VIEWPOINT DEPENDENT MODEL FOR MULTI-VIEW OBJECT RECOGNITION

THESIS REPORT

by

L. F. M. de Lange

Master of Science
in Mechanical Engineering

at the Delft University of Technology,
April 19, 2016

Thesis committee: Prof. dr. ir. Pieter Jonker
Dr. Emile Hendriks
Ir. Aswin Chandarr
Ir. Boris Lenseigne

TU Delft Delft University of Technology

CONTENTS

ABSTRACT

The life expectancy of humans increases due to better medical care, food quality and personal hygiene. The demand for domestic robots performing (simple) household chores will increase as consequence to this trend. An important task of these robots is recognizing objects within an environment. Object models are constructed using multiple views from various viewpoints in order to increase the recognition robustness and reduce the influence of noise such as variation in illumination. The objective of this thesis is the construction of a novel viewpoint dependent model for multi-view object recognition.

A benchmark is created in order to evaluate novel object recognition models for multi-view object recognition. The acquisition of object data is replaced by a scientific captured object dataset, which reduces the influence of external noise (i.e. illumination) and unequal view distributions. Edges, corners, shape, color and texture features are extracted, which describe the object data mathematically. Artificial noise is added for the construction of query objects, which are classified via the Euclidean distance measure.

Based on sequence alignment, a novel viewpoint dependent object model is proposed and evaluated. Each view in a viewpoint dependent object is represented by a single value based on the location of that view in a kd-tree. The proposed sequence alignment algorithm matches objects via the optimal alignment of view sequences. The benchmark results show that multi-view sequence alignment has a higher object recognition rate compared to single-view object recognition.

Novel object data is captured by moving a hand held Kinect camera around an object. A visual odometry method is implemented, which estimates the camera egomotion from registered depth and color. The ground plane is extracted from the depth data for the initial camera alignment and object segmentation. A viewpoint dependent object is constructed from the estimated egomotion and segmented object data. For object recognition each object is captured twice, where object variation occurs by changes in lighting and unequal distributed views. The scientific object datasets are replaced by the novel captured object datasets for evaluation of the object recognition performance. The benchmark results show an increased object recognition rate for the sequence alignment algorithm, despite the influence of external noise factors.

1

INTRODUCTION

Humans tend to live longer due to better medical care, food quality and personal hygiene. The demand for household help may increase due to an extended life span. A solution for the support of elderly and disabled is found in the field of robotics. Domestic robots should be able to perform (simple) household tasks by interaction with the environment. Object detection and localization are important tasks of such robots. This thesis focuses on object recognition of household objects with a single digital camera.

The orientation of objects within an environment is arbitrary. Shape, color and texture change in each viewpoint. Evaluating multiple views per object increases recognition robustness and reduces the influence of noise factors such as illumination. A novel viewpoint dependent multi-view object recognition approach is proposed in this thesis.

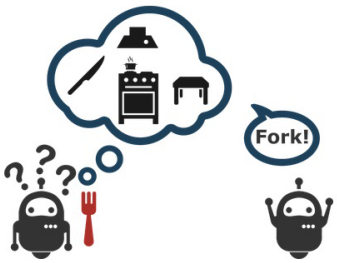

Figure 1.1: Object recognition for domestic robots

A general overview of object recognition is covered in the first part of chapter 2. Existing multi-view object models are treated in the second part, followed by a problem description and a list of design requirements. Based on an object recognition pipeline a benchmark is created for performance evaluation of novel object models in chapter 3. The influence of external variables are reduced by evaluating scientific object datasets for which the recognition performance is determined. A novel viewpoint dependent object model based on sequence alignment is presented and evaluated in chapter 4. Subject of the last two chapters is construction of viewpoint dependent object models from color and depth information. Viewpoint estimation is denoted in chapter 5, which is used to construct viewpoint dependent object models described in chapter 6. The performance of a home made object dataset is evaluated using the benchmark from chapter 3. In the final chapter 7 the conclusion is written.

2

OBJECT RECOGNITION

A general introduction to object recognition is written in section 2.1, where the process from data acquisition to object classification is evaluated. Existing multi-view object models for object recognition are subject of section 2.2. A problem description is formulated in section 2.3, followed by a design requirements list. The conclusion is written in section 2.4.

2.1. OBJECT RECOGNITION PIPELINE

Recognizing an object from a scene is a complex action. Figure 2.1 shows a general overview of object recognition. A digital (depth)camera acquires depth and color data. Objects are segmented and described mathematically by extracting features. These features are are used for object classification. A detail explanation of each step in object recognition is written in the following subsections.

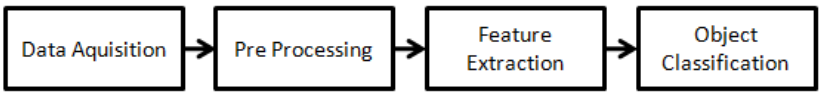

Figure 2.1: Object recognition pipeline

2.1.1. DATA ACQUISITION

Digital cameras are available in all sorts and sizes with different quality and price range (figure 2.2). Expensive range cameras such as laser scanners make use of the time of flight principle, where depth information is obtained from the time a light bundle leaves the emitter and reaches the sensor. Low-cost range cameras emit a pattern of structured light and obtains depth via triangulation of the refracted light. In this thesis, a low cost Kinect sensor is chosen as it is widely used in many affordable domestic robots. The challenge is to obtain a good recognition performance with low quality images and challenging environment conditions.

Figure 2.2: HD webcam, Laser scanner, Microsoft Kinect

2.1.2. PRE-PROCESSING

The acquired data is large in quantity and contains many color and depth values. Irrelevant background data is removed by object segmentation. The object data is re-scaled and placed at the center of a blank canvas. This produces a consistent and uniform appearance of all object data.

2.1.3. FEATURE DESCRIPTION

An important step in object recognition is feature description. In order to perform mathematical operations on the data, features are extracted. These features are an abstract representation of the data based on mathematical concepts. An overview of feature descriptor properties are listed below. Descriptor categories include edges/corners, texture, shape and color, which are summarized briefly in the remaining of this subsection.

- Descriptive power: The capability to discriminate dissimilar objects
- Robustness: The influence of noise on a descriptor
- Invariance: The influence of linear transformations (rotation, translation and scaling) on a descriptor
- Complexity: The execution time of the feature extraction process

EDGES AND CORNER DESCRIPTORS

Edges and corners are detected via the second derivative of an image known as the image Laplacian. Convolution of an image with a Laplacian kernel (figure 2.3) shows an increased response at locations containing edges and corners. Researchers try to approximate the image Laplacian in order to reduce the descriptor extraction time. Two of the most cited edge and corner descriptors are Scale Invariant Feature Transform (SIFT) from Lowe [17]) and Speeded-Up Robust Features (SURF) from Bay et al. [2]. SURF approximates the image Laplacian using boxed kernels (figure 2.3). Dynamic programming is applied to get fast filter responses, which are independent of the kernel size. SIFT approximates the image Laplacian via the Difference Of Gaussian method, which is explained in section 3.3.

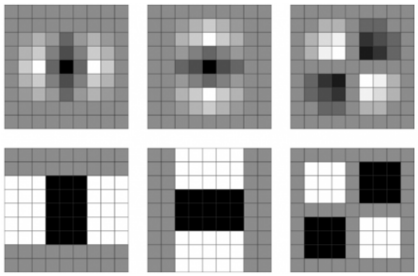

Figure 2.3: Laplacian kernel (top) and approximation box kernel (bottom)

TEXTURE DESCRIPTORS

The descriptive power of some objects is captured by their texture. Texture descriptors are based on local binary patterns (LBP), which was first described by Ojala et al. [19]. A circular neighborhood is evaluated around each pixel (figure 2.4). A binary pattern is obtained by comparing each neighborhood pixel with its center pixel. All binary patterns are combined in a histogram, which represents the texture of an object. Based on this LBP algorithm, many extensions are proposed to improve the descriptor properties.

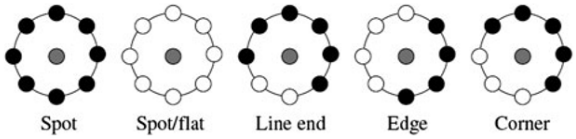

Figure 2.4: LBP descriptors

SHAPE DESCRIPTORS

In some cases the shape of an object is a unique and distinctive feature. The paper of Zhang and Lu [28] gives a review of the most used shape descriptors. A distinction is made between contour and region-based shape descriptors. Circularity, eccentricity and compactness are example shape factors, which describe an object by its contour. Image moments and skeleton are examples of region-based shape descriptors.

The addition of depth information from range cameras makes it possible to describe the 3D surface of an object. Surface normals are estimated in a local neighborhood and used to obtain the surface curvature. An alternative method proposed by Wohlkinger and Vincze [27] uses random sampled points. This so called ensemble of shape descriptor is shown in figure 2.5 for different distance measures.

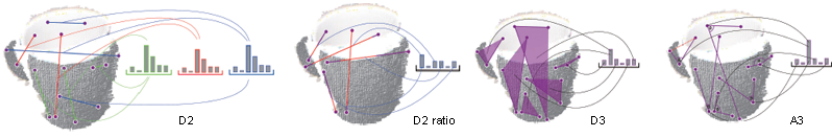

Figure 2.5: 3D shape descriptor, ensemble of shape functions

COLOR DESCRIPTORS

Color in an image is represented by three channels containing the primary colors red, green and blue (RGB). A more natural representation of color is found in the alternative Hue, Saturation and Value (HSV) colorspace. HSV is by design invariant to color shift and scaling. Figure 2.6 shows three color images and their unique color histogram output.

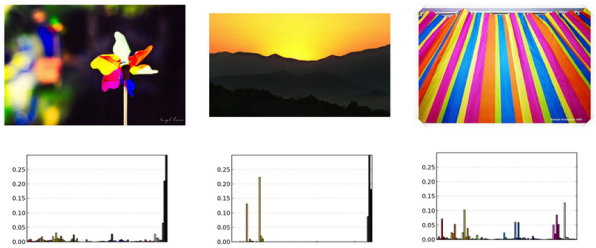

Figure 2.6: Example of color histogram using hue color component

MULTI MODAL DESCRIPTORS

In recent years the development of novel feature descriptors focuses on combining existing algorithms. The aim is construction of descriptors, which have good descriptor properties. Hinterstoisser et al. [11] for example developed *Linemod*, which is a hybrid descriptor combining color gradient with surface normals (figure 2.7).

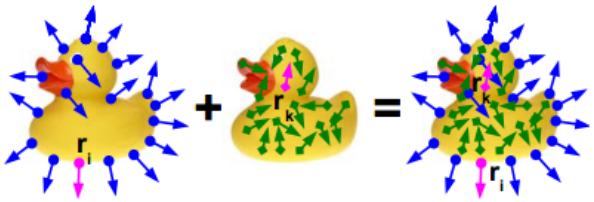

Figure 2.7: Linemod combination of color gradients with surface normals

2.1.4. OBJECT CLASSIFICATION

Classification of query object views is the final step in object recognition. Euclidean, Mahalanobis and Covariance are example distance measures, which directly measure the relation between feature vectors. More complex classification algorithms search for patterns within the feature vector data. These patterns are either linear or non-linear based on the chosen classifier. Discriminant analysis and support vector machine are examples of linear classifiers. Neural networks and the k-nearest neighbor algorithm are non-linear classifiers. Ensemble classification algorithms are based on weak classifiers, which in a particular hierarchy form a very powerful classifier. Well known ensemble algorithms include cascaded HAAR-transform, decision trees and random forests.

2.2. EXISTING MULTI-VIEW OBJECT MODELS

A digital camera captures 2D data of a 3D object. In this process shape, color and texture information of the complete object is lost. Multi-view object models gather more information from an object in order to improve the object recognition rate. These models combine multiple views of an object based on a mathematical relation. In the literature, object models range from 3D point clouds to a more abstract clustered feature approach. This section evaluates the exisiting multi-view object models: Computer Aided Design (CAD), Potemkin, Canonical parts and Hypergraphs.

