Feature Regression for Continuous Pose Estimation of Object Categories

Von der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades

Doktor-Ingenieur

(abgekürzt: Dr.-Ing.)

genehmigte

Dissertation

von

M.Sc. Michele Fenzi

geboren am 13. November 1982

in Prato, Italien

2016

1. Referent:

Prof. Dr.-Ing. Jörn Ostermann
Gottfried Wilhelm Leibniz Universität Hannover, Germany

2. Referent:

Prof. Dr. Tinne Tuytelaars
Katholieke Universiteit Leuven, Belgium

Tag der Promotion:

20.11.2015

Abstract

Continuous pose estimation of object categories has recently become very popular thanks to its application in fields like autonomous driving and robotics. When the number of objects becomes large or not all objects are known in advance, it becomes necessary to develop methods to solve the pose estimation problem for a generic object of a given class. However, the generality of the problem makes this task much more challenging, as difficulties related to the variability in appearance and geometry arise. Many works propose solutions that rely on 3D information, with the underlying assumption that only the 3D geometry of the class allows for a continuous estimation of the pose. However, 3D-based methods depend on the availability of 3D training data or the feasibility of 3D reconstruction. Other authors have proposed methods based only on 2D training information. However, the price to pay to avoid an explicit 3D class representation is very often an estimation of the pose limited to a few discrete viewpoints. In this thesis, we propose a solution that bridges this gap by devising an innovative method that provides a real-valued pose by relying only on a 2D-based class model.

The key intuition behind the method proposed in this thesis is that the variation in the components of a feature descriptor is smooth as a function of the viewpoint change. On this basis, we first introduce our basic block, the *generative feature model* (GFM), *i.e.*, a regression function that predicts the appearance of a patch under a given viewpoint. We show how to aggregate the large amount of generative feature models that results when many training objects are considered. As each cluster is a collection of GFMs, we illustrate how to combine them to build a *generative cluster model* (GCM). Finally, we embed our class representation based on GCMs in a probabilistic framework in order to compute the posterior distribution of the object pose in a single image. Secondly, we propose to enrich our approach by introducing geometry into the matching process. We reformulate the matching step as a graph matching problem, where the spatial arrangement between matching features is taken into account. We show that this is beneficial for the accuracy of the proposed method. Finally, we illustrate how to extend our algorithm to video sequences in order to apply temporal constraints. We build a graph by sampling from the posterior distributions estimated at each frame and we find the pose trajectory that best explains the pose observations by solving a linear program.

Overall, we show that the innovative concept of generative feature models and the way we propose to use them permits to tackle the problem of continuous pose estimation of object categories without resorting to 3D information.

Keywords: pose, features, regression

Zusammenfassung

Kontinuierliche Posenschätzung von Objektkategorien hat in jüngster Zeit aufgrund der Anwendbarkeit etwa im Autonomen Fahren und der Robotik weite Verbreitung gefunden. Wenn hierbei die Anzahl der Objekte groß wird oder nicht alle Objekte im Voraus bekannt sind, werden Verfahren nötig, die die Pose eines generischen Objekts einer gegebenen Klasse zu schätzen in der Lage sind. Allerdings wird das Problem der Posenschätzung durch die hohe Variabilität von Aussehen und Geometrie sehr anspruchsvoll.
Viele in der Literatur vorgeschlagene Ansätze basieren auf 3D-Informationen. Die Grundannahme ist hierbei, dass allein die 3D-Geometrie einer Klasse eine kontinuierliche Posenschätzung ermöglicht. 3D-basierte Methoden können jedoch nur zum Einsatz kommen, wenn 3D-Trainingsdaten verfügbar sind oder eine 3D-Rekonstruktion möglich ist. Andere Ansätze kommen mit 2D-Trainingsdaten aus. Der Nachteil bei der Vermeidung einer expliziten 3D-Klassendarstellung ist dann jedoch meist, dass die Schätzung auf wenige diskrete Posen limitiert ist. In dieser Dissertationsschrift wird hingegen ein innovatives, auf einem 2D-Klassenmodell basierendes Verfahren vorgeschlagen, das einen kontinuierlichen Posenwinkel liefert.
Die Grundprinzip hinter dem vorgeschlagenen Verfahren ist, dass sich die Komponenten des Merkmalsdeskriptors bei einer Veränderung des Blickwinkels stetig ändern. Darauf aufbauend wird erstens ein Funktionsbaustein, das *generative feature model* (GFM), eingebracht. Hierbei handelt es sich um eine Regressionsfunktion, die das Aussehen eines Patchs bei einem gegebenen Blickwinkel berechnet. Es wird dabei gezeigt, wie sich die GFMs, die sich aus mehreren Trainingsobjekten ergeben, aggregieren lassen. Da jedes entstandene Cluster eine Menge von GFMs ist, ist es möglich, diese in einem *generative cluster model* (GCM) zu kombinieren. Schließlich betten wir unsere auf GCMs basierte Klassendarstellung in ein Wahrscheinlichkeits-Framework ein, um die A-posteriori-Verteilung der Objektposen in einem Einzelbild zu berechnen.
Zweitens wird dieser Ansatz modifiziert, indem die Geometrie in den Matching-Prozess eingebracht wird. Dieser Matching-Schritt wird dann als ein Graph-Matching-Problem umformuliert, bei dem die räumliche Anordnung zwischen den Merkmalen berücksichtigt wird. Es wird gezeigt, dass so die Leistungsfähigkeit des vorgeschlagenen Verfahrens gesteigert wird. Schließlich wird der vorgestellte Algorithmus auf Videosequenzen erweitert, indem zeitliche Randbedingungen eingeführt werden. Dabei wird durch Sampling der A-posteriori-Verteilungen jedes Bildes ein Graph konstruiert. Mit einem Simplex-Verfahren wird dann diejenige Posentrajektorie gefunden, die die Posenbeobachtungen am besten erklärt.
Zusammenfassend wird gezeigt, dass das vorgeschlagene innovative Konzept der GFMs und ihre Anwendung das Problem der kontinuierlichen Posenschätzung von Objektkategorien ohne 3D-Information löst.

Schlagworte: Pose, Merkmalen, Regression

Contents

List of Symbols

$\mathbf{A}$ Attribute matrix

$\boldsymbol{\alpha}$ Viewpoint

$\boldsymbol{\alpha}^*$ Estimated viewpoint

$\boldsymbol{\alpha}^{\mathbf{gt}}$ Ground-truth viewpoint

$*$ Convolution operator

$\mathbf{C}$ Cost matrix for dynamic time warping

$\mathbf{c}$ Representative feature of a generative cluster model

$C(\boldsymbol{\alpha})$ Generative Cluster Model

$C_{\mathbf{in}}$ Entrance cost in Linear Programming

$C_{\mathbf{out}}$ Exit cost in Linear Programming

$C_{\mathbf{sc}}$ Score cost in Linear Programming

$C_{\mathbf{t}}$ Transition cost in Linear Programming

$\mathbf{D}$ Degree matrix

$d_{\mathbf{1NN}}$ Euclidean distance to the first nearest neighbor feature

$d_{\mathbf{2NN}}$ Euclidean distance to the second nearest neighbor feature

Δf Frame distance

$DTW(T^i, T^j)$ Cost of aligning feature tracks T^i and T^j using dynamic time warping

D_{xx} Second derivative of D with respect to x

$D(x, y, \sigma)$ Difference-of-Gaussian image

E Edge set

$\mathbf{e}$ Regression error

$\mathbf{f}$ Feature descriptor

$F(\boldsymbol{\alpha})$ Generative Feature Model

$\mathcal{F}$ Set of Generative Feature Models

f_{in} Entrance flag in Linear Programming

f_{out} Exit flag in Linear Programming

f_{sc} Score flag in Linear Programming

f_{t} Transition flag in Linear Programming

$\mathbf{F}$ Ideal fundamental matrix

$\tilde{\mathbf{F}}$ Estimated fundamental matrix

$\mathbf{G}$ Matrix of the Gaussian-weighted distances

$G(\boldsymbol{\alpha}^i, \boldsymbol{\alpha}^j)$ Gaussian-weighted distance between two viewpoints

γ Distance weight between a test feature descriptor and a representative feature model

$\mathcal{G}$ Connected graph

$G(x, y, \sigma)$ Gaussian kernel with width σ

$\mathbf{H}$ Hessian matrix

$\mathbf{I}$ Identity matrix

$\mathcal{I}$ Set of images

$I(x,y)$ Image

K Cluster of generative feature models

$\mathcal{K}$ Set of clustered generative feature models

$\mathbf{L}$ Normalized Laplacian matrix

λ Regularization parameter

$\mathbf{L}_u$ Unnormalized Laplacian matrix

$L(x, y, \sigma)$ Image filtered with a Gaussian kernel of width σ

$\mathbf{M}$ Affinity matrix in graph matching

o Object

$\mathbf{o}$ Pose observation

$\mathcal{O}$ Set of objects

$\mathcal{P}$ Matching set resulting from graph matching

$\mathcal{Q}$ Set of test feature descriptors extracted from an image

$\mathbf{q}$ Test feature descriptor

$\mathbf{R}$ Covariance matrix of a generative feature model

$\mathbf{r}$ Representative feature of a generative feature model

$\mathcal{R}$ Set of representative features $\mathbf{r}$

ρ Length of an oriented segment

$\mathbf{S}$ Pairwise similarity matrix between two sets of tracks

T Augmented feature track

τ Feature distance threshold

$\mathbf{T}$ Matrix of the normalized rows of $\mathbf{U}$

$\mathcal{T}$ Set of augmented feature tracks

θ Angle between x-axis and oriented segment

$\mathbf{U}$ Matrix of the eigenvectors of the Laplacian matrix

$\mathbf{u}$ Eigenvector of the Laplacian matrix

V Vertex set

$\mathbf{W}$ Matrix of coefficient vectors of the Radial Basis Function Network

$\mathbf{w}$ Coefficient vector of the Radial Basis Function Network

$\mathbf{x}$ 2D point in homogeneous coordinates

$\mathbf{Z}$ Matrix of features

List of Abbreviations

1NN First Nearest Neighbor

2D Two-Dimensional

2NN Second Nearest Neighbor

3D Three-Dimensional

4D Four-Dimensional

5NN Five Nearest Neighbors

Adaboost Adaptive Boosting

BRIEF Binary Robust Independent Elementary Features

CAD Computer-Aided Design

DLT Direct Linear Transform

DoF Degrees of Freedom

DoG Difference of Gaussians

DPM Deformable Part Model

DTW Dynamic Time Warping

FH Fast Hessian

GT Ground Truth

HOG Histogram of Oriented Gradients

IQP Integer Quadratic Problem

KDE Kernel Density Estimation

KL Kullback–Leibler

LoG Laplacian of Gaussian

LOO Leave One Out

LP Linear Programming

MAE Mean Absolute Error

MAP Maximum A Posteriori

N3M Natural 3D Marker

NB Naive Bayes

NN Nearest Neighbor

OCR Optical Character Recognition

PnP Perspective-n-Point

PCA Principal Component Analysis

RANSAC RANdom SAmple Consensus

RBF Radial Basis Function

RGB Red Green Blue

SfM Structure from Motion

SIFT Scale-Invariant Feature Transform

SURF Speeded Up Robust Feature

SVM Support Vector Machine

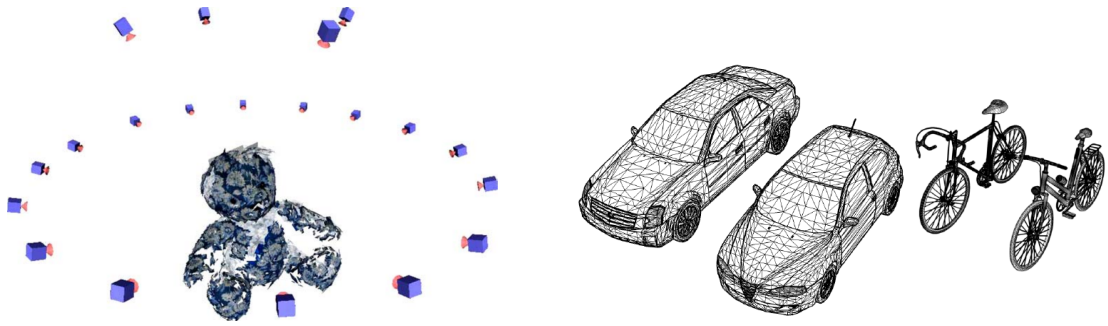
Chapter 1

Introduction

Object pose estimation is the task of determining the orientation and location of an object in an image or a video sequence with respect to some fixed coordinate system. The need for pose estimation methods comes as a natural consequence of the loss of depth information when a three-dimensional scene is projected on the two-dimensional image plane. While the inherent ambiguity is exploited by artistic techniques like optical illusions and *trompe-l'œil*, recovering the pose of an object by removing the projection uncertainty is actually an important challenge for the computer vision researcher.
More importantly, pose recovery is a key ingredient required by complex visual systems for the accomplishment of highly advanced tasks. Scene understanding and augmented reality applications as well as robotic and autonomous driving systems all benefit from an accurate knowledge of the pose of the objects in the scene. For example, the orientation of other vehicles is a strong cue for autonomous driving systems to make correct and safe decisions, likewise the knowledge of spatial object orientation is necessary for a robotic arm to perform precise manipulation. Although pose estimation is one of the most studied problems in computer vision, recovering the pose of an object in an image or a video is still an open problem. This is also confirmed by the plethora of different methods that have been proposed in the literature. The complexity of application requirements and scenarios as well as the availability of training data play an important role in the development of a pose estimation algorithm. Aspects such as

- Availability of a 3D model
- Discreteness or continuity of the required pose
- Knowledge of the object identity or class

are crucial in the development of a pose estimation algorithm, as solutions devised for simpler situations are not easily applicable to harder cases.
A 3D object model turns out to be extremely advantageous when coping with two-dimensional ambiguities due to the loss of depth information. The registration of



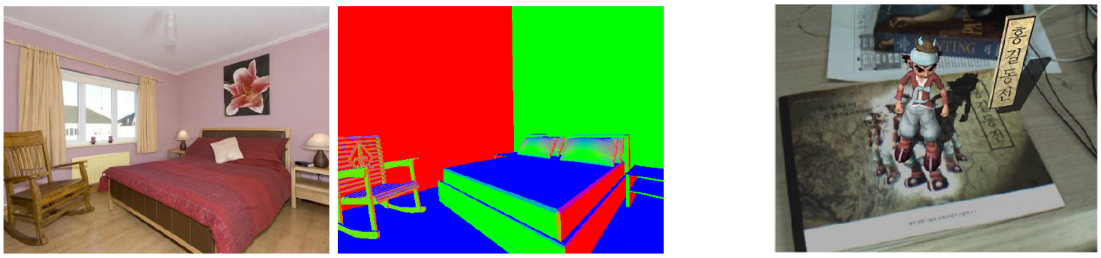
(a) (b)

Figure 1.1: (a) A 3D sparse model reconstructed with a multi-view geometry method. The position of the training views is also estimated and indicated by sketched cameras. (b) Dense CAD models for two types of cars and bicycles, respectively. (Pictures taken from [102, 72].)

the model on the image data, in terms of individual landmarks, silhouette, inner surfaces or a combination of the above, leads directly to the estimation of the object pose. As illustrated in Figure 1.1, different kinds of model may be available. 3D sparse models are the result of a reconstruction based on multi-view geometry, while 3D dense models are the output of computer-assisted design (CAD) systems. Nonetheless, the availability of 3D training models cannot be given for granted. The capability of 3D reconstruction methods depends strongly on the amount of object texture and training data, while CAD models are often expensive and may not be available for the specific object at hand.

Now, let us consider an application where a discrete value for the returned pose is sufficient, such as scene understanding. Given an image of a domestic environment, the estimation of surface normals simplifies to a classification problem, because walls and furniture surfaces usually share three orthogonal directions. But the same algorithm cannot be used for augmented reality applications, as an accurate estimation of the surface orientation is needed for a correct and realistic placement of virtual objects. Examples of these two applications are depicted in Figure 1.2.

While the aforementioned aspects are related to practical or applicative issues, one aspect that is intrinsically related to the pose estimation problem is the object identity. The knowledge of the object identity is advantageous from many points of view, such as the possibility to create a precise model of the object. However, for all the applications we have mentioned above and for many others, this is practically not feasible. The number of different object types combined with their potential variation in appearance and geometry becomes so large that storing all the models in a database is unfeasible, as an enormous amount of memory would



(a) (b)

Figure 1.2: (a) Image of a bedroom (left), and surface normal estimation (right). Surface colors identify the normal orientations. (b) In a digital book application, virtual animated characters must be realistically placed. For this purpose, it is necessary to estimate the book surface with high accuracy. (Pictures taken from [104, 58].)

be needed independently of the adopted model format. Furthermore, the database should be updated every time a new object becomes available and this might not be possible.
As a consequence, the capability of recovering the pose of an unknown object, when the class membership is the only available information, is of large interest. However, the generality of the problem makes the task more challenging, as the method must cope with difficulties such as:

- definition and construction of a 3D class model
- intra-class variability in both appearance and geometry
- pose mapping from model to instance.

Whereas the construction of a precise 3D model is theoretically possible for individual objects, building a 3D class representative model is a harder challenge, as class instances, from which the model should be learned, differ in appearance and geometry. Although the range of variations depends on the extent of the class at hand so that fine- and coarse-grained problems have recently started to be addressed separately, a 3D class model must be built upon different exemplars. Therefore, the algorithm must identify and aggregate similar patterns in the training data to construct a model that is characteristic of that class without overlooking the *generality-vs.-distinctiveness* trade-off. That is, the method must be general enough to accommodate for variations in appearance and geometry, so that the pose of unseen objects can be correctly recovered. But, at the same time, it must be discriminative enough to differentiate between the poses. Finally, the pose transformation mapping the model to the instance must be defined in

agreement with the model itself, and this becomes non-trivial when the model is not geometrically well-defined.
Some works have shown how to generate and use plausible 3D class models for pose estimation, but they still rely on CAD design and multi-view geometry, thus presenting the same issues highlighted before. In this thesis, we present an innovative method for pose recovery that relies only on a 2D sparse class model learned from image data.
The underlying motivation behind the usage of 3D training data is that it is needed to provide a real-valued pose. According to this, the use of only two-dimensional training information seems to restrict the performance of pose estimation algorithms to the mere recovery of a discrete pose. In this thesis, we counter this claim by showing that our method provides a real-valued pose without relying on any 3D information. The key ingredient is a new and interesting way to exploit a standard tool used for pose estimation, appearance-based features.

1.1 Contributions and Organization

As we have explained above, methods that rely on 3D class models overcome the lack of a well-defined three-dimensional class geometry by resorting to synthetic or reconstructed models. In this thesis, we show that this assumption is not necessary as 2D training data can be sufficient to correctly infer three-dimensional information. Moreover, the pose returned by our method is continuous, thus countering the underlying assumption of these methods, where 3D training data is considered necessary to provide a real-valued pose. We now detail the organization of the thesis as well as the main contributions.

Chapter 2 We start by introducing the general paradigm adopted by feature-based methods to solve the pose estimation problem. We present the advantages and disadvantages of using features in recovering the pose of an individual, specific object as well as of an unknown object belonging to a known class. Then, we provide a comprehensive illustration of related works for both situations, with more emphasis on the class case, as it is the one addressed in the present thesis.

Chapter 3 In this chapter, we first introduce appearance-based features from a historical perspective highlighting the reasons behind their success. Then, we explain in depth the two different features that are extensively used in this thesis.

Chapter 4 Here, the core of the method proposed in this thesis is detailed [29]. We first show that appearance-based features are not invariant to out-of-plane rotations, and then we explain how to exploit this apparent weakness for our benefit.

The key insight is that the variation in the components of feature descriptors is smooth as a function of the viewpoint change. Therefore, we build a prediction function, the so-called *generative feature model*, that predicts the descriptor components given an unknown viewpoint. We then demonstrate how to use generative feature models to solve the pose estimation problem in a maximum a posteriori fashion. We show through experiments that our method is effective in the single object case.
We then move to the class case, and we show how to aggregate generative feature models that have been collected from different training instances of the same class. We define a pairwise distance and, on this basis, we find groups of similar feature models. Given a test image of an unknown object of a certain class, we match the test features against the clustered set of generative models and we recover the object pose by extending in a straightforward manner the formulation developed for the single case. We show that the application of our method to the class case is also effective, thus supporting the thesis' claim that a correct real-valued pose can be estimated without resorting to any 3D model.

Chapter 5 By analyzing the results of the previous chapter, we show that our algorithm strongly depends on the correctness of the matches established between test and model features. The approach detailed in Chapter 4 is purely based on appearance, as generative feature models predict only the descriptor components and the matching step does not take feature location into account. However, spatial information is an extremely important cue that should not be neglected.
In this chapter, we explain how to improve the quality of the matches by introducing geometry into the process [28]. More specifically, we reformulate the matching step as a graph matching problem, where the spatial relation between matching features is taking into account. As a result, we enforce that pairs of matching features share geometrical consistency in their respective spatial locations. We show that the introduction of geometrical cues in the matching step is beneficial for the accuracy of the proposed method.

Chapter 6 First, we show the results obtained when our method is applied on a frame-by-frame fashion to a video sequence. We show that the algorithm performance mainly alternates between bursts of good and wrong estimations. Furthermore, we demonstrate that even in the case of wrong estimations the posterior distribution contains a strong evidence of the correct pose.
Therefore, we extend our algorithm to videos in order to apply temporal constraints on the sequence of estimated poses [30]. More specifically, we assume that the pose of an object can only have a small and smooth change over consecutive frames. We build a graph by sampling from the posterior distributions estimated in each

frame and we find the pose trajectory that best explains the pose observations. We show that the embedding of spatial and temporal constraints permits to the proposed method to provide a much more accurate pose when video sequences are at hand.

Chapter 7 Our final chapter is dedicated to the conclusions of the thesis, and it is closed by a presentation of future research directions.

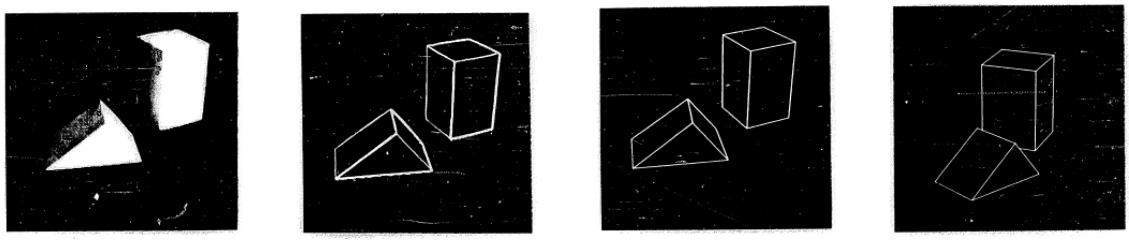
Chapter 2

Feature-based pose estimation

In this chapter, we describe in detail how pose estimation can be solved by means of appearance-based features. We start with a brief overview of representative pose estimation methods that have been presented over the years. We first illustrate some early methods that addressed the pose estimation problem for simple volumetric shapes as well as for free-form objects. Then, we move to intermediate approaches based on the decomposition in 2D views and on geometric invariants. At the end of this overview, a discussion over the unifying thread among these approaches motivates the introduction of pose estimation methods using appearance-based features. We first explain the reasons behind their development, and then we illustrate a general paradigm alongside related works of pose estimation methods for single objects. In the last part of the chapter, we discuss the difficulties that occur when appearance-based features are used to handle the pose estimation problem for object classes. To conclude, we provide a comprehensive overview of related works highlighting similarities and differences with respect to the method proposed in this thesis.

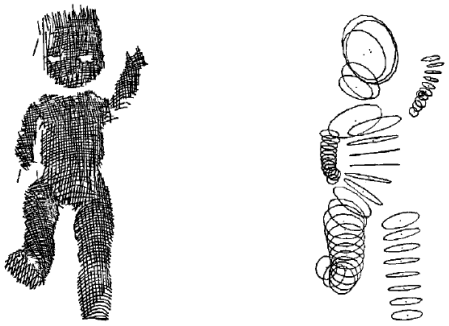
2.1 Historical perspective

At the very beginning of the computer vision era, pose estimation algorithms were purposely developed for simplified situations, where primitive untextured polyhedrons were to be detected over an empty background, as shown in Figure 2.1. The establishment of rigorous mathematical foundations for the treatment of simplified problems was deemed necessary before moving to more difficult situations. A famous example is the work proposed in [100], where simple polyhedrons, the so-called *block models*, are fit to the test image to estimate the object pose. More specifically, the pose is determined by finding the transformation that best aligns the junction points detected in the edge image and those of a CAD model.
While these early methods worked well for objects with simple shapes, the possibility of recovering the pose for free-form objects was first achieved with the introduction of *generalized cylinders* [3]. The key insight is that any arbitrary ob-



(a) (b) (c) (d)

Figure 2.1: (a) A picture with block models. (b) Edge extraction. (c) Correct matching of polyhedrons. (d) The scene is depicted from a different point of view to show accurate pose estimation. (Pictures taken from [100].)

