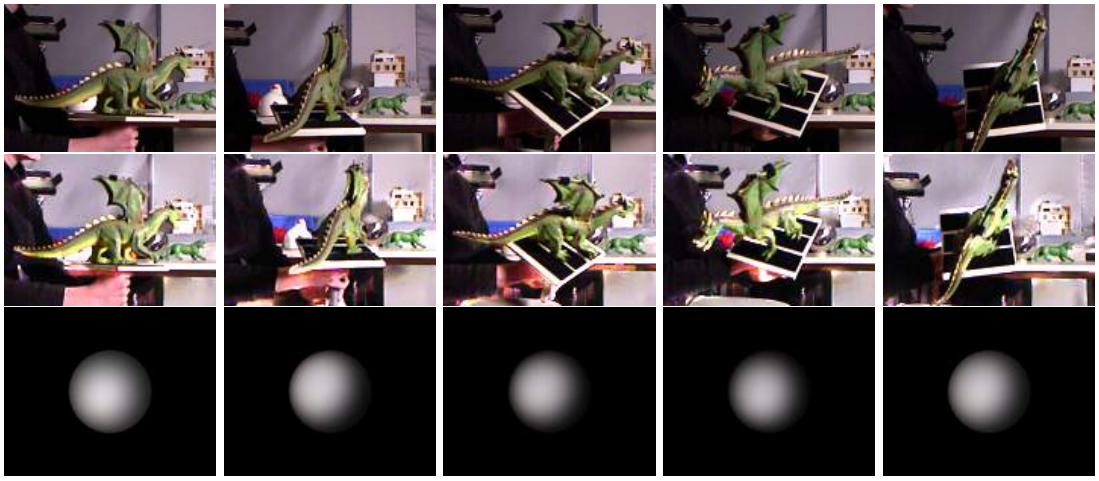

Figure 4.8: Top Row: Original images. **Middle Row:** Estimated albedos. **Bottom Row:** Estimated spherical harmonics.

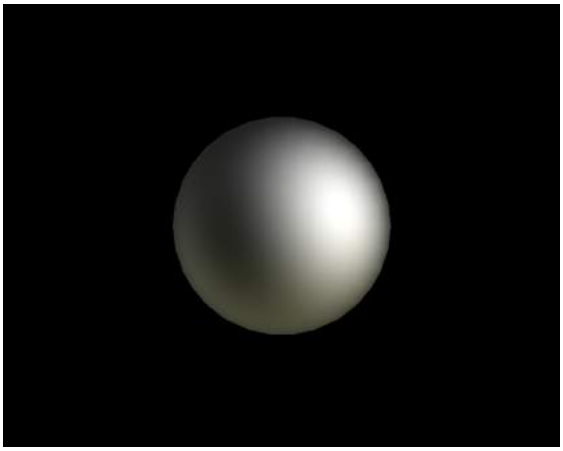

Figure 4.9: Ground truth spherical harmonic estimation of the lighting condition.

The depicted images are taken from the training set of the Dragon and thus only differ in the orientation of the object. The rest of the image is kept static. The perfect albedo estimation would yield images which contain the same color for the same part of the object. The color of the albedos from the intrinsic image algorithm varies significantly between the different images. Looking at the lighting estimation, the orientation of the light is estimated on the left part of the sphere. This contradicts the ground truth lighting condition. Looking at the complete dataset (not depicted) the estimated lighting conditions tend to have a skew to the left side of the sphere. The reason for this behavior might be the discrete set of lighting conditions which

are used for the optimization in the intrinsic image algorithm. Adapting this dataset of lighting conditions in the intrinsic image algorithm might yield better estimation results.