2.2.1. COMPUTER AIDED DESIGN

Computer Aided Design (CAD) models are created using modeling software and contain no flaws such as outliers or bad registration (figure 2.8). Research mainly focuses on mathematical concepts for optimal recognition given a perfect 3D point cloud model. Object recognition is done either via 2D projection or by registration to another point cloud model.

Figure 2.8: Object models from SHREC14-sketch database

2.2.2. POTEMKIN

CAD models can contain thousands of points, which requires storage space and processor power to evaluate. Chiu [5] describes in his PHD thesis an object model, which approximates the geometry of an object by basic geometrical shapes. A schematic overview of this Potemkin model is shown in figure 2.9. The algorithm is divided in three phases, which are described below.

1. *Generic transforms*
   Synthetic views of basic geometrical shapes are obtained via projection. This phase is object independent and only has to be executed once.

2. *Part transforms*
   Multiple views of an object are manually labeled using the generic transforms from phase one. Individual part transforms are obtained from the labeled views, which describe the relation of a part across all object views.

3. *Object skeleton*
   The labeled views and part transforms from the first two phases are used to estimate the object skeleton. The object skeleton is an abstract 3D representation of an object based on basic geometrical shapes. 2D synthetic views are obtained via projection and used for object recognition

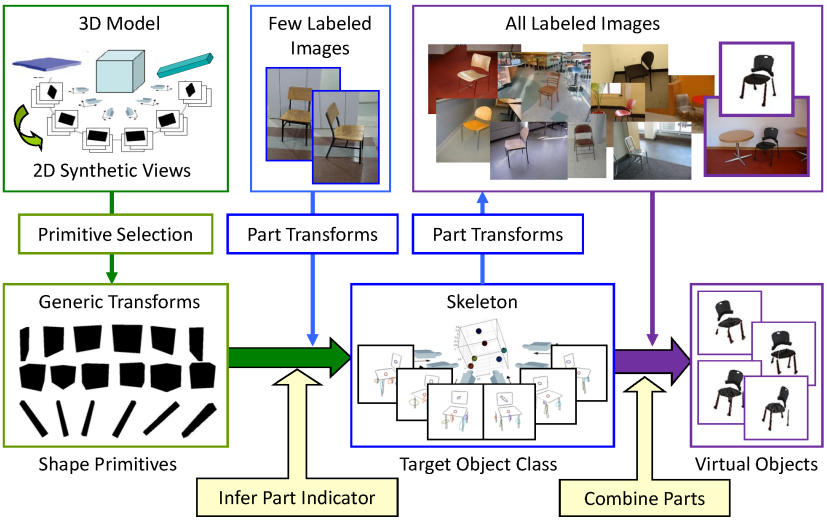

Figure 2.9: Architecture of the Potemkin model

2.2.3. CANONICAL PARTS

A more abstract approach is proposed by Savarese and Fei-Fei [21], who formulated a 2.5D canonical part based model. Edges and corners are extracted from multiple views of an object. Canonical parts are obtained based on the consistency and appearance of the found features across multiple views. A view which shares multiple canonical parts is called a canonical view. The object model is constructed from local relations between canonical parts sharing equal canonical views.

2.2.4. HYPERGRAPH

Pairwise relationships are commonly used in object recognition algorithms. The true relationship between objects are more complex, especially when multiple views are considered. Zhou et al. [29] proposed the use of hypergraphs in order to capture these complex relations. A hypergraph is composed of a vertex set and an edge set (figure 2.10). In case of multi-view object recognition the vertex set contains multiple views of different objects. Features are extracted from each view and clustered to form the edges of the hypergraph. Zhou et al. [29] applied spectral clustering and transductive inference for object recognition.

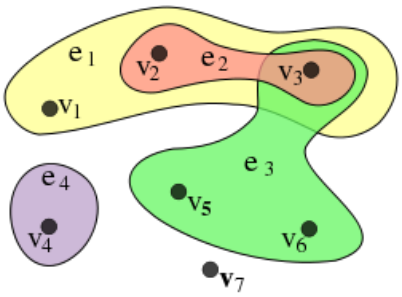

Figure 2.10: Example hypergraph containing 7 vertexes and 3 edges

2.3. PROBLEM DESCRIPTION

The multi-view object models from the previous section are summarized in table 2.1. The CAD and Potemkin object models both have global viewpoint dependency as synthetic views are obtained via projection. Construction of these object models is complex and is done (semi) supervised. No supervision is needed for the canonical part based model. This model consists of local relations between parts that share equal views. Hypergraphs are created unsupervised, but have no viewpoint dependency.

It is hard to compare the recognition performance of object models, because different data is used to validate each object model. A global viewpoint dependent object model captures the full geometry of an object and is more likely to perform better in object recognition than having no view relations. Existing global viewpoint object dependent models are complex to create and need supervision. In this thesis an object model is proposed, which has global viewpoint dependency and does not need supervision to generate viewpoint dependent object models.

Table 2.1: Summary multi-view object models

Model	Description	Supervision	Viewpoint dependency
CAD	3D Point cloud	+	Global
Potemkin	Approximation of geometry using basic 3D shapes	++	Global
Canonical parts	Linked parts sharing equal views	-	Local
Hyper graph	Multi dimensional clustered features	-	None

2.3.1. DESIGN REQUIREMENTS

The field of object recognition is very broad. Research ranges from 3D objects to scene semantics and from objects containing multiple instances such as cups or cars to objects with only one unique instance. Each research area requires a different approach, therefore it is of importance to define a design requirements list.

- Object dimensions are limited to household objects and varies between 10 and 50$\mathtt{cm}^3$
- Novel objects should be learned unsupervised
- Run-time object retrieval is mandatory
- The geometry of an object does not change over time
- The viewpoint dependency should be integrated in the object model
- One object is considered at a time in order to avoid the use of tracking and detection algorithms

2.4. CONCLUSION

A general overview of object recognition is presented at the beginning of this chapter. Data is acquired using a digital (depth)camera. Objects are segmented from the background by removal of irrelevant data. A mathematical representation of the object data by extracting features makes it possible to compare views in the classification step.

Existing multi-view object models include CAD, Potemkin, Canonical parts and Hypergraph. These models range from 3D point clouds to more abstract clustered features and from supervised to unsupervised. Different data is used to validate the object models, which makes it hard to compare the object recognition performance. Based on the existing multi-view object models a novel unsupervised object model with global viewpoint dependency is proposed. A design requirement list is defined beforehand to set boundaries in the broad field of object recognition. The next chapter introduces a systematic approach for the evaluation of multi-view object models in object recognition.

3

MULTI-VIEW OBJECT RECOGNITION BENCHMARK

A benchmark is developed in order to evaluate and compare novel object models for multi-view object recognition. The object recognition pipeline from figure 2.1 is used as guidance in this chapter. Acquisition of object data is described in section 3.1. Feature extraction is subject of section 3.2 and section 3.3 is devoted to object classification. The benchmark performance is measured for single-view object models in section 3.4 by conducting multiple experiments. A conclusion is denoted in section 3.5.

3.1. OBJECT DATA

The benchmark describes a method, which evaluates the object recognition performance of different object models. External variables, which have influence on object recognition should therefore be reduced. Object data is acquired by digital cameras as described in section 2.1.1. A systematic approach is preferred, which minimizes the influence of image noise and/or unequal distributed views. For this reason two existing scientific object datasets are used. Specifications of SOIL-47 by Burianek et al. [4] and the RGB-D by Lai et al. [16] dataset are listed in table 3.1 and view examples are shown in figure 3.1.

Table 3.1: Dataset specifications

SOIL-47 dataset

- Dataset contains cereal boxes
- 20 objects
- 21 views per object
- Fixed elevation angle
- Captures 180 degrees
- Dataset captured twice under different lighting condition

RGB-D Kinect dataset

- Dataset contains household objects
- 51 objects
- 3 times 40 - 50 views per object
- Elevation angles at 40, 45 and 60 degrees
- Captures 360 degrees

Irrelevant background data is removed using object segmentation masks, which are provided for the SOIL-47 and RGB-D dataset. Data normalization is not applied as all data is captured consistent under equal conditions. Each object is placed at the center of a blank canvas. The center of an object is calculated from the segmentation mask boundaries (equation 3.1).

$$c_x = \frac{x_{left} - x_{right}}{2} \quad \text{and} \quad c_y = \frac{y_{top} - y_{bottom}}{2} \tag{3.1}$$

Figure 3.1: Object sample views from SOIL-47 (top) and RGB-D (bottom)

3.2. FEATURE EXTRACTION

A mathematical representation of the data makes it possible to compare views and objects. Section 2.1.3 gives an overview of feature descriptor in different categories. Edges/corners, texture, shape and color features are described via numerous mathematical approaches. From each category one descriptor is used for the object recognition benchmark. These descriptors are obtained from the MATLAB toolbox *vlfeat* by Vedaldi and Fulkerson [25] and are listed below.

1. Scale Invariant Feature Transform
2. Histogram of Oriented Gradients
3. Local Binary Patterns
4. Hue, Saturation and Value
5. Neural Network

3.2.1. SCALE INVARIANT FEATURE TRANSFORM (SIFT)

SIFT by Lowe [17] is based on the detection and description of corners and edges using the Difference of Gaussian approach. The first phase of this algorithm detects points of interest in an image. An input image is convoluted with a Gaussian kernel.

$$G(x,y,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x^2+y^2)/2\sigma^2} \tag{3.2}$$

Where $\sigma = n * \sqrt{2}$ denotes scale and (x,y) the image coordinates. Different scales are applied to construct a scale space representation of the input image. Each consecutive scale produces a more smoothed version of the previous scale. In order to make SIFT scale invariant this process is repeated with a scaled input image. Each so called octave contains a scale space of smoothed images and forms an image pyramid as shown in figure 3.2.

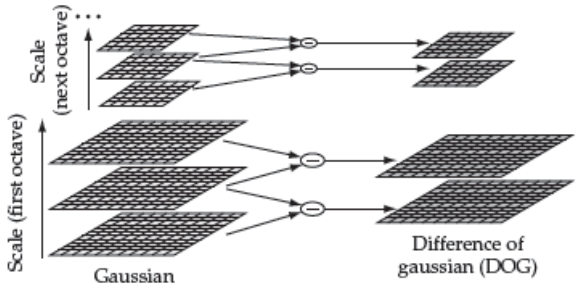

Figure 3.2: DOG construction using scale space

From the image pyramid, points of interest are obtained by approximating the image Laplacian. Within each octave two consecutive scales are subtracted, which is called difference of gaussian (DOG).

$$\Delta G = G(x,y,\sigma+\sqrt{2}) - G(x,y,\sigma) \tag{3.3}$$

Maxima and minima in the DOG images are marked as keypoint. Each pixel is compared to its direct 26 neighbors from which 18 are located at the DOG scales above and below the evaluated pixel. A pixel is marked as maxima or minima when it is respectively the highest or lowest value.

Description of the detected interest points is done in the second phase of the SIFT algorithm. A local neighborhood of 4 by 4 bins is evaluated around each keypoint. The bin size dependents on the octave in which the evaluated keypoint is found. For each bin a spatial histogram of gradients is constructed. Equation 3.5 and 3.4 are used to determine the orientation and magnitude of every pixel within a bin.

$$m(x,y) = \sqrt{((I(x+1,y) - I(x-1,y))^2 + (I(x,y+1) - I(x,y-1))^2} \tag{3.4}$$

$$\phi(x,y) = \tan^{-1}\left(\frac{I(x,y+1) - I(x,y-1)}{(I(x+1,y) - I(x-1,y)}\right) \tag{3.5}$$

For each bin the magnitudes are combined and normalized in one histogram, which holds 8 orientations. The final dimension of a SIFT feature descriptor is $4 * 4 * 8 = 128$. An example SIFT keypoint descriptor is illustrated in figure 3.3.