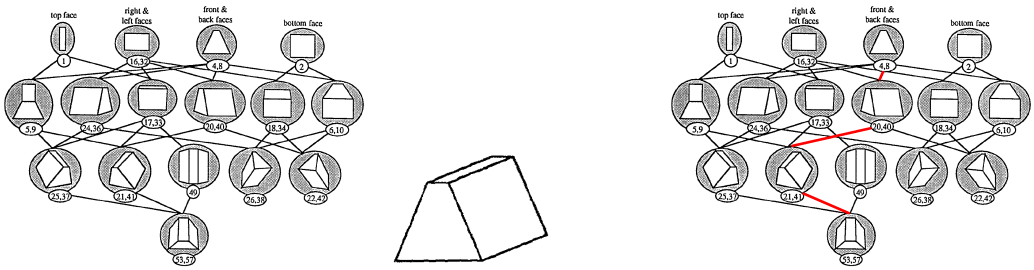


(a) (b)

Figure 2.2: (a) Range image of a doll. (b) Representation based on generalized cylinders. Several cylinders are traced in the image representing different body parts. (Pictures taken from [3].)

ject can be represented by a combination of curved axes and cross-section functions defined on these axes, as illustrated in Figure 2.2. These algorithms try to determine the object's medial axis and several secondary axes by growing cylinders with varying cross-sections around them. The pose was determined by matching the current axial structure against a set of models stored in a database, with the potential usage of further cues like spatial arrangement and lengths of the secondary parts.

The 3D models employed by many of these early methods were often built by means of computer-aided design (CAD) systems used by professional drawers. The time consumption and the money cost to have such models available was proportional to the complexity of the objects to be modeled. In addition, each model was tailored to a single, specific object, thus limiting the availability and applicability of the developed frameworks.



(a) (b) (c)

Figure 2.3: (a) The representation of a wedge as an aspect graph. (b) Test view of the object. (c) Its pose identified by the sub-graph in red in the aspect graph. (Pictures taken from [23].)

(a) (b) (c)

Figure 2.4: (a) Profile of a surface of revolution and convergence of corresponding points to vanishing point under projective transformation. (b) Picture with several surfaces of revolution. (c) Extracted profiles with axes computed automatically (Pictures taken from [132].)

This difficulty was circumvented by the introduction of *aspect graphs* [121]. The key idea is that a 3D model can be represented by a decomposition into several 2D views that are related to each other in a graph structure, where edges represent boundaries shared between surfaces. The pose of an object view can be estimated by matching test and model surfaces in terms of number of edges, edge orientations and neighboring surfaces in the graph, as shown in Figure 2.3.

While methods based on aspect graphs and alike were inherently two-dimensional, a return to a full 3D point of view was marked by the introduction and investigation of *geometric invariants* [17]. The key idea was to define and study geometric properties that do not vary with respect to the viewpoint. For example, measures that are based on the collinearity of three points or the cross-ratio of four points are invariant under perspective transformations. The object pose is recovered by first computing invariants from extracted edges, then by finding matches with model invariants and by computing the projective transformation that best aligns the corresponding locations, as illustrated in Figure 2.4

In this short perspective, we can see that all these methods share a common thread. That is, they are all based on some sort of primitive feature in order to solve the recognition and pose estimation problems, *e.g.*, junction points, cylinders, surfaces, and geometric invariants. As a matter of fact, this is strongly supported by what neuroscientists affirm about our understanding of the 3D world on the basis of the images impressed on the retina. Since the introduction of the Gestalt theory, visual perception studies have confirmed that the human visual understanding of the world is based on the organization and grouping of small primitives extracted from the perceived data [80]. Computer vision researchers have, directly or not, followed this paradigm by developing top-down/bottom-up algorithms that could solve visual tasks by building upon low-level primitives, also known as *features*. The term features is generally used to indicate relatively small, "interesting" image structures, that can be extracted and re-detected in other images of the same scene.

The other aspect that can be traced in this overview is the increasing importance given to the development of robust methods. Whereas early approaches were concerned about reliable algorithms for simplified situations, later research moved to the analysis of increasingly real scenarios by including viewpoint changes, occlusions and background clutter. As a consequence of this shift, the demand of features, that could be robust or even invariant to typical image transformations and external factors, led eventually to the introduction of geometric invariants [132]. While approaches based on invariants were elegant and theoretically correct, their performance was often unreliable and their application was limited to few categories of objects, *e.g.*, revolution surfaces, prisms and canals. The reason is that geometric invariants were computed on the basis of edge and ridge information extracted from images, which is imprecise and undistinctive. The large employment of pre- and post-processing, such as edge grouping and chaining, was useful yet not decisive in coping with the inherent weaknesses and ambiguities of edge extraction.

At the end of the 1990's, the introduction of appearance-based features permitted to simultaneously get rid of all the extra processing and dramatically improve recognition and pose estimation results [109]. The key idea was to apply the "invariance philosophy" to a cue which was so far considered unreliable and non-representative, *i.e.*, object texture. The result was the development of features that were much more distinctive and robust than edge-based features to a large set of image transformations and external factors. The far-reaching importance of appearance-based features is also proven by their current role as a standard tool for many visual tasks, including pose estimation.

As features are at the basis of the method proposed in this thesis, we detail their implications and usage in the following sections. We illustrate how pose estimation can be solved by means of texture-based features as well as the advantages and

disadvantages of using features for recovering the pose of single objects and object categories, respectively. In Chapter 3, we will explain the algorithmic details of the features that are used in this thesis.

2.2 Appearance-based Pose Estimation

Unlike earlier features, appearance-based features are texture-based descriptions of localized structures of interest in the image. In general, they result from the combination of two tools:

- an interest point detector to find characteristic object structures
- a discriminative descriptor of the interest point neighborhood to re-identify the same structure in a new image.

Texture information is used in both detection and description steps, *i.e.*, to extract stable points from the image and to describe the point neighborhood. Detection is usually performed by finding locally spatial maxima of first or second order image derivatives convolved with a Gaussian filter at multiple scales [48, 75]. According to the operator used, interest points located at a corner or in the center of a blob-like region are returned. The description is usually a summary of the neighboring texture information computed in terms of histograms of gradients. That is, a feature is a localized vector generated by the concatenation of the (x,y) location of the interest point, of the gradient-based histograms computed on neighboring sub-regions, and possibly additional information. The reason why objects can be well described by spatial distributions of gradients is that gradients are highly informative, as also motivated by neuroscience studies, and spatial aggregation reduces sensitivity to small variations.
Before explaining how appearance-based features are used for pose estimation, we first illustrate the two reasons for their huge success. First of all, both detection and description can be designed to be robust to many specific transformations and factors. More specifically, features may be directly invariant towards scale changes by applying the detector at several scales, and towards translation, in-plane rotation and illumination changes by properly designing the descriptor. In addition, the influence of background clutter and occlusions is limited by the discriminative description of the features as well as their local nature. Finally, the transformation of images into a large collection of spatially localized feature vectors is beneficial. Spatial information can be used to enforce a geometrical context in the process and the vectorial structure allows for a straightforward application of algebraic methods.
Appearance-based features have been first introduced as a tool for solving object recognition problems with a typical two-step paradigm. First, a feature-based

description of the object, which acts as an object model, is gathered from one or more pictures. At test time, features are extracted from the query image and compared with the object description. These are very fast procedures, as feature extraction can be implemented in hardware and matching amounts to finding the feature nearest neighbor, for which fast approximate techniques can be used. The consistency of the matches is finally evaluated in terms of their number and geometrical arrangement by estimating the transformation that best maps the model features on the query features. As a result, a decision on the presence of the object in the scene is taken.

The application of appearance-based features to pose estimation problems follows the tight connection between this task and object recognition. Since objects change their appearance with respect to viewpoint, algorithms that have been developed for object recognition can be easily transferred to pose estimation and automatically benefit from the same advantages. As a matter of fact, the object recognition paradigm already contains the recovering of the object pose. When correspondences are checked for their geometric consistency, a registration of the model features to the image features is performed, and the underlying transformation is nothing but the pose of the object. Therefore, the paradigm outlined above can be directly adopted by changing the focus from finding the object to estimating its pose.

The use of appearance-based features for pose estimation presents additional advantages from those highlighted before. First of all, model building and pose estimation techniques are performed on the basis of the same features. This favors the cascading or joint coupling with other tasks that are also performed with the same features. Additionally, a system architecture based on appearance features is transparent in the choice of the specific feature, as most of them have a point-like nature with vectorial description. Finally, feature detection and description are two completely independent steps, so that different detectors and descriptors can be combined according to necessity.

Since appearance-based features are point-like entities, they can be arranged either in planar or volumetric structures depending on the object at hand. While the planar arrangement results directly from feature extraction, the 3D case involves a volumetric object reconstruction from training images. For this purpose, methods based on projective and multi-view geometry are used, whose introduction in the computer vision community traces back to the time of geometric invariants. In Figure 2.5, we summarize the two algorithmic paradigms used for model construction and pose estimation according to the object dimensionality.

While 3D pose estimation algorithms are needed whenever the object depth cannot be neglected, their two-dimensional counterparts find application in several tasks, such as target tracking or augmented reality. In the former, a planar object, like

2D Pose Estimation:

Model A set of features extracted from a picture of the object can be taken as model representation. Additional descriptions from different viewpoints may be aggregated in order to be robust to viewpoint changes.

Matching Test and model features are set in correspondence according to descriptor distance.

Pose The object pose is given by the *homography* transformation that best maps the model features onto the corresponding query features.

3D Pose Estimation:

Model Features collected from several views of the object are used to create a 3D object reconstruction.

Matching Test and representative model features are set in correspondence according to the descriptor distance.

Pose A *projective* transformation is estimated, such that 3D feature points are best mapped onto the 2D query features.

Figure 2.5: 2D and 3D paradigms for the pose estimation of single objects.

a logo or a checkerboard pattern, must be found and tracked in a video, while in the latter a virtual (possibly 3D) object is to be placed upon a surface, whose precise orientation must be estimated. In the following section, we provide a broad overview of methods for 2D and 3D pose estimation in the case of single objects that have been recently proposed.

2.3 Pose Estimation for Single Objects

According to the paradigm outlined in Figure 2.5, pose estimation for single objects amounts to one off-line task, model creation, and two on-line tasks, feature matching and pose computation. This general paradigm encompasses several difficulties, such as finding correct matches and guaranteeing real-time performance, so we have opted to present the related works according to the difficulties they address.

Figure 2.6: (Left) Test image. (Right) Objects have been recognized and the models are reprojected on the image with the estimated pose. (Pictures taken from [102].)

Figure 2.7: According to the estimated pose of the textured mug, a teapot is placed above it. (Pictures taken from [41].)

3D Models While 2D models are simply a set of features extracted from an object view, 3D models need a proper construction procedure, which is generally borrowed from multi-view geometry methods [49]. These methods deliver a 3D reconstruction of the object starting from a set of pictures that fully cover the object in the view space. The difficulty lies mainly in the reconstruction procedure and how to handle the multiple features that describe the same 3D point from different viewpoints. In [101, 102], the authors build a model off-line using features extracted by an affine-invariant detector and described using a SIFT descriptor. The model is built by iteratively merging partial models constructed from pairs of views. At test time, image and model features are set in correspondence on the basis of color histograms and SIFT features. Finally, the pose is recovered by taking the projection matrix computed from the minimal subset of matches that has the largest consensus, as represented in Figure 2.6.

In [77], the model is constructed by clustering training images in a small number of model views according to the number of matching features and by linking them on the basis of the shared features. Given a test image, features are extracted and matched in a probabilistic fashion by taking into account location, scale and orientation of each matching feature. If the overall matching probability

Figure 2.8: The method of [53] applied to images where the object is deformed or strongly occluded. (Picture taken from [53].)

exceeds a predefined threshold, the object is considered found and the best matching model view is returned as the estimate of its pose. In [40, 41], the authors propose a method for placing a virtual object on top of another one for augmented reality applications, as illustrated in Figure 2.7. First, they build a 3D model using multiple-view geometry techniques applied to features collected from a set of training images. Then, they extract features from the test image and match the descriptors to the model in order to find correspondences. The pose of the object is obtained by finding the best projective transformation using model fitting techniques and non-linear optimization to remove the influence of spurious matches.
An alternative method that does not rely on 3D reconstruction, but it exploits CAD models is proposed in [51]. Giving a 3D textured model of the object, features are detected by rendering the model from different viewpoints. So-called N3M's (Natural 3D Markers) are identified as quadruplets of points with high repeatability and equal spatial distribution over the object. For each point inside a N3M, a classifier is learned using randomized trees. At run time, test points are matched to N3M's using the corresponding classifiers, and a local pose is computed. Each local pose is eventually verified, and consistent N3M's are used for the final estimation.

Feature Matching When the amount of clutter is high, many erroneous correspondences can be established, thus leading to a wrong pose estimation regardless of the robustness of the method used to compute the pose. Therefore, effective matching techniques must be introduced to replace nearest neighbor matching, which is easily prone to errors. In this regard, [68] interprets matching as a classification problem. A known 2D model is learned in terms of PCA-reduced features and for each feature a classifier is trained by randomly warping the original patch. At test time, the feature identity of the query patches is found by means of the model classifiers. Then, the homography is computed by solving a linear least squares problem. A variation on the previous method is proposed by the same authors in [69], where randomized trees are used instead of classifiers based on

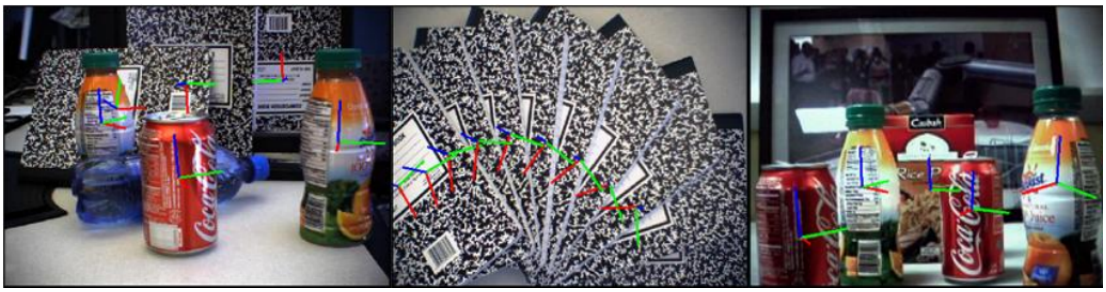


Figure 2.9: Three different scenes with repeated objects. The pose of the object is depicted by the three colored axes. (Pictures taken from [14].)

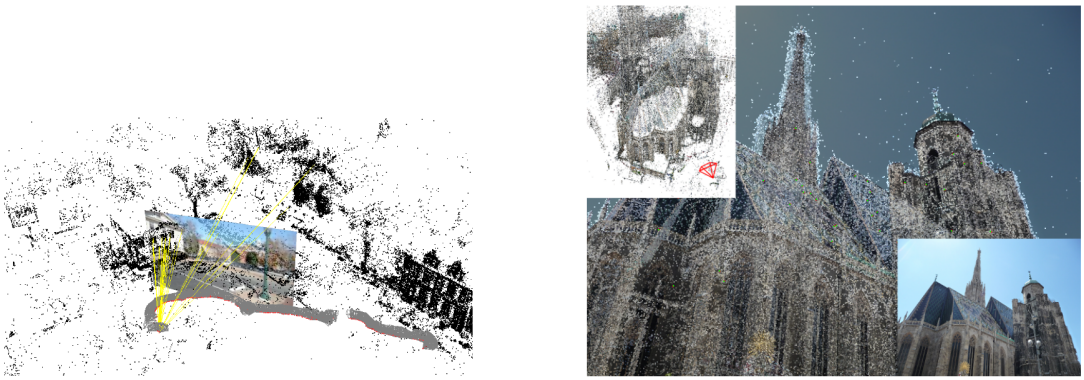
(a) (b) (c)

Figure 2.10: (a) and (b) Images of the four sides of a juice carton. Please note that two sides are only differentiated by the presence of the red lid. (c) Two instances of the same object are detected in a cluttered image and opposite sides are correctly identified. (Pictures taken from [55].)

K-means, and the decision thresholds are learned on the basis of the difference between pixel values. By following the same direction, [53] proposes a method based on two feature classifiers. A first classifier based on Ferns is learned for each feature identity. A second classifier is learned for each feature orientation from descriptors computed on synthetically warped patches. The output of the pair of classifiers is the feature identity and an initial guess for the pose of the patch. The final pose is estimated by recursively refining the initial guess by means of a cascade of linear predictors, as shown in Figure 2.8. In the follow-up of this method [52], the authors propose to learn only one linear classifier that combines keypoint identity and pose, instead of two cascaded classifiers. In a second follow-up [54], they replace the linear classifier with a set of mean patches. By decomposing all patches into a set of PCA components and warping them, the method can perform in real time.

Repetition of objects and structures An important problem closely related to matching is the presence of multiple object instances and repetitive structures. The repetition of objects in the scene and of patterns on the object surface requires a careful handling of the resulting matching ambiguities. A solution for the problem of multiple instances is addressed in [14], where a 3D point cloud is reconstructed from a set of training views of the object. Then, SIFT features are extracted from the test image and matched against the model features. In this case, test features are clustered with mean shift and each cluster undergoes an individual pose estimation procedure, as shown in Figure 2.9. The repetitive structure problem is addressed in [55], where features are enriched with synthetic versions obtained through patch warping and then clustered in a hierarchical way. At test time, features are extracted from the test image and matched against each hierarchical level of model features without discarding ambiguous matches. The pose is finally estimated using RANSAC and a view constraint based on point co-visibility, as illustrated in Figure 2.10. In this way, the ambiguity is retained and solved only in the last part of the pose estimation paradigm, where geometry provides useful cues.

Real Time Applications such as on-line tracking and augmented reality have usually strong real-time requirements, thus calling for the introduction of efficient techniques. These techniques must be tailored upon the type of feature used and the average amount of features extracted from the scene. In [58] and [85], two methods for real-time performance of augmented reality applications, such as digital books and board games, are proposed. In [58], multi-threading is introduced to provide real-time performance, where one thread is dedicated to frame-to-frame tracking, while the other has the task to recognize the book illustrations and produce 3D animation effects. In [85], the playing board is recognized and the homography transformation is estimated using BRIEF, which is a fast feature descriptor. Then, pawns are detected by looking only at specific board positions for additional time saving. Eventually, pawns can be moved on the board and their position is tracked over time. In [122, 123], the authors propose a real-time tracking method based on the usage of keyframes as well as frame-by-frame matching, thus achieving high speed and avoiding both drifting and jitter problems. In [92, 59], methods for augumented reality are proposed for a high number of objects. The idea is to distribute object detection over consecutive frames. Each newly detected object is dedicated a frame-by-frame tracking as soon as computational resources are available. For this purpose, tracking is only performed on so-called “temporal keypoints”, instead of the typical tracking-by-detection approach, which is more time consuming. In the remaining time, detection is still performed to prevent tracking losses and drift.



(a) (b)

Figure 2.11: The current position of the camera is found in a given 3D point-based reconstruction of a large object like a square (a) or a monument (b). (Pictures taken from [56, 105].)

Model dimensions Another hard challenge occurs when the dimensionality of the model grows from few hundreds to millions of features, as in the case of image localization tasks. Given a 3D model of a large object, like a monument or even a city area, the goal is to relocate a test image with respect to the model, as represented in Figure 2.11. Several works have proposed heuristic methods to speed up the re-localization performance. In [56], the approach is inspired by document retrieval as a vocabulary tree is built from all the descriptors. Given a query image, SIFT features are extracted and dropped down the tree. Each set of 3D points or "document" is given a score. Once the best documents are recovered, pose verification is performed by creating 2D-3D matches between the query features and the 3D points of the set and then solving the Perspective-n-Point (PnP) problem. In [71], image localization is sped up by using priority matching. For each connected component of the reconstruction, a set of most representative 3D points is computed by solving an image coverage problem. For the localization, model features are matched against the image by giving high priority to points with high visibility, co-visibility and spatial distance in the model. Once 3D-2D correspondences are available, pose is estimated using a direct linear transform (DLT). In [105], location recognition is accelerated by using direct 2D-to-3D matching, feature quantization and prioritized search. More specifically, visual words are obtained by clustering the 3D points of a given reconstructed city scene. At test time, features are extracted from the query image and assigned to the nearest visual word. Correspondences in which the visual word contains few model features are analyzed first. The search stops after N correspondences are found and the pose estimation is performed by solving the PnP problem.

2.4 Pose Estimation for Object Classes

Up to now, our discussion has described how features can be employed to solve the pose estimation problem for the case of individual objects. When we address the pose estimation problem for object classes, where only the category membership of the object is known, like in this thesis, specific difficulties arise, as already highlighted in Chapter 1.

The first problem comes from a design property of features that is generally advantageous in object recognition tasks, *i.e.*, distinctiveness. Feature descriptors are designed to be as discriminative as possible, so that false positive detections are kept to a minimum. However, when we have to estimate the pose of an unknown object, this property is not advantageous anymore, as the difference between a query descriptor and the descriptors of analogous structures in the training set is possibly large. Therefore, countermeasures must be taken to reduce feature specificity by aggregating descriptors of similar parts, so that a more general description of each structure is created. Since feature generality and distinctiveness are in contrast, the clustering algorithm must be carefully chosen in order to find a satisfactory trade-off.

The second difficulty is related to the point-like nature of appearance features. While the feature spatial arrangement is used in the single object case to build a precise model, this cannot be done directly in the class case, as feature clusters do not have a precise spatial location. Therefore, when a 3D class model is of interest, 3D locations for feature clusters must be defined. For example, a spatial distribution of parts can be fit to individual 3D reconstructed models or feature clusters can be manually linked to the 3D points of a CAD model.

The third problem is related to the transformation that aligns the model features to the test features. Irrespective of the dimensionality of the class model, this problem is not trivial to solve. On the one hand, if the class model is three-dimensional, the different arrangement between model points and query points must be taken into account when computing the mapping transformation. On the other hand, if the model is defined as a view classifier, no model-to-instance mapping is actually possible, and thus the pose is just given by the label of the classifier with the highest score.

While the first problem is treated by all methods in the literature with a feature clustering step, the last two problems turn out to be more critical. In this regard, two different trends can be identified in the literature according to the dimensionality of the model employed. Methods that opt for a 3D model have a large price to pay in terms of complexity and availability of suitable training data, not to mention ambiguities in model construction and in the alignment transformation. The important advantage deriving from full 3D geometry is that the pose can be estimated as a continuous value. On the contrary, methods that rely on 2D

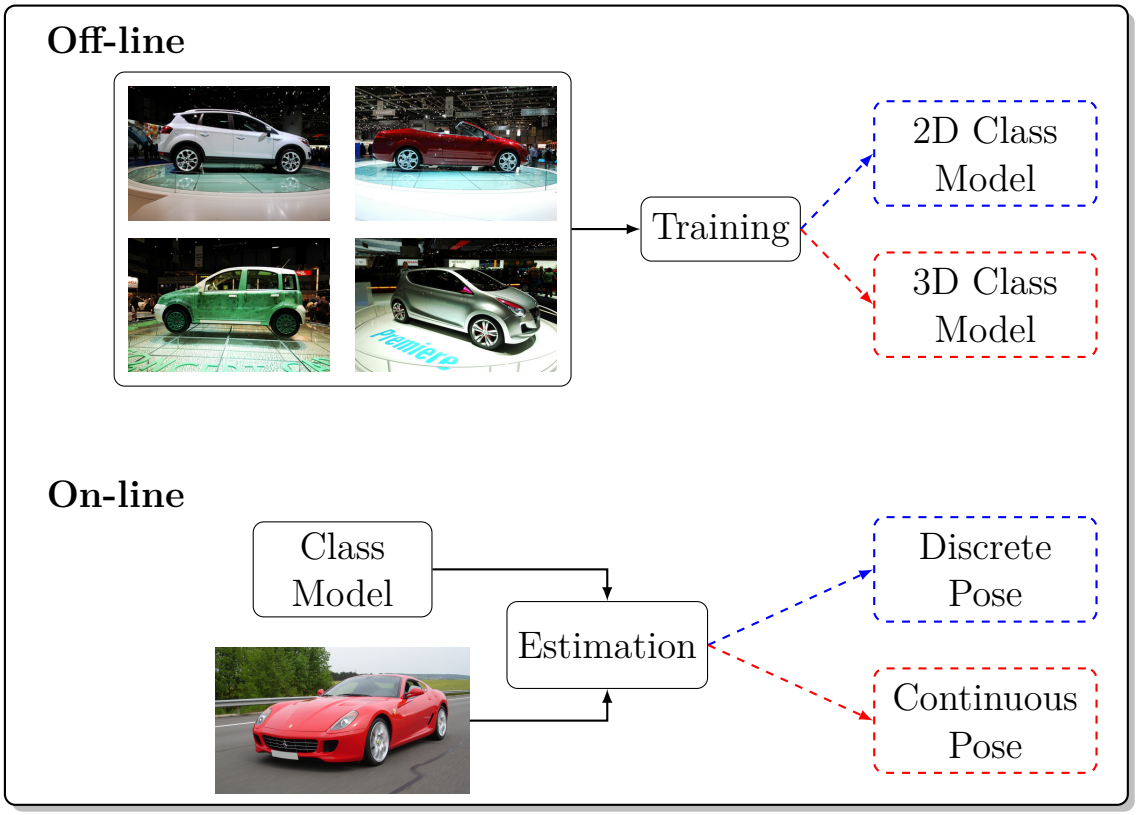


Figure 2.12: Paradigm for pose estimation of object categories. In the off-line step, a class model (2D/3D) is trained or built according to a set of training images or videos. In the on-line step, the class model is employed by the estimation algorithm to determine the pose of the object in the query image. The pose is either discrete or continuous according to the model dimensionality.

models are intrinsically simpler to build and to handle, often amounting to a set of view classifiers. However, they have the strong disadvantage of providing only a discrete, often coarse, value for the pose. Like for pose estimation methods for single objects, those addressing the case of object classes can be also framed in one paradigm, yet more general, as illustrated in Figure 2.12.
As highlighted before, both strategies suffer from strong limitations. 3D models can be hard to build and are complex to treat, while strategies providing a discrete pose may not be suitable for many applications. In very recent years, several methods have been proposed to bridge this gap, *i.e.*, to provide a continuous pose estimation on the basis of only 2D training information. Part of these works propose a solution based on a discrete-to-continuous strategy. That is, they first estimate a rough viewpoint, usually by means of view classifiers, and then refine

Figure 2.13: 3D textured CAD models for cars and motorbikes. (Pictures taken from [73].)

the pose with continuous precision. On the contrary, other approaches, like the one proposed in this thesis, directly estimate a real value for the pose without going through any rough intermediate estimation. The important advantage is that they do not have to rely on a hard decision. That is, they avoid the problem of an initially wrong viewpoint estimation that would irrecoverably undermine the correctness of the final pose.
In the following, we provide a comprehensive discussion of related works for pose estimation of object classes. We structured this discussion according to the three different strategies that we have highlighted above. At the end of this overview, we will provide a comparison between the method we propose in this thesis and the most similar approaches available in the literature. We will motivate our technical choices and discuss the advantages of our approach.