4.3 Performance Tests of the Light Estimation Methods

This section features a comparison of the different inference methods to estimate the correct lighting condition present in the test images. Per object, approximately 600 test images are used. For each light estimation method, the rate of correctly detected lighting conditions is measured. Figure 4.10 shows that the prior light estimation

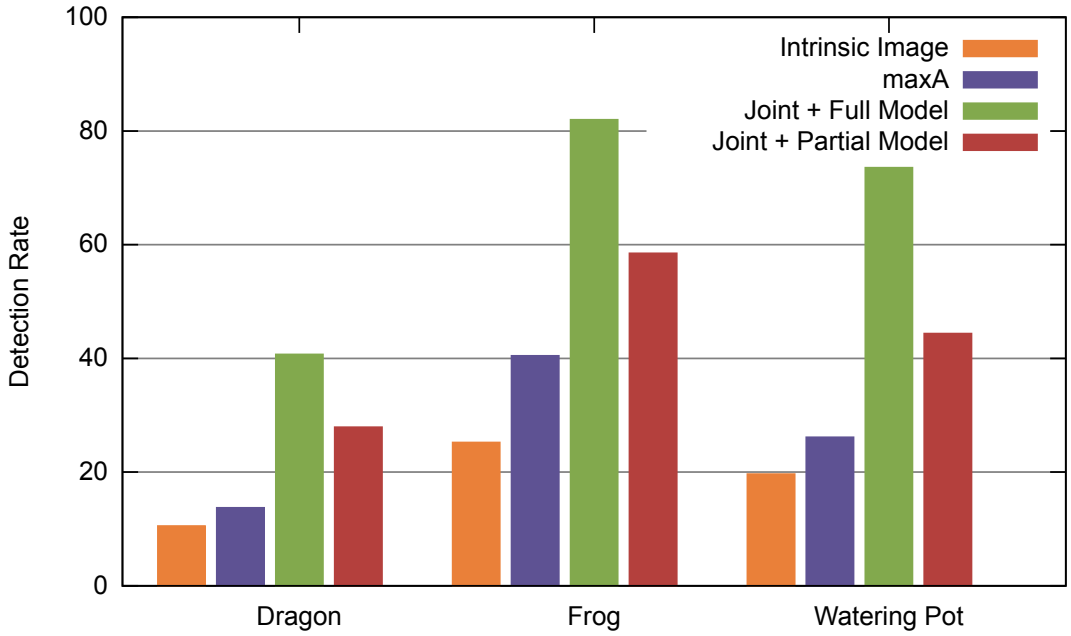

Figure 4.10: The detection rates of the different light estimation methods on every object.

techniques only accomplish a very poor performance on this dataset. For the Dragon, the intrinsic image approach accomplishes a detection rate of only $10,5\%$. The maxA approach achieves $13,7\%$. This shows that these methods are not better than a random selection of the lighting condition per image. A uniform random sampling would yield a performance of $12,5\%$ because eight lighting conditions are possible. The best

results are obtained for the Frog: 25% for the intrinsic image approach and 40,45% for the maxA method. Both estimation methods are not suitable for the performance improvement of the conditional random forest. In order to improve the pose estimation task, their detection rate must be higher than the detection rate of the joint inference method.

The results for the light detection using the joint inference method are the best among the different estimation methods. Using this approach, the Full Model outperforms the Partial Model by a large margin. Although this method obtains the best results, some problems are connected with it. Because the complete pose estimation pipeline has to be performed for every lighting condition in order to obtain a lighting estimation, the runtime scales linearly with the amount of lighting conditions. This could cause some problems for future iterations of the system which take more lighting conditions into account. Striking is the strong drop in performance regarding the Dragon. This object is the most colorful and textured of the three, so the conditioning approach might have problems on this type of object. Another reason for the bad light estimation of the Dragon object could be the difficulty of the recorded test set. The poses of the test set might contain slight inplane rotation which was not covered by the training set. The forest might not be able to generalize these new poses well enough.

4.4 Detection Rates

The purpose of this chapter is the performance comparison between the different methods to construct the conditional random forest and the baseline method. I also want to investigate the influence of the depth features on the performance of the different systems. To compare the performance, the rate of correctly detected poses must be measured. For this task I use the same error measure as [6]. The original model is transformed with the ground truth as well as the detected pose hypothesis. The error is the summed minimum distances between the transformed vertices of the model.

$$e_{pose} = \frac{\sum_{v \in V} \min_{v'} (|H_{gt}\vec{v}' - H^*\vec{v}|)}{d|V|}, \tag{4.10}$$

where $v \in V$ enumerates all the vertices of the object, H_{gt} is the ground truth transformation, d is the diameter of the object's bounding box and H^* is the detected hypothesis.