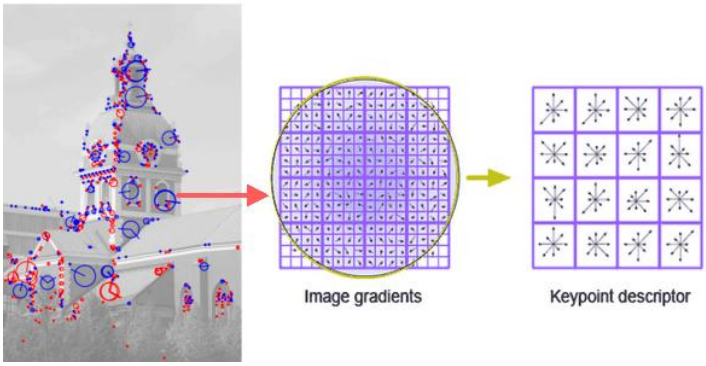

Figure 3.3: Construction of SIFT keypoint from oriented gradient histogram

3.2.2. HISTOGRAM OF ORIENTED GRADIENTS (HOG)

Keypoints in SIFT are described by image gradients in a local neighborhood. The HOG algorithm implemented in *vlfeat* is based on the work of Dalal and Triggs [7] and describes the gradient of a complete image. The first derivative or image gradient $\nabla I(x,y)$ is computed by Sobel kernels. The gradients are assigned to an orientation in the range $[0, 2\pi]$ (figure 3.4). An input image is divided into a number of prescribed sized cells (figure 3.6). Orientations are accumulated for each cell in four blocks, as is done in SIFT (figure 3.3). This results in a histogram h_d in which directions and orientations are captured. A second histogram h_u of undirected orientations of half the size is obtained by folding h_d in two. Given a HOG cell for each of the 4 blocks a normalization factor is obtained as the inverse norm of a stacked h_u histogram. Finally the h_d histograms are normalized using the 4 normalization factors, which is stored as the descriptor of one cell.

Figure 3.4: Histogram of oriented gradients

3.2.3. LOCAL BINARY PATTERNS (LBP)

The texture descriptor from Ojala et al. [19] is obtained for each image pixel. A circular equally spaced pixel neighborhood with radius R and P pixels is selected around a center pixel g_c (figure 3.5). Illumination invariance (grayscale) is achieved by subtracting the neighborhood pixels from the center pixel. The assumption is made that the difference $g_P - g_c$ is independent of g_c.

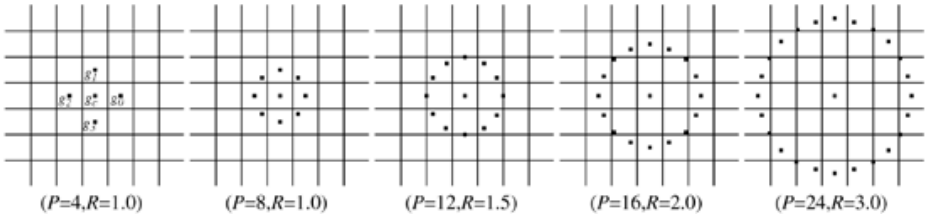

Figure 3.5: LBP descriptors with varying neighborhood sizes

The signed differences $g_P - g_c$ are evaluated to achieve illumination and rotation invariance.

$$T = t(g_c)\,t(g_0 - g_c, g_1 - g_c, ..., g_{P-1} - g_c) \tag{3.6}$$

A sign function assigns a 1 if there is a positive difference and 0 otherwise.

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x \leq 0 \end{cases} \tag{3.7}$$

Rewriting equation 3.6 gives a LBP descriptor

$$LBP_{P,R} = \sum_{P=0}^{P-1} s(g_P - g_c)2^P \tag{3.8}$$

Where the multiplication term 2^P denotes the binary position for each evaluated point P. Rotation invariance is achieved by measuring the number of transitions from $0 \rightarrow 1$ and vice versa. Ojala et al. [19] demonstrated that a good discrimination is found by looking at fundamental or uniform patterns. Uniform patterns have a maximum of two transitions and function as micro templates for spots and edges of varying curvature.

$$U(LBP_{P,R}) = |s(g_{P-1} - g_c) - s(g_0 - g_c)| + \sum_{P=1}^{P-1} |s(g_P - g_c) - s(g_{P-1} - g_c)| \tag{3.9}$$

$$LBP_{P,R}^{riu2} = \begin{cases} \sum_{P=0}^{P-1} s(g_p - g_c) & \text{if } U(LBP_{P,R}) \leq 2 \\ P+1 & \text{otherwise,} \end{cases} \tag{3.10}$$

Vlfeat implements a LBP descriptor with a local neighborhood of 3×3 pixels and 58 uniform binary patterns. The image is divided into cells of a prescribed size. A binary pattern is obtained for each pixel inside a cell and aggregated into a histogram (figure 3.6).

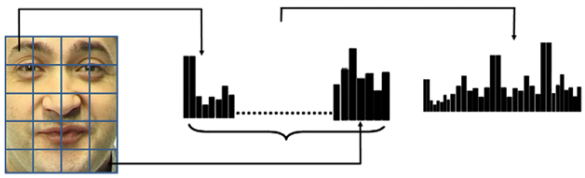

Figure 3.6: Schematic construction of HOG/LBP descriptor from image cells.

3.2.4. HUE, SATURATION AND VALUE

A simple feature descriptor is used to capture the color of an image in a histogram. RGB pixel values are transposed to the Hue, Saturation and Value (HSV) colorspace. From all pixel values in each HSV layer a histogram of predefined dimensions is constructed (figure 2.6). The 3 histograms are concatenated in combination with their mean and standard deviation to form a single feature vector.

3.2.5. NEURAL NETWORK (NN)

Vedaldi and Lenc [26] developed a toolbox for MATLAB named *MatConvNet*, which implements convolutional neural networks. The *imagenet-vgg-verdeep-16* pre-trained neural network by Simonyan and Zisserman [22] is used for image description in the object recognition benchmark (figure 3.7).

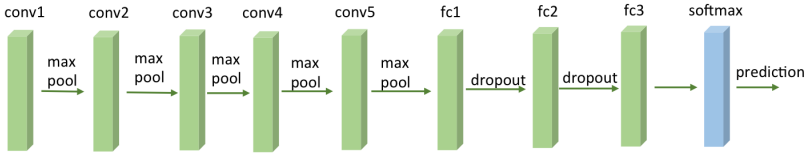

Figure 3.7: Schematic layer overview of imagenet neural network

This neural network consists of 16-layers. A stack convolutional layers with varying depth is followed by three fully connected layers. Two of those layers have 4096 channels each and the third performs a 1000-way ILSVRC classification. The final layer is a soft-max layer, which assigns a class to an input image. A feature vector is obtained from the second full connected layer. The neural network input layer requires a 224×224 RGB image.

3.3. VIEW CLASSIFICATION

The final step in object recognition is object classification as stated in section 2.1.4. The benchmark objective is evaluation of novel object models. External influences that may affect the recognition performance should therefore be reduced. For this reason object classification is done in a basic form by finding the minimal Euclidean distance (equation 3.11).

$$d_{eucl}(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{3.11}$$

Where p and q are feature vectors. HOG, LBP, HSV and NN describe a complete image by one feature vector and are called global descriptors. Finding the closest feature vector in a large database is computational demanding. In order to quickly solve this nearest-neighbor problem a kd-tree is used. This is a hierarchical structure built by partitioning the data recursively along the dimension of maximum variance.

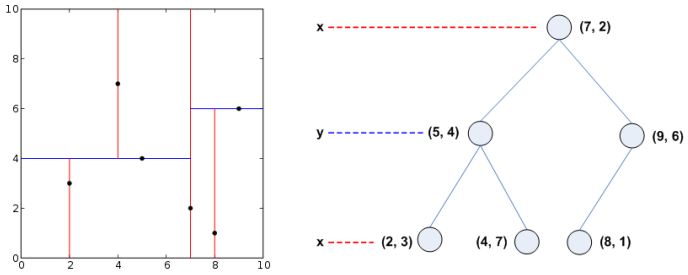

Figure 3.8: Kd-tree decomposition for the pointset (2,3), (5,4), (9,6), (4,7), (8,1), (7,2)

Figure 3.8 illustrates an example kd-tree. The median along the first dimension splits the data in two partitions at the point (7,2). From both remaining partitions median points are selected along the second dimension. A horizontal line is drawn through the points (5,4) and (9,6). This process is repeated until no points are left. A query point is classified in a maximum of 3 steps by this kd-tree, which is more efficient than evaluating each of the 8 points. Figure 3.9 shows the execution time of HOG feature matching with varying database dimensions.

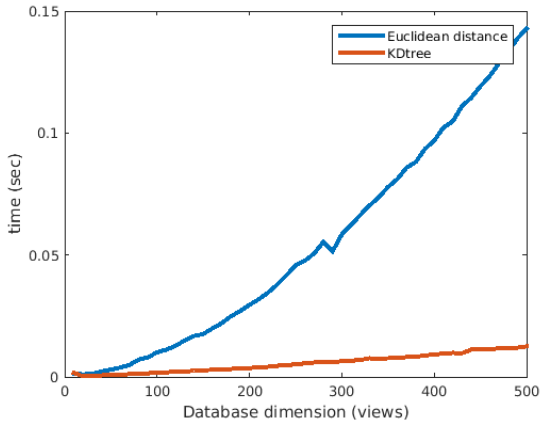

Figure 3.9: Euclidean vs KDtree HOG features matching time with varying database dimensions

SIFT is a local descriptor, which describes (multiple) edges and corners within an image. Matching is done by comparing all features from one image to the features of another image. Each feature vector of an image is matched against all feature vectors from another image using Euclidean the distance. A distance threshold determines if there is a true match between two vectors.

3.4. BENCHMARK PERFORMANCE

This section describes how the benchmark object recognition performance is evaluated. The first subsection 3.4.1 describes the construction of query object models via data manipulation. The benchmark performance results are presented in subsection 3.4.2 for single-view object recognition. Multiple experiments are conducted, which measure object/view recognition and execution time.

3.4.1. QUERY OBJECTS

The object recognition performance is determined by query objects. These query objects can't be exact copies of objects in the object dataset, because the extracted features will be identical. Query objects are therefore constructed by manipulation of the objects based on the feature properties list from section 2.1.3. Three different noise models are used for the construction of the query objects.

1. Data manipulation by adding noise to the data. Equation 3.12 shows how noise is added in MATLAB using a Gaussian function.

$$I_{x,y} = I_{x,y} + \sqrt{\sigma} * randn(1) + \mu \tag{3.12}$$

   Where randn returns a single random scalar drawn from the standard normal distribution. μ and σ are respectively the mean and variance. Figure 3.10 shows the addition of Gaussian noise with $\sigma = 0.05$ and $\mu = 0$.

2. Data manipulation via an affine transformation. A rotation matrix maps points from one frame into a second over rotation angle θ (equation 3.13). The third picture in figure 3.10 illustrates a rotation over 10 degrees.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.13}$$

3. The SOIL-47 dataset has captured each view twice under different lighting conditions (figure 3.10 last image).

Figure 3.10: Data manipulation: Original, Gaussian, Illumination, Affine

3.4.2. SINGLE-VIEW OBJECT RECOGNITION PERFORMANCE

Figure 3.11 shows a schematic overview of the object recognition benchmark. Multiple experiments are conduced for each descriptor $\{HOG, HSV, LBP, NN, SIFT\}$. Features are extracted from views in object dataset SOIL-47 or RGB-D. For fast classification a kd-tree is constructed from all extracted features. Query views are obtained by $\{gaussian, affine, illumination\}$ data manipulation. The object recognition performance is finally determined by classification of selected query views.