3D Model When approaches rely on 3D models, two alternatives are of common usage. On the one hand, one or more CAD models of objects from the class at hand are designed privately or acquired from Internet repositories [1]. CAD models are appealing for several reasons: they have a precise geometry, they give the possibility of texture renderings, they contain no noise. The two main disadvantages are related to availability and cost. On the other hand, 3D models can be reconstructed by employing Structure-from-Motion techniques, which revert the pros and cons of using CAD models. Annotated image databases are ubiquitous in computer vision and they are free and extremely varied. However, 3D modeling suffers from errors in the geometric reconstruction as well as from the introduction of spurious information in the model. However, irrespective of the model chosen, the knowledge of the 3D geometry allows to compute a projective mapping from the model to the query instance, and thus permits to deliver a continuous value for the pose. In the following, we will illustrate different strategies where 3D models are used to compute the pose of an unknown object of a given class.
In [73], a general class model is learned from accurate 3D CAD models, like those shown in Figure 2.13. More specifically, the authors render a set of images from

3D CAD models by sampling the pose space in terms of azimuth, elevation and distance. SURF features are extracted from the rendered images and labeled with class, viewpoint and 3D position. Features are clustered in a codebook using k-means in order to cope with intra-class variability. At test time, query features are matched against the codebook, and each entry cast a vote in a Hough space. All the maxima are eventually verified by computing a projection matrix on the basis of the 2D-3D matches. The matrix, and thus the viewpoint, with the highest consensus is returned. In their follow-up paper [72], the authors use 3D CAD models with a different spirit, by using appearance to estimate a coarse viewpoint and then geometry for continuous refinement. In details, they rely on a spatial pyramid of DAISY histograms to describe each object. For each discrete viewpoint and each part, a SVM classifier is learned. A geometric model is built out of 3D CAD models by learning a distribution for the location of the 3D points belonging to each part. The probability is defined in terms of an adaptive mixture of Gaussians. At test time, in a sliding window fashion, parts are detected and a coarse vote in the Hough space is collected. Finally, the continuous pose is found by maximizing the projection of the image to the 3D parts of the model.

A recent work in which CAD models are used in training is that of [126], where a method for viewpoint and part layout is proposed. Given a set of 3D CAD models, training is performed by learning a set of aspect parts, so that each part is represented as a 3D rectangle with a center location and a HOG feature representation. Given an image, the probability that the object is depicted from a certain viewpoint is expressed through an energy-based formulation. The formulation takes into account both unary potentials for the appearance of the individual parts and pairwise potentials for the reciprocal positions of pairs of parts. By training a Structured SVM [120], the pose is found as the one that maximizes the output of the classifier. The layout of the parts comes automatically from the estimation of the viewpoint and the 3D model. The authors have also proposed an extension for this method in [4], where detection and pose estimation is performed simultaneously on a set of images depicting the object. In this case, individual parts with low scores can benefit from two different evidences. One given by the other parts of the model that are simultaneously visible in the same image, and the other given by the same part in other images to which a strong match can be found.

In [128], a method for 3D object detection and pose estimation based on visual words and SfM reconstruction is described. On the basis of a set of images of different class instances, SIFT features are extracted and clustered in visual words. For each instance, a visual model is created by using SfM techniques, so that a correspondence between 3D coordinates and visual words can be determined. At test time, SIFT features are extracted from the query image and matched against several models of each class. Through over-segmentation, small regions of the

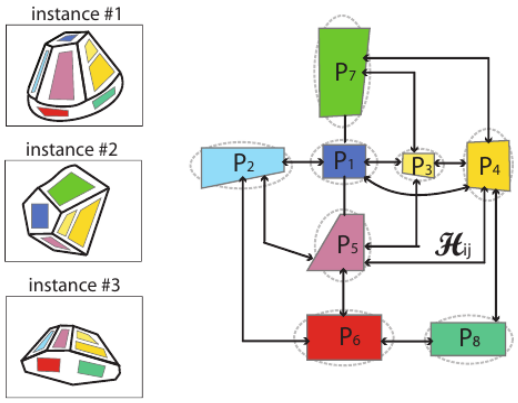


Figure 2.14: (Left) Instances of a hypothetical object, where the colored parts represent sets of small neighboring features. Parts with the same color across different instances are found to be in correspondence. (Right) The model of the object category is a graph where the nodes are the object parts P_i and the edges are the connecting homographies (H_{ij}). (Pictures taken from [106].)

image can be matched to the model and 2D-3D correspondences are considered to compute a projection matrix. Eventually, the model is reprojected on the image and each projection matrix is ranked according to its consensus. The matrix with the highest consensus is returned as the estimated pose.
As in the previous work, the authors in [39] collects individual 3D models from a set of training pictures. In this case, models are merged during training by simply aggregating all the features in one large, redundant 3D class model. Given a test image, HOG features are extracted and multiple matches to the 3D model features are found. Now, for each match, the authors assume that the query feature shares the same viewpoint and scale as the matching model feature. Thus, the projective transformation amounts to a simple shift in the 2D location of the feature, which is cast as a vote in a dimensionality-reduced Hough space. Finally, a 6 DoF projective transformation that fits all the features that cast a vote in the neighborhood of the Hough maximum is returned.

2D Model - Discrete estimation In [106, 107, 108], the authors propose a method for pose estimation of object classes, where the model is constituted by 2D parts linked through homographies, as illustrated in Figure 2.14. Model parts are selected from a larger pool of training parts, where each part is defined by a set of quantized SIFT features and a bounding box. The linkage structure between two model parts relies on the homography that projects one part in the best view of the

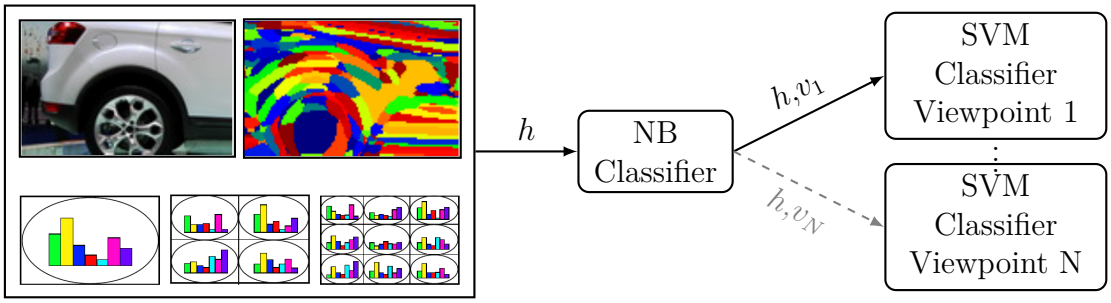


Figure 2.15: DAISY dense features are extracted from the given window and quantized. A histogram pyramid is built by computing histograms over smaller spatial cells. The concatenated histograms are input to a NB classifier and the viewpoint with the highest score is determined. The histogram is then input to the corresponding SVM classifier for a binary decision about the object and the viewpoint. (Picture taken from [90].)

other. At test time, each model part is compared against the test features and the five best matches per part are retained. Part matches are eventually evaluated by scoring their pairwise location and appearance according to a view synthesis criterion. The combination of parts with the highest score simultaneously identifies the object location and the object pose in terms of the nearest model view.

While the previous approaches rely on 2D parts connected by two-dimensional transformations, many others bluntly interpret the pose estimation problem as a classification problem. In these works, the model has no real dimensionality and the accuracy of the provided pose depends on the number of views on which the classifier is trained.

One work in this direction has been presented in [90], where the main idea is to consider different views of an object as different objects. In order to estimate the pose, the authors train two types of classifiers: one Naive Bayes (NB) classifier and N SVM classifiers, where N is the number of possible output viewpoints. The NB classifier is trained with a pyramid of spatial histograms that are computed on the basis of quantized DAISY features, as illustrated in Figure 2.15. The NB classifier is used to learn a mapping from the spatial histograms to the probability of each viewpoint. At test time, as illustrated in Figure 2.15, histograms are extracted from the given bounding box and the viewpoint with the highest probability is determined by the NB classifier. On the basis of the chosen viewpoint, the corresponding SVM classifier is run. If the classification is positive, the viewpoint is returned as the estimated pose.

In [98], the choice of the classifier falls upon the recently introduced Hough Forests [36]. The authors leverage the single-view object detector described in [35] and

provide a multi-view extension. For each view, a Hough forest is trained using RGB patches taken from view-labelled training images. At run-time, each test patch is dropped down the forest and a set of 4D votes is cast, where the four dimensions are: 2D location, scale, and viewpoint. The Hough image results by considering the average for each tree and the contribution of each patch. The view component of the maxima detected in the Hough image provides the estimation of the object pose.
The previous approach indirectly relies on the concept of feature sharing, as different features can be present in multiple views and vote for similar center locations. A work that is focused on feature sharing for multi-view detection is presented in [119]. In this paper, the authors propose a method based on a variation of Adaboost [33], called "gentleboost" [34], to perform view classification of cars. The idea is to learn a classifier based on features[1] that simultaneously classifies a patch with respect to multiple views. Each set of multiple views is chosen in a greedy fashion as the one that minimizes the current misclassification error. The view for which the accumulated score over all shared classifiers is the highest is returned as the estimated pose of the object.

2D Model - Continuous estimation In this paragraph, we describe approaches that rely on 2D training information, yet are able to provide a continuous pose, like the method proposed in this thesis. In this way, they take advantage of the simplicity of creating and handling a 2D model, without losing the capability of outputting a real-valued pose. This is obtained by employing different strategies, like regression, Taylor expansion or kernel density estimation.
The authors in [118] leverage the approach presented in [116, 117] and provide an extension for continuous pose estimation using regression. The two original works presented a method for object classification based on projecting training SIFT features on a lower dimensional manifold. In [118], the embedding of training features from different instances is constrained not only by the intra-image location distance and inter-image feature distance, but also by the viewpoint distance. Regression is performed with a radial basis function network, where the centers are a subset of the training images. The regression coefficients are learned by solving a linear least-squares problem where the interpolation matrix has entries in terms of the percentile-based Hausdorff distance between two projected center images. At test time, test features are embedded on the manifold using a previously learned out-of-sample mapping function. Linear regression in the embedded image space is performed to output a continuous estimate for the pose.
The work in [47] proposes the usage of K-ary regression forests to perform pose

[1] In boosting literature, the term feature commonly identifies a binary test to be applied to the image patch

estimation. Instead of a typical binary regression forest, the authors provide a mechanism to train an adaptive K-ary regression forest, where the value of K is chosen at every splitting node. More specifically, K is found by minimizing the Bayesian Information Content at each splitting node with respect to the number of clusters. In this way, the partition of the training HOG features is performed in the optimal way according to the labels that reach every node. At test time, a test patch is dropped down the forest, and the reached leaves provide a regressed value for the pose that is finally averaged over all forest trees.
In [44] the authors extend the object detector in [26, 27] to provide a continuous pose. The idea is to train a set of part-based SVM classifiers with HOG features extracted from positive and negative training images, where each classifier is dedicated to a certain discrete viewpoint. Then, the optimization function is modified by making the classifier templates dependent on a pose offset through a Taylor expansion. The sum of the coarse viewpoint and the offset maximizing the new optimization function is returned as the continuously estimated pose.
Another work that follows the same strategy of first providing coarse pose candidates and then refining the estimation can be found in [50]. Here, object localization and pose estimation are coupled in a single classifier. As the search space is huge and the objective function is non-convex, the key idea proposed by the authors is to parameterize the classifier with respect to the pose variable. In this way, it is possible to instantiate an arbitrary amount of sub-classifiers, where each classifier is dedicated to a specific pose. First, this set of classifiers is used to select several location candidates with an associated coarse pose. Then, the proposals are evaluated on the original objective function in a continuous way, but on a reduced search space. The maximum of the objective function is finally returned as the estimated pose.
A continuous extension of the work presented in [98] is proposed in [99], as illustrated in Figure 2.16. Here, Hough Forests are used to determine only the candidate locations for the object at the maxima of the Hough voting space for the (x,y) coordinates. The continuous extension for the pose estimation is obtained by first considering all the votes in a small neighborhood around the pose maximum. And then, a probability distribution for the pose is built by using a Gaussian Kernel Density Estimation (KDE) centered around the pose votes. The pose that maximizes this distribution is the returned pose.

2.5 Comparison to our method

Our method can be framed in the third category mentioned above, *i.e.*, our class model is built using only 2D training information and a continuous value for the object pose is provided. Like all these approaches, we also use appearance-based

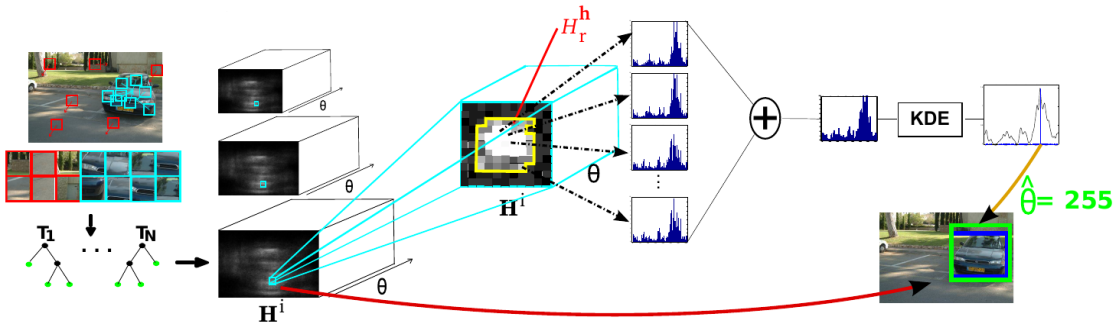


Figure 2.16: Each patch in the test image is dropped down a Hough Forest, whose votes are accumulated in Hough images at multiple scales. The votes in the neighborhood of the maxima are averaged and a KDE is fit to the discrete distribution in order to find a continuous pose. (Picture taken from [99].)

features, but we propose to use them in an innovative way by intentionally exploiting one of their seeming weaknesses.
Whereas features are not invariant to out-of-plane rotations, the way their descriptor varies with respect to a change in viewpoint is smooth. This gives us the inspiration for learning a regression function for each feature that can predict its descriptor given a query pose. Each regression function, that we named *generative feature model*, is learned from an arbitrarily large set of features describing the same patch under different viewpoints. Since pose estimation is ultimately a continuous problem, any classification method providing only a discrete pose is inherently inaccurate, and additionally, classification methods proposed so far achieve accurate estimations only for coarse quantizations. Our motivation to rely on regression functions to estimate a continuous value for the pose is thus a natural choice. In this regard, our work shares some similarity to [118], as they also use regression on local features. While they apply regression to an entire set of training features as an atomic unit, we propose in this thesis to apply it individually to each feature. This has the advantage, confirmed by experiments, that the pose of an unknown object can be better estimated by aggregating evidence from individual patches of different training instances, rather than combining whole instances together.
As already mentioned, a feature clustering step is necessary to reduce discriminativeness and create a more general description of the grouped feature tracks. We resort to spectral clustering, a technique that permits to perform grouping on the basis of the elements connectivity instead of their geometrical proximity, like k-means, which is the clustering method of choice of all the approaches mentioned before. Furthermore, the use of spectral clustering is beneficial for two different

reasons. First, we do not need to assume any special arrangement for the feature tracks to be clustered, as it can handle non-convex sets, unlike k-means. Secondly, k-means cannot be applied to sets whose elements have a different dimensionality, like our feature tracks. A feasible alternative would be k-medoids, but the non-convexity issue would still remain. As a similarity score for clustering, we propose to score pairs of generative feature models by using an algorithm inspired by dynamic time warping. Dynamic time warping is a technique from audio processing that permits to align signals of different length, and so its application in our problem is straightforward by replacing signals with feature tracks and time with viewpoint.

At the end of the clustering step, we learn a class representation that is based on the grouping of generative feature models, where each cluster is analogously named *generative cluster model*. Each cluster model is assigned a regression function that is the linear combination of the regression functions belonging to the elements of the cluster. Finally, on the basis of the generative cluster models, a posterior distribution of the pose of the query instance is estimated. Unlike many methods mentioned above, we treat the problem in a probabilistic fashion. In this way, we can apply a *maximum a posteriori* reasoning if we need to provide a pose value for each single frame, but we can also combine the posterior distributions at consecutive frames to infer more information about the variation of the pose over time. We show this capability of our method in Chapter 6, where we sample from the posterior distribution at each frame and we connect these samples to yield the optimal pose trajectory over an entire video sequence.

Chapter 3

Appearance-based features

In this chapter, we first give a brief introduction of the main ideas behind feature design by examining technical details that make features highly distinctive and invariant to image transformations. Then, we will describe in detail the algorithms of two specific features that are extensively used in this thesis. This deep treatment can be considered as a prerequisite for the presentation of the fundamentals of our method in Chapter 4.

As mentioned in Chapter 2, standard feature design encompasses two steps: keypoint detection and keypoint description, so that each feature is characterized by a location and a vector description. The feature detection process should identify repeatable and precise locations, so that the same features can be extracted in another image of the same object. As edge points are hard to re-localize in other images, the focus moves towards points that exhibit strong changes in two directions. These points are located at the local extrema of the image response to a Gaussian first or second order derivative filter. Gaussian filtering is performed to smooth the image so that high frequencies are removed. The use of image derivatives permits to identify corners and blobs, which are repeatable structures characterized by a precise location. This operation is usually performed at different scales, *i.e.*, by changing the size of the Gaussian filter, so that scale invariance is guaranteed. In addition to translation and scale invariance, features are also assigned a main orientation, so that invariance to in-plane rotation is also provided. This assignment is usually based on the estimation of the main gradient orientation in the neighborhood of the point. As features must be distinctive, the feature descriptor must be computed in such a way that features can be matched without ambiguity. The descriptor is a vector computed on the basis of the neighborhood information, for example by concatenating a number of histograms calculated over a predefined set of sub-regions around the keypoint.

Actually, many fundamental concepts of this two-step design have been introduced much earlier. For example, the idea to describe an image by a set of local interest points can be traced back to the Moravec and Harris corner detectors [86, 48]. Invariance to scale was studied by Lindeberg in his work [74], where the so-called

scale-space theory was presented. The distinctiveness of a description based on local derivatives was already pointed out in [61], where interest points are described by partial derivatives of different orders, the so-called *local jet*. The first work where this two-step design is entirely employed is the one presented in [109]. In this paper, the authors use a corner detector with a multi-scale approach to identify characteristic locations, at which they compute a local jet descriptor in a rotation-invariant way. This seminal work paved the way to the active development of many invariant features over the last fifteen years, of which SIFT and SURF features are possibly the most famous. Since they constitute the fundamental tool behind the approach developed in this thesis, we will provide a detailed description of these two features in the following two sections.

3.1 SIFT - Scale Invariant Feature Transform

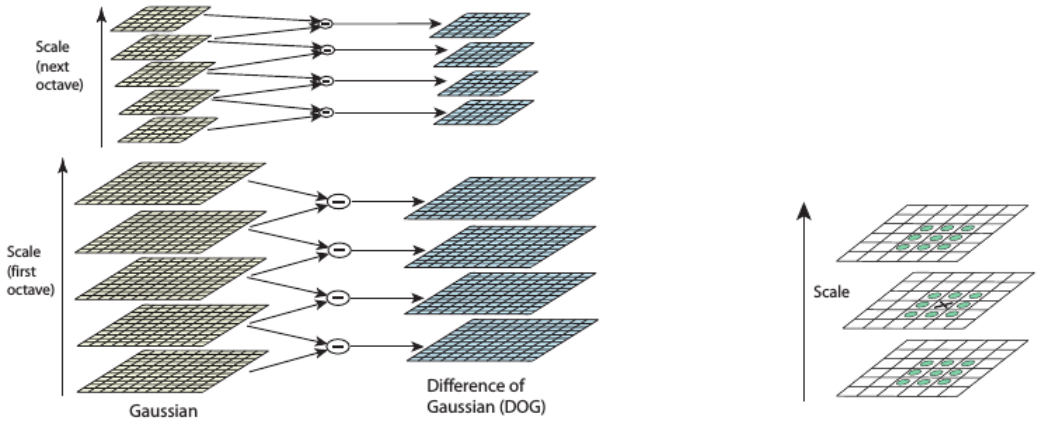
With the SIFT algorithm, the image is transformed into a set of features that are invariant to translation, in-plane rotation and scaling, and they are partially invariant to changes in illumination and viewpoint. The algorithm to compute SIFT features can be split into four steps, each of which is described in detail in the following. In the first two steps, candidate interest points are first identified as extrema of the image scale space, and then are precisely localized by fitting a quadratic model. Furthermore, the orientation of the interest point is computed so that all the future operations are performed on image data that has been transformed according to the estimated keypoint orientation, scale and location. Finally, the descriptor is calculated by binning the gradient orientations of the neighboring pixels.

Scale-space extrema detection

The first step of the algorithm involves searching for candidate image locations that are stable across different scales. This is achieved by applying the *scale-space* theory. More specifically, an image set is obtained by repeatedly filtering the original image with a smoothing Gaussian[2] derivative kernel of increasing size.
Each Gaussian-filtered image $L(x, y, \sigma)$ in the scale space is the result of the convolution between the original image $I(x, y)$ and a Gaussian derivative kernel $G(x, y, \sigma)$, that is,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y).$$

[2] In [60], it is shown that the scale-space representation must satisfy the heat diffusion equation. Since the Gaussian function is the Green's function of the heat diffusion equation on an infinite domain, the only possible kernel for generating the scale space is a Gaussian function or its derivatives.



(a) (b)

Figure 3.1: (a) Scale-space pyramid (b) The extremum is found by comparison with its neighbors at its own scale as well as the two adjacent scales. (Pictures taken from [78].)

According to the order of derivation used, different structures can be found at the extrema of the filtered responses. First order derivatives are used to find corners, while second order derivatives are used to identify blobs. In [84], the scale-normalized Laplacian of Gaussian (LoG) operator, which is a second order derivative operator, is found to be the most stable. In SIFT, an approximation of this operator, namely the Difference-of-Gaussian (DoG) operator, is used. The DoG is defined as the difference of two Gaussian functions with scales differing by a factor of k. By applying the DoG operator to the image, a band-pass filtered version of the image $D(x, y, \sigma)$ is computed as

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{3.1}$$

The choice of the DoG function benefits from two facts. First, the DoG operator is closely related to the scale-normalized LoG function. Secondly, the DoG works by performing a simple image subtraction instead of a more computationally demanding differentiation. For greater efficiency, the DoG pyramid is built by convolving the original image repeatedly with Gaussian filters of increasing size, where the size factor between consecutive DoG images is k. Once a complete octave has been processed, *i.e.*, when the filter size has doubled its starting value, the image is down-sampled by a factor of 2 and the process is repeated for a fixed amount of octaves, as exemplified in Figure 3.1. In order to detect the extrema, a simple comparison of each pixel to its 26 neighbors in a $3 \times 3 \times 3$ cube is done, as shown

in Figure 3.1.

Keypoint localization

The candidate locations found in the previous step suffer from errors due to the discretization of the image domain as well as of the scale space. In order to increase the accuracy in location and scale, a 3D quadratic function is fit to the nearby data. This is beneficial not only because of the increased accuracy, but it also permits to reject points that have a low contrast or that are weakly located on an edge. More specifically, a Taylor expansion of the DoG image $D(x, y, \sigma) = D(\mathbf{x})$, shifted so that the origin is at the candidate point location, results in

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}.$$

By setting the derivative of this expansion to zero, the function extremum is located at $\hat{\mathbf{x}} = -\left(\frac{\partial^2 D}{\partial \mathbf{x}^2}\right)^{-1}\frac{\partial D}{\partial \mathbf{x}}$. By substituting this value into the expansion, we obtain the value of the function at $\hat{\mathbf{x}}$ as $D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}}$.
Subsequently, points that have a low contrast are rejected by comparing $D(\hat{\mathbf{x}})$ with a predefined threshold. By mimicking the Harris corner algorithm, the Hessian matrix of the DoG image is computed at the candidate point to reject edge structures. More specifically, the Hessian matrix $\mathbf{H}$ is defined as

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}.$$

The matrix eigenvalues α and β are proportional to the principal curvatures of D. Since the goal is to reject points located on an edge, the computation of the eigenvalue ratio $r = \alpha/\beta$ suffices. The ratio can be computed directly from the matrix entries, according to the following formula

$$\frac{\mathrm{Tr}(\mathbf{H})}{\mathrm{Det}(\mathbf{H})} = \frac{D_{xx} + D_{yy}}{D_{xx}D_{yy} - (D_{xy})^2} = \frac{\alpha + \beta}{\alpha\beta} = \frac{(r+1)^2}{r}$$

where Tr and Det are the matrix trace and determinant, respectively. Therefore, points for which this ratio is smaller than a predefined threshold are discarded. To summarize, the candidate points that remain at the end of this step are strongly localized at the center of blob-like structures and show a high contrast.