A detection counts as an inlier if the calculated error e_{pose} is smaller than 0.1. The rate of regression inliers is also considered in these tests. This rate measures the deviation of each object coordinate prediction from the ground truth object coordinates. To calculate this rate, the predicted and rendered object coordinates are compared at each pixel of the ground truth mask. Deviations below 20 mm are counted as inliers. Furthermore, all the presented tests are performed using the joint inference method from Subsubsection 3.4.2.1 because the other two presented inference methods were not able to reliably predict the correct lighting condition. All the forests are trained using the parameter setting of [6]. At each node, 1000 randomly sampled features are evaluated. The recursive splitting of the nodes stops if 50 or less samples arrive in the node. The number of training samples differs between the tests. The used number of samples per object is explicitly stated for each test.

4.4.1 Dataset from Brachmann et al.

Brachmann et al. incorporated a test of their pose estimation system regarding different lighting conditions in [6]. They recorded twenty objects of their dataset under three different lighting conditions. The random forest is trained using the training images from two of the three lighting conditions. The testing was performed on all the lighting conditions. The result of the test was, that the detection rate dropped for testimages of unseen lighting conditions by about 7% on average. For the seen lighting conditions no real performance drop was noticed. This dataset differs from my own recordings, because the camera-light configuration is not kept static. The images are taken by moving around the object while keeping the lighting condition constant. Because of that, the prior light estimation using intrinsic images can not be applied to this dataset. I will restrict the tests on this dataset to the joint inference method using the Full Model. The dataset consists of one dark lighting condition with the room light turned off, one bright lighting condition with the lights turned on and one spotlight. I trained one regular random forest with all lighting conditions as the baseline. For the construction of the Full Model, I trained three specialized forests on the distinct lighting conditions. All tests are performed only on the bright test dataset using RGB and depth features. The different forests were trained using 600,000 training samples per object for all training images.

	Regular RF	Conditional RF	Oracle
Avg. Pose Inliers	96,31%	96,05%	96,96%
Avg. Regression Inliers	52,74%	54,27%	56,49%

Table 4.1: Detection rates for RGB and depth features.

The Oracle describes the forest which was trained exclusively on the bright dataset. It simulates the perfect light estimation. The result of this forest gives insight about the possible upper boundary of the performance of the conditional random forest. The evaluation using RGB and depth features Table 4.1 shows no distinct performance difference between the different forest methods. This results from the powerful depth features. During the training of the forests, the depth features are chosen roughly twice as frequently as color features for the split of the samples. This means that the depth features are more likely to achieve a high information gain in the splits. The more depth features are used, the forest becomes less affected by the color deviations caused by the different lighting conditions. The second test aims to evaluate the RGB features.

	Regular RF	Conditional RF	Oracle
Avg. Pose Inliers	72,72%	81,55%	87,07%
Avg. Regression Inliers	39,47%	43,34%	45,82%

Table 4.2: Detection rates with only RGB features.

Table 4.2 shows a general drop of the performance because the descriptive contribution of the depth features is missing. With this prerequisite, the conditional random forest shows a significant performance boost with regards to the regular random forest. The high performance of the Oracle shows that the correct pose estimation is dependent on the correct estimation of the lighting condition. This suggests the incorporation of a prior light estimation step which could potentially accomplish a performance boost. The conditional random forest estimates the correct lighting condition in 85% of the cases on this dataset. The prior light estimation step must exceed this score in order to improve

the pose estimation. As a last experiment on this dataset, I trained a forest only on depth features. This forest was still able to achieve a detection rate of 95%. This shows the importance of the depth features for the regression step.

4.4.2 Own Dataset

This dataset covers the three different objects under eight different lighting conditions. The first test on this dataset compares the baseline method with the Full Model and the Partial Model of the conditional random forest. Table 4.3 shows the results of this test. Each forest of the Full Model, as well as the regular random forest are trained using 1 million samples. The Partial Model is trained using 4 million samples. RGB and depth features are used for this test. The Oracle measures the performance a conditional random forest would have, if it was provided with a perfect prior light estimation step. Again, the forests show no significant differences regarding the detection rates. This marginal difference is the result of the incorporation of the depth features. However, looking at the regression inlier rates, the regular random forest outperforms both conditional models. This might be the case, because the regular random forest learned from a wider range of poses than each of the conditional forests.