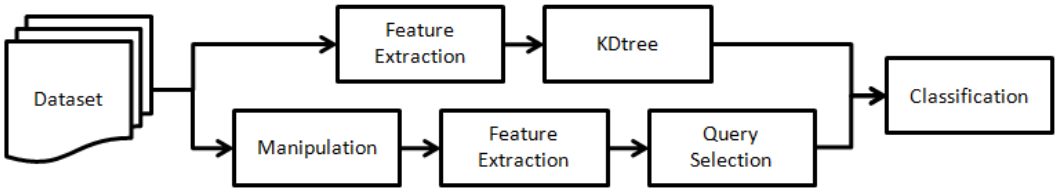

Figure 3.11: Flow diagram single-view object recognition

OBJECT RECOGNITION

Given a query object, a sequence of views is selected for object recognition. This is done in order to simulate a realistic approach, where a digital camera captures a sequence of images by moving around an object. Each query view is matched individually against the complete object dataset for object classification. Figure 3.12 shows a query sequence of three views (bottom) with corresponding best matches indicated by the arrows.

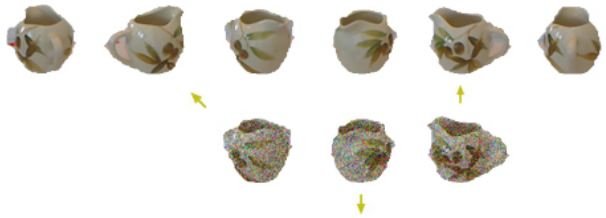

Figure 3.12: A query sequence (bottom) is matched against all object models
In this example 2 out of 3 views correctly recognize the object

Object recognition performance is measured systematically by increasing the query sequence length. The location of sequences in query objects are determined randomly. Figure 3.13 shows the results of single-view object recognition for the SOIL-47 and RGBD-dataset. The object recognition rate is determined as the individual query views, which correctly recognize an object divided over all query views.

Figure 3.13: Single-view object recognition results
The left plots are obtained from the RGB-D dataset (50 objects, 40 views) and right plots from SOIL-47 dataset (17 objects, 20 views)
The standard deviation of Gaussian datamanipulation is set to 0.05 and an affine rotation of 10 degrees is used

Quality and object diversity of both datasets have influence on the descriptor performance. Objects from SOIL-47 contain more details compared to the objects of the RGB-D dataset, therefore more SIFT keypoint are detected and a higher object recognition rate is achieved. Dataset SOIL-47 contains similar looking cereal boxes. This affects the object recognition performance of HOG and LBP descriptors, which are obtained directly from the pixel values in a small neighborhood.

Based on the benchmark results there is chosen to compare and evaluate novel multi-view object models by the RGB-D object dataset in combination with Gaussian data manipulation. The RGB-D dataset contains various household objects, which is preferred over the similar looking cereal boxes of the SOIL-47 object dataset. There is chosen for Gaussian data manipulation, because some of the feature descriptors are by design invariant to affine data manipulation.

VIEW RECOGNITION

The objective of this thesis is the development of a viewpoint dependent object model. For this reason the influence of individual views on object recognition is investigated. The number of equally spaced views in each object is determined by view discretization (figure 3.14).

Figure 3.14: Data discretization on two scales with equally spaced views

The object view spacing is increased systematically to determine the view recognition performance on different scales. Figure 3.15 shows the results for Gaussian data manipulation. The ratio of correct views/objects depicts the chance a view is correctly recognized given that the corresponding object is recognized. Both plots show a ratio decrease for small discretization angles. This indicates that the recognition of single-views is harder for small view spacing. Choosing a view spacing has influence on the execution time of the object recognition process, because it defines the number of views in each object. The optimal view spacing is determined by a high ratio correct views/objects and small discretization angle.

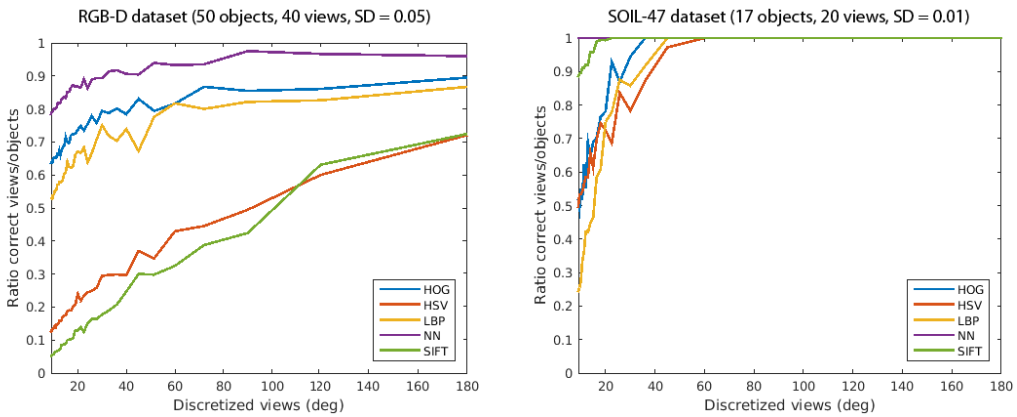

Figure 3.15: Single-view view recognition using Gaussian data manipulation

EXECUTION TIME

The descriptor matching time is obtained from the single-view object recognition experiment. Figure 3.16 shows the matching time of one query object to all objects in a dataset. There exists a linear relationship between the descriptor matching time and the query sequence size. The matching time of SIFT descriptors is higher, because each image contains multiple descriptors.

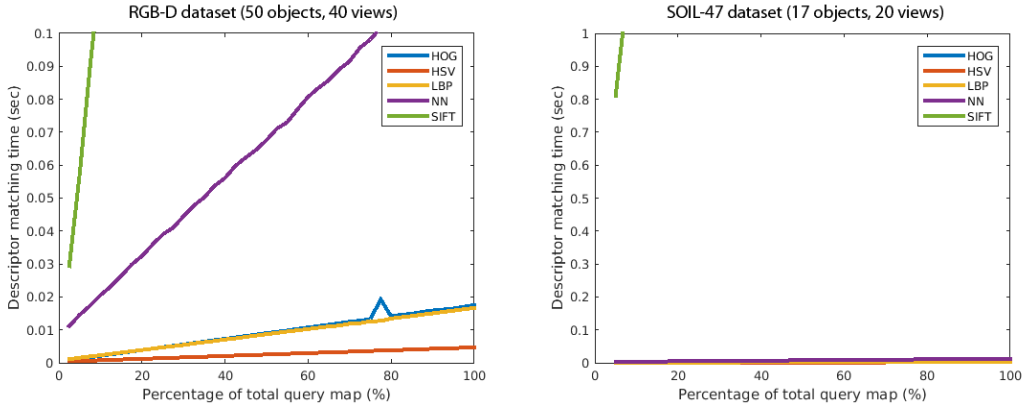

Figure 3.16: Descriptor matching time

3.5. CONCLUSION

In this chapter a benchmark is developed, which measures the recognition performance of object models for multi-view object recognition. A scientific object dataset minimizes the influence of external noise factors on object recognition. Feature descriptors based on different mathematical concepts are extracted from the data, which are classified via the Euclidean distance. A kd-tree solves the nearest neighbor problem for the global feature descriptors HOG, LBP, NN and HSV.

Query objects are obtained by adding Gaussian, affine or illumination noise to the object data. For each query object a view sequence is randomly selected. The view sequence length is hereby increased systematically, such that the object recognition rate is obtained for a specific number of views per object. The benchmark results show that the quality and object diversity of the datasets have high influence on the descriptor recognition performance. The influence of view spacing on object recognition is investigated for the development of a viewpoint dependent object model. Objects are discretized to obtain the view recognition rate for different view spacing. The results indicate that the view recognition rate decreases for small view spacing.

A novel viewpoint dependent object model based on sequence alignment is introduced in the next chapter. This model replaces the single-view object model from section 3.4.2.

4

VIEWPOINT DEPENDENT MULTI-VIEW OBJECT RECOGNITION

This chapter introduces a novel object recognition approach, where the relative spatial relation between multiple views are captured in an object model. Viewpoint dependency is subject of section 4.1. A novel viewpoint dependent object recognition algorithm based on sequence alignment is proposed in section 4.2. The conclusion of this chapter is written in section 4.3.

4.1. VIEWPOINT DEPENDENT OBJECT MODEL

Integration of viewpoint dependency in an object model is one of the design requirements as stated in section 2.3.1. A viewpoint relation describes the relative transformation, which maps points from one view into another. Figure 4.1 shows the extension of general object recognition with viewpoint dependency.

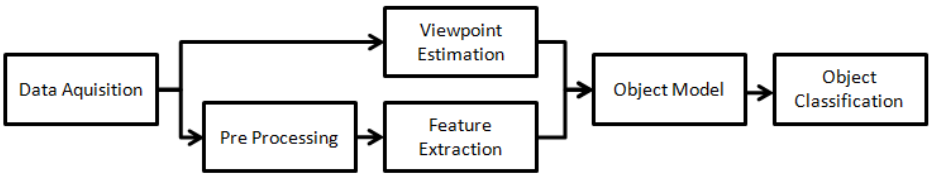

Figure 4.1: Viewpoint dependent object recognition pipeline

Viewpoints are obtained by point cloud registration or egomotion estimation, which is subject of the next chapter. For the development of a novel viewpoint dependent object model, datasets SOIL-47 and RGB-D are used. These datasets are captured systematically with known view spacing. SOIL-47 object data is acquired by a moving robot arm at intervals of approximately 9 degrees between $[0, \pi]$. For the RGB-D dataset a turntable is used, which is spun at constant speed. A video sequence is recorded for elevation angles 30, 45 and 60 degrees. Views are obtained from ground truth pose angles between $[0, 2\pi]$ by tracking red markers on a turntable. A reference pose is chosen such that pose angles are consistent across video sequences. Figure 4.2 gives a schematic impression of the viewpoint relations for objects in both datasets.

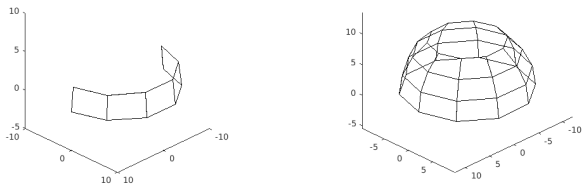

Figure 4.2: Schematic object viewpoints for SOIL-47 (left) and RGB-D dataset (right)

4.2. VIEWPOINT DEPENDENT OBJECT RECOGNITION ALGORITHM

This section introduces a novel object recognition approach using sequence alignment. Sequence alignment fundamentals are subject of the first subsection. The second subsection describes how sequence alignment can be used for viewpoint dependent object recognition.

4.2.1. SEQUENCE ALIGNMENT

The field of bioinformatics applies sequence alignment to sequences of DNA, RNA and proteins. These complex molecules consist of arranged nucleic acid sequences. The international union of pure and applied chemistry (IUPAC) Comm [6] formalized the use of nucleic acid notation. The four nucleotides commonly found in DNA are represented by the characters G, C, A and T. Regions of similarity within DNA, RNA and proteins are identified by sequence alignment, which may be a consequence of function, structural or evolutionary relationships. A sequence alignment of mammalian histone proteins is shown in figure 4.3.