Orientation assignment

The third step envisages the computation of a main orientation for each candidate point. As the final descriptor will be calculated relatively to this main orientation,

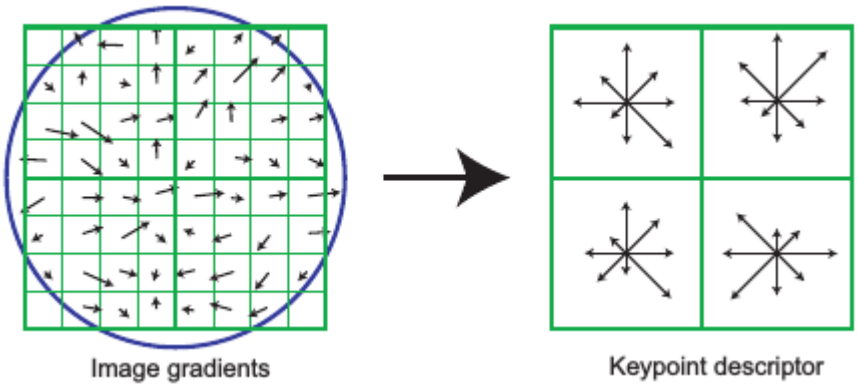


Figure 3.2: (Left) Gradients computed in a square neighborhood of a keypoint weighted by a Gaussian window shown in blue. (Right) Histogram binning of gradient orientations over the four 4×4 sub-regions. The number of sub-regions is reduced for ease of visualization. (Pictures taken from [78].)

rotation invariance is guaranteed. More in detail, the scale of the keypoint is used first to identify the closer Gaussian image $L(x,y)$ in scale space. Then, for each pixel $L(x,y)$ within a circular neighborhood around the keypoint, gradient magnitude and orientation are computed, according to

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

An orientation histogram is formed by finely binning the gradient orientations weighted by their magnitude as well as by a circular Gaussian window, whose size is 1.5 times the size of the keypoint scale. The peak of the orientation histogram identifies the main keypoint orientation and it will be used in the following step to compute the feature descriptor in a rotation-invariant manner. If the orientation histogram envisages multiple significant peaks, a corresponding number of different keypoints will be created.

Keypoint descriptor

Up to now, the interest point has an accurate location at (x, y, σ) and is provided with a main orientation. In this last step, a descriptor for the keypoint is assembled. The key idea is to represent the information in the neighborhood of the point in another, coarser gradient orientation histogram. More precisely, gradient

magnitudes and orientations are collected from a 16×16 patch around the keypoint. The patch is rotated with respect to the main orientation and its size is proportional to the keypoint scale. In each 4×4 sub-region, a 8-bin histogram of gradient orientations is computed, as illustrated in Figure 3.2. The resulting descriptor is formed by concatenating the 16 histograms for a total size of 128 elements. The benefit of aggregating gradient information over regions is that the descriptor will be more robust to small image perturbation and noise in the gradient computation.

3.2 SURF - Speeded Up Robust Features

Similar to SIFT, SURF transforms the image into a set of features that are invariant to translation, in-plane rotation and scaling. The SURF algorithm can be split into the same four steps as SIFT: candidate point identification, refined localization, computation of the main interest point orientation, and descriptor calculation. The differences lie in the technical tools used to compute each step. SURF uses an approximation of the Hessian matrix instead of the DoG to find candidate points, while it uses the same algorithm to refine their location. Instead of using local gradients for the computation of the main orientation and the final descriptor, SURF is based on the patch responses to a set of oriented Haar wavelets.

Scale-space extrema detection

By mimicking the LoG approximation introduced in SIFT, the keypoint detector used in SURF employs an approximation of another blob-detector, the Hessian matrix. This matrix is based on the responses of the image after filtering with a Gaussian second order derivative, that is,

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

where $L_{xx}(\mathbf{x}, \sigma) = \frac{\partial^2 G(\sigma)}{\partial x^2} * I(\mathbf{x})$.
The approximation is performed by discretizing and cropping the Gaussian second order derivative filters to obtain simpler *box filters*, as illustrated in Figure 3.3.
The advantage of using box filters is that they can be coupled with integral images for a fast computation. As defined in [124], an integral image $I_\Sigma(x, y)$ is defined as

$$I_\Sigma(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i, j),$$

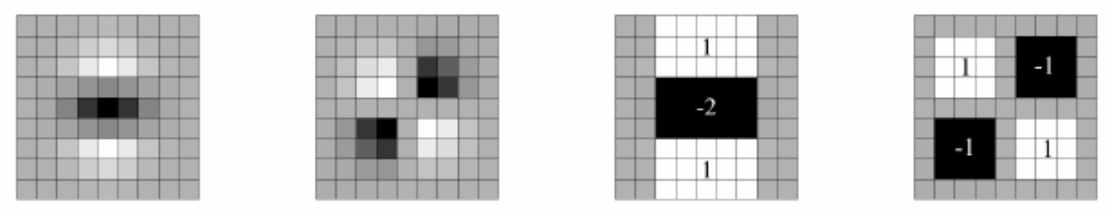


Figure 3.3: Gaussian second order partial derivative in y-direction (L_{yy}) and xy-direction (L_{xy}). The corresponding box filters. Each entry in the black regions has weight -1, in the white regions $+1$, and in the grey regions has value 0. (Picture taken from [7].)

that is, the accumulation of all the pixel values in the rectangle that has the image origin as upper left corner and the point (x, y) as bottom right corner. The advantage of using integral images is that the sum of intensities over a rectangular area of any size can be computed in only 4 additions. Therefore, its direct application for box filtering makes SURF computationally very efficient.
The algorithm starts with a filter box of size 9×9 that corresponds to a discretized Gaussian second order derivative of $\sigma = 1.2$. In order to keep the filter weights balanced after discretization, the diagonal components of the Hessian matrix are multiplied by a ratio of $\frac{|L_{xy}(1.2)|_F |D_{xy}(9)|_F}{|L_{xx}(1.2)|_F |D_{xy}(9)|_F} = 0.9$. Following the scale-space theory, increasingly filtered versions of the original image must be created. The image is not repeatedly filtered as in SIFT, but the scale-space is obtained by simply increasing the filter size. Again, the usage of box filters and integral images is computationally beneficial, as the size of the filter does not affect the operation cost. More specifically, several filtering octaves are envisaged, where now the octave identifies an interval between two filters whose sizes are in a ratio of 2:1.
Unlike SIFT, the candidate keypoints are detected as those points at which the determinant of the Hessian achieves a maximum in a $3 \times 3 \times 3$ neighborhood. The location and scale is accurately refined using the same algorithm explained in the SIFT section.

Orientation assignment

As in SIFT, each keypoint must be provided a reproducible orientation. The difference is that in SIFT gradients in the neighborhood were binned in a fine histogram, while in SURF the main orientation is estimated using Haar wavelets and a sliding orientation window.
More specifically, for each sampled point in a circular window around the keypoint, the responses to x- and y-oriented Haar wavelets are computed. Then, these responses are weighted with a Gaussian centered at the keypoint location and

(a) (b)

Figure 3.4: (a) Haar wavelets in the x and y direction. (b) Sliding orientation window identifies the main orientation of the Gaussian weighted responses. (Pictures taken from [7].)

represented as points in a diagram by mapping the corresponding responses with respect to two orthogonal axes. As shown in Figure 3.4, a sliding window of size $\pi/3$ is used to compute the sum-of-responses vector. The main orientation is chosen as the longest sum-of-responses vector.

Keypoint descriptor

A square patch around the keypoint is rotated with respect to the main orientation, and for each sampled pixel in this patch four terms are computed: dx, dy, $|dx|$ and $|dy|$. The first two terms are the signed responses to the same two Haar wavelets, now rotated with respect to the main orientation, while the second two are their unsigned counterparts. The patch is divided in sixteen 4×4 sub-regions and the 4-dimensional vectors are accumulated. The final descriptor is the concatenation of these accumulations for a total length of 64 elements.

Chapter 4

Pose Estimation with Feature Regression

In this chapter, we introduce the core of the method proposed in this thesis. First, we present the building block of our approach, the so-called generative feature model, *i.e.*, a regression function that predicts the feature descriptor components as a function of the viewpoint. Then, we describe how to use generative feature models in order to solve the pose estimation problem for a single, specific object.
In the second part of this chapter, we extend the formulation developed for the single case to the class case, where only the object membership, but not its identity, is known. In order to do so, we first show how to measure the similarity between generative feature models collected from different objects and how to effectively aggregate them into meaningful clusters. Then, we formulate the pose estimation problem in terms of the model clusters by embedding them into a probabilistic framework that permits to obtain a posterior distribution for the estimated pose. Finally, we substantiate the effectiveness of our method with a set of experiments on two publicly available datasets.

4.1 Feature Regression and Generative Feature Models

When we have introduced appearance-based features in Chapter 2, we have highlighted that one of their main advantages is that they are designed to be robust to many image transformations. In Chapter 3, we presented two specific appearance-based features, *i.e.*, SIFT and SURF, and we showed in technical details how these features are made invariant towards scale changes, translation, in-plane rotation, and illumination changes.
As a matter of fact, all appearance-based features that have been proposed in the literature up until the time of writing are not invariant to perspective transforma-

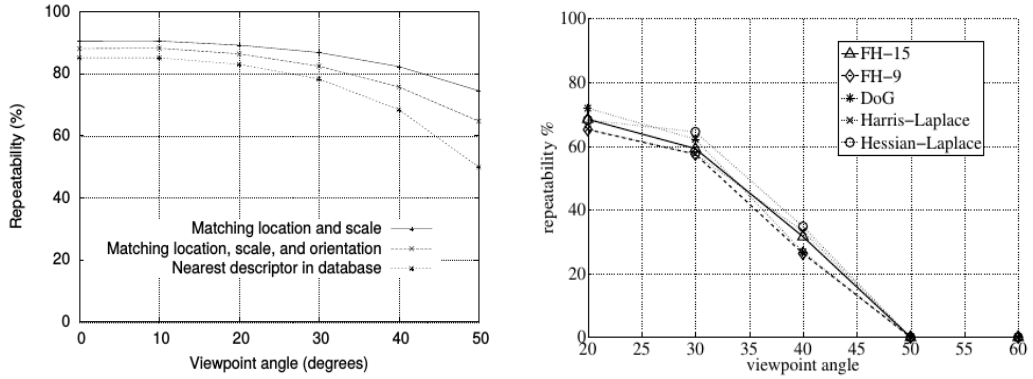


Figure 4.1: Repeatability for SIFT (left) and SURF (right) features. In the right pictures, FH-9 and FH-15 refer to the Fast Hessian detector employed by SURF with initial filter of size 9×9 and 15×15, respectively. (Pictures taken from [78] and [6].)

tions[3]. That is, when the object undergoes an out-of-plane rotation, either due to its own motion or to a change of the camera position/orientation, the descriptors of the corresponding features in the two images will not be equal.

In addition, the detector employed by features like SIFT or SURF can guarantee only some small tolerance to out-of-plane rotations. This tolerance is measured in terms of the so-called repeatability, *i.e.*, the percentage of re-detected features as a function of the viewpoint change. As shown in Figure 4.1, repeatability decreases to around 70-80% after a viewpoint rotation of only 20° for both features, and it often depends on the evaluation dataset.

The fact that appearance-based features are not perspective invariant is a strong limitation *per se*. However, there is an additional, experimental fact about the way they change with respect to a perspective transformation that turns this seeming weakness into a useful cue. Let us focus our attention to the way features descriptors change with respect to viewpoint. As an example, Figure 4.2 shows the behavior of the components of the SIFT descriptor of a certain patch with respect to camera rotation. The variation in the components amplitude appears smooth when the camera orientation changes. So, it comes natural to think about the possibility to predict the feature descriptor components as a function of the viewpoint, if a sufficient amount of view-labelled training descriptors for the same patch is given. On the other hand, if we reverse the task, *i.e.*, we want to estimate

[3]In the work of Köser *et al.* [62], the authors provide a method for the generation of 3D perspectively invariant features on the basis of a depth map and a set of views of a planar or panoramic scene. Nevertheless, the assumption of having a depth camera or a stereo rig is not considered here, where only single RGB images are used.

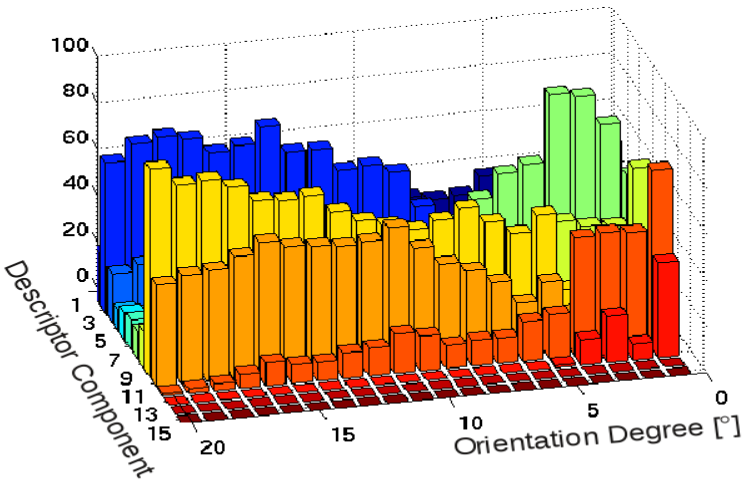


Figure 4.2: The first 15 components, each represented in a different color, of a SIFT descriptor of the same patch undergoing a rotational motion. The variation in the component magnitude as function of the pose change appears to be smooth.

the viewpoint from which a certain descriptor has been observed, we can think that the most likely viewpoint is the one that yields the smallest error between the predicted and the current descriptor. This fundamental intuition will turn out to be extremely advantageous for our pose estimation task and is at the basis of the building block of our method, the generative feature model.

4.2 Generative Feature Models

The generative feature model is a regression function that models the feature descriptor of a certain patch as a function of the viewpoint. In developing the generative feature models, we drew inspiration from the work of [115], where regression models are learned for small, individual, grey-valued patches. In that work, the authors consider the specific task of robot pose estimation, the so-called visual odometry, for autonomous robot navigation. On the contrary, we propose in this thesis to learn regression models of local features, because this is a more general, robust, and effective way to describe and predict the appearance of a generic object than mere raw patches. By doing so, we are able to go beyond the original framework, as we provide a method that not only estimates the orientation of an object by means of the learned regression models, but can also generalize to other objects that belong to the same class.

Each generative feature model describes the appearance of a single patch, and it is learned from a set of training pairs, where each training pair is composed of a feature descriptor $\mathbf{f}$ and a viewpoint $\boldsymbol{\alpha}$ from which the descriptor has been extracted[4].

Let T^i be the augmented feature track for the 3D planar patch P^i, *i.e.*, a set of n pairs composed by the features $\{\mathbf{f}^i\}_{i=1}^n$ extracted from the same patch under different viewpoints and the viewpoint labels $\{\boldsymbol{\alpha}^i\}_{i=1}^n$ themselves, *i.e.*,

$$T^i = \{(\mathbf{f}_1^i,\boldsymbol{\alpha}_1^i),(\mathbf{f}_2^i,\boldsymbol{\alpha}_2^i),\ldots,(\mathbf{f}_n^i,\boldsymbol{\alpha}_n^i)\}, \tag{4.1}$$

where $\mathbf{f}_j^i$ is the k-dimensional descriptor representing the neighborhood of patch P^i when the viewpoint is $\boldsymbol{\alpha}_j^i$.

On the basis of the smoothness that feature descriptors exhibit with respect to a change in viewpoint, it is possible to design a vector-valued function, our *generative feature model*, $F^i : \mathbb{R}^m \rightarrow \mathbb{R}^k$, where m is the dimensionality of the pose space and k is the dimensionality of the feature descriptor. The function F acts as a mapping between pose and feature descriptor space, *i.e.*, it predicts the feature descriptor of the corresponding patch given a viewpoint as input. Thus, by learning such a function, we are able to yield a descriptor estimate under the viewpoint $\boldsymbol{\alpha}$, $F^i(\boldsymbol{\alpha}) = \hat{\mathbf{f}}^i$, for each training track T^i. In order to learn this regression function, we turn to the Radial Basis Function (RBF) network theory [9].

Radial basis function networks are simple artificial neural networks that have been developed for function approximation, but have also found application in other fields like classification, time series prediction, and system control. In a nutshell, the function approximation given by a RBF network is expressed as a weighted linear combination of n non-linear basis functions, whose weights are learned in a training procedure. In our case, the basis functions, also known as kernels, have a Gaussian shape, so that the output of the RBF network for each patch P^i can be written as

$$F^i(\boldsymbol{\alpha}) = \sum_{j=1}^{n} \mathbf{w}_j^i G(\boldsymbol{\alpha}, \boldsymbol{\alpha}_j^i) \tag{4.2}$$

where $\mathbf{w}_j^i$ are the k-dimensional vector coefficients estimated from T^i during the learning of the RBF network, and $G : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ is the Gaussian kernel of the RBF network defined as,

$$G(\boldsymbol{\alpha},\boldsymbol{\alpha}_j^i) = G(\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_j^i\|) = \exp\left(-\frac{\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_j^i\|^2}{\sigma^2}\right) \tag{4.3}$$

where $\|\cdot\|$ represents a suitable pose distance metric, and σ is the kernel bandwidth. In the following section, we describe in details how to train the RBF network in order to compute the coefficients $\mathbf{w}_j^i$ of the generative feature model.

[4]For sake of generality, we assume that $\boldsymbol{\alpha}$ is a m-dimensional vector, where $m \in \{1,2,3\}$, although in most of this thesis $\boldsymbol{\alpha}$ will be a scalar α.

4.2.1 Generative Feature Model as Radial Basis Function Network

In the RBF network literature, training the network means to find the multidimensional surface that best fits the training data and to express it as a weighted combination of activation functions that act as the *basis* functions of the network. Additionally, we impose a further training constraint that will be justified in the following, *i.e.*, we impose that the interpolating surface must pass through all training data points, which leads us to a strict, multi-variable interpolation problem. Now, let us see how we can learn our generative feature model F by training the corresponding RBF network[5].
On the basis of an augmented track T, we are given a set of n different training viewpoint labels $\{\boldsymbol{\alpha}_j\}_{j=1}^n \subset \mathbb{R}^m$ and a corresponding set of n feature descriptors $\{\mathbf{f}_j\}_{j=1}^n \subset \mathbb{R}^k$. Training the RBF network means to find a function $F : \mathbb{R}^m \to \mathbb{R}^k$, such that

$$F(\boldsymbol{\alpha}) = \sum_{j=1}^{n} \mathbf{w}_j G(\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_j\|) \qquad \text{and} \qquad F(\boldsymbol{\alpha}_j) = \mathbf{f}_j \tag{4.4}$$

where the known n training viewpoint labels are taken to be the centers of the radial basis functions. We can combine the two conditions in Equation (4.4) and express them in matrix form as

$$\mathbf{GW} = \mathbf{Z} \tag{4.5}$$

where

$$\mathbf{G} = \begin{bmatrix} G_{11} & \cdots & G_{1n} \\ \vdots & \ddots & \vdots \\ G_{n1} & \cdots & G_{nn} \end{bmatrix}, \tag{4.6}$$

with $G_{pq} = G(\|\boldsymbol{\alpha}_p - \boldsymbol{\alpha}_q\|)$, and

$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nk} \end{bmatrix}, \qquad \mathbf{Z} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_n \end{bmatrix} = \begin{bmatrix} f_{11} & \cdots & f_{1k} \\ \vdots & \ddots & \vdots \\ f_{n1} & \cdots & f_{nk} \end{bmatrix} \tag{4.7}$$

If $\mathbf{G}$ is non-singular, the matrix of training coefficients $\mathbf{W}$ can be found as

$$\mathbf{W} = \mathbf{G}^{-1}\mathbf{Z}. \tag{4.8}$$

The non-singularity of $\mathbf{G}$ is guaranteed by the following theorem [83]:

[5] We have dropped the superscript i for sake of clarity, but everything described in this section refers to a single generative feature model F^i.

Theorem 1: Micchelli's theorem. *Let* $\{\mathbf{x}_i\}_{i=1}^n$ *be a set of distinct points. Then, the* $n \times n$ *interpolation matrix* Φ *where* $\phi_{ij} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$ *is non-singular.*

Many functions are covered by Micchelli's theorem and are commonly used in RBF networks, such as

- Multiquadrics: $\phi(r) = \sqrt{r^2 + c^2}$, for some $c > 0$.
- Inverse multiquadrics: $\phi(r) = \frac{1}{\sqrt{r^2+c^2}}$, for some $c > 0$.
- Gaussian: $\phi(r) = e^{-\frac{r^2}{\sigma^2}}$, for some $\sigma > 0$.

As we said in Section 4.2, we opt for a Gaussian kernel in this thesis. Therefore, the output of the RBF network $F(\boldsymbol{\alpha})$ is expressed by a linear combination of the vector weights learned in Equation (4.8) and the Gaussian radial basis functions. The plain least-square approach described above is not usually a good strategy, because unavoidable noise in the training data increases the uncertainty in the estimated input-output mapping. In our case, training data can be affected by camera noise, feature descriptor quantization, imperfect pose labelling, and outliers in the feature track T.
A well-known countermeasure is based on the introduction of a regularization term, whose goal is to stabilize the solution by means of an additional functional that contains some prior information on F. A very common prior, that we also adopt in this thesis, is that the input-output mapping is smooth. According to Tikhonov's regularization theory [94], the optimal values for $\mathbf{w}_j$ can be obtained from

$$(\mathbf{G} + \lambda\mathbf{I})\mathbf{W} = \mathbf{Z} \quad \Rightarrow \quad \mathbf{W} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{Z} \tag{4.9}$$

where λ is the regularization parameter and $\mathbf{I}$ is the $n \times n$ identity matrix.
From a different perspective, RBF networks can also be considered as fully connected artificial neural networks with only three layers:

- The input layer is composed of a set of source nodes, where each node takes the value of the corresponding input component.
- The second layer is the hidden layer in the network and it is composed by the activation functions that apply a non-linear transformation to the input signal.
- The third layer provides the response of the network to the input as a weighted linear combination of the hidden node responses.

An example of a RBF network as an artificial neural network is graphically represented in Figure 4.3.

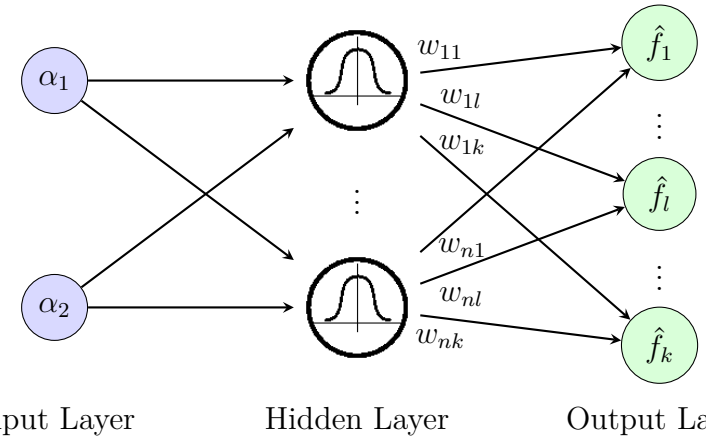


Figure 4.3: A 2-dimensional vector $\boldsymbol{\alpha} = (\alpha_1,\alpha_2)$ is the input of the depicted three-layer RBF network with Gaussian kernel and n hidden nodes. The RBF network output is a k-dimensional vector $\hat{\mathbf{f}} = \{\hat{f}_1, \ldots, \hat{f}_l, \ldots, \hat{f}_k\}$.

4.3 Estimate the Pose of a Single Object Instance

In this section, we focus on estimating the pose of a single, known object. This can be considered as an introductory test to verify the assumption that features are informative with respect to viewpoint and generative feature models can capture and reproduce this informativeness. As discussed in Chapter 2, we first introduce our training stage, where we learn the generative feature models for the object at hand, then we proceed to estimate its pose in the testing stage.

Training stage

We assume that we are given a set of training images $\mathcal{I} = \{I_i\}_{i=1}^{n}$, where the object is depicted from different points of view and that we know the viewpoint of each image. The goal of the training stage is to collect a set of augmented feature tracks from the training images and build a generative feature model for each augmented track.

In order to collect the augmented feature tracks, we first start by matching each pair of images on the basis of the chosen feature. That is, we extract two sets of features, one per image, and we find for each feature in the first set the nearest neighbor feature in the second set. This simple matching strategy can lead to many wrong matches as many background features will also be included in the matching set. As suggested in [78], an empirically effective solution is to filter the resulting matches with a ratio test based on the distance that each feature in the first set has to its first and second nearest neighbor in the second set. That is, we

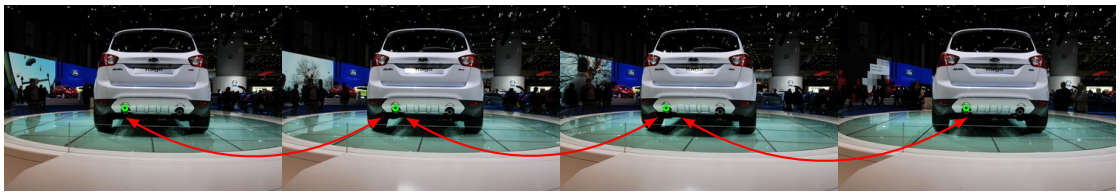


Figure 4.4: One feature track T^i associated to different views of the left exhaust pipe (green circles). T^i contains four feature descriptors and relative viewpoint labels.

reject all the matches for which

$$\frac{d_{1\text{NN}}}{d_{2\text{NN}}} > \tau \tag{4.10}$$

where $d_{1\text{NN}}$ and $d_{2\text{NN}}$ are the distances to the first and the second nearest neighbor, respectively, and τ is a threshold that [78] suggests to set at 0.7.
In order to further reduce the number of wrong matches, we apply a geometric filter that rejects wrong feature matches on the basis of epipolar geometry. Given the set of matches, we compute the fundamental matrix $\mathbf{F}$, that is the matrix that verifies the following equality for each match

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0 \tag{4.11}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the locations of the matching features of the first and second image, respectively, expressed in homogeneous coordinates. Since matches are contaminated by noise in the feature location as well as outliers, Equation (4.11) has normally no solution. We circumvent this by turning to a robust parameter estimation method, RANSAC [31], that finds the fundamental matrix $\widetilde{\mathbf{F}}$ with the largest consensus among the matches. On the basis of this fundamental matrix, we discard the matches whose reprojection error, $\mathbf{x}_1^T \widetilde{\mathbf{F}} \mathbf{x}_2$, is larger than a small threshold (3 pixels).
By connecting matches that share one of the two matching features, we build a set of feature tracks, where each feature track T^i is a set composed of matching feature descriptors, $T^i = \{\mathbf{f}_1^i, \mathbf{f}_2^i, \ldots, \mathbf{f}_n^i\}$. We finally obtain the set of *augmented* feature tracks, by pairing each descriptor with the viewpoint of the image in which it has been extracted, that is,

$$T^i = \{(\mathbf{f}_1^i, \boldsymbol{\alpha}_1^i), (\mathbf{f}_2^i, \boldsymbol{\alpha}_2^i), \ldots, (\mathbf{f}_n^i, \boldsymbol{\alpha}_n^i)\}, \tag{4.12}$$

as also defined Section 4.2. An example of a feature track is shown in Figure 4.4.