	Object	Regular RF	Full Model	Partial Model	Oracle
Avg. Pose Inliers	Dragon	98,17%	97,56%	98,17%	98,62%
	Frog	100%	100%	98,93%	100%
	Watering Pot	99,35%	99,67%	99,67%	99,67%
Regression Inliers	Dragon	56,92%	51,46%	56,31%	
	Frog	79,17%	71,87%	63,37%	
	Watering Pot	54,08%	49,171%	48,67%	

Table 4.3: Detection rates for RGB and depth features.

The second test is performed using only RGB features. Table 4.4 shows the detection

rates for this test. The Full Model achieves a $10 - 20\%$ performance gain on the test set of the Frog and the Watering Pot. These are the objects where the correct lighting condition is found frequently. This encourages the use of conditional random forests for the modeling of the light. For the Dragon, no significant performance gain is achieved.

	Object	Regular RF	Full Model	Partial Model	Oracle
Pose Inliers	Dragon	42,83%	43,59%	28,50%	51,14%
	Frog	70,90%	91,36%	63,93%	92,62%
	Watering Pot	81,77%	91,77%	72,09%	89,15%
Regression Inliers	Dragon	16,39%	17,57%	12,53%	
	Frog	38,95%	49,62%	37,55%	
	Watering Pot	25,83%	31,86%	25,89%	

Table 4.4: Detection rates for RGB features only.

This results from the fact that the correct lighting condition is rarely found by the joint inference approach for this object (see Figure 4.10). A better inference method would solve this problem, because the Oracle shows that the performance of the conditional system can be improved when the correct light condition is chosen. The Partial Model is inferior to the Full Model in these tests. Interestingly, the Oracle achieves a worse detection rate than the Full Model on the Watering Pot. This anomaly is probably caused by the different pose coverage of the different conditional forests. This means that sometimes the forest of the wrong lighting condition achieves a higher score, because the pose in the test image was closer to one of the training images of this forest.

4.4.3 Analysis of the Full Model

To gain a deeper understanding of the conditional model, I evaluated the pose estimation performance of every forest from the Full Model on every lighting condition. This

test aims to obtain some insights about the generalization of each conditional forest to the other lighting conditions. The tables Table 4.5, Table 4.6 and Table 4.7 show the full matrix of each lighting condition tested with each conditional forest. The columns show the performance of each forest which was trained on a different lighting condition on the test set of one lighting condition. The rows present how one forest performs on all the different lighting conditions.

	Light 1	Light 2	Light 2 & 4	Light 2 & 6	Light 3	Light 4	Light 5	Light 6
Light 1	**52,08**%	44,77%	**40,0**%	44,57%	36,48%	0,0%	23,25%	33,33%
Light 2	36,45%	**62,68**%	32,63%	31,32%	27,02%	2,22%	13,95%	34,25%
Light 2 & 4	36,45%	49,25%	29,47%	40,96%	18,91%	4,44%	18,6%	34,25%
Light 2 & 6	50,0%	50,74%	**40,0**%	**57,83**%	39,18%	0,0%	**58,13**%	**55,55**%
Light 3	29,16%	41,79%	23,15%	32,53%	**51,35**%	0,0%	23,25%	40,74%
Light 4	17,70%	19,40%	12,63%	18,07%	27,02%	**46,66**%	4,65%	28,7%
Light 5	38,54%	20,89%	29,47%	16,86%	21,62%	6,66%	**58,13**%	43,51%
Light 6	28,12%	19,40%	22,1%	18,07%	24,32%	12,22%	25,58%	50,92%

Table 4.5: Detection rates for every forest of the Full Model on the Dragon object. Only RGB features are used. The columns describe the performance of each forest on one lighting condition. The rows show the performance of one forest on all the lighting conditions. The highest values in every column are highlighted.