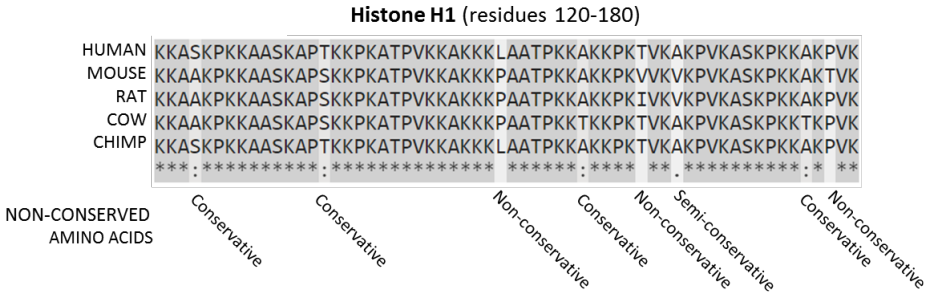

Figure 4.3: Sequence alignment of mammalian histone proteins

A local sequence alignment algorithm was first proposed by Smith and Waterman [23]. A guaranteed optimal local alignment is found via dynamic programming. Score matrix H is initialized given two sequences.

$$H(i,0) = 0, 0 \leq i \leq m, \quad H(0,j) = 0, 0 \leq j \leq n \tag{4.1}$$

Where m and n are the length of both sequences. A score is obtained for each position in matrix H based on the maximum of the following four conditions.

$$H(i,j) = max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + s(a_i, b_j) & Match/Mismatch \\ max_{k \geq 1} \{H(i-k,j) + wk\} & Gap \\ max_{l \geq 1} \{H(i,j-l) + wl\} & Gap \end{Bmatrix}, 1 \leq i \leq m, 1 \leq j \leq n \tag{4.2}$$

In which $s(a,b)$ is a similarity function based on the alphabet and w is a gap penalty. Equation 4.3 shows the score matrix H for sequences 'signal' and 'align'. Matches are rewarded with +2, mismatches -1 and gaps -0.5. The maximum condition of equation 4.2 for each position in $H(i,j)$ is indicated by an arrow in the trace back matrix T.

$$H = \begin{pmatrix} & - & s & i & g & n & a & l \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 & 0 & 0 & 2 & 1.5 \\ l & 0 & 0 & 0 & 0 & 0 & 1.5 & 4 \\ i & 0 & 0 & 2 & 1.5 & 1 & 1 & 3.5 \\ g & 0 & 0 & 1.5 & 4 & 3.5 & 3 & 3 \\ n & 0 & 0 & 1 & 3.5 & 6 & 5.5 & 5 \end{pmatrix} \quad T = \begin{pmatrix} & - & s & i & g & n & a & l \\ - & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a & 0 & 0 & 0 & 0 & 0 & \nwarrow & \leftarrow \\ l & 0 & 0 & 0 & 0 & 0 & \uparrow & \nwarrow \\ i & 0 & 0 & \nwarrow & \leftarrow & \leftarrow & \uparrow & \uparrow \\ g & 0 & 0 & \uparrow & \nwarrow & \leftarrow & \leftarrow & \uparrow \\ n & 0 & 0 & \uparrow & \uparrow & \nwarrow & \leftarrow & \leftarrow \end{pmatrix} \tag{4.3}$$

In case of local alignment the maximum value of H is selected. The optimal alignment is retrieved using the trace back matrix T. For sequences 'signal' and 'align' in equation 4.3 this becomes.

$$\begin{matrix} \_ & s & i & g & n & a & l \\ a & l & i & g & n & \_ & \_ \end{matrix} \tag{4.4}$$

4.2.2. SEQUENCE ALIGNMENT IN OBJECT RECOGNITION

In the previous subsection complexly arranged nucleic acids were represented by single characters and used for sequence alignment. Feature vectors in a viewpoint dependent object model are in similar way complex descriptions of object views. Representation of these vectors by single values is possible by evaluating their location in a kd-tree (section3.3). Algorithm 1 describes how this process is done.

```
Data: Object database
Result: Single value object database, kd-tree
begin
    for each object in database do
        for each view in object do
            Extract features using a descriptor;
        end
    end
    Construct kd-tree from all feature vectors;
    for each object in database do
        for each view in object do
            Represent view by a single value based on the feature vector location in kd-tree;
        end
    end
end
```

Algorithm 1: Single value object representation

The object recognition performance is measured using the feature vectors, kd-tree and single value representation of objects from algorithm 1. A query view sequence is represented by single values and compared to each database object for optimal alignment. For each object alignment the total Euclidean distance is calculated between feature vectors of the aligned view sequence and query view sequence. The object that contains the alignment with minimal Euclidean distance is selected for object recognition. Algorithm 2 shows the pseudo code of sequence alignment object recognition.

```
Data: Query views, output Algorithm 1
Result: Object recognition
begin
    for each query view do
        Extract features using a descriptor;
        Represent query view by a single value based on the feature vector location in kd-tree;
    end
    for each object in database do
        if if the object and query object share equal single values then
            Find optimal alignment position;
            Calculate the total Euclidean distance between the aligned (query) object feature vectors;
        else
            Evaluate next object;
        end
    end
    The object with smallest total Euclidean distance is selected for object recognition
end
```

Algorithm 2: Object recognition sequence alignment algorithm

4.2.3. ALGORITHM PERFORMANCE

A flow diagram of the proposed algorithm is illustrated in figure 4.4. The object recognition performance is evaluated using the benchmark described in section 3.4. The proposed algorithm uses global feature vectors, which are represented by single values. For this reason the local feature descriptor SIFT cannot be evaluated as there are multiple features per view.

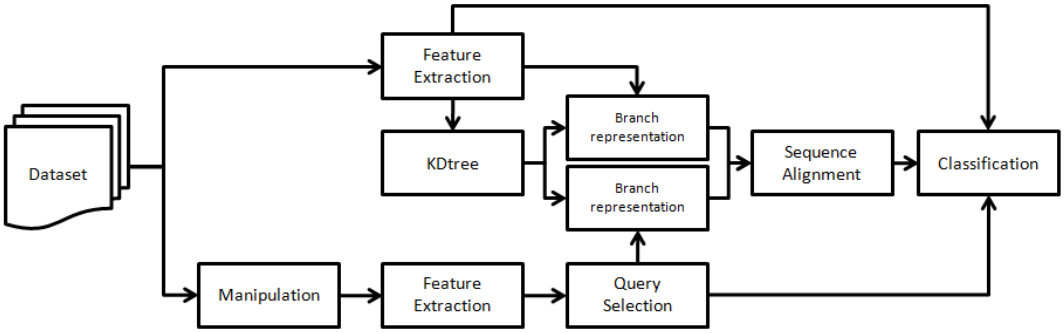

Figure 4.4: Flow diagram sequence alignment object recognition

OBJECT RECOGNITION

Sequence alignment compares two strings for optimal alignment. Objects captured at multiple elevation angles are for this reason split and evaluated at each elevation angle. When an optimal alignment is found between two view sequences the total Euclidean distance is obtained for the complete query object and database object as described in algorithm 2.

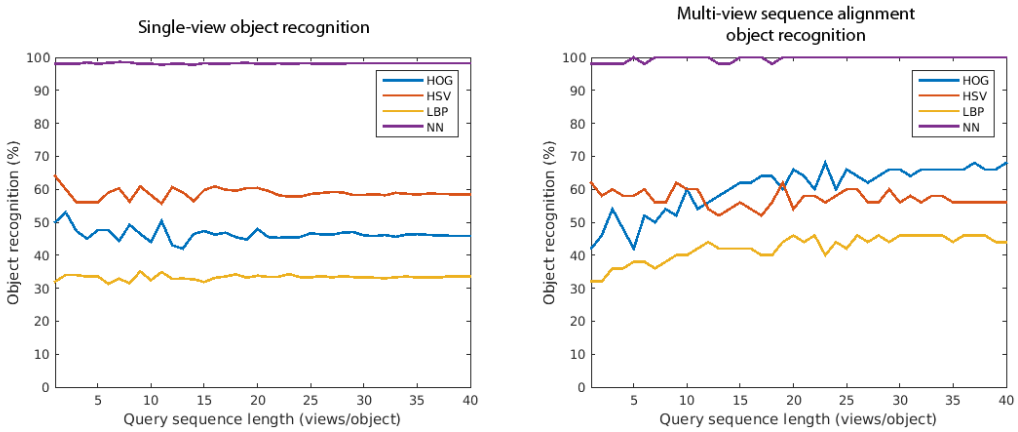

Figure 4.5: Object recognition RGB-D dataset and Gaussian data manipulation (50 object, 40 views, $\sigma = 0.01$)

Figure 4.5 shows the comparison of single-view and multi-view sequence alignment object recognition. Both plots are obtained from the same (manipulated) object data from the RGB-D dataset. Feature descriptors NN, HOG and LBP show an increased recognition performance for increasing query sequence length. In case of the color descriptor HSV the recognition performance is slightly decreased with respect to single-view object recognition. This difference can be declared by evaluating the view recognition rate shown in table 4.1. In case of HSV only 4.5% of all views are correctly recognized. Sequence alignment is only beneficial compared to single-view object recognition when views are correctly recognized.

Table 4.1: View recognition rate (40 views) per descriptor

Descriptor	HOG	HSV	LBP	NN
View recognition rate	32.4%	4.5%	15.2%	72.6%

EXECUTION TIME

The descriptor matching time for single-view and multi-view sequence alignment object recognition is shown in figure 4.6. In multi-view sequence alignment object recognition finding the optimal alignment is responsible for additional execution time.

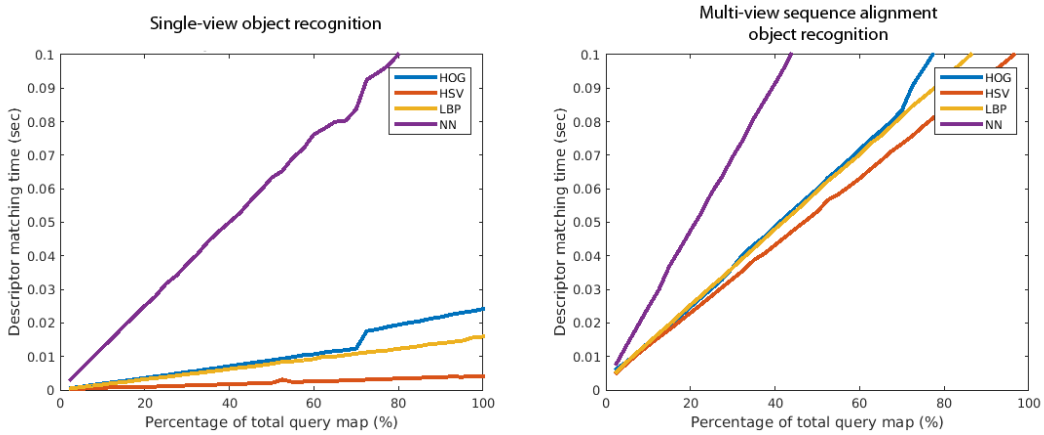

Figure 4.6: Descriptor matching time RGB-D dataset(50 object, 40 views)

LIMITATIONS

The sequence alignment algorithm as presented in this section has some limitations. Each view is represented by a different branch in a kd-tree. This might be a problem for objects with similar views from multiple viewpoints (i.e. balls, plates and bowls). Multiple branches may represent similar views, which has influence on the sequence alignment performance.

Each view is represented by only one value based on the closest feature vector in a kd-tree. The sequence alignment algorithm does not take the second or third closest feature vectors into account.

4.3. CONCLUSION

A novel object recognition algorithm is proposed based on sequence alignment. View sequences are obtained from viewpoint dependent objects and are represented by single values. Sequence alignment finds the optimal alignment between a query object and all database objects. The benchmark from previous chapter is used to obtain and compare the recognition performance. The results show a relation between the view recognition rate of the feature descriptors and the increased sequence alignment object recognition rate in comparison to single-view object recognition.

The benchmark uses an object dataset, which minimizes the influence of external noise on object recognition. The next chapter estimates the relative relation between views in order to construct a viewpoint dependent object model from data obtained by a Kinect camera.

5

VIEWPOINT ESTIMATION

In the previous chapters data acquisition is replaced by the SOIL-47 and RGB-D object datasets. This reduces the influence of external variables such as noise and unequal view distribution. The construction of novel object models from data obtained by a Kinect camera is subject of chapters 5 and 6. This chapter focuses on viewpoint estimation for the construction of a viewpoint dependent object model. The experimental setup is denoted in section 5.1. The alignment of depth and color data via depth registration is subject of section 5.2. Viewpoint estimation methods are the subject of section 5.3. In section 5.4 a method based on egomotion estimation is implemented. The performance of the algorithm is measured in section 5.5. Finally a conclusion is denoted in section 5.6.

5.1. EXPERIMENTAL SETUP

A movie still of the experimental setup is shown in figure 5.1. One object is placed at the center of an empty surface to avoid the use of object tracking and detection algorithms as stated in the design requirements list from section 2.3.1. Depth and RGB data are obtained by moving a hand held Kinect camera around the object. The gathered data is used to construct a viewpoint dependent object.