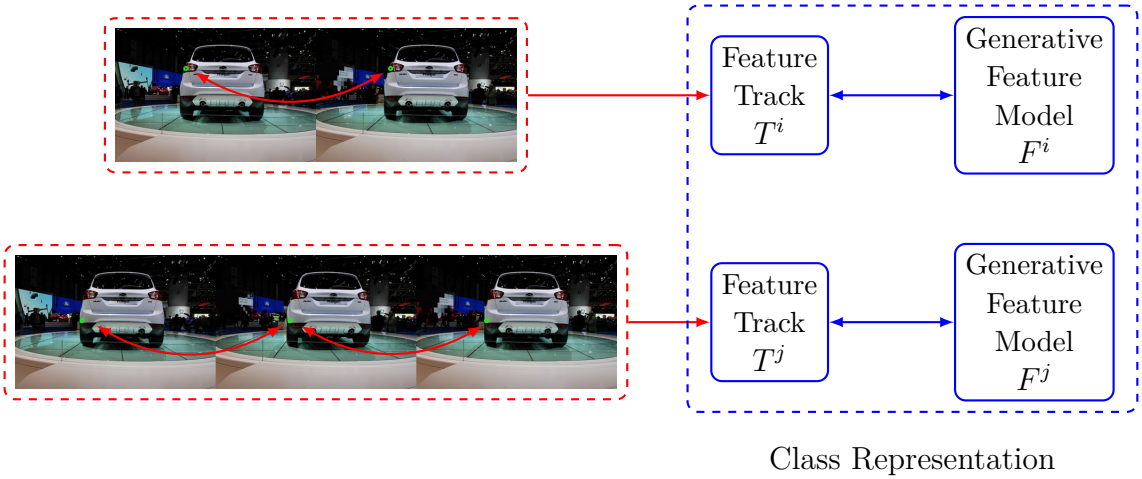


Figure 4.5: The training stage in case of a single object. Features are tracked over training images and collected in augmented feature tracks T. For each feature track, a generative feature model F is computed. The whole set of augmented feature tracks and associated generative feature models form our class representation.

The final training step amounts to compute a generative feature model F^i for each augmented track T^i. In order to do so, we use the method explained in Section 4.2.1. That is, we compute the following RBF network,

$$F^i(\boldsymbol{\alpha}) = \sum_{j=1}^{n} \mathbf{w}_j^i G(\boldsymbol{\alpha}, \boldsymbol{\alpha}_j^i) \tag{4.13}$$

where, again, $\mathbf{w}_j^i$ are the vector weights estimated during learning, and G is the Gaussian kernel of the network. As mentioned in Section 4.2.1, we use all the training labels as centers of the radial basis function. This is motivated by the fact that tracks have usually a short length, as feature detectors have a low repeatability when the object undergoes an out-of-plane rotation. A block diagram of the whole training stage is depicted in Figure 4.5.

Testing stage

In the testing stage, we assume that we are given a query image I_q that depicts the object at hand in an unknown pose. According to our assumption that feature descriptors are informative about the viewpoint of the object, we can think that if the set of feature descriptors $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^N$ is found in the current query image, we can define a probability function $p(\boldsymbol{\alpha}|I_q) = p(\boldsymbol{\alpha}|\mathcal{Q})$ that expresses the likelihood of the pose $\boldsymbol{\alpha}$ when $\mathcal{Q}$ is observed.

Applying Bayes' Rule to $p(\boldsymbol{\alpha}|\mathcal{Q})$, we obtain the following

$$p(\boldsymbol{\alpha}|\mathcal{Q}) = \frac{p(\mathcal{Q}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(\mathcal{Q})}. \tag{4.14}$$

Our pose estimation problem amounts to finding $\boldsymbol{\alpha}^*$ that maximizes the expression above,

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|\mathcal{Q}) = \arg\max_{\boldsymbol{\alpha}} \frac{p(\mathcal{Q}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(\mathcal{Q})}. \tag{4.15}$$

As $p(\mathcal{Q})$ does not depend on $\boldsymbol{\alpha}$, we obtain the following maximum a posteriori (MAP) estimation problem

$$\begin{aligned} \boldsymbol{\alpha}^* &= \arg\max_{\boldsymbol{\alpha}} \frac{p(\mathcal{Q}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(\mathcal{Q})} \\ &= \arg\max_{\boldsymbol{\alpha}} p(\mathcal{Q}|\boldsymbol{\alpha})p(\boldsymbol{\alpha}) = \arg\max_{\boldsymbol{\alpha}} \prod_{i=1}^{N} p(\mathbf{q}_i|\boldsymbol{\alpha})p(\boldsymbol{\alpha}), \end{aligned} \tag{4.16}$$

where we have assumed full independence between features in the rightmost equality. It is also important to note that this formulation allows to embed a prior on the pose through the term $p(\boldsymbol{\alpha})$.

The likelihood term $p(\mathcal{Q}|\boldsymbol{\alpha}) = \prod_{i=1}^{N} p(\mathbf{q}_i|\boldsymbol{\alpha})$ in Equation (4.16) can be expressed in terms of the generative feature models. In the training stage, we have computed the set of generative feature models $\mathcal{F} = \{F^j\}_{j=1}^{M}$, that is, our representation of the object in terms of feature regressors. If we put each query feature $\mathbf{q}_i$ in correspondence with a generative feature model, we can think that the likelihood of the pose $\boldsymbol{\alpha}$ is reflected by the norm of the regression errors $\mathbf{e}_i^j(\boldsymbol{\alpha}) = \mathbf{q}_i - F^j(\boldsymbol{\alpha})$. That is, if a generative model predicts the feature $\mathbf{q}_i$ with a small error for a certain viewpoint $\boldsymbol{\alpha}$, we assume that $\boldsymbol{\alpha}$ has a high likelihood to be the query viewpoint according to that generative feature model.

In order to put the query feature descriptor $\mathbf{q}_i$ in correspondence with a generative model, we define a representative feature $\mathbf{r}^j$ for each generative model F^j, so that we can define $\mathcal{R} = \{\mathbf{r}^j\}_{j=1}^{M}$ as the set of all the representative features of the generative feature models[6]. Each representative feature is defined as the feature descriptor whose viewpoint is the closest to the center of the viewpoint interval covered by the generative model. Another possible choice could be the average of the feature descriptors contained in the model, but the difference in

[6]Recall that each generative model has been built from a feature track comprising a set of feature descriptors of the same patch extracted at different viewpoints. Therefore, we will sometimes refer to features of the generative model, meaning the features contained in the track from which the generative model has been built. This generates no ambiguity as feature tracks and corresponding generative models are in a one-to-one relationship.

matching performance is negligible as the number of descriptors per track is small and descriptors are discriminative. With this definition for the generative model representatives, we can rewrite the probability $p(\mathbf{q}_i|\boldsymbol{\alpha})$ by including $\mathbf{r}^j$ as

$$p(\mathbf{q}_i|\boldsymbol{\alpha}) = \sum_{j=1}^{M} p(\mathbf{q}_i, \mathbf{r}^j|\boldsymbol{\alpha}) \tag{4.17}$$

We use the regressor function F^j of the generative model to provide an estimation of the likelihood of the feature descriptor $\mathbf{q}_i$ being observed from viewpoint $\boldsymbol{\alpha}$ when it is set in correspondence to $\mathbf{r}^j$. We define $p(\mathbf{q}_i, \mathbf{r}^j|\boldsymbol{\alpha})$ on the basis of the error $\mathbf{e}_i^j(\boldsymbol{\alpha}) = \mathbf{q}_i - F^j(\boldsymbol{\alpha})$ between the query feature and the descriptor estimated by the corresponding generative feature model as well as a compatibility term between $\mathbf{q}_i$ and $\mathbf{r}^j$. Therefore, the observation error is defined as

$$p(\mathbf{q}_i, \mathbf{r}^j|\boldsymbol{\alpha}) = \gamma_{\mathbf{q}_i\mathbf{r}^j} \exp\left(-\frac{1}{2}(\mathbf{e}_i^j(\boldsymbol{\alpha}))^T(\mathbf{R}^j)^{-1}\mathbf{e}_i^j(\boldsymbol{\alpha})\right) \tag{4.18}$$

where $\mathbf{R}^j$ is the error covariance matrix of the j-th regressor estimated by leave-one-out cross-validation from the training samples. In order to reduce the number of tentative representative models, we rely on feature discriminativeness by defining $\gamma_{\mathbf{q}_i\mathbf{r}^j}$ as

$$\gamma_{\mathbf{q}_i\mathbf{r}^j} = \begin{cases} 1 & \text{for } \tilde{\mathbf{r}}^i \text{ s.t. } \|\mathbf{q}_i - \tilde{\mathbf{r}}^i\| = \min_{\mathbf{r}^j} \|\mathbf{q}_i - \mathbf{r}^j\| \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

To summarize, we find for each feature in $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^N$ the nearest neighbor model representative in $\mathcal{R} = \{\mathbf{r}^j\}_{j=1}^M$, so that we have a correspondence between each query feature $\mathbf{q}_i$ and a generative model through its representative feature $\tilde{\mathbf{r}}^i$. Then, we define the following maximum a posteriori estimation for the viewpoint $\boldsymbol{\alpha}^*$ by replacing Equation (4.17) in Equation (4.16).

$$\begin{aligned} \boldsymbol{\alpha}^* &= \arg\max_{\boldsymbol{\alpha}} p(\mathcal{Q}|\boldsymbol{\alpha})p(\boldsymbol{\alpha}) \\ &= \arg\max_{\boldsymbol{\alpha}} \prod_{i=1}^{N}\sum_{j=1}^{M} p(\mathbf{q}_i, \mathbf{r}^j|\boldsymbol{\alpha})p(\boldsymbol{\alpha}) = \arg\max_{\boldsymbol{\alpha}} \prod_{i=1}^{N} p(\mathbf{q}_i, \tilde{\mathbf{r}}^i|\boldsymbol{\alpha})p(\boldsymbol{\alpha}). \end{aligned} \tag{4.20}$$

A graphical interpretation of the testing stage is provided in Figure 4.6.

4.3.1 Introductory Experiment

Here, we will perform an introductory experiment in order to prove the efficacy of the method proposed above. The main purpose of this short experiment is to show that the trained generative feature models correctly regress feature descriptors and

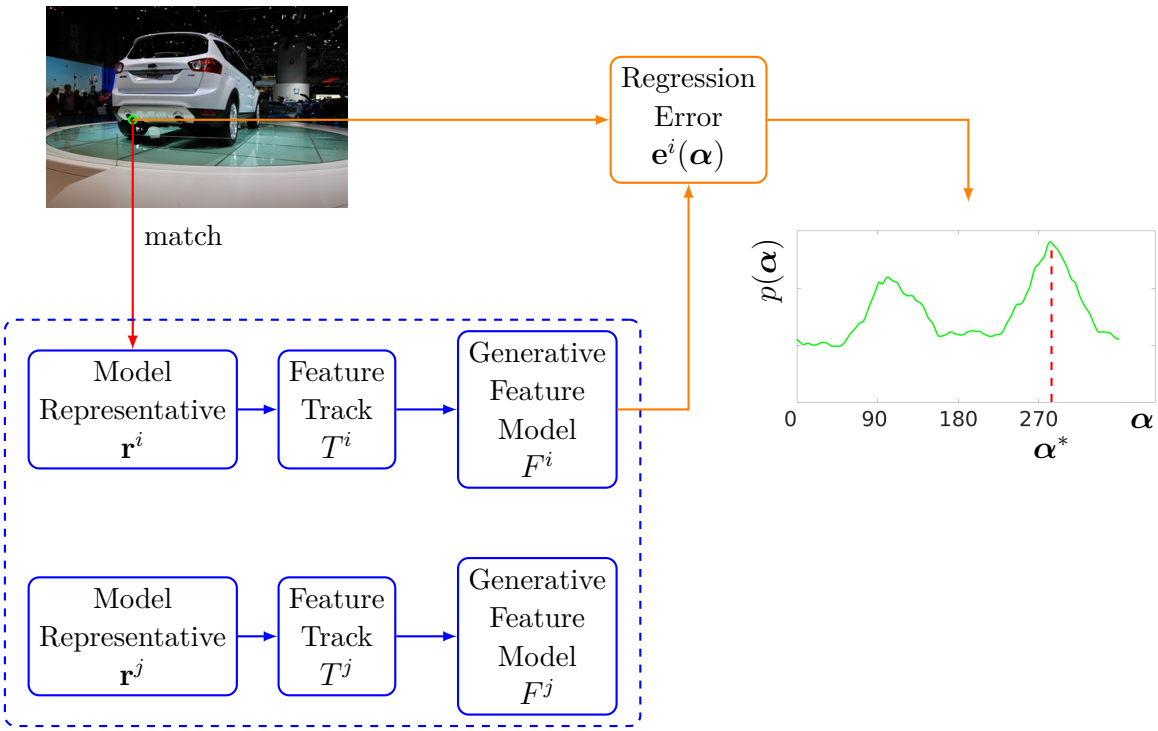


Figure 4.6: The testing stage in case of a single object. Query features are extracted from the test image and each query descriptor is matched to its nearest neighbor track T^i through the model representative $\mathbf{r}^i$. The corresponding generative feature model predicts through regression an estimate of the query feature descriptor as a function of the putative pose $\boldsymbol{\alpha}$. The norm of the error $\mathbf{e}^i(\boldsymbol{\alpha})$ between the query descriptor and the predicted descriptor is used to build a posterior probability distribution, whose maximum is returned as the estimated pose of the object.

this permits to reliably estimate the pose of a single, specific object in a query image, as described in Section 4.3.
In order to do so, we consider the EPFL multi-view car dataset [90]. This dataset provides pose-labelled sequences of cars rotating on a floor pedestal. The sequences are densely sampled in the pose space, so it is possible to collect several features that represent the same patch under different viewpoints.
We consider the first 10 car sequences and, for each, we learn a set of generative models using a 33% split, *i.e.*, one image every three is used for learning and the rest for testing. By doing so, we want to evaluate if the generative models that we learn from a subset of the training images permit to evaluate the pose of the same object in the remaining images.
As explained in Section 4.3, we extract a set of local features from the training images and we form a set of model tracks $\mathcal{T}$ by tracking the features over the training views. Matches are verified and filtered through epipolar geometry with a very low threshold in order to reduce the number of outliers to a minimum. For each track, a generative feature model is learned as described in Section 4.2. At run-time, a set of features $\mathcal{Q}$ is extracted from the query image and matched against the set of model representatives defined in Section 4.3. Finally, we return the maximum a posteriori estimation of the pose by using Equation (4.20). In this experiment, the pose prior is assumed to be uniformly distributed over the entire pose space.
We evaluated the performance of our method by using two different feature descriptors and detectors, SIFT [78] and SURF [6], in all four possible configurations:

(a) SIFT detector + SIFT descriptor

(b) SIFT detector + SURF descriptor

(c) SURF detector + SIFT descriptor

(d) SURF detector + SURF descriptor

In Table 4.1, we present an evaluation of our method with respect to the four configurations in terms of the mean absolute error (MAE). The MAE is defined as the average of the absolute difference between the returned value and the ground truth calculated on each sequence, *i.e.*

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{\alpha}_i^* - \boldsymbol{\alpha}_i^{\mathrm{gt}}\| \tag{4.21}$$

where n is the number of images in the sequence, $\boldsymbol{\alpha}^*$ is the pose returned by our algorithm, and $\boldsymbol{\alpha}^{\mathrm{gt}}$ is the ground-truth pose.

Table 4.1: Performance of the proposed method with respect to four configurations of feature detector and descriptor. The configuration SURF detector + SIFT descriptor obtains the best mean absolute error (MAE).

	Configurations			
	SIFT+SIFT	SIFT+SURF	SURF+SIFT	SURF+SURF
Sequence 1	1.09	6.30	**1.01**	1.33
Sequence 2	1.38	20.30	**1.11**	1.53
Sequence 3	**0.90**	3.18	**0.90**	1.17
Sequence 4	0.75	4.20	**0.65**	0.93
Sequence 5	10.12	27.70	**1.82**	1.95
Sequence 6	**1.08**	3.21	1.22	1.80
Sequence 7	**0.80**	8.88	0.82	1.03
Sequence 8	0.66	10.80	**0.65**	0.83
Sequence 9	3.94	9.55	**1.07**	1.24
Sequence 10	2.81	18.43	**0.69**	1.07
MAE	2.35	11.26	**0.99**	1.29

In order to put the performance of our method in perspective, we must say that each sequence is different in length, ranging from 60 to 140 images, and thus the distance between consecutive training samples ranges from 7.5° to 18°, approximately. As the dataset presents only a one-dimensional pose variation, we just evaluate the mean absolute error for the azimuth component.
As we can see in Table 4.1, the combination of SURF detector and SIFT descriptor achieves the best performance with an average MAE over the whole set of sequences of 0.99°. The reason for this is that the SURF detector provides more keypoints with respect to the SIFT counterpart, while the description of the gradient distribution employed by SIFT is finer, and thus more precise than that of SURF. In comparison to [118], where only the result for the first sequence is provided, we improve the pose accuracy by approximately 45% (1.01° vs. 1.84°).

4.4 From Single Instance Prediction to Object Class Prediction

We proved in the last section that the proposed method achieves state-of-the-art results for the pose estimation of single, specific objects. In this section, we generalize from a single instance to a whole class, *i.e.*, we learn a representation that permits to compute the pose of an unknown object of a given class. We

assume that the features describing analogous structures of objects belonging to the same class are similar. For example, we assume that the features describing car wheels are sufficiently similar independently from the specific model under consideration. This underlying assumption has been adopted and experimentally verified by a plethora of approaches [25, 16, 112, 43, 66].
In order to be robust against intra-class variations in appearance and geometry, we should aggregate different instances of the same class. For example, if our goal is to estimate the pose of an unknown car, our training dataset should comprise different car models ranging from city cars and sedans to station wagons and sport cars. When multiple training instances are considered, the huge number of augmented feature tracks makes the class representation described in Section 4.2 increasingly harder to handle. Therefore, a more compact class representation becomes necessary.
For this purpose, we propose to cluster feature tracks in a consistent way, *i.e.*, we group feature tracks that represent similar patches under similar viewpoints. This means that tracks in the same cluster will have similar feature descriptors as well as a similar viewpoint interval. Consequently, we will obtain clusters that represent local object structures in a more general way, and the clustered representation will be more robust against intra-class variations of appearance and geometry. We resort to spectral clustering, a technique that permits to perform grouping on the basis of the elements connectivity instead of their geometrical proximity. Before illustrating the clustering algorithm, we first need to define a similarity score that measures the affinity between each pair of feature tracks. We leverage from the dynamic time warping technique to compute a similarity score for our feature tracks. In the following, we present a brief treatment of dynamic time warping and how we use it to compute track similarity. Then, we will present the spectral clustering algorithm with its theoretical fundamentals.

4.4.1 Dynamic Time Warping for Track Similarity

Dynamic time warping (DTW) is a well-known technique from time-series analysis that allows to find an optimal alignment between two input sequences [103]. DTW was firstly employed in applications for automatic speech recognition, as it can cope automatically with speakers having different speeds [96]. While it has been used since thirty years in domains such as signal and audio processing as well as data mining, DTW has only recently become popular in the Computer Vision community. Until now, it has been employed in several fields, such as human behavior alignment [91, 130], human motion alignment [131], curve alignment [111], handwriting recognition [97], and novelty detection [2].
Let us consider two input sequences $\mathcal{X} = (x_1, \ldots, x_n, \ldots, x_N)$ of length N and $\mathcal{Y} = (y_1, \ldots, y_m, \ldots, y_M)$ of length M. Let us also assume that the components

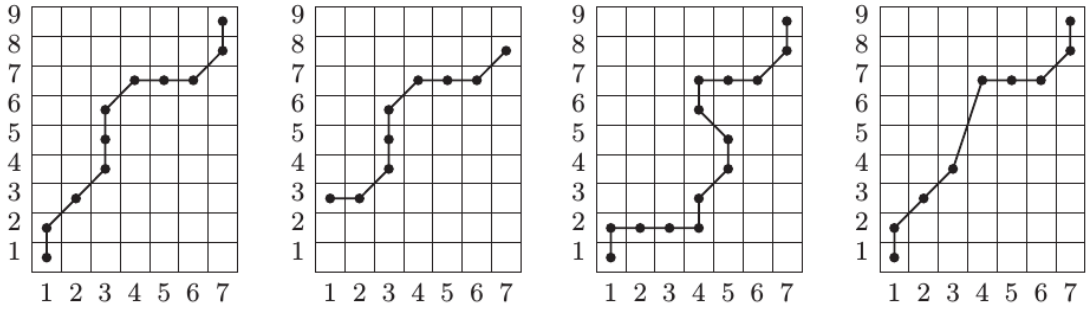


Figure 4.7: Four possible paths within the same cost matrix $\mathbf{C}$ that align two sequences $\mathcal{X}$ of length $N = 9$ and $\mathcal{Y}$ of length $M = 7$. Each alignment is defined by the set of index pairs (n,m), where n and m are the corresponding coordinates of the black dots in the cost matrix.

of $\mathcal{X}$ and $\mathcal{Y}$ are elements of a feature space F, *i.e.*, $x_n, y_m \in F \subset \mathbb{R}^k$. The best alignment of the two input sequences is achieved when each component of the first sequence is assigned the best corresponding component of the second sequence according to a global alignment cost. Let us define the cost to align each pair of components through a dissimilarity measure $c : F \times F \to \mathbb{R}$. That is, $c(x,y)$ outputs a small value if x and y are similar to each other, otherwise $c(x, y)$ outputs a large value. The precise definition of the cost function c is not necessary for a general treatment of the dynamic time warping algorithm, and we will define it later according to our application. The only property that is normally assumed is that the cost function c is symmetric, so that, as a consequence, the alignment does not depend on the order of the input sequences.
If we define a $N \times M$ cost matrix $\mathbf{C}$, such that $C(i,j) = c_{x_i,y_j}$, any possible sequence alignment defines a warping path (or *valley*) within the cost matrix $\mathbf{C}$, such as those depicted in Figure 4.7. More formally, a warping path p is a sequence $p = (p_1, \ldots, p_l, \ldots, p_L)$ with $p_l = (n_l,m_l) \in \{1, \ldots, N\} \times \{1, \ldots, M\}$ for $l \in \{1, \ldots, L\}$. For each warping path p, there is a corresponding alignment cost c_p, defined as follows,

$$c_p(\mathcal{X}, \mathcal{Y}) = \sum_{l=1}^{L} c(x_{n_l},y_{m_l}). \tag{4.22}$$

The definition of warping path is very general and it allows for alignments that are usually not acceptable, *e.g.*, warping paths that do not match the first components of each sequence, like the second warping path shown in Figure 4.7. In order to enforce some consistency in the warping path, we add the following conditions to the definition above:

Boundary condition: $p_1 = (1,1)$ and $p_L = (N,M)$, *i.e.*, the first elements of $\mathcal{X}$ and $\mathcal{Y}$ as well as the last elements of $\mathcal{X}$ and $\mathcal{Y}$ are aligned to each other.

Monotonicity condition: $n_1 \leq \ldots \leq n_l \leq \ldots \leq n_L$ and $m_1 \leq \ldots \leq m_l \leq \ldots \leq m_L$, *i.e.*, the alignment cannot turn back on itself, as matching indexes either stay the same or increase.

Step size condition: $p_{l+1} - p_l \in \{(1,0),(0,1),(1,1)\}$ for $l \in \{1,\ldots,L-1\}$, *i.e.*, no element in $\mathcal{X}$ and $\mathcal{Y}$ is left out from the alignment and all index pairs contained in p are pairwise distinct.