In most of the cases, the forest corresponding to the correct lighting condition achieves the best performance. Particulary striking is the bad performance of every forest on the Light 4 test set. Even the corresponding Light 4 forest has a lower performance than all other forests on their corresponding test sets among all objects. This means that some lighting conditions are hard to handle for the system even if the correspondence is known. Coincidentally, Light 4 is the darkest of the light sources, so the noise of the camera is higher than for the other lighting conditions. The appearance variability caused by the noise might be one of the reasons why the forest of this lighting condition performs so poorly.

	Light 1	Light 2	Light 2 & 4	Light 2 & 6	Light 3	Light 4	Light 5	Light 6
Light 1	**92,85**%	64,93%	65,93%	87,01%	29,72%	0,0%	53,24%	56,86%
Light 2	51,19%	**87,01**%	49,45%	54,54%	40,54%	0,0%	33,76%	31,37%
Light 2 & 4	65,47%	**87,01**%	**100,0**%	64,93%	63,51%	5,12%	40,25%	51,96%
Light 2 & 6	86,90%	**87,01**%	59,34%	**96,10**%	32,43%	0,0%	50,64%	59,80%
Light 3	41,66%	79,22%	85,71%	49,35%	**90,54**%	0,0%	20,77%	51,96%
Light 4	0,0%	7,79%	6,59%	0,0%	5,40%	**87,17**%	23,37%	17,64%
Light 5	17,85%	24,67%	14,28%	37,66%	4,05%	6,41%	**96,10**%	88,23%
Light 6	33,33%	33,76%	15,38%	33,76%	5,40%	2,56%	83,11%	**91,17**%

Table 4.6: Detection rates for every forest of the Full Model on the Frog object, Only RGB features are used,

	Light 1	Light 2	Light 2 & 4	Light 2 & 6	Light 3	Light 4	Light 5	Light 6
Light 1	**98,88**%	90,32%	65,0%	96,29%	72,85%	0,0%	63,63%	58,58%
Light 2	67,77%	87,09%	68,75%	83,95%	67,14%	1,38%	53,03%	54,54%
Light 2 & 4	91,11%	**100,0**%	**91,25**%	**97,53**%	67,14%	12,5%	60,60%	54,54%
Light 2 & 6	87,77%	88,70%	73,75%	95,06%	62,85%	0,0%	62,12%	59,59%
Light 3	70,0%	83,87%	46,25%	88,88%	**95,71**%	2,77%	50,0%	52,52%
Light 4	4,44%	16,12%	0,0%	1,23%	14,28%	**62,5**%	21,21%	24,24%
Light 5	44,44%	51,61%	27,5%	61,72%	42,85%	9,72%	**100,0**%	**87,87**%
Light 6	50,0%	50,0%	7,5%	54,32%	28,57%	6,94%	86,36%	82,82%

Table 4.7: Detection rates for every forest of the Full Model on the Watering Pot object, Only RGB features are used,

Also, this light source is the only LED light source and has a slightly different tone than the other light sources. A trend regarding the proximity of the light can also be observed in these tables. The further the angluar distance of the light sources is, the lower the detection rate will be, most of the time. Regarding the combined lighting conditions, the forests of Light 2 & 4 and Light 2 & 6 yield comparable results to the corresponding forest of Light 2 on the test set of Light 2. This is not true for the test sets of Light 4 and Light 6. This implies that Light 2 dominates the other light sources in these combined lighting conditions.

4.4.4 Test on Unseen Lighting Conditions

This test aims to compare the Full Model using the joint inference method to the regular random forest regarding their generalization capabilities. Both models are trained using the training images of all lighting conditions but Light 5. The test is then performed

	Dragon	Frog	Watering Pot
Full Model	20.93%	87.01%	93.93%
Regular Random Forest	30.23%	77.92%	71.21%

Table 4.8: Generalization comparison of the Full Model and the baseline method. This table shows the detection rates of the conditional random forest and the regular random forest on the test set for the lighting condition Light 5. Both forests were trained without the training images of Light 5. Only RGB features are used.

only on the test set of Light 5. Table 4.8 shows the resulting detection rates. Surprisingly, the conditional random forest achieves worse results for the Dragon, but outperforms the regular random forest by a large margin on the other objects. The conditional random forest chooses Light 6, the nearest lighting condition to Light 5 80.3% of the time for the Watering Pot and 90.9% of the cases for the Frog. This explains the high detection rates of the Full Model for these objects. For the Dragon, the forest chooses Light 6 only in 18% of the cases. This suggests that the appearance variability of the Dragon is higher under the different lighting conditions than the appearance variability of the Frog or the Watering Pot.