Figure 5.1: Object model construction of a banana

5.2. DEPTH REGISTRATION

The Microsoft kinect camera captures data via an infrared and RGB camera. Misalignment between depth and color occurs as consequence of different lens properties and camera locations. An example of misalignment is shown in the left plot of figure 5.2. Depth registration aligns data from both cameras given intrinsic and extrinsic camera parameters. A linear relation exists between the disparity (depth) image and inverse depth.

$$q(d_{uv}) = k_1 d_{uv} + k_2, \quad z = \frac{1}{q(d_{uv})} \tag{5.1}$$

Where u, v are image coordinates and $k_1 = -0.00307$ and $k_2 = 3.33095$ are based on experimental results conducted by Khoshelham and Elberink [15]. The intrinsic camera matrix maps points from world to image coordinates.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

In which f is the focal point and c the principle point offset of the camera. Using the inverse intrinsic matrix a point cloud is constructed from image coordinates and depth information (equation 5.1).

$$pc_{ir} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = z \left(K_{ir}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) \tag{5.3}$$

Transformation of the point cloud from the infrared sensor to the RGB camera is done via the extrinsic rotation and translation matrix.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

The translation vector in this rotation matrix should indicate that both sensors are not located at the same position. Multiplication of the extrinsic matrix by the point cloud visualizes the point cloud in RGB camera perspective.

$$pc_{rgb} = R_{(ir \to rgb)} pc_{ir} \tag{5.5}$$

Finally the inverse depth point cloud is projected on the RGB camera sensor to obtain registered depth data.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K_{rgb} \left(\frac{pc_{rgb}}{pc_{rgb,z}} \right) \tag{5.6}$$

For depth registration the default (factory) intrinsic and extrinsic camera parameters are used. The right plot of figure 5.2 shows depth and color after registration.

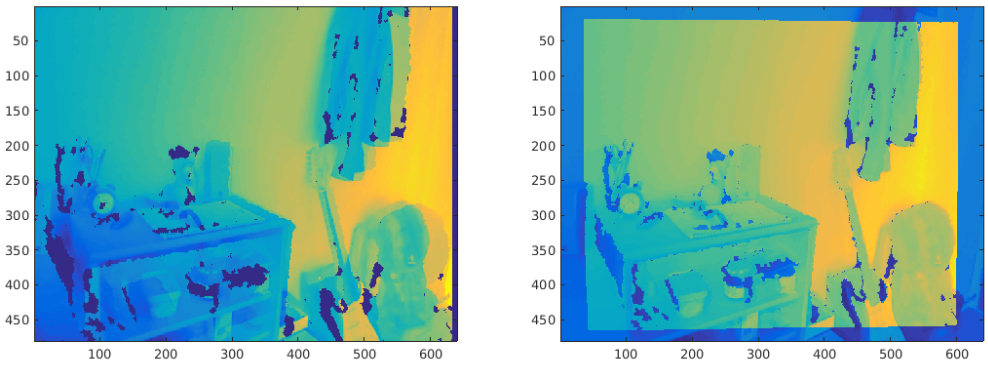

Figure 5.2: Depth registration made visual by projection of depth data over RGB sensor data
Left: unregistered, Right: registered

5.3. VIEW REGISTRATION METHODS

The alignment of an image pair with shared visual and depth information is called view registration. One of the design requirements in section 2.3.1 states the geometry of an object does not change over time. This implies that view registration is obtained via a rigid transformation with rotation matrix R and translation vector T (figure 5.3). A global iterative closest point registration technique is subject of the first subsection. Subsection 5.3.2 denotes registration techniques, which account for small rotations and translations.

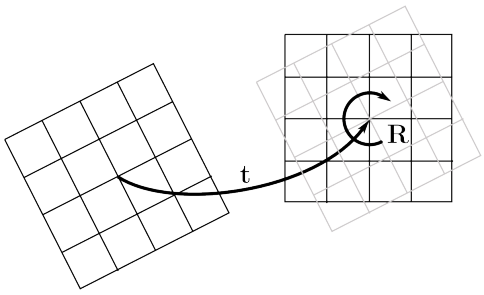

Figure 5.3: Rigid view transformation

5.3.1. GLOBAL VIEW REGISTRATION

Registration of point clouds via iterative closest point (ICP) was first described by Besl and McKay [3]. The ICP algorithm works by minimizing the mean squared error of two arbitrary rotated and translated point clouds. The registration is done in 5 steps listed below.

1. Translate the centroid of both point clouds to the origin
2. Compute the point cloud covariance matrix
3. Apply singular value decomposition to obtain a rotation and translation estimate
4. Update point set with obtained rotation and translation estimate
5. Terminate iteration when the update falls below a threshold, else repeat the algorithm

The execution time of ICP is slow, because all points have to be evaluated. Rusinkiewicz and Levoy [20] improved the algorithm by selecting image features (section 2.1.3). The selected features in two point clouds are matched and given a distance dependent weight. An optimal point cloud registration is found by ICP as described above.

5.3.2. VISUAL ODOMETRY

Consecutive views do not differ much when a video sequence is analyzed frame by frame. In example only small rotations δR and translations T occur in small time Δt intervals. View registration of a video sequence gives a 6 DoF estimate of the camera position and orientation relative to the scene. In computer vision this is known by visual odometry or egomotion estimation. The paper of Fang and Zhang [9] gives an evaluation of RGB-D visual odometry methods. The methods are divided in three categories according to what kind of data mainly is used. Image-based, depth-based and hybrid-based methods.

IMAGE-BASED VISUAL ODOMETRY

Huang et al. [13] uses sparse image features (i.e SIFT, SURF) to find correspondences between consecutive frames. Depth information of the feature locations is used to obtain a transformation matrix by minimizing the re-projection error (figure 5.4). Kerl et al. [14] proposed an alternative dense feature approach, which uses all pixels within a frame to estimate the transformation matrix. This method assumes a world point observed by two frames has the same brightness. The goal is obtaining a transformation that best satisfies the photo-consistency constraint over all pixels.

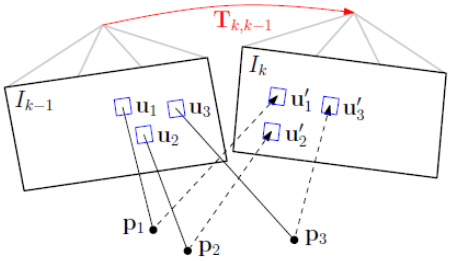

Figure 5.4: Optimization by minimization of re-projection error

DEPTH-BASED VISUAL ODOMETRY

An extension of the sparse feature based method detects and extracts planes, lines or points. Tang et al. [24] uses these depth features to estimate a transformation matrix by minimizing the re-projection error between features of two consecutive frames. Registration of a dense point cloud is possible as described earlier by the ICP algorithm.

HYBRID-BASED METHODS

In hybrid-based methods an initial registration guess is obtained from sparse image features. Dense point registration techniques are used to refine the registration (Andreasson and Stoyanov [1], Dryanovski et al. [8]).

5.4. EGOMOTION ESTIMATION

View registration based on global ICP always finds the optimal alignment of two point clouds. Visual odometry estimates small rotations and translations by linearization of the motion model. This introduces small motion errors, which are accumulated across views. Due to the fact that the execution time of a linearized model is faster, there is chosen to implement a visual odometry algorithm. In this section a lightweight sparse visual odometry image-based method is introduced, partly based on the work done by Lui et al. [18].

The first step is selection of points in 2 to be aligned frames. Keypoints are extracted by approximation of the image Laplacian as denoted in section 2.1.3. Points can either be matched or tracked across multiple frames to form point pairs. The algorithm of Lui et al. [18] obtains egomotion by considering small rotations and translations, such that $sin(\theta) = \theta$ and $cos(\theta) = 1$. The relationship between matched features in consecutive frames is expressed as

$$p_{t+1} = [\delta R|\delta T]^{t+1} p_t \tag{5.7}$$

Where $M = [\delta R|\delta T]$ is the 6DoF small rotation and translation approximation model

$$M = \begin{bmatrix} 1 & -\theta_z & \theta_y & t_x \\ \theta_z & 1 & -\theta_x & t_y \\ -\theta_y & \theta_x & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

The motion model in equation 5.7 is rewritten using linear velocities. The matrix Jacobian is obtained from parital derivatives of M with respect to the unkown motion parameters $\Theta = (t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$.

$$p_{t+1} = J\hat{\Theta} + p_t \tag{5.9}$$

With point pair Jacobian

$$J_p = \begin{bmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{bmatrix} \tag{5.10}$$

The total Jacobian is constructed by stacking each point pair on top of each other such that $J = [J_{p,1}, J_{p,2}, ..., J_{p,n}]$. Rewriting equation 5.9 gives an estimation of the motion parameters.

$$\hat{\Theta} = (J^T J)^{-1} J^T \begin{bmatrix} p_{1,t+1} - p_{1,t} \\ p_{2,t+1} - p_{2,t} \\ ... \\ p_{n,t+1} - p_{n,t} \end{bmatrix} \tag{5.11}$$

Where $(J^T J)^{-1} J^T$ is the pseudo inverse of J. From the estimated motion parameters equation 5.7 is updated and iterated until the projection error falls below a threshold

$$p_t - [\delta R|\delta T]^{t+1} p_t \leq threshold \tag{5.12}$$

The least squares solution presented above is not robust in the presence of outliers. Figure 5.5 shows a histogram of absolute point pair distances after one iteration. A M-estimator is used to reduce the influence of outliers on motion estimation by assigning weights to each point pair.

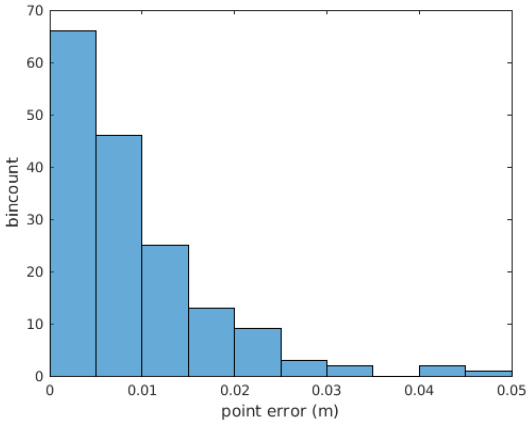

Figure 5.5: Example histogram of point pair distances after first iteration

The absolute distance of a point pair is calculated via

$$\Delta r = \sqrt{\sum \left(\begin{bmatrix} x_{t+1} & y_{t+1} & z_{t+1} \end{bmatrix}^T - \begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^T \right)^2} \tag{5.13}$$

The standard deviation σ of all point pair distances is calculated and used to assign weights to each individual point pair

$$W = 1 - \frac{\Delta r^2}{\sigma^2 + \Delta r^2} \tag{5.14}$$

Adding the weights to equation 5.11 gives

$$\hat{\Theta} = (J^T W J)^{-1} J^T W \begin{bmatrix} p_{1,t+1} - p_{1,t} \\ p_{2,t+1} - p_{2,t} \\ \cdots \\ p_{n,t+1} - p_{n,t} \end{bmatrix} \tag{5.15}$$

The camera's pose at T^{t+1} relative to its starting pose is updated from its previous pose, T^t by some small incremental motion described by $M(\hat{\Theta})^{t+1}$ with $T^0 = I$ a 4x4 identity matrix.

$$T^{t+1} = M(\hat{\Theta})^{t+1} T^t \tag{5.16}$$

This equation expresses frame T^{t+1} in the coordinates of T^t. The camera location in world coordinates is obtained by the matrix inverse $(T^{t+1})^{-1}$.

5.5. PERFORMANCE

The accuracy of the egomotion algorithm is measured by comparing the estimated camera motion to the real camera motion. A rope is attached to the Kinect camera and a fixed point in the scene, which introduces a constraint on the camera motion. Figure 5.6 shows the estimated motion versus the real camera motion for repeated movement over $\pi/4$. Violation of the linearized motion model by fast or complex camera movements introduces small motion errors. These errors are accumulated in each frame and causes deviation of the estimated motion compared to the real camera motion. Another influence on the accuracy of egomotion is the number of evaluated point pairs in each frame.