The problem of finding the best alignment can thus be reformulated as the one of finding the warping path with the minimum cost within $\mathbf{C}$ that respects the aforementioned conditions. That is, the best alignment is obtained when the total cost c_p defined in Equation (4.22) is minimized, that is,

$$c_{p^*}(\mathcal{X},\mathcal{Y}) = \min_p c_p(\mathcal{X},\mathcal{Y}). \tag{4.23}$$

While testing all possible warping paths is computationally unfeasible, as the complexity is exponential in the sequence lengths, dynamic time warping permits to find the optimal warping path in $O(NM)$ time by relying on a *dynamic programming* formulation.
Let the prefix sequences be $\mathcal{X}(1:k) = (x_1,\ldots,x_k)$ for $k \in \{1,\ldots,N\}$ and $\mathcal{Y}(1:l) = (y_1,\ldots,y_l)$ for $l \in \{1,\ldots,M\}$. We can define the *optimal* accumulated cost for the index pair (k,l) as $D(k,l) = c_{p^*}(\mathcal{X}(1:k),\mathcal{Y}(1:l))$. Therefore, the $N \times M$ matrix $\mathbf{D}$ is the full optimal accumulated cost matrix, and its bottom right entry $D(N,M)$ equals the optimal warping cost $c_{p^*}(\mathcal{X},\mathcal{Y})$. The matrix $\mathbf{D}$ can be computed efficiently by using the DTW algorithm[7] shown in Algorithm 1.
In our framework, temporal sequences are replaced by feature tracks, and this implies that the temporal index is replaced by a pose index, whereas scalar values are replaced by feature descriptors. Nonetheless, dynamic time warping can be applied in a straightforward way. The only thing that we have to define is a suitable cost function for assessing the dissimilarity between feature tracks.
Let two tracks be $T^i = \{(\mathbf{f}_1^i,\boldsymbol{\alpha}_1^i),\ldots,(\mathbf{f}_n^i,\boldsymbol{\alpha}_n^i)\}$ and $T^j = \{(\mathbf{f}_1^j,\boldsymbol{\alpha}_1^j),\ldots,(\mathbf{f}_m^j,\boldsymbol{\alpha}_m^j)\}$, and let us assume that we ordered the pairs in each track according to the pose label, so that $\boldsymbol{\alpha}_1^i < \ldots < \boldsymbol{\alpha}_n^i$ and $\boldsymbol{\alpha}_1^j < \ldots < \boldsymbol{\alpha}_m^j$.
In order to cluster feature tracks that are similar both in appearance and viewpoint, we define the following cost function between two augmented feature descriptors $(\mathbf{f}_k^i,\boldsymbol{\alpha}_k^i) \in T^i$ and $(\mathbf{f}_l^j,\boldsymbol{\alpha}_l^j) \in T^j$ as

$$C(k,l) = G(\boldsymbol{\alpha}_k^i,\boldsymbol{\alpha}_l^j)^{-1}\|\mathbf{f}_k^i - \mathbf{f}_l^j\|_2 \quad \forall k \in \{1,\ldots,n\}, l \in \{1,\ldots,m\} \tag{4.24}$$

where $G(x,y) = e^{-\frac{\|x-y\|}{2\sigma^2}}$ and $\|\cdot\|_2$ is the L^2-norm. This definition of the cost function gives a low alignment cost for tracks that represent similar patches from

[7] The computational complexity and the respect of the conditions given above are proven in [87].

Algorithm 1 Dynamic Time Warping Algorithm

Require: Two sequences $\mathcal{X} = (x_1, \ldots, x_N)$ and $\mathcal{Y} = (y_1, \ldots, y_M)$
Declare $\mathbf{D}$ as a $N \times M$ matrix;
$D(1,1) = c(x_1, y_1)$
for $i = 2 \rightarrow N$ **do**
  $D(i,1) = D(i-1,1) + c(x_i, y_1)$;
end for
for $j = 2 \rightarrow M$ **do**
  $D(1,j) = D(1,j-1) + c(x_1, y_j)$;
end for
for $i = 2 \rightarrow N$ **do**
  for $j = 2 \rightarrow M$ **do**
    $D(i,j) = \min(D(i-1,j), D(i,j-1), D(i-1,j-1)) + c(x_i, y_j)$;
  end for
end for
return Alignment cost $DTW(\mathcal{X},\mathcal{Y}) = D(N,M)$

similar viewpoints. As explained above in Algorithm 1, the cost of aligning the two feature tracks $DTW(T^i,T^j)$ is given by the last entry of the accumulation matrix $D(n,m)$.
If the visibility intervals of the two tracks do not overlap, we directly set

$$DTW(T^i,T^j) = \infty. \tag{4.25}$$

This will turn out to be beneficial in the clustering step, as it leads to a sparse similarity matrix.

4.4.2 Spectral Clustering for Track Grouping

In Equation (4.24), we defined a dissimilarity measure between augmented descriptors and consequently a dynamic programming algorithm that permits to find the alignment cost for two augmented tracks. Now, we show how to use this dissimilarity score to cluster the tracks so that each cluster will contain tracks representing similar patches from overlapping viewpoints.
Regarding the clustering method, we resort to spectral clustering, a technique that permits to perform grouping on the basis of the elements connectivity. This is a crucial advantage of spectral clustering, as it can properly handle non-convex sets. Since we cannot assume any special arrangement for the feature tracks, this motivates our choice in favor of spectral clustering over other popular clustering methods. For example, k-means clustering is based on the geometrical proximity

of the elements, and thus it would not work properly with non-convex sets. In the following, we will present the spectral clustering algorithm that we use in our approach as well as insights into the algorithm performance.
Let us consider our set of tracks $\mathcal{T} = \{T^i\}_{i=1}^n$ and let us compute the DTW distance between each pair of tracks as defined in Section 4.4.1. Whereas DTW returns a positive dissimilarity score, spectral clustering works with positive similarity scores. In order to adapt to this, we transform the dissimilarity score $DTW(T^i,T^j)$ into the similarity score s_{ij} according to the following transformation,

$$s_{ij} = \exp(-DTW(T^i,T^j)). \tag{4.26}$$

Thus, by considering the similarity scores of all pairs of tracks, we obtain the following similarity matrix $\mathbf{S}$,

$$\mathbf{S} = \begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nn} \end{bmatrix}. \tag{4.27}$$

We can also represent the training tracks in the form of a weighted similarity graph $G = (V, E)$. Each feature track is represented by a vertex $v_i \in V$ and each edge $e_{ij} \in E$ connecting vertexes v_i and v_j has a weight $w_{ij} = s_{ij}$.
The problem of clustering can now be expressed by referring to the similarity graph. We want to partition the graph in such a way that edges between different groups have very low weights, which means that feature tracks in different clusters are dissimilar from each other, and edges within a group have high weights, which means that feature tracks within the same cluster are similar to each other.
We define the weighted adjacency matrix of the graph as $\mathbf{W} = \mathbf{S}$. If the vertices v_i and v_j are not connected by an edge, then $w_{ij} = 0$. This happens when the two tracks do not overlap in the pose domain (cf. Equations (4.25) and (4.26)). The degree of a vertex $v_i \in V$ is defined as $d_i = \sum_{j=1}^n w_{ij}$, and the degree matrix $\mathbf{D}$ is defined as the diagonal matrix with degrees $d_1, \ldots, d_n$ on the diagonal. On the basis of the weighted adjacency matrix $\mathbf{W}$ and the degree matrix $\mathbf{D}$, we can define the unnormalized graph Laplacian as

$$\mathbf{L}_u = \mathbf{D} - \mathbf{W} \tag{4.28}$$

and the normalized graph Laplacian[8] $\mathbf{L}$ as

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{L}_u\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{4.29}$$

[8]In the literature, there is another definition of the normalized graph Laplacian. We use the algorithm proposed in [89], where this definition for the graph Laplacian is considered. The other well-known spectral clustering algorithm has been proposed in [113], where the definition of the normalized graph Laplacian is slightly different, and the algorithm changes accordingly.

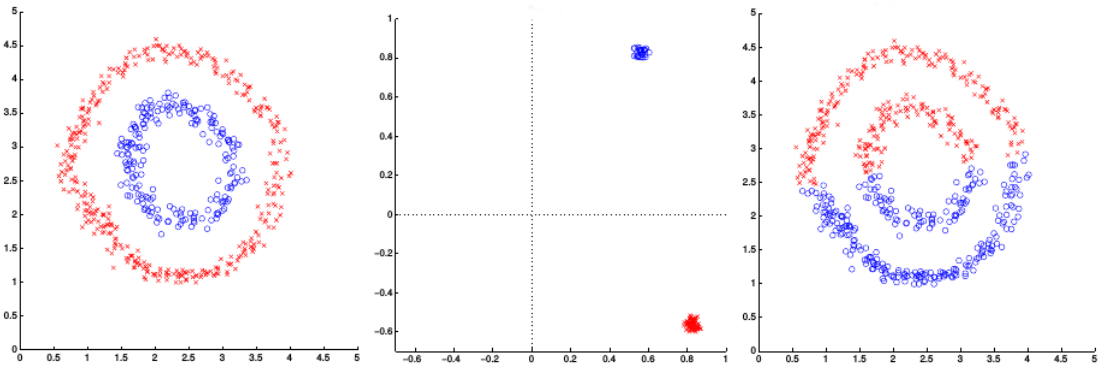


Figure 4.8: Results obtained with spectral clustering (left) and k-means (right) with $k = 2$. As the two point sets are not convex in their original domain k-means fails. After mapping, they form tight clusters instead, and k-means is successful (middle). (Pictures taken from [89].)

The matrix $\mathbf{L}$ and its eigenvalues have a special importance with respect to the connected components in the graph, as shown in Algorithm 2.

Algorithm 2 Spectral Clustering Algorithm

Require: Similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, number k of clusters to construct

1: Compute the normalized Laplacian $\mathbf{L}$, as defined in Equation (4.29) with $\mathbf{W} = \mathbf{S}$
2: Compute the smallest k eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of $\mathbf{L}$
3: Let $\mathbf{U} \in \mathbb{R}^{n \times k}$ be the matrix containing the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ as columns
4: Form the matrix $\mathbf{T} \in \mathbb{R}^{n \times k}$ by normalizing the rows of $\mathbf{U}$ to have unit norm
5: For $i = \{1, \ldots, n\}$, let $\mathbf{y}_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of $\mathbf{T}$
6: Cluster the points $(\mathbf{y}_i)_{i=1}^n$ with the k-means algorithm into clusters $C_1, \ldots, C_k$
7: **return** Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \mathbf{y}_j \in C_i\}$

The advantage of spectral clustering with respect to k-means lies all in the mapping step, *i.e.*, steps 3 and 4 in Algorithm 2. While data samples may not form convex sets in their original domain, the mapping to a higher dimensional domain often reveals tight clusters that can be easily identified by a later application of k-means, as shown in Figure 4.8.
As a further insight into the performance of the algorithm, we will apply it to an ideal case, where we have three clusters S_1, S_2, and S_3 of sizes n_1, n_2 and

n_3, respectively, and all points in different clusters are infinitely far apart. Let us also assume that the data points are ordered according to which cluster they are in, so that the first n_1 points are in cluster S_1, the next n_2 in S_2, and so on. Since points in different clusters are infinitely far apart this means that the corresponding element w_{ij} is null. Therefore, $\mathbf{W}$ and $\mathbf{L}$ are block-diagonal:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}^{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{W}^{33} \end{bmatrix} \quad \mathbf{L} = \begin{bmatrix} \mathbf{L}^{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}^{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{L}^{33} \end{bmatrix} \tag{4.30}$$

We find the first 3 eigenvectors of $\mathbf{L}$. Since $\mathbf{L}$ is block-diagonal, its eigenvalues and eigenvectors are the union of the eigenvalues and eigenvectors of its blocks, and we stack the latter in column order in $\mathbf{U}$ as

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}^{11} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}^{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}^{33} \end{bmatrix} \tag{4.31}$$

Next, we normalize each row of $\mathbf{U}$ to have unit norm, so that we obtain:

$$\mathbf{T} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{4.32}$$

If we apply k-means to the rows of $\mathbf{T}$, there are k mutually orthogonal points on the surface of the unit k-sphere around which the rows of $\mathbf{T}$ will cluster. This exactly represents the true clustering of the original data.

4.5 Class Probabilistic Formulation

In this section, we show how to compute the pose of an unknown object of a given class on the basis of a set of training instances. As already done in Section 4.3, we will first present the training stage, and then we will focus on the testing part.

Training stage

We assume that we are given a set of O objects $\mathcal{O} = \{o^i\}_{i=1}^{O}$ that comprises different instances of the class of interest. Furthermore, we assume that for each object o^i, we are given a set of N_i training images $\mathcal{I}^i = \{I^{ij}\}_{j=1}^{N_i}$, where the object is depicted from different viewpoints. Finally, we assume that we are also given a set of viewpoint labels $\{\boldsymbol{\alpha}^{ij}\}_{j=1}^{N_i}$ for each object sequence $\mathcal{I}^i$, which describe the object orientation with respect to a common coordinate system. The goal of the training stage is to create a class representation by clustering generative feature

models. Each generative feature model is built upon an augmented feature track that has been extracted from one of the training instances.
We start by collecting augmented feature tracks from the training sequences. For each object o^i, we collect a set of feature tracks $\mathcal{T}^i = \{T^{ij}\}_{j=1}^{M_i}$ by tracking feature descriptors as described in Section 4.3. Each element $T^{ij} \in \mathcal{T}^i$ is a set of pairs, where each pair is formed by a feature descriptor $\mathbf{f}^{ij}$ and the corresponding viewpoint label $\boldsymbol{\alpha}^{ij}$, *i.e.*,

$$T^{ij} = \{(\mathbf{f}_1^{ij},\boldsymbol{\alpha}_1^{ij}),(\mathbf{f}_2^{ij},\boldsymbol{\alpha}_2^{ij}),\ldots,(\mathbf{f}_n^{ij},\boldsymbol{\alpha}_n^{ij})\}. \tag{4.33}$$

Then, for each augmented track T^{ij} we compute the corresponding generative feature model F^{ij} as described in Section 4.2.1. That is, we compute the following RBF network,

$$F^{ij}(\boldsymbol{\alpha}) = \sum_{k=1}^{n} \mathbf{w}_k^{ij} G(\boldsymbol{\alpha}, \boldsymbol{\alpha}_k^{ij}). \tag{4.34}$$

Finally, we collect all the feature tracks resulting from different objects in one single set

$$\mathcal{T} = \bigcup_{i=1}^{O} \mathcal{T}^i. \tag{4.35}$$

On the basis of $\mathcal{T}$, we build the similarity matrix $\mathbf{S}$ according to the method explained in Section 4.4.1. Each element of this matrix represents the degree of similarity of two feature tracks in terms of feature descriptors and viewpoint interval. Finally, we cluster the feature tracks by applying the feature clustering algorithm described in Section 4.4.2.
The clustering step returns a set of clusters $\mathcal{K} = \{K^k\}_{k=1}^{L}$, where each cluster contains a set of feature tracks,

$$K^k = \{T^{jk}\}_{j=1}^{J_k}, \tag{4.36}$$

where J_k is the number of feature tracks T^{jk} grouped in cluster K^k. Since generative feature models are in a one-to-one relationship with the feature tracks, we also know that each cluster identifies a corresponding group of generative feature models. A block diagram of the training stage is given in Figure 4.9.

Testing stage

Let us assume we are given a query image I_q depicting the object. We can define a probability function $p(\boldsymbol{\alpha}|I_q)$ that expresses the likelihood of the object being seen under viewpoint $\boldsymbol{\alpha}$ if the object is observed in the current query image.
By applying Bayes' Rule to $p(\boldsymbol{\alpha}|I_q)$, we obtain the following

$$p(\boldsymbol{\alpha}|I_q) = \frac{p(I_q|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(I_q)}. \tag{4.37}$$

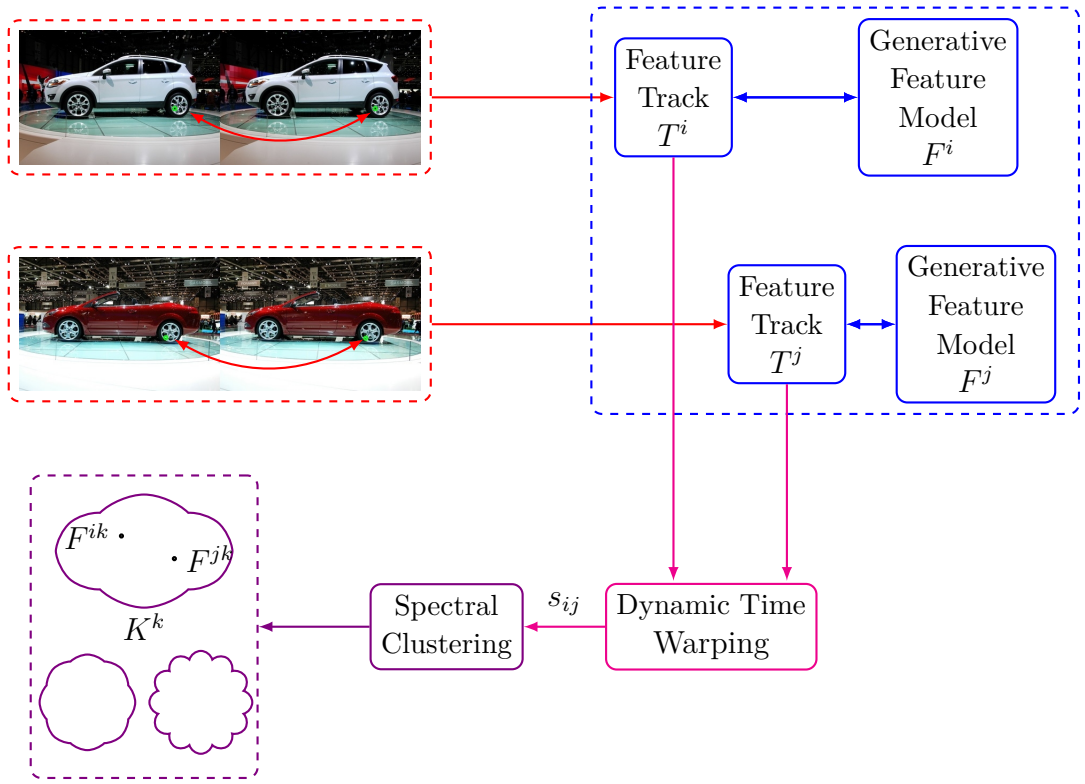


Figure 4.9: The training stage in case of object categories. Features are tracked over training sequences and collected in augmented feature tracks. For each feature track T, a generative feature model F is computed. The similarity of each pair of augmented feature tracks is evaluated by the DTW algorithm in a score s. On the basis of the scores, spectral clustering is applied to create clusters of feature tracks and associated generative feature models. Feature tracks representing similar structures, like a wheel part, will be clustered together if they occur in overlapping viewpoint intervals.

Our pose estimation problem amounts to finding $\boldsymbol{\alpha}^*$ that maximizes the expression above,

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} p(\boldsymbol{\alpha}|I_q) = \arg\max_{\boldsymbol{\alpha}} \frac{p(I_q|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(I_q)}. \tag{4.38}$$

As $p(I_q)$ is independent from $\boldsymbol{\alpha}$, we obtained the following maximum a posteriori (MAP) estimation problem

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \frac{p(I_q|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(I_q)} = \arg\max_{\boldsymbol{\alpha}} p(I_q|\boldsymbol{\alpha})p(\boldsymbol{\alpha}). \tag{4.39}$$

If no prior information on the pose is available, this prior probability will be set to a uniform distribution over the pose domain, turning the maximum a posteriori estimation into a maximum likelihood estimation.
The conditional term $p(I_q|\boldsymbol{\alpha})$ in Equation (4.39) can be expressed as,

$$p(I_q|\boldsymbol{\alpha}) = p_d(I_q|\boldsymbol{\alpha})p_g(I_q|\boldsymbol{\alpha}) \tag{4.40}$$

where $p_d(I_q|\boldsymbol{\alpha})$ is a a discriminative term and $p_g(I_q|\boldsymbol{\alpha})$ is a generative term. The discriminative term was not introduced in the single object case presented in Section 4.3, as the generative feature models were built from features belonging to the same object. On the contrary, when the object is not part of the training set is advisable to inject some discriminativeness into the approach in order to disambiguate between the viewpoints. The generative term can be expressed in terms of the set of features $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^N$ extracted from I_q as

$$p_g(I_q|\boldsymbol{\alpha}) = p(\mathcal{Q}|\boldsymbol{\alpha}) = \prod_{i=1}^{N} p(\mathbf{q}_i|\boldsymbol{\alpha}). \tag{4.41}$$

In the training stage, we have clustered the generative feature models into a set of clusters $\mathcal{K} = \{K^k\}_{k=1}^L$, and for each cluster K^k we define a representative feature $\mathbf{c}^k$ as the average feature descriptor of all the clustered descriptors. Let us now consider a query feature $\mathbf{q}_i$ and put it in correspondence with a cluster through its representative feature $\mathbf{c}^k$. Since the cluster is composed of generative feature models, we can think that the likelihood of the pose $\boldsymbol{\alpha}$ is reflected by the norm of the regression error generated by the matching cluster regressors. In order to perform feature regression at a class level, we create a cluster regressor, known as *generative cluster model*, by linearly aggregating the individual generative feature models contained in the cluster. That is, we define the generative cluster model $C(\boldsymbol{\alpha})$ for the cluster K^k as

$$C^k(\boldsymbol{\alpha}) = \sum_{j=1}^{J_k} w_i^{jk} F^{jk}(\boldsymbol{\alpha}) \tag{4.42}$$

where J_k is the number of generative feature models contained in the matching cluster and F^{jk} is an individual generative feature model in cluster K^k. The weights w_i^{jk} are computed as

$$w_i^{jk} = \frac{d_i^{jk}}{\sum_j d_i^{jk}} \tag{4.43}$$

where $d_i^{jk} = \|\mathbf{q}_i - \mathbf{r}^{jk}\|$ is the distance between the query descriptor $\mathbf{q}_i$ and each generative feature model representative[9] $\mathbf{r}^{jk}$. In this way, a more similar generative feature model has a higher contribution to the prediction error, which is defined as

$$\mathbf{e}_i^k(\boldsymbol{\alpha}) = \mathbf{q}_i - C^k(\boldsymbol{\alpha}) = \mathbf{q}_i - \sum_{j=1}^{J_k} w_i^{jk} F^{jk}(\boldsymbol{\alpha}). \tag{4.44}$$

Furthermore, we decompose the overall prediction error $\mathbf{e}_i^k(\boldsymbol{\alpha})$ as a summation of errors generated by the generative feature models forming the generative cluster model in the following way

$$\mathbf{e}_i^k(\boldsymbol{\alpha}) = \sum_{j=1}^{J_k} w_i^{jk}(\mathbf{q}_i - F^{jk}(\boldsymbol{\alpha})) = \sum_{j=1}^{J_k} w_i^{jk}\mathbf{e}_i^{jk} \tag{4.45}$$

If a cluster predicts the feature $\mathbf{q}_i$ with a small error for a certain viewpoint $\boldsymbol{\alpha}$, we assume that $\boldsymbol{\alpha}$ has a high likelihood to be the query viewpoint according to that cluster. Therefore, we can write the probability $p(\mathbf{q}_i|\boldsymbol{\alpha})$ in terms of the cluster representatives as

$$p(\mathbf{q}_i|\boldsymbol{\alpha}) = \sum_{k=1}^{L} p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha}). \tag{4.46}$$

We define $p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha})$ on the basis of the error between the query feature and the descriptor estimated by the matching generative cluster model as well as a compatibility term between $\mathbf{q}_i$ and $\mathbf{c}^k$. Therefore, the observation likelihood for a query feature $\mathbf{q}_i$ and $\mathbf{c}^k$ is defined as

$$p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha}) = \gamma_{\mathbf{q}_i\mathbf{c}^k} \exp\left(-\frac{1}{2}\sum_{j=1}^{J_k}(\mathbf{e}_i^{jk}(\boldsymbol{\alpha}))^T(\mathbf{R}^{jk})^{-1}\mathbf{e}_i^{jk}(\boldsymbol{\alpha})\right) \tag{4.47}$$

where $\mathbf{R}^{jk}$ is the error covariance matrix of the j-the generative feature model in the matching cluster K^k. In order to reduce the number of tentative representative

[9]The generative feature model representative has been introduced in Section 4.3 as the feature descriptor with the most central viewpoint among those composing the generative feature model.

models, we rely on feature discriminativeness by defining $\gamma_{\mathbf{q}_i\mathbf{c}^k}$ to be

$$\gamma_{\mathbf{q}_i\mathbf{c}^k} = \begin{cases} 1 & \text{for } \tilde{\mathbf{c}}^i \text{ s.t. } \|\mathbf{q}_i - \tilde{\mathbf{c}}^i\| = \min_{\mathbf{c}^k} \|\mathbf{q}_i - \mathbf{c}^k\| \\ 0 & \text{otherwise} \end{cases}. \tag{4.48}$$

We will see in Chapter 5 how to change the definition of $\gamma_{\mathbf{q}_i\mathbf{c}^k}$ in order to take into account several model representatives for each query feature $\mathbf{q}_i$.
Now, let us return to the full set of features that we extracted from the query image and the set of all the representative features of the generative models $\mathcal{C} = \{\mathbf{c}^k\}_{k=1}^{L}$. We find for each feature in $\mathcal{Q}$ the nearest neighbor model representative in $\mathcal{C}$, so that we have a one-to-one correspondence between each query feature $\mathbf{q}_i$ and a generative model through its representative feature $\tilde{\mathbf{c}}^i$. Therefore, the maximum a posteriori estimation of $\boldsymbol{\alpha}$ is defined as

$$\begin{aligned} \boldsymbol{\alpha}^* &= \arg\max_{\boldsymbol{\alpha}} p_d(I_q|\boldsymbol{\alpha}) p(\mathcal{Q}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) = \arg\max_{\boldsymbol{\alpha}} p_d(I_q|\boldsymbol{\alpha}) \prod_{i=1}^{N} \sum_{k=1}^{L} p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) \\ &= \arg\max_{\boldsymbol{\alpha}} p_d(I_q|\boldsymbol{\alpha}) \prod_{i=1}^{N} p(\mathbf{q}_i, \tilde{\mathbf{c}}^i|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}). \end{aligned} \tag{4.49}$$

4.6 Experimental Evaluation

In this section, we provide an experimental evaluation of our proposed method. In order to test it, we consider two datasets: the EPFL multi-view car dataset [90] and the Pointing'04 face dataset [42]. The first dataset provides pose-labelled sequences of cars rotating on a floor pedestal. Sequences are densely sampled in the pose space, but cars are uncommonly shaped, thus making model generalization very challenging. The second dataset contains human faces with variable yaw and pitch values. In this case, the sampling in pose space is more sparse but there is less variability in the exemplars.
We will first analyze the performance of our algorithm on the car dataset by providing results as well as insight into the performance of the individual parts of our method. Subsequently, we will show the results of our method on the face dataset.