4.4.5 Precision Comparison

In this section, I want to investigate whether the conditional random forest achieves a better precision than the regular random forest. I chose to investigate only the joint inference method using the Full Model because this combination achieved the best performance rates. The pose error is calculated as in Equation 4.10. Figure 4.11 shows the histograms of the pose error for each object. They show the amount of poses which

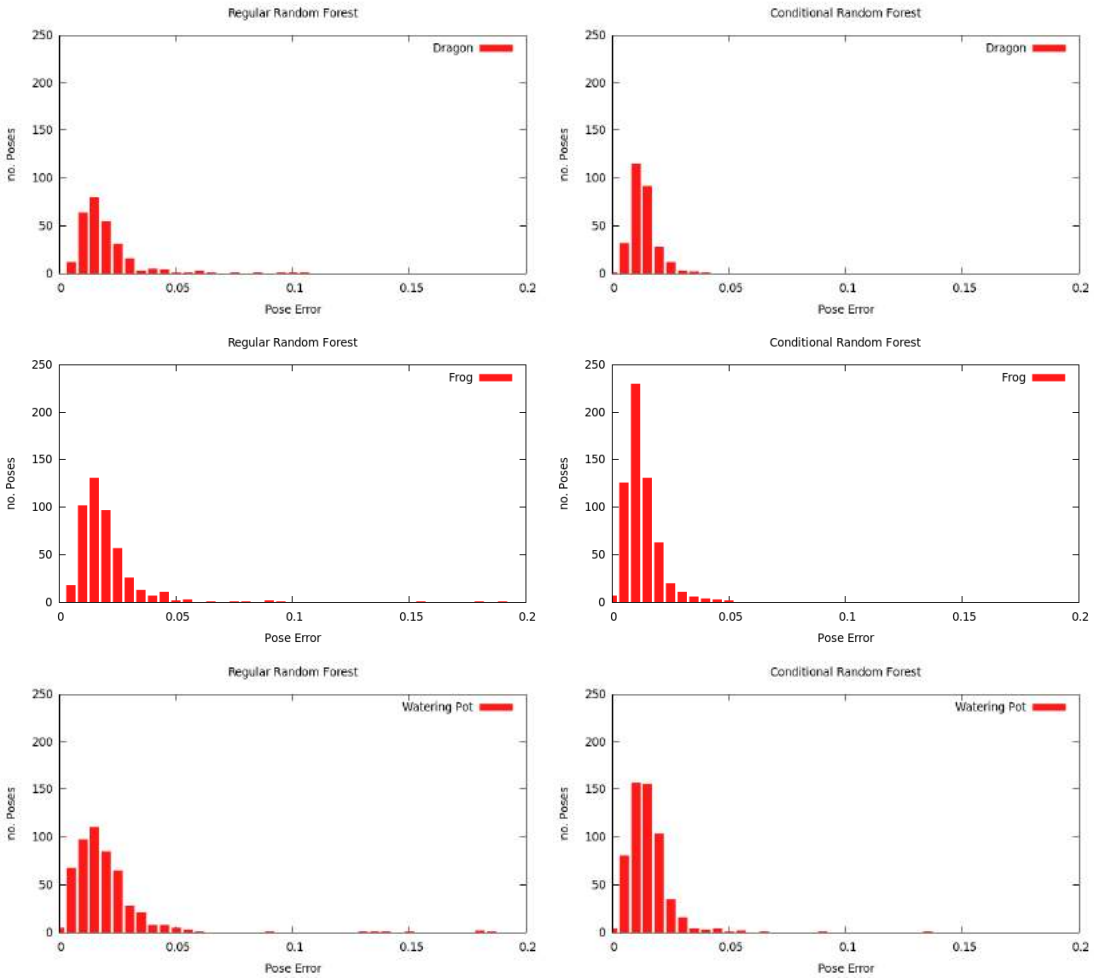

Figure 4.11: Precision comparison of the Full Model and the baseline method. Only RGB features are used.

achieved a certain pose error. The size of the bins is set to 0.005. The pose outliers are not shown in these graphs. This visualization shows, that the conditional random forest also achieves a more precise pose estimation result for each object, because the

peaks of the histograms of the conditional random forests correspond to a lower pose error than the peaks of the regular random forest.