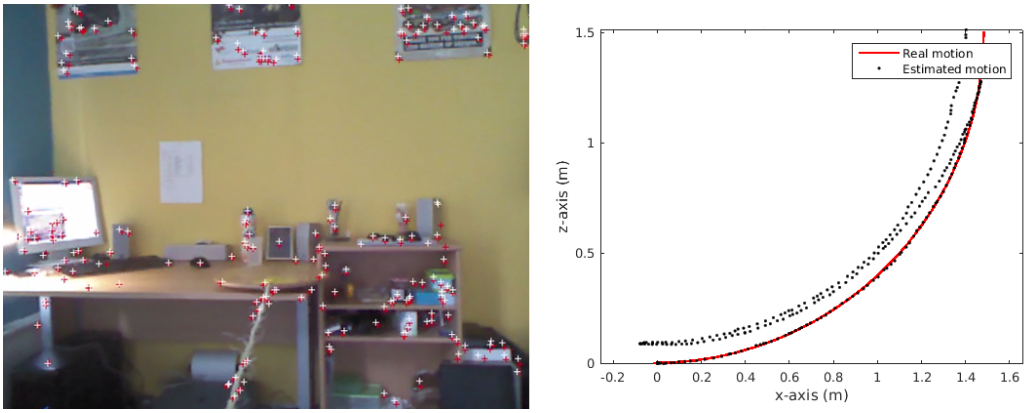

Figure 5.6: Estimated egomotion with a circular constraint

Viewpoint dependent objects will be created from the estimated egomotion. The experimental setup described in section 5.1 is used to reproduce objects from the RGB-D object dataset. A hand held Kinect camera is moved around an object on three different elevation angles. The estimated egomotion from this experiment is visualized in figure 5.7. This figure clearly shows three circles on separate heights, despite the accumulated motion error. The results indicate that it is possible to construct a viewpoint dependent object model similar to the models from the RGB-D object dataset.

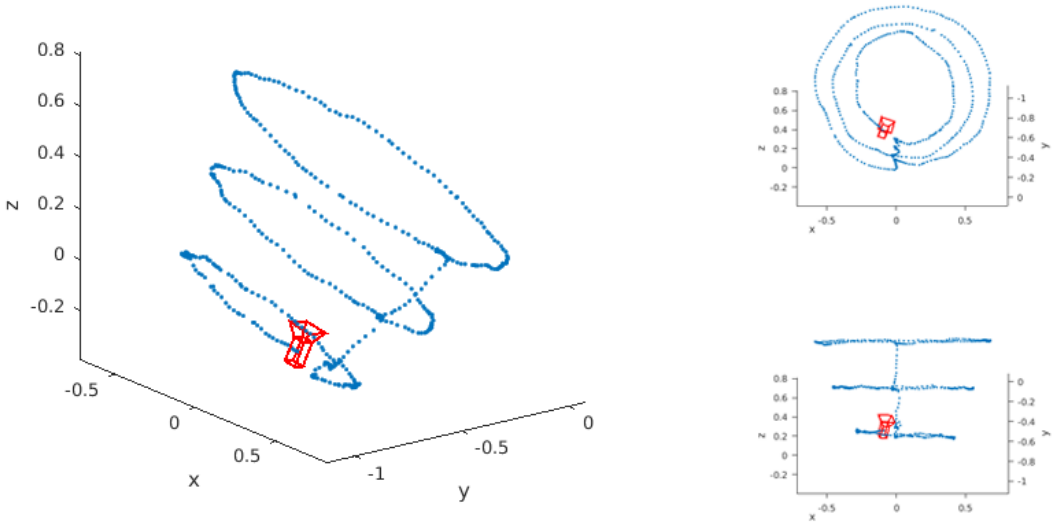

Figure 5.7: Egomotion estimation of three rotations on different heights

5.6. CONCLUSION

Depth and color data from multiple camera sources are aligned via the intrinsic and extrinsic camera parameters. View registration aligns two consecutive frames with shared visual and depth information. A visual odometry method is implemented, where sparse image features are used to find a 6 DoF relation between consecutive frames. Linearization of the motion model introduces small estimation errors, which are accumulated in each frame. This causes a deviation of the estimated motion with respect to the real camera motion. An experimental result showed that despite the accumulated motion errors it is possible to create a viewpoint dependent object. Object segmentation and integration of visual odometry in object data is the subject of the next chapter.

6

INTEGRATION VISUAL ODOMETRY IN OBJECT RECOGNITION

Viewpoint dependent object models are constructed from the captured data and estimated egomotion. Object segmentation is subject of section 6.1. Section 6.2 describes how objects are constructed by initial alignment of the camera with respect to the ground plane. The object recognition performance of novel captured object data is measured in section 6.3. Final section 6.4 denotes a conclusion.

6.1. OBJECT SEGMENTATION

Removal of irrelevant background data is done by extracting the (dominant) ground plane from the point cloud data. A segmented object remains, when only points above the ground plane are evaluated. A plane is defined by the following equation.

$$n_x x + n_y y + n_z z + h = 0 \tag{6.1}$$

Where n is the surface normal and h the plane height. Random Sample Consensus (RANSAC) described by Fischler and Bolles [10] is the most used algorithm, which fits a plane to point cloud data. Plane parameters are estimated by random sampling of point cloud data. A brief explanation of RANSAC is stated below.

1. Random subset of points are selected from an input point cloud
2. Plane parameters are estimated by fitting a plane model to this subset
3. All points in the point cloud are evaluated whether they fit the estimated plane model or not
4. The above steps are repeated, until enough points fit the model

Multiple RANSAC iterations are required in order to find the dominant plane in one frame. One of the design requirements stated that the object recognition algorithm should be performed in run-time. For this reason a faster plane detection approach by Holz et al. [12] was implemented. This algorithm operates directly on the plane equation by segmentation in normal space followed by refinement in distance space.

6.1.1. INITIAL SEGMENTATION IN NORMAL SPACE

A surface normal is obtained for each 3D point p by taking the cross product of 2 vectors tangential to the surface.

$$n(u,v) = (p(u+1,v) - p(u-1,v)) \times (p(u,v+1) - p(u,v-1)) \tag{6.2}$$

Where point p has image coordinates u, v. The tangential vectors are obtained between the left and right and between the upper and lower neighboring pixels. Convolution with a smoothing filter reduces the influence of noise on the normal space.

Based on a predefined grid size, s edges $e = 1, 2, ..., s$ are assigned to data in normal space ranging from -1 to 1. The normal space is mapped to a 3D voxel grid.

$$v = e_x + (e_y - 1)s + (e_z - 1)s^2 \tag{6.3}$$

Initial clusters are formed by voxels containing (multiple) surface normals. An initial guess of the dominant plane orientation is found by selecting the voxel with the highest normal count. The average surface normal of each initial cluster is compared to the average surface normal of the dominant plane by calculating the vector distance.

$$d = \sqrt{\sum (d_{avg} - c_{avg})^2} \tag{6.4}$$

Where d_{avg} and c_{avg} are the average surface normals. When the distance falls below a threshold two neighboring voxels are merged. The dominant plane orientation $(n_x, n_y, n_z)^T$ is found as the average surface normal of the merged voxels. The left plot in figure 6.1 shows segmentation in normal space.

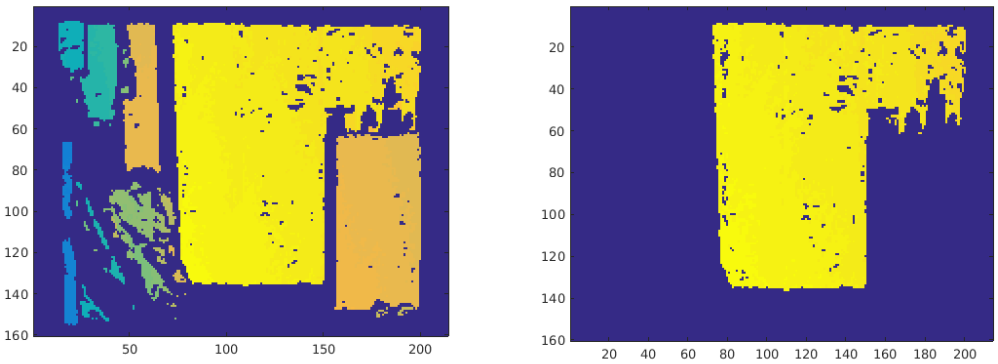

Figure 6.1: Left: plane segmentation in normal space, Right: refinement in distance space

6.1.2. SEGMENTATION REFINEMENT IN DISTANCE SPACE

The remaining unsolved variable in equation 6.1 is the plane height, which is solved by refinement in the distance space. The points after initial segmentation represent multiple planes with similar normal orientation. For each point the shortest distance is calculated to a plane through the origin with a dominant surface normal found in the previous step.

$$h = p_x n_x + p_y n_y + p_z n_z \tag{6.5}$$

A histogram with edges of 0.025m is created from the initial segmented data, which ranges from 0 to 2 meter. The highest edge count of the histogram is selected as the dominant plane height (figure 6.1). Segmented object data is obtained by evaluating all points in front of the found dominant plane.

$$p_x n_x + p_y n_y + p_z n_z - h < -0.025 \tag{6.6}$$

The resulting object mask can contain noise, which is reduced by the morphological filers opening and border fill. Figure 6.2 shows segmented object data results.

Figure 6.2: Segmented object data representing bananas, milk carton and a shoe

6.2. VIEWPOINT DEPENDENT OBJECT MODEL

The segmented object data is used in combination with the estimated egomotion to construct viewpoint dependent objects. The egomotion estimation as described in section 5.4 is initialized with a 4x4 identity matrix. This implies no rotations or translations from camera body perspective. In other words the camera is located at the center of the world frame. Alignment of the dominant plane normal to the world y-axis, gives the relative camera orientation with respect to the plane. A schematic representation of this process is shown in figure 6.3.

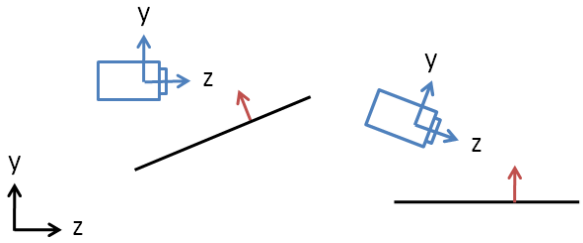

Figure 6.3: Camera orientation before and after alignment of the ground plane with the world y-axis

The angle between the world y-axis and plane normal is obtained via

$$\phi = \arccos\left(\begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T \cdot \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T\right) \tag{6.7}$$

Construction of a rotation axis by taking the cross product of two vectors

$$v = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T \times \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \tag{6.8}$$

The Rodriguez rotation formula is used to compute a rotation matrix of the rotation ϕ over axis v

$$r = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \sin(\phi) \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} + (1 - cos(\phi)) \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}^2 \tag{6.9}$$

The initial camera position is set as the height between plane and camera as described in subsection 6.1.2. The alignment is completed by replacement of the initial motion matrix with the aligned motion matrix.

$$T^0 = \begin{bmatrix} r & r & r & 0 \\ r & r & r & h \\ r & r & r & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

The viewpoint variables azimuth and elevation are easily obtained from the estimated egomotion matrix. Elevation is the angle between camera y-axis and the world y-axis, azimuth is the camera rotation angle around the world y-axis. The elevation and azimuth angles are discretized for the construction of object models at respectively 10 and 20 degrees. Figure 6.4 shows the result of a captured object model.