4.6.1 EPFL Multi-view Car Dataset

With regard to the EPFL car dataset, we use the testing framework proposed in [90], as this is the original work where the dataset has been introduced. We evaluated our method using two different splits of training and testing sequences:

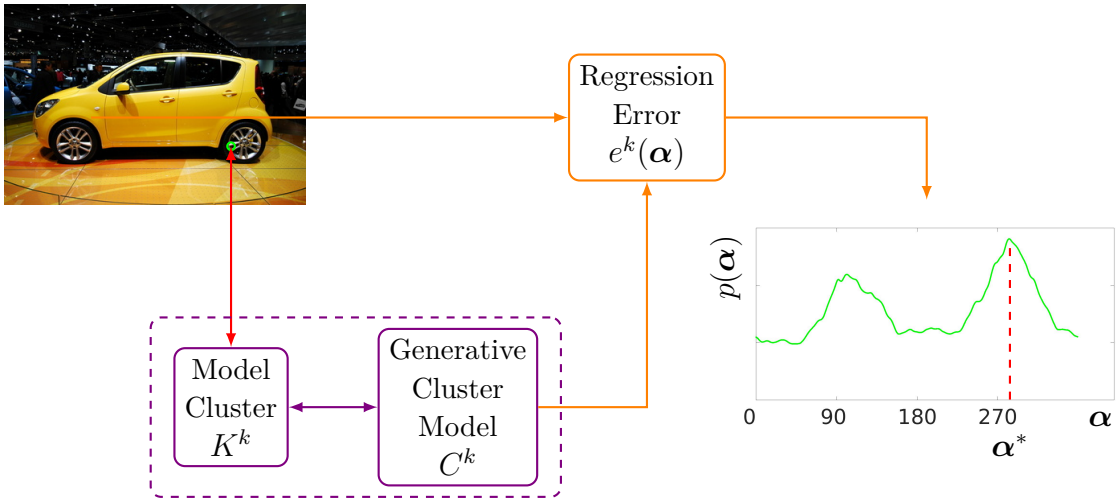


Figure 4.10: The testing stage in case of object classes. Query features are extracted from the test image and each query descriptor is matched to its nearest neighbor cluster K^k. The corresponding generative cluster model linearly combines the output of its internal regressors to predict the query feature descriptor as a function of the putative pose $\boldsymbol{\alpha}$. The norm of the error $\mathbf{e}^k$ between the query descriptor and the predicted descriptor is used to build a posterior probability distribution, whose maximum is returned as the estimated pose of the object.

50% split: Training is performed on all images of the first 10 sequences, and testing is performed on all images of the second 10 sequences.

Leave One Out (LOO) split: Training is performed on all images of 19 sequences in turn, and testing is performed on all images of the remaining one.

In order to train our class representation, we use the method described in Section 4.5, as pictured in Fig. 4.10. We use a combination of SURF detector coupled with a SIFT descriptor, because this combination obtained the higher accuracy in the introductory experiment of single object pose estimation presented in Section 4.3.1. In order to estimate the pose of the target car in each query image, we use the method described in Section 4.5 with a discriminative term determined by a coarse pose classifier [76].
The authors in [76] extend and improve the Deformable Part Model (DPM) object detector introduced by [27] for pose estimation purposes. The idea exploited in [76] is to learn the mixture model of the DPM detector as a function of specific viewpoint intervals. For this purpose, DPM training is modified according to a *semi*-latent strategy, where only the filter components are treated as latent whereas pose labels are fixed. The pose classifier is usually trained to return a discrete pose estimate over 4, 8 or 16 intervals. For this experiment, we have used its 16-interval implementation, thus receiving an initial pose interval estimation of 22.5° in width. Therefore, the discriminative term is defined as a uniform distribution over the interval returned by the classifier and null elsewhere.
We considered the method proposed in [90] as the baseline method. We also compare against the work proposed in [118], as it also proposes a regression-based pose estimation approach (cf. Section 2.4). On our side, we provide an evaluation of our full method against two partial implementations of it. The implementation variants were chosen to highlight the beneficial contribution of each part of our method, as follows:

Track 5NN (w/o regression and clustering) We match each query feature with the 5 nearest neighbors among the generative feature model representatives. The returned pose is the mode among these viewpoints. This is a naive implementation that provides results only in terms of the viewpoint of the five nearest neighbor tracks. Five was chosen because it represents the most frequent cluster size in the full method implementation.

Regression 5NN (w/o clustering) We find again the 5 nearest neighbors among the generative feature model representatives for each query feature. The pose is estimated by using the track regression functions as described in the single exemplar case. This second variant shows the benefit of introducing regression for pose estimation.

Table 4.2: Pose Estimation on the EPFL dataset. Comparison among full and partial implementations of our method, [90] and [118].

Method	MAE [°] 90^{th} percentile	MAE [°] 95^{th} percentile	MAE [°]
Ozuysal *et al.* [90] (Baseline)	-	-	46.48
Torki *et al.* [118] - 50% split	19.4	26.7	33.98
Track 5NN [29] - 50% split	15.33	23.64	32.08
Regression 5NN [29] - 50% split	15.17	23.49	31.93
Fenzi *et al.* [29] - 50% split	**14.51**	**22.83**	**31.27**
Torki *et al.* [118] - LOO split	23.13	26.85	34.90
5NN Track [29] - Leave One Out split	15.17	23.48	31.92
Regression 5NN [29] - LOO split	15.10	23.42	31.85
Fenzi *et al.* [29] - LOO split	**14.41**	**22.72**	**31.16**

The results of each variant of our method as well as of [90] and [118] are compared in Table 4.2 in terms of the Mean Absolute Error (MAE, cf. Section 4.3.1). In Table 4.2, we see an improvement in pose accuracy over the state of the art of approximately 10%. In order to explain these large estimation errors, we must take into account an issue also present in [90] and [118], which strongly affects the performance of all methods. In certain orientations, mainly front and rear as well as left and right side, cars show a strong similarity due to symmetry. This potentially generates 180° errors that have a non-negligible effect on the overall average.

In order to provide more insight into the "real" performance of the methods, we also present results in terms of their 90^{th} and 95^{th} percentile, so that the influence of "flipping" errors is at least partially removed. In the leftmost two columns in Table 4.2, our method shows to perform better than state of the art, with an improvement given by the full method in the order of 25%. Furthermore, even if the 16-bin classifier we use for the discriminative term has a worse performance than [118] (66% vs. 70.31% accuracy for 16 bins), we are still able to obtain smaller errors. We want to highlight the fact that our method provides better results even when the naive 5NN implementation is used. This can be taken as a further confirmation of the superiority of local approaches for pose estimation over global ones. In Figure 4.11, we provide visual examples of the performance of our method on the multi-view EPFL dataset.

For a deeper insight into the performance of our method, we give in Table 4.3 the results of a side experiment performed using the ground-truth bin instead of the actual classifier output. A very naive "baseline" can be obtained by randomly

Figure 4.11: Results of our method on the EPFL car dataset. In the top left corner, a diagram representing the car orientation, where the car front is labeled by 0°. For each picture, the ground truth value (green) and the estimated one (red) are given in a box and a red arrow is overlaid on the car. In the last row, two examples of flipped estimation are provided. The pose of the car in the bottom left corner is hard to guess also for a human being.

Table 4.3: Pose estimation using a ground-truthed classifier

Method	MAE [°]
5NN Track - 50% split	6.87
Regression 5NN - 50% split	6.64
Our Full Method - 50% split	**5.64**
5NN Track - Leave One Out split	6.63
Regression 5NN - Leave One Out split	6.53
Our Full Method - Leave One Out split	**5.48**

guessing the viewpoint inside the correct bin, and this has a MAE of 7.5°. Our Track 5NN implementation performs only slightly worse than the Regression 5NN implementation as the training images are very finely sampled in pose. On the contrary, the improvement in the accuracy given by our full method comes out more evidently with a 20% and 15% increase, respectively. The better performance compared to the only-regression variant is due to the use of consistent clusters having appropriate size, instead of the m nearest neighbours.

Pointing'04 face dataset

With regard to the Pointing'04 face dataset, we use the same testing framework proposed in [42], as this is the original work where the dataset has been introduced. Evaluation is performed with a 5-fold cross validation, where each sample set is tested in turn using a model trained on the remaining 80% of the samples.
For our method, we have evaluated three variants with a similar spirit and implementation as in the previous experiment on the EPFL car dataset. This time, we used only the first nearest neighbor (1NN) because samples of the same person can be found both in training and testing. We report the results in Table 4.4.
Our method shows to have a better or comparable performance with respect to the state of the art on this dataset. The slightly worse performance in the pitch estimation for all methods is due to the coarser sampling of the pitch poses. Unlike the previous experiment, the benefit of using feature regression comes out more evidently when the pose sampling is coarser with an increase in pose accuracy of 42%. Again, the full method outperforms its variants by 57% and 25%, respectively. We can think that each generative model explains the query pose by combining the descriptors of the same person in neighbouring poses and the descriptors belonging to other persons, potentially available at the same viewpoint of the query image. In Figure 4.12, we provide visual examples of the performance of our method on the Pointing'04 face dataset.

Table 4.4: Pose estimation on the Pointing'04 dataset

Method	Yaw MAE [°]	Pitch MAE [°]
Gourier *et al.* [42]	10.1	15.9
Haj *et al.* [46]	6.56	**6.61**
Track 1NN [29]	13.82	19.27
Regression 1NN [29]	7.89	9.43
Fenzi *et al.* [29]	**5.94**	6.73

Figure 4.12: Results of our method on the Pointing'04 face dataset. In this case, the returned pose value is two-dimensional, as it is given in terms of pitch and yaw. The ground truth pose (green) and the estimated one (red) are given in a box.

Chapter 5

Enforcing Geometrical Constraints

As shown in Section 4.6, our algorithm is an effective solution for the problem of pose estimation for object categories, but it seems to be prone to errors due to symmetry. When class objects present similar appearance from two different viewpoints, *e.g.*, the right and left view of a vehicle, one viewpoint may be mistaken for the other. Therefore, the results of our algorithm are either very accurate or they suffer from a so-called "flipping" error. For example, in the case of vehicles, we often observe a flipping error of approximately 180°, that strongly affects the results when the mean absolute error is taken as evaluation metrics. An example of this ambiguity is shown in Figure 5.1.

After a careful analysis, we identified two different causes, both contributing to these errors. The first cause is due to the hard decision taken by the discrete pose estimator that we used as discriminative term. If the pose classifier decision is wrong, our algorithm has no possibility to invert it, thus leading to an unavoidably incorrect estimation. Since many discrete pose classifiers, like the one we used, suffer from flipping errors themselves, our algorithm is also automatically affected.

The second cause is more general and is related to the lack of geometrical information in our algorithm. As we have seen in Chapter 4, the feature regressors are built on the basis of the feature descriptors alone without taking the feature location into account. In addition, no geometrical information is used when query features are matched against the generative cluster models.

There are two possible steps in which we can introduce geometrical information in our method. The first one is in the feature descriptor, *i.e.*, we could augment the feature descriptor with the feature location, possibly normalized to take translation effects into account. As a result, our generative feature model would perform regression over both appearance and location. Although this choice would be a natural solution, it has the disadvantage that features would still be treated individually, whereas taking an approach based on the spatial arrangement of the features would provide a stronger cue to solve the pose estimation problem.

The other step is during matching, when we find correspondences between query features and cluster representatives. Since the precise geometry of the query ob-

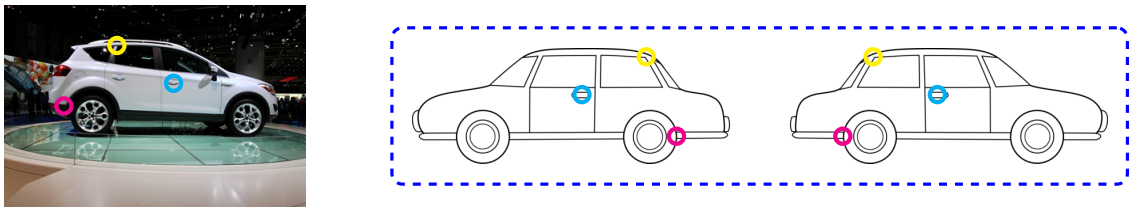


Figure 5.1: (Left) test image, (right) our class representation depicted as two sketched cars with opposite views. We represented three different features in magenta, yellow and cyan and their matching representative models in the class representation. If the features in the left model "view" are selected by nearest neighbor matching due to a slightly smaller distance to the query descriptors, the pose estimated by our algorithm would lead to a flipping error. It is clear from this picture that the integration of geometry in the matching process would help to solve ambiguous situations.

ject is unknown and is probably different from that of the training instances, we cannot introduce constraints based on epipolar geometry as in the pose estimation paradigm for single objects (see Chapter 2). Nonetheless, most objects of many classes show some *loose* similarity in the geometric arrangement of their parts, *e.g.*, wheels are always located in the lower part and windows in the upper part of any vehicle. Therefore, a technique that permits a loose mapping between the query instance and the model would be beneficial for our method.

As a matter of fact, there are many Computer Vision tasks, such as shape matching and 2D-3D registration, whose solution is based on determining a set of correspondences between two sets of features that do not belong to the same, specific object. Another typical example is optical character recognition (OCR) for handwritten texts, where one has to match handwritten characters to a predefined set of digits and letters. In many of these applications, a technique known as graph matching is employed to solve the correspondence problem.

On the basis of their generality, graphs are commonly used to describe complex visual concepts, like scenes or objects, by representing their parts as nodes of a graph. While each node contains information about some local object property, the graph is also characterized by a set of weighted edges that represent second-order relationships between nodes. The advantage of graph matching techniques is that they exploit both first- and second-order relationships between object parts

by introducing a pairwise term in the matching problem formulation[10]. In simple words, graph matching aligns two graphs in such a way that they "look most similar" with respect to unary and binary correspondences between graph nodes. Graph matching appears to be a very suitable solution for the matching step of our algorithm for four different reasons:

- Loose object-to-model mapping
- Suitable for point-like entities
- Inter-relations between multiple features
- Soft matching

First of all, graph matching permits to find a loose, yet consistent mapping between a query graph representing the query instance and a model graph representing our class representation. Given the point-like nature of features, the interpretation of the query instance and our class representation as graphs is straightforward. In Section 5.2, we will show how to interpret our problem in terms of the graph matching formulation. Furthermore, graph matching permits to take into account inter-relations between multiple features. In this way, stronger geometrical constraints can be introduced in place of a simpler coordinate regression. Finally, graph matching results can be interpreted in probabilistic terms as a soft assignment. We will show in Section 5.3 how we easily integrate this soft matching probability into the probabilistic framework that we presented in Section 4.4. At the end of this Chapter, we will experimentally show how the introduction of graph matching in our algorithm leads to a higher accuracy in the pose estimation performance. This means that we have successfully and simultaneously dealt with the two sources of error highlighted at the beginning of this Chapter: wrong decisions due to the pose classifier and lack of geometrical information.
In the following section, we will present an introduction to graph matching along with references to related works where graph matching is employed for Computer Vision applications.

5.1 Graph Matching Fundamentals

Since the early days of Computer Vision [32, 5], graphs have been considered as a useful tool to interpret many problems, where local structures, like image regions or features, are envisaged in the problem formulation. According to the graph

[10] When the order of the considered relationships is greater than 2, the terms *hypergraph* and, consequently, *hypergraph matching* are used.

framework, image structures are represented by graph nodes and the structural relationship between each pair of structures is encoded by a graph edge. Therefore, tasks such as object classification or recognition, where image regions or parts described by features are to be matched, can be naturally solved by graph matching techniques. The goal of graph matching is thus to find the best match between the two graphs, *i.e.*, to set the nodes of the two graphs in correspondence in such a way that the unary information of the nodes and the pairwise information of the edges are preserved as much as possible.
Since graph matching is a NP-hard problem, the solution becomes more computationally demanding as soon as the number of nodes increases. This is usually the case when graph nodes represent features like in this thesis, as the number of features extracted from an image can easily reach several thousands. For this reason, a lot of research has focused on approximations and heuristics to reduce the computational burden. Many different solutions have been proposed, such as those based on spectral decomposition [67, 125, 12, 10], and relaxation labeling [11, 13, 70]. Since we also employ a graph matching algorithm based on spectral decomposition, we will provide a deeper analysis of related works that are focused on this technique.

5.1.1 Related works

In a nutshell, graph matching algorithms based on spectral decomposition perform matching on the basis of the similarity between the spectra of the adjacency matrix of the two graphs. Spectral graph matching algorithms have wide applicability because they do not impose any constraint on the type of node-to-node as well as edge-to-edge affinities. The main advantage lies in its efficiency, as the integer constraints on the solution are dropped. This relaxation, also known as *linear relaxation*, makes the problem tractable when the number of nodes increases, as the complexity of the problem is reduced from NP-hard to a low-order polynomial. Another important aspect of spectral-based graph matching lies in the intuitive interpretation of its solution, which is based on the structure of the affinity matrix. In fact, the spectral formulation directly shows why correct matches, that respect the geometric alignment of the two graphs, are favored with respect to wrong matches that produce "random" alignments.
In [67], the authors present a spectral-based graph matching technique to solve object recognition and object categorization problems. The spectral method proposed bases its efficiency on a double relaxation, as the authors relax both the integer constraint as well as the one-to-one matching constraint, *i.e.*, when matching should result in a bijective mapping of the nodes of the two graphs. The latter constraint is eventually re-introduced in the solution in a later discretization step, where a greedy algorithm selects the best assignment on the basis of the graph

matching results. In [15], the authors present a spectral relaxation technique that incorporates one-to-one or one-to-many constraints within the relaxation scheme. On an abstract level, the proposed solution parallels that of [67], as the affine constrains are embedded by directly modifying the affinity matrix according to the problem at hand, and discretization is again performed as a post processing step. More importantly, the authors propose an additional normalization for the graph matching scoring function by transforming the original affinity matrix into a compatibility matrix and by applying a bistochastic normalization on it.
Several works have explored hypergraph matching techniques to enforce constraints of higher order. In [129], each feature set is modeled by a hypergraph, where the complex relations are represented by hyper-edges. Consequently, a match between feature sets is then modeled as a hypergraph matching problem. First, a soft matching criterion is formalized, as opposed to previous methods where soft matching just results from a relaxation of the hard matching problem. Second, the authors establish an algebraic relation between the hyper-edge weight matrix and the desired vertex-to-vertex probabilistic matching, that finally results in a convex optimization problem. In [21], a method for matching two sets of points using high-order tensors and geometric constraints is proposed. The authors build a three-dimensional tensor that capture the affinity between triplets of points. Then, they derive an approximate solution by using a power iteration algorithm to find the main eigenvector of the affinity matrix in a similar fashion to [67]. Very importantly, the solution obtained is only guaranteed to converge to a stationary point, and not to the global optimum. In [22], the authors propose a method to match two images on the basis of an energy functional that considers both feature matching and distance potentials. First, SIFT features are extracted from a dense grid, concatenated on small neighbourhoods and coded through sparse coding. Max pooling is eventually applied to larger image regions to obtain the region descriptor. The energy functional is defined as a sum of unary terms depending on the region descriptor and pairwise term that depends on the distance between the regions. The energy is minimized with a variation of the Ishikawa method [57] to accommodate for the non-convexity of the current problem. Even though hypergraph matching-based approaches have shown to give better results, the price to pay is a much higher computational burden due to the introduction of tensors and a greater implementation complexity.

5.1.2 Formulation

Let us consider two fully connected graphs $\mathcal{G}$ and $\mathcal{G}'$, representing two different sets of elements of possibly different cardinality, where each element is a node in the corresponding graph. Let us also define an affinity measure $S_1(i, i')$, where $i \in \mathcal{G}$ and $i' \in \mathcal{G}'$, that evaluates the similarity between two nodes of the two

graphs according to some criterion. The solution to a graph matching problem is the mapping $\mathcal{P}$ between the two graphs, $\mathcal{P} = \{(i, i') | i \in \mathcal{G}, i' \in \mathcal{G}'\}$, such that the global similarity score

$$S = \sum_{(i,i') \in \mathcal{P}} S_1(i, i') \tag{5.1}$$

is maximized. Usually, additional constraints can be added to the problem. Typical constraints require that each element in $\mathcal{G}$ has one single match in $\mathcal{G}'$ with or without multiple assignments, commonly known as *many-to-one* and *one-to-one* mapping, respectively.
In its simplest form, graph matching amounts to finding the best correspondence for each node in the first graph only on the basis of its individual similarity to nodes of the second graph and in accordance with the matching constraints. This problem is also well-known under the name of *bipartite graph matching*. The general graph matching paradigm extends this simple formulation by taking into account not only matches between individual nodes but also matches between m-tuples. For sake of simplicity, we will describe the case of $m = 2$, but the extension to higher orders is straightforward by using tensors in place of two-dimensional matrices.
Let us define for each pair of assignments (i, i') and (j, j'), an additional affinity measure $S_2(i, j, i', j')$, possibly different from the individual affinity measure, that evaluates the similarity between the pair of nodes (i, j) in the first graph and the pair of nodes (i', j') in the second graph. We can summarize the individual and pairwise affinities in a single $N \times N$ matrix $\mathbf{M}$ with $N = n \times n'$, where n and n' are the number of nodes in $\mathcal{G}$ and $\mathcal{G}'$, respectively. Each element of the two-dimensional matrix $\mathbf{M}$ will be identified by the four indexes of the corresponding graph nodes, as follows

- $\mathbf{M}_{ii',ii'}$, *i.e.*, a diagonal element, contains the individual node affinity $S_1(i, i')$ between node $i \in \mathcal{G}$ and node $i' \in \mathcal{G}'$.
- $\mathbf{M}_{ii',jj'}$, *i.e.*, a off-diagonal element, contains the pairwise affinity $S_2(i, j, i', j')$ of the node pair (i, j) of the first graph and the node pair (i', j') of the second graph.

Furthermore, without loss of generality, we assume that $\mathbf{M}$ is symmetric, *i.e.*, $\mathbf{M}_{ii',jj'} = \mathbf{M}_{jj',ii'}$, and we finally require that S_1 and S_2 yield non-negative affinities. On the basis of the affinity matrix $\mathbf{M}$, we can consider the assignments (i, i') and (j, j') as nodes of an auxiliary undirected graph $\mathcal{H}$, where the pairwise affinity $\mathbf{M}_{ii',jj'}$ is the weight on the edge that connects the two nodes, while the individual score $\mathbf{M}_{ii',ii'}$ and $\mathbf{M}_{jj',jj'}$ can be considered as weights of the corresponding loop edges at the nodes. Therefore, $\mathbf{M}$ represents the adjacency matrix of $\mathcal{H}$, and so the graph matching problem has now a second-order formulation.

In order to solve the problem, we have to find the set $\mathcal{P}$ of matches (i, i') that maximizes the total score

$$S = \sum_{ii',jj' \in \mathcal{P}} \mathbf{M}_{ii',jj'} \tag{5.2}$$

such that the mapping constraints are met. We can represent $\mathcal{P}$ by an indicator vector $\mathbf{x}$, such that $\mathbf{x}_{ii'} = 1$ if $(i, i') \in \mathcal{P}$ and zero otherwise. Therefore, we can rewrite the total score as:

$$S = \sum_{ii',jj' \in \mathcal{P}} M_{ii',jj'} = \mathbf{x}^T \mathbf{M} \mathbf{x} \qquad \text{such that } \mathbf{x} \in \{0,1\}^N \text{ and } \mathbf{C}\mathbf{x} < \mathbf{b} \tag{5.3}$$

where $\mathbf{C}\mathbf{x} < \mathbf{b}$ represents a set of constraints expressed in an affine inequality form that can be imposed on the solution. The optimal solution $\mathbf{x}$ is the binary vector $\mathbf{x}^*$ that maximizes the score, given the mapping constraints

$$\mathbf{x}^* = \arg\max \mathbf{x}^T \mathbf{M} \mathbf{x} \qquad \text{such that } \mathbf{x} \in \{0,1\}^N \text{ and } \mathbf{C}\mathbf{x} < \mathbf{b} \tag{5.4}$$

The maximization problem in Equation (5.4) belongs to the family of Integer Quadratic Problems (IQP), which are known to be NP-hard. Therefore, different approximate solutions have been proposed according to the application at hand and the mapping constraints [8, 79]. Here, we describe in details the relaxation that we adopt in this thesis and that has been proposed in [67].
According to this,

- The mapping constraint is such that each node of $\mathcal{G}$ is mapped to only one node in $\mathcal{G}'$, but the same node in $\mathcal{G}'$ can have one or more corresponding nodes in $\mathcal{G}$, *i.e.*, a many-to-one mapping. In mathematical terms, this constraint is expressed as $\|\mathbf{x}\|^2 = n$.
- The integral constraint is dropped. That is, we allow $\mathbf{x}$ to take real values in $[0, 1]$.