5 Discussion and Future Work

This thesis presented multiple methods to model the lighting in order to improve the pose estimation task. All methods were based on conditioning the estimator on a discrete set of lighting conditions. The random forest framework was extended to a conditional random forest using the Full Model and Partial Model as different construction methods. The Full Model was able to outperform the Partial Model in my experiments. This contradicts the results of [22]. The reasons for this might be my approach for the learning of the Partial Model. I used a new labeling approach which incorporated the lighting information into the learning procedure. [22] explain that they use the standard learning method for regular random forests which does not employ the lighting information. Regarding future work, a test on a normally trained Partial Model should be employed.

For the inference of the final object pose using these conditioned systems, I investigated two prior light estimation approaches. The maxA method relies on the output of the regression step from the Partial Model to infer a light estimation. [22] proposed this method. They had a segmented depth image of a human pose available as the input for their system. This segmentation made the maxA approach viable. In the object pose estimation task the segmentation is not given beforehand. This causes the background to interfere with the light estimation step too much. To make this approach viable, it would be a reasonable approach to condition the background on the lighting condition as well. This would result in a more reliable estimation of the ligthing condition at the background pixels. The problem with this approach would be the limitation to a small amount of possible backgrounds during testtime.

The method using the intrinsic image algorithm by Barron et al. as implemented in this thesis was not able to reliably predict the correct lighting condition. Reasons for this behavior might be the lacking tuning of the system for this use case. The internal lighting condition estimation of the intrinsic image algorithm for example is dependent

on a learned dataset of lighting conditions. Moreover, it is not clear how well the spherical harmonic representation is suited for the modeling of the lighting condition because the limitation to infinitely far away light sources is unreasonable for many real-world applications.

The joint inference of the lighting condition and the pose proved to increase the detection rates by a significant amount when only RGB features are used for the regression. This result is of great interest for the further development of the pose estimation system because one of the problems with the existing framework is the dependency on a depthmap as an additional input. The solution to this problem would make the system more applicable for real-world tasks where the depthmap often can not be provided. Even though the joint inference method presents promising results on the investigated datasets, some problems need to be overcome.

One problem is the discrete representation of the lighting conditions. Any real-world application using this approach would need to cover a huge number of possible lighting conditions. The complete space of possible lighting conditions is hard to cover using this discrete representation even if just one light source is allowed in the scene. If the camera was moving around freely, the relative position of the light source would change with respect to the camera. This fact necessitates the dataset to contain every direction of the incoming light relative to the camera. That means that for every light orientation one conditional random forest needs to be trained. The less constraints can be imposed on the lighting conditions at test time, the more forests need to be trained. An unconstrained real-world application which works for every lighting condition is thus not feasible using this approach. Another problem moving entailed by this method is the runtime. The joint inference is scaling linearly with respect to the lighting conditions. This will result in unfeasible runtimes if the amount of lighting conditions increases. The incorporation of a sophisticated prior lighting estimation will be necessary for futher development.

Provided with such a sophisticated prior light estimation method, the inference method could be extended to a mixing method as described by [9]. From every conditional forest, trees might get added to the new random forest proportional to the probability of the latent variable which is provided by the pre-estimation step. The inference then could be performed conventionally.

One aspect which should receive some attention is the extension of the dataset to more

textured objects. The Dragon showed a strong drop in performance with regard to the textureless objects. More textured objects should be examined to confirm or refute the hypothesis that lighting is harder to estimate on textured objects.