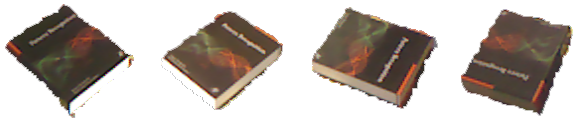

Figure 6.4: Hand held Kinect object model data
Elevation angle 40-50, Azimuth angles 0, 90, 180, 270

6.3. OBJECT RECOGNITION PERFORMANCE

The experimental setup from section 5.1 is used to capture RGB and depth data by moving a hand held Kinect camera around an object. Viewpoint dependency is achieved by finding the relative viewpoint with respect to the object via egomotion estimation. Segmented object data is stored at discretized viewpoint angles. Figure 6.5 shows the schematic overview for the construction of novel viewpoint dependent object model.

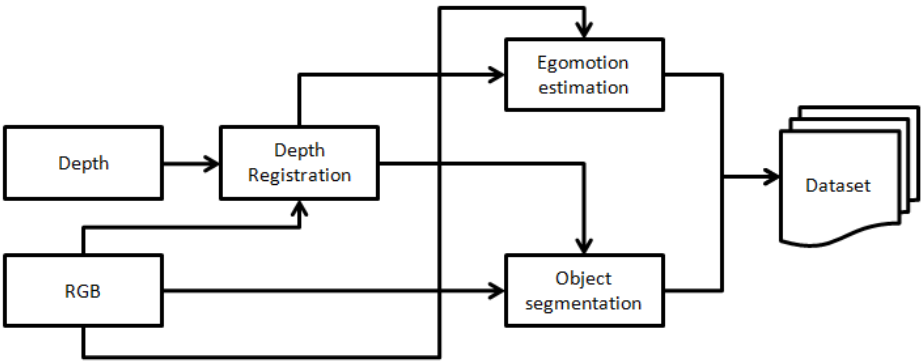

Figure 6.5: Flow diagram object dataset construction

Two object datasets are constructed to determine the object recognition performance. Each object is captured after which the orientation of that object is manually changed in order to construct the query object. The first object dataset differs from the second query object dataset by change in lighting and discretized view distribution (figure 6.7). The elevation angle is kept equal across all objects for consistency. Keeping an equal elevation angle while walking around an object with a hand held camera appeared to be difficult, therefore a fixed elevation angle is chosen instead of multiple. Specifications of the (query)dataset are stated below. The average frame rate of depth registration, egomotion estimation and object segmentation is 4.5.

- Dataset contains household objects (figure 6.6)
- 36 objects
- 18 views per object, azimuth discretization of 20 degrees
- Fixed elevation angle between 40 and 50 degrees
- Captures 360 degrees
- Dataset captured twice by manual change of object orientation

Figure 6.6: Selection of household objects from the object dataset

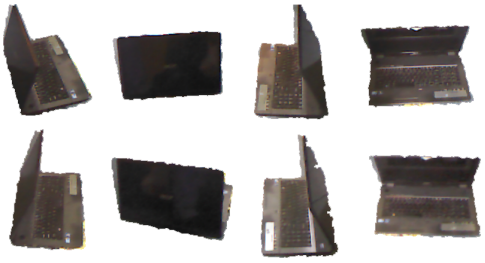

Figure 6.7: Unequal view distribution and lighting noise in an (query) object model after manual rotation

The object recognition performance is evaluated using the benchmark described in section 3.4. Figure 6.8 shows the result of single-view versus multi-view sequence alignment object recognition. The obtained results are similar to the object recognition performance found in section 4.2.3. Unequal view distributions in the (query) objects are caused by small errors in the estimated egomotion and view discretization. Changing the orientation of an object for the construction of a query object introduces lighting noise. Despite these noise factors an increase in object recognition is found for the sequence alignment method. The influence of view recognition on the results can't be investigated, because the orientation of objects are arbitrary changed for the construction of query object models.

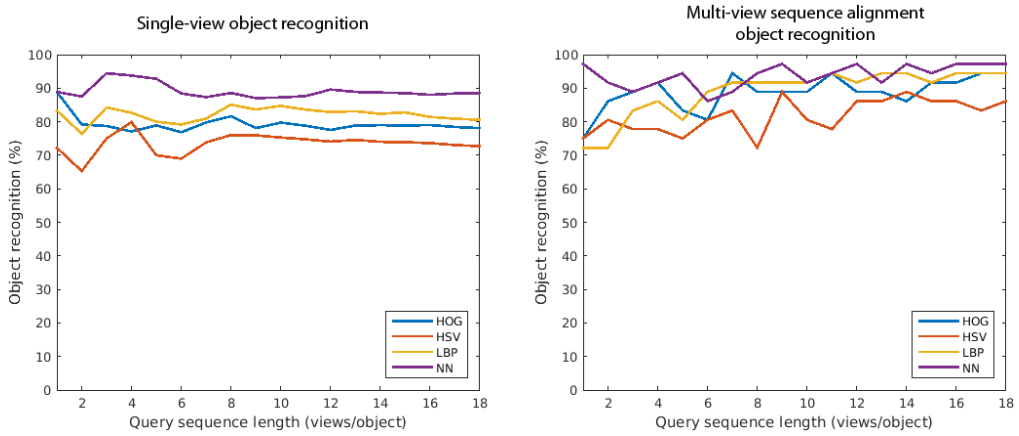

Figure 6.8: Object recognition manual captured datasets

6.4. CONCLUSION

A viewpoint dependent object is constructed from estimated egomotion. Dominant plane parameters are obtained by segmentation of depth data in normal space followed by refinement in distance space. The dominant plane orientation is used for initial camera alignment and removal of irrelevant background data. The object recognition performance is determined from two object datasets, where object variation occurs by lighting changes and unequal view distributions. The results show that multi-view sequence alignment has a higher object recognition rate compared to single-view object recognition. This result is similar to what is found in section 4.2, where the systematic captured RGB-D dataset is used. In conclusion the proposed multi-view viewpoint dependent object model based on sequence alignment has a higher object recognition rate compared to single-view object recognition.

7

CONCLUSION AND RECOMMENDATIONS

The objective of this thesis is the development of a viewpoint dependent object model for multi-view object recognition. This is done based on the object recognition pipeline presented in chapter 2 (figure 2.1). For data acquisition a Kinect camera is used, which captures RGB and depth data. Objects are segmented from the background and described mathematically by extracting features. The feature vectors are classified in order to determine the object recognition performance. The field of object recognition is very broad, therefore a list of design requirement is defined beforehand.

Exisiting multi-view object models include CAD, Potemkin, Canonical Parts and Hypergraph. The CAD and Potemkin models need supervision and are complex to create. Both models use 2D projection to obtain synthetic views for object recognition. The Canonical parts, Hypergraph and Aspectgraph models don't need supervision, but have local or lack viewpoint dependency. Different object data is used for the validation of each object model, which makes it hard to compare the object recognition performance.

In chapter 3 a benchmark is created in order to evaluate and compare novel object recognition models for multi-view object recognition. A systematical approach is preferred, which minimizes the influence of external noise and unequal view distribution. For that reason data acquisition is replaced by the systematic captured object datasets RGB-D and SOIL-47. Feature descriptors from different categories are used to represent the object data mathematically. Edges, corners, texture, shape and color are described by SIFT, HOG, LBP, HSV and NN. Classification of query objects is done by finding the object with minimal Euclidean distance. The query objects are obtained by adding Gaussian, affine or illumination noise to the object data. A sequence of views is randomly selected from each query object to obtain the single-view object recognition performance. The benchmark results show that the quality and object diversity of both datasets have high influence on the descriptor recognition performance.

A novel viewpoint dependent object recognition algorithm based on sequence alignment is introduced in chapter 4. Each view is represented by a single value based on the branch position in a kd-tree. Sequence alignment finds the optimal alignment between a query view sequence and all objects. The benchmark results show a relation between the percentage recognized views of a feature descriptor and the improved sequence alignment object recognition rate in comparison to single-view object recognition.

In chapters 5 and 6 novel object data captured by a hand held Kinect camera is used to construct viewpoint dependent objects. RGB and depth data from different sources are registered via the intrinsic and extrinsic camera parameters. Viewpoint dependency is achieved by finding the 6DoF relation between consecutive frames. A visual odometry method is implemented, where sparse features are used to obtain the camera egomotion. Experimental results showed that despite the accumulation of small motion errors it is possible to create a viewpoint dependent object model.

Equations of the dominant ground plane are obtained by segmentation of depth data in normal space followed followed by refinement in distance space. The plane parameters are used to segmented object data and align the initial Kinect camera orientation to the ground plane for the construction of a viewpoint dependent object model. For each object an object is constructed after which the orientation is manually changed to obtain a second query object model. The object recognition performance is determined from the two object datasets, where object variation occurs by lighting changes and unequal discretized view distributions. The benchmark results show that viewpoint dependent multi-view sequence alignment has a higher recognition rate compared to single-view object recognition.

Recommendations for further research are found in the limitations of the proposed sequence alignment algorithm. Global features descriptors are represented by a single value based on the branch position in a kd-tree. The second and third closest feature vector are not evaluated, which might increase the object recognition performance.

All views of an object are used for the construction of a kd-tree. Objects with similar views from multiple viewpoints such as balls or plates are assigned different branch values for each view. A solution might be found by pruning the kd-tree or merging of similar view descriptors.

BIBLIOGRAPHY

[1] Andreasson, H. and Stoyanov, T. (2012). Real time registration of rgb-d data using local visual features and 3d-ndt registration. In *SPME Workshop at Int. Conf. on Robotics and Automation (ICRA)*.

[2] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer.

[3] Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics.

[4] Burianek, J., Ahmadyfard, A., and Kittler, J. (2000). Soil-47, the surrey object image library. *Centre for Vision, Speech and Signal processing, Univerisity of Surrey.[Online]. Available: http://www. ee. surrey. ac. uk/Research/VSSP/demos/colour/soil47*.

[5] Chiu, H.-P. (2009). *Models for multi-view object class detection*. PhD thesis, Massachusetts Institute of Technology.

[6] Comm, I.-I. (1970). Abbreviations and symbols for nucleic acids, polynucleotides, and their constituents. *Biochemistry*, 9(20):4022–4027.

[7] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

[8] Dryanovski, I., Valenti, R. G., and Xiao, J. (2013). Fast visual odometry and mapping from rgb-d data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2305–2310. IEEE.

[9] Fang, Z. and Zhang, Y. (2015). Experimental evaluation of rgb-d visual odometry methods. *International Journal of Advanced Robotic Systems*, 12.

[10] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[11] Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. (2012). Gradient response maps for real-time detection of textureless objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):876–888.

[12] Holz, D., Holzer, S., Rusu, R. B., and Behnke, S. (2011). Real-time plane segmentation using rgb-d cameras. In *RoboCup 2011: robot soccer world cup XV*, pages 306–317. Springer.

[13] Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2011). Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, volume 2.

[14] Kerl, C., Sturm, J., and Cremers, D. (2013). Robust odometry estimation for rgb-d cameras. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3748–3754. IEEE.

[15] Khoshelham, K. and Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454.

[16] Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE.

[17] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee.

[18] Lui, W. L. D., Tang, T. J. J., Drummond, T., and Li, W. H. (2012). Robust egomotion estimation using icp in inverse depth coordinates. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1671–1678. IEEE.

[19] Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987.

[20] Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE.

[21] Savarese, S. and Fei-Fei, L. (2007). 3d generic object categorization, localization and pose estimation. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

[22] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[23] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197.

[24] Tang, T. J. J., Lui, W. L. D., and Li, W. H. (2011). A lightweight approach to 6-dof plane-based egomotion estimation using inverse depth. In *Australasian Conference on Robotics and Automation*.

[25] Vedaldi, A. and Fulkerson, B. (2008). VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/.

[26] Vedaldi, A. and Lenc, K. (2015). Matconvnet – convolutional neural networks for matlab.

[27] Wohlkinger, W. and Vincze, M. (2011). Ensemble of shape functions for 3d object classification. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2987–2992. IEEE.

[28] Zhang, D. and Lu, G. (2004). Review of shape representation and description techniques. *Pattern recognition*, 37(1):1–19.

[29] Zhou, D., Huang, J., and Schölkopf, B. (2006). Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, pages 1601–1608.