For the recording of the dataset, a more advanced method should be employed as well. For this thesis I did the pose coverage of the objects per hand for each lighting condition. Moving forward to a bigger set of lighting conditions, a rendering procedure for the creation of the training images should be used. This would result in a low effort method for the creation of the dataset. Only the 3D model of the real object would have to be recorded. The lighting and rotation would be done by the rendering software. Many different lighting conditions could be modeled using this approach. The inplane roation would also be easy to incorporate. However, the mapping from the rendered training images to the real-world test images is a non-trivial task.

Another interesting approach for future work in this area is the albedo forest. Provided with the ground truth spherical harmonic illumination, it is possible to recover the albedo of the object for each training image. This allows the trained forest to be completely invariant towards lighting. During testtime, the albedo of the scene must be inferred. Using this albedo as input for the forest, the correct pose could be estimated more reliably. The benefit of this method is that only one forest needs to be trained in order to cover all possible lighting conditions. The problem arises in the albedo estimation step during test time. This could be solved by a better tuned intrinsic image system.

List of Figures

List of Tables

Literature

[1] D. Anastasia, F. Comida, and D. Kaneider. *Luminance HDR Software*. (accessed April 29, 2015). URL: http://qtpfsgui.sourceforge.net/.

[2] I. Badami, J. Stückler, and S. Behnke. "Depth-Enhanced Hough Forests for Object-Class Detection and Continuous Pose Estimation". In: *SPME*. 2013.

[3] J. Barron and J. Malik. "Intrinsic Scene Properties from a Single RGB-D Image". In: *CVPR*. 2013.

[4] J. Barron and J. Malik. "Shape, Albedo, and Illumination from a Single Image of an Unknown Object". In: *CVPR*. 2012.

[5] R. Basri and D. Jacobs. "Lambertian reflectance and linear subspaces". In: *TPAMI*. 2003.

[6] Eric Brachmann et al. "Learning 6D Object Pose Estimation using 3D Object Coordinates". In: *ECCV*. 2014.

[7] L. Breiman. "Random Forests". In: *Machine learning*. 2001.

[8] A. Criminisi, J. Shotton, and E. Konukoglu. *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*. Technical report. Microsoft Research, 2011.

[9] M. Dantone et al. "Real-time Facial Feature Detection using Conditional Regression Forests". In: *CVPR*. 2012.

[10] P. Debevec. "Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography". In: *SIGGRAPH*. 1998.

[11] J. Gall et al. "Hough forests for object detection, tracking, and action recognition". In: *TPAMI*. 2011.

[12] S. Hinterstoisser et al. "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes". In: *ACCV*. 2012.

[13] S. Izadi et al. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera”. In: *UIST*. 2011.

[14] W. Jarosz. *Spherical Harmonics Appendix*. (accessed April 29, 2015). URL: http://www.cs.dartmouth.edu/~wjarosz/publications/dissertation/appendixB.pdf.

[15] S. Knorr and D. Kurz. “Real-time Illumination Estimation from Faces for Coherent Rendering”. In: *ISMAR*. 2014.

[16] M. Martinez, A. Collet, and S. Srinivasa. “MOPED: A Scalable and Low Latency Object Recognition and Pose Estimation System”. In: *ICRA*. 2010.

[17] J. Mitchell. “Discrete Uniform Sampling of Rotation Groups Using Orthogonal Images”. In: *SISC*. 2007.

[18] R. Newcombe et al. “KinectFusion: Real-time Dense Surface Mapping and Tracking”. In: *ISMAR*. 2011.

[19] R. Ramamoorthi and P. Hanrahan. “On the Relationship between Radiance and Irradiance: Determining the Illumination from Images of a Convex Lambertian Object”. In: *JOSA A*. 2001.

[20] J. Shi and J. Malik. “Normalized Cuts and Image Segmentation”. In: *TPAMI*. 2000.

[21] J. Shotton et al. “Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images”. In: *CVPR*. 2013.

[22] M. Sun, P. Kohli, and J. Shotton. “Conditional Regression Forests for Human Pose Estimation”. In: *CVPR*. 2012.

[23] C. Wu et al. “3D Model Matching with Viewpoint-Invariant Patches (VIP).” In: *CVPR*. 2008.