Therefore, the new graph matching problem that we need to solve is expressed as follows,

$$\mathbf{x}^* = \arg\max \mathbf{x}^T \mathbf{M} \mathbf{x} \qquad \text{such that } \mathbf{x} \in [0,1]^N \text{ and } \|\mathbf{x}\|^2 = n \tag{5.5}$$

As the norm is fixed, we can simply rewrite the equation above as

$$\mathbf{x}^* = \arg\max \frac{\mathbf{x}^T \mathbf{M} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \qquad \text{such that } \mathbf{x} \in [0,1]^N \text{ and } \|\mathbf{x}\|^2 = n \tag{5.6}$$

thus transforming the graph matching problem into a Rayleigh's quotient problem. The relaxation of the integer constraint implies that all the components of

the solution $\mathbf{x}^*$ may be non-zero, *i.e.*, the solution set $\mathcal{P}$ can contain up to all candidate matches. In this case, the solution components $\mathbf{x}^*_{ii'}$ represent the degree of association of (i, i') with the solution set $\mathcal{P}$. The larger the value of $\mathbf{x}^*_{ii'}$, the higher the confidence about the correctness of the match. As a further consequence, the integer relaxation affects also the mapping constraint by transforming the many-to-one mapping to a many-to-many mapping constraint with fixed norm.

As the mapping is now many-to-many with fixed norm, only the relative values between the elements matter. Thus, we can fix the norm of $\mathbf{x}$ to 1 without loss of generality.

The solution to this problem is known to be the principal eigenvector of $\mathbf{M}$, where the principal eigenvector $\mathbf{v}_{\max}$ is the eigenvector that corresponds to the largest eigenvalue $\lambda_{\max}$ of the square matrix $\mathbf{M}$. We can show that this solution respects the mapping constraint by means of the Perron-Frobenius theorem,

Theorem 2: Perron-Frobenius' theorem. *Let $A = (a_{ij})$ be an $n \times n$ positive matrix: $a_{ij} \geq 0$ for $1 \leq i,j \leq n$. Then the following statements hold*

1. *There is a positive real number r, called the Perron root or the Perron–Frobenius eigenvalue, such that r is an eigenvalue of A and any other eigenvalue λ is strictly smaller than r in absolute value, i.e., $r > |\lambda|$.*

2. *The eigenvector $\mathbf{v}$ of A with eigenvalue r have all positive components: $\mathbf{v}_i \geq 0$ for $1 \leq i \leq n$*

Since $\mathbf{M}$ has non-negative elements, the Perron-Frobenius theorem guarantees that the principal eigenvector has all positive components. If we combine this with the unit norm constraint, we are guaranteed that the components of $\mathbf{x}^*$ will be in the interval $[0, 1]$. From an intuitive point of view, a consistently large set of correctly matching nodes will be identified in the spectral solution by high-score components. On the contrary, spurious matches due to random similarity will probably end up unconnected with the main cluster, thus resulting in low-score components.

In closing this section, we want to make some consideration on the complexity of the solution proposed here. Since eigenvalue problems have a general complexity of $O(n^3)$ for dense matrices, the size of $\mathbf{M}$ cannot be too large without making the problem intractable. One simple solution to work with large graphs, that we will also use, is to set thresholds on the pairwise affinity score in order to make $\mathbf{M}$ as sparse as possible. This has the effect to reduce the complexity of the problem to $O(n^{3/2})$, and is simultaneously beneficial for data storage, as sparse matrices have compressed storage formats.

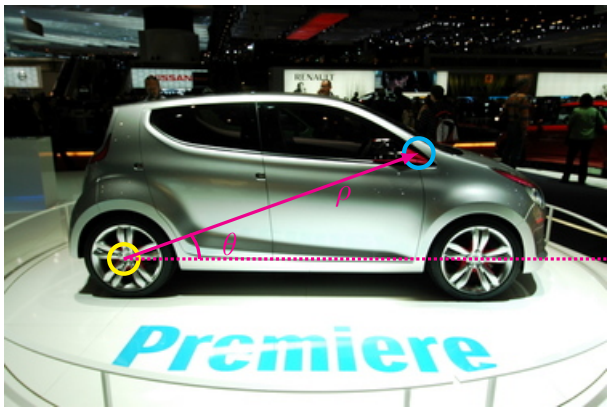


Figure 5.2: The off-diagonal attribute $A_{ij} = (\theta,\rho)$ on a sample image. θ is defined as the angle between the horizontal axis and the directed segment connecting vertices i and j, whereas ρ is defined as the length of the segment.

5.2 Graph Matching Interpretation of Our Problem

In this section, we describe how we integrate graph matching into our approach in order to favor geometrically consistent poses and simultaneously remove the negative effect of an inaccurate discriminative term.

Let us consider two separate graphs, one for the query image and one for our class representation defined in Chapter 4. More specifically, the two graphs are attributed graphs defined by the triplets $\mathcal{G} = (V, E, \mathbf{A})$ and $\mathcal{G}' = (V', E', \mathbf{A}')$, respectively. V and V' are the sets of vertices, where each vertex in V corresponds to a query feature and each vertex in V' corresponds to a generative cluster model representative. E and E' are the sets of edges connecting all vertices in V and V', respectively. The matrices $\mathbf{A}$ and $\mathbf{A}'$ are the attribute matrices. For $i \neq j$ and $i' \neq j'$, each entry A_{ij} and $A'_{i'j'}$ is a multi-dimensional attribute for the edges $e_{ij} \in E$ and $e_{i'j'} \in E'$, respectively, that represents some *relationship* between the corresponding vertices. For $i = j$ and $i' = j'$, *i.e.*, for the loops e_{ii} and $e_{i'i'}$, A_{ii} and $A_{i'i'}$ are also defined as multi-dimensional attributes with a different definition with respect to off-diagonal attributes.

With regard to the attribute matrix $\mathbf{A}$, the following assignment is considered

$$A_{ij} = \begin{cases} \mathbf{q}_i & \text{for } i = j \\ (\theta_{ij},\rho_{ij}) & \text{for } i \neq j \end{cases} \tag{5.7}$$

where $\mathbf{q}_i$ is the test feature descriptor, θ_{ij} is the angle between the x-axis and the directed segment P_{ij} connecting the locations of two test descriptors $\mathbf{q}_i$ and $\mathbf{q}_j$,

ρ_{ij} is the length of P_{ij}. A visualization of the off-diagonal attribute is given in Figure 5.2.
Regarding the model graph, we first define for each cluster representative its 2D location as the average location of all the feature descriptors contained in the cluster, and the model feature descriptor as the cluster representative $\mathbf{c}$. Therefore, the model attribute matrix $\mathbf{A}'$ is defined, similarly to $\mathbf{A}$, as follows

$$A'_{i'j'} = \begin{cases} \mathbf{c}^{i'} & \text{for } i' = j' \\ (\theta_{i'j'}, \rho_{i'j'}) & \text{for } i' \neq j' \end{cases} \tag{5.8}$$

where $\theta_{i'j'}$ is the angle between the x-axis and the directed segment $P_{i'j'}$ that connects the 2D location of cluster representative $\mathbf{c}^{i'}$ and $\mathbf{c}^{j'}$, wheareas $\rho_{i'j'}$ is the length of $P_{i'j'}$.
Correspondingly, each entry of the affinity matrix $\mathbf{M}$ is defined as follows:

$$M_{ii',jj'} = \begin{cases} \exp\left(-\frac{d_{ii'}}{m}\right) & \text{if } i = j \text{ and } i' = j' \\ \left(1 - \frac{\beta}{\tau_1}\right)\left(\frac{\tau_2 - \phi}{\tau_2 - 1}\right) & \text{if } \beta \leq \tau_1 \text{ and } 1 \leq \phi \leq \tau_2 \text{ and } (i \neq j \text{ or } i' \neq j') \\ \left(1 - \frac{\beta}{\tau_1}\right)\left(\frac{\tau_2 \phi - 1}{\tau_2 - 1}\right) & \text{if } \beta \leq \tau_1 \text{ and } \frac{1}{\tau_2} \leq \phi < 1 \text{ and } (i \neq j \text{ or } i' \neq j') \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

The first line refers to the diagonal entries of the attribute matrix, and it takes only appearance into account as in standard feature matching. Since $d_{ii'} = \|\mathbf{q}_i - \mathbf{c}^{i'}\|$ is the distance in descriptor space and $m = \max d_{ii'}$, a high entry is assigned to feature pairs that are close in descriptor space. The remaining three lines involve the enforcement of the geometric structure, where $\beta = |\theta_{ij} - \theta_{i'j'}|$ is the absolute angular distance and $\phi = \frac{\rho_{ij}}{\rho_{i'j'}}$ is the Euclidean distance ratio. The absolute orientation difference and the length ratio of the two segments are compared to two thresholds, τ_1 and τ_2, and a matching score is defined accordingly. A high entry is assigned to feature pairs whose locations are geometrically consistent, both in orientation and length.
According to the graph matching formulation in Section 5.1, we know that the assignment that maximizes the graph matching score $\mathbf{x}^*$ is such that

$$\mathbf{x}^* = \arg\max \mathbf{x}^T \mathbf{M} \mathbf{x} \qquad \text{such that } \mathbf{x} \in [0,1]^N \text{ and } \|\mathbf{x}\|^2 = 1 \tag{5.10}$$

and $\mathbf{x}^*$ is the main eigenvector of $\mathbf{M}$.
Given the relaxation in Section 5.1, each entry $x^*_{a,b}$ of the solution $\mathbf{x}^*$ is not integer anymore. Even though a greedy discretization approach could be used to identify

a set of one-to-one matches (cf. [67]), we opt to keep the real-valued entries in view of a later probabilistic formulation. As a matter of fact, each entry x^*_{ab} can be taken to represent the degree of association of the potential match to the main matching set, *i.e.*, the confidence that we have on the correctness of the match. Furthermore all entries in $\mathbf{x}^*$ are in $[0, 1]$, so that we can embed the graph matching results in our probabilistic formulation in a straightforward way.

5.3 Graph Matching Integration in Our Probabilistic Formulation

In Section 4.5, we set our pose estimation algorithm in a probabilistic framework by assuming that each query descriptor $\mathbf{q}_i$ was informative about the viewpoint $\boldsymbol{\alpha}$ under which the object is observed. More specifically, we considered a probability function $p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha})$, which we reformulated in Equation (4.46) and (4.47) as

$$p(\mathbf{q}_i|\boldsymbol{\alpha}) = \sum_{k=1}^{L} p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha}) = \sum_{k=1}^{L} \gamma_{\mathbf{q}_i\mathbf{c}^k} \exp\left(-\frac{1}{2}\sum_{j=1}^{J_k}(\mathbf{e}_i^{jk}(\boldsymbol{\alpha}))^T(\mathbf{R}^{jk})^{-1}\mathbf{e}_i^{jk}(\boldsymbol{\alpha})\right). \tag{5.11}$$

Since we matched each query descriptor $\mathbf{q}_i$ to the closest generative cluster model representative $\tilde{\mathbf{c}}^i$, we set $\gamma_{\mathbf{q}_i\tilde{\mathbf{c}}^i} = 1$ and 0 otherwise. This had the disadvantage that potentially good clusters with a slightly greater distance to the query descriptor were immediately discarded.

Thanks to the replacement of nearest neighbor matching with graph matching, we can take into account multiple matches for the same query feature $\mathbf{q}_i$ in a consistent way, as all the matches are scored according to their compatibility. We will show in the following how our probabilistic formulation can be changed in order to accommodate for graph matching.

As in Chapter 4, we first extract a set of features $\mathcal{Q} = \{\mathbf{q}_i\}_{i=1}^N$ from the query image. Differently from Chapter 4, where we matched each query feature separately against the set of generative cluster model representatives, we now use both sets as input to the matching algorithm. That is, we build the query graph $\mathcal{G}$ from the set of query features $\mathcal{Q}$ and we build the model graph $\mathcal{G}'$ from the set of generative cluster model representative $\mathcal{C} = \{\mathbf{c}^k\}_{k=1}^L$. On the basis of the two graphs, we perform graph matching as described in Section 5.1 and Section 5.2.

After the graph matching step, for each feature $\mathbf{q}_i$ we define the probability $p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha})$ in a straightforward way. The term $p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha})$ is expressed as in Section 4.5

$$p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha}) = \gamma_{\mathbf{q}_i\mathbf{c}^k} \exp\left(-\frac{1}{2}\sum_{j=1}^{J_k}(\mathbf{e}_i^{jk}(\boldsymbol{\alpha}))^T(\mathbf{R}^{jk})^{-1}\mathbf{e}_i^{jk}(\boldsymbol{\alpha})\right) \tag{5.12}$$

where J_k is the number of generative feature models in cluster K^k.
$\mathbf{e}_i^{jk}(\boldsymbol{\alpha}) = \mathbf{q}_i - F^{jk}(\boldsymbol{\alpha})$ is the prediction error made by the j-th generative feature model in cluster k, and $\mathbf{R}^{jk}$ is its covariance matrix. We derive the term $\gamma_{\mathbf{q}_i\mathbf{c}^k}$ from the graph matching step.
Since $\|\mathbf{x}\| = 1$ and $x_{ik}^* \in [0, 1]$, we can interpret the square of each score as a probability term ranking the compatibility between $\mathbf{q}_i$ and C^k

$$\gamma_{\mathbf{q}_i\mathbf{c}^k} = (x_{ik}^*)^2 \tag{5.13}$$

In a nutshell, the graph matching step has been used to rank the compatibility of each query feature $\mathbf{q}_i$ with each regressor C^k on the basis of the overall geometric arrangement of the matches.
In Equation (4.40), we introduced the discriminative term $p_d(I_q|\boldsymbol{\alpha})$ to include a pose classifier. The classifier is used to generate a searching interval that is limited to the output bin. Given the strength of the graph matching step and the potential errors due to the hard decision made by the pose classifier, we just let the graph matching results drive the maximization by removing the discriminative term. Experimentally, this proves to be beneficial, as we are able to decrease the mean absolute error by 25%, as shown in Section 5.4.
Now, the posterior distribution for $\boldsymbol{\alpha}$ is obtained from Equation (4.49) by eliminating the discriminative term and introducing the graph matching term as

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \prod_{i=1}^{N} \sum_{k=1}^{L} p(\mathbf{q}_i, \mathbf{c}^k|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) \tag{5.14}$$

5.4 Experimental Results

In this section, we show that the introduction of geometrical context in the algorithm proposed in Chapter 4 is beneficial, as our method obtains now much more accurate results. We test our modified approach on two car datasets: the EPFL multi-view car dataset [90] and the PASCAL VOC 2006 dataset [24]. The EPFL dataset has already been used in Chapter 4 for evaluation, and thus we reuse it here to directly show that the introduction of the graph matching step is beneficial. We use the PASCAL VOC 2006 dataset to prove that the graph matching step permits to obtain a more accurate pose estimation even when the pose delivered by the classifier is 100% correct.

5.4.1 EPFL multi-view car dataset

Before showing the better performance of our method in estimating the pose for object categories, we compare it to our baseline in the same preliminary experiment we performed in Section 4.3.1.

Table 5.1: Performance of the new method and the method proposed in Section 4.3.1 with respect to the four configurations of feature detector and descriptor (cf. Table 4.1). The configuration SURF detector + SIFT descriptor obtains again the best mean absolute error (MAE).

	Configurations			
	SIFT+SIFT	SIFT+SURF	SURF+SIFT	SURF+SURF
Fenzi *et al.* (2013) [29]	**2.35**	11.26	0.99	**1.29**
Proposed [28]	2.41	**10.40**	**0.96**	**1.29**

5.4.2 Single Instance Pose Estimation

We test our algorithm on the first 10 car sequences of the EPFL dataset with a 33% split between training and testing, *i.e.*, one image every three for learning and the rest for testing. For each sequence, we track features over the training images and we compute a regression function for each track, as described in Chapter 4. The class model in this case coincides with the exemplar model itself and no actual clustering is performed.

For each test image, we extract a set of query features and, for each feature, we find the two nearest neighbor tracks in the model. Then, we apply our graph matching-based approach to score the candidate matches, as explained in Section 5.3. Finally, we obtain the estimation from Equation (5.14).

We evaluate the performance of our new method using the four combinations that we also used in Section 4.3.1. We compare the mean absolute error (MAE) in degrees between the ground truth orientation and the result returned by each algorithm.

In three of the four configurations considered, our new method has practically the same performance as our old method, indicated by "Fenzi *et al.* (2013)", as reported in Table 5.1. On the contrary, the improvement for the configuration SIFT+SURF is larger, as this configuration suffers more from flipping errors. We show in Figure 5.3 how the graph matching step is beneficial with respect to the original method.

5.4.3 EPFL multi-view car dataset

In this section, we compare our method enriched with the graph matching step to the method we proposed in Chapter 4, indicated again as "Fenzi *et al.* (2013)" and to [118, 90]. We used the same testing framework, *i.e.*, two different splits between training and testing

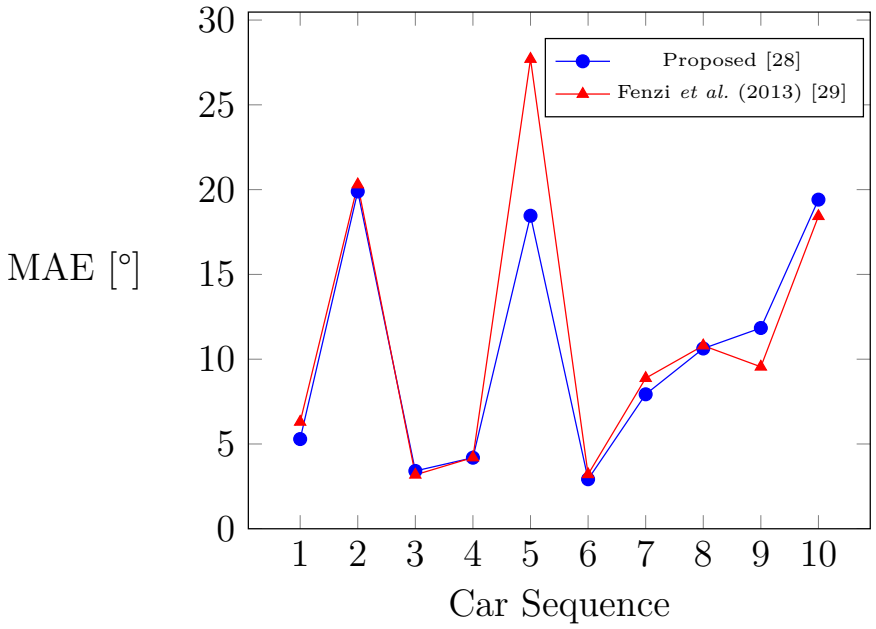


Figure 5.3: Single instance. Our new method outperforms [29] by approximately 5% on average for the configuration SIFT+SURF, with an improvement of approximately 33% on the hardest sequence.

50% Split: training the model on the first 10 sequences and testing it on the second 10;

Leave One Out (LOO): training the model on 19 sequences and testing it on the remaining one.

We build our model according to Chapter 4. Then, we extract a set of features $\mathcal{Q}$ from each query image and, for each feature, we find its five nearest neighbor clusters in the model. Then, we use our graph matching-based approach to assign a score to the candidate matches, as explained in Section 5.3. Finally, we estimate the car pose as in Equation (5.14).
In Table 5.2, we can see that our new method outperforms all others. In particular, our absolute MAE is 25% smaller with respect to the results presented in Chapter 4. This experimentally shows that introducing a soft geometric match is beneficial with respect to a hard decision based on the pose classifier [76]. In the two leftmost columns of Table 5.2, we also provide results of our method in terms of the 90^{th} and the 95^{th} percentile. Even without considering most of the large errors due to 180° flipped estimations, our method still obtains a better accuracy. As in [118] and in our baseline, the performance of our method with 50% and LOO splits is similar, showing that the model relies on the first 10 sequences to

Table 5.2: EPFL dataset. Our method compared to [90], [118] and [29].

Method	MAE [°] 90^{th} percentile	MAE [°] 95^{th} percentile	MAE [°]
Ozuysal *et al.* [90]	-	-	46.48
Torki *et al.* [118] - 50% split	19.4	26.7	33.98
Fenzi *et al.* (2013) [29] - 50% split	14.51	22.83	31.27
Proposed [28] - 50% split	**12.67**	**17.77**	**23.38**
Torki *et al.* [118] - LOO split	23.13	26.85	34.90
Fenzi *et al.* (2013) [29] - LOO split	**14.41**	22.72	31.16
Proposed [28] - LOO split	15.53	**19.27**	**24.53**

Table 5.3: PASCAL VOC 2006 dataset. Our extended method compared to our original method, without and with [76].

Method	MAE [°]
Fenzi *et al.* (2013) [29] without pose classifier	28.50°
Fenzi *et al.* (2013) [29] with pose classifier [76]	14.70°
Proposed [28]	**14.49°**

estimate the final pose, while the second 10 sequences seem to introduce a small amount of noise in the model. More precisely, we noticed in this experiment that the estimation of the pose is usually driven by two or three training cars belonging to the first 10 training sequences, which are more similar in appearance to the current test object. Since the second half is mainly comprised of uncommonly shaped cars, their inclusion in the training set is actually slightly disadvantageous.

5.4.4 PASCAL VOC 2006 dataset

In this section, we compare our extended method to the one proposed in Chapter 4 in order to compare what is the real strength of the graph matching step with respect to the pose classifier that we used before. In order to make the experiment more challenging, we used the pose classifier proposed in [76] in its best configuration, *i.e.*, when the pose interval is divided in only 4 bins.
We consider a subset of the images available in the PASCAL VOC 2006 test dataset. The subset comprises all the pictures where the car is in one of the following four annotated orientations: front, rear, left side, right side. Some sample pictures taken from this dataset are shown in Figure 5.4.
As shown in Table 5.3, our approach performs better than our old method by a

Figure 5.4: Sample images from the Pascal VOC 2006 car dataset.

factor of 2 when our baseline is used without any pose classifier. More importantly, the performance is still better even when the classifier is used. Unlike in the EPFL experiment, where the pose classifier has a substantial influence on the final error of our baseline method, its performance on this dataset is almost perfect (96% accuracy). Therefore, our method not only recovers the correct orientation over the whole pose range (360°), instead of the smaller (90°) correct interval given by the classifier, but it is also more accurate. We would like to highlight that our method is trained with the first 10 sequences of the EPFL dataset. Since the appearance of the cars for our training dataset is quite different from that of the cars depicted in the PASCAL VOC 2006 dataset, we indirectly show that our method can generalize and cope well with a change in domain.

Chapter 6

Enforcing Temporal Constraints

As shown in Chapter 5, the introduction of geometrical context in our algorithm permits to increase its performance by reducing the overall mean error from 31.27° to 23.38°. In spite of the improvement achieved, there is still room to enhance our algorithm in order to obtain a larger gain in performance.
For this purpose, we first analyzed how the estimation produced by our algorithm changes with respect to the ground-truth orientation. Secondly, we investigated on the shape of our pose posterior distribution in order to understand what happens when the returned pose is wrong. From this analysis, we can draw the following two important conclusions:

1. The performance of our algorithm depends on the ground truth viewpoint, *i.e.*, certain viewpoints are harder to estimate than others.

2. The posterior distribution always contains a strong evidence of the correct viewpoint.

In Figure 6.1, we give a visual example of the first fact. We plot the performance of our algorithm in terms of the mean absolute error for each frame of a test sequence, in which the object rotates around its central axis. We can see that the algorithm performance is very good on the great majority of the frames with the exception of two bursts of large errors, the first from frame 35 to frame 40 and the second from frame 46 to frame 56. This phenomenon occurs in almost all sequences where large errors seem to be concentrated around specific viewpoints, at which our algorithm provides a poor performance.
In Figure 6.2, we visualize the second fact by plotting the pose posterior distribution estimated by our algorithm for a single frame. As it can be seen, the posterior distribution is multi-modal with two strong, largely separated peaks. Often, our algorithm estimates a bi-modal distribution as a consequence of the large similarity in appearance that an object class may share in different views, as we already noted in Chapter 5. When the geometrical context is not powerful enough to disambiguate these situations, the largest mode returned by our algorithm may be

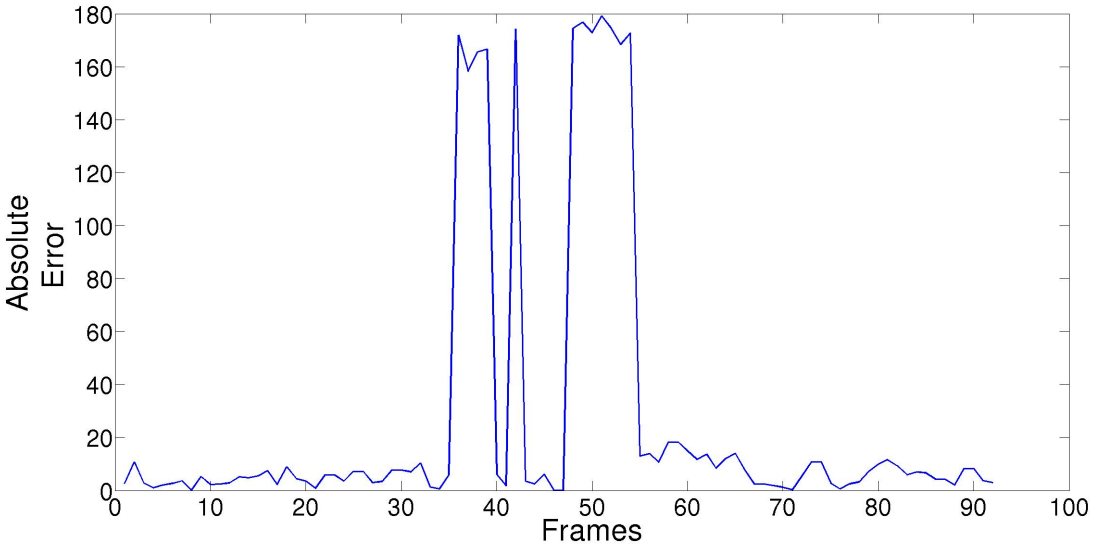


Figure 6.1: Performance of our algorithm on a car sequence from the EPFL dataset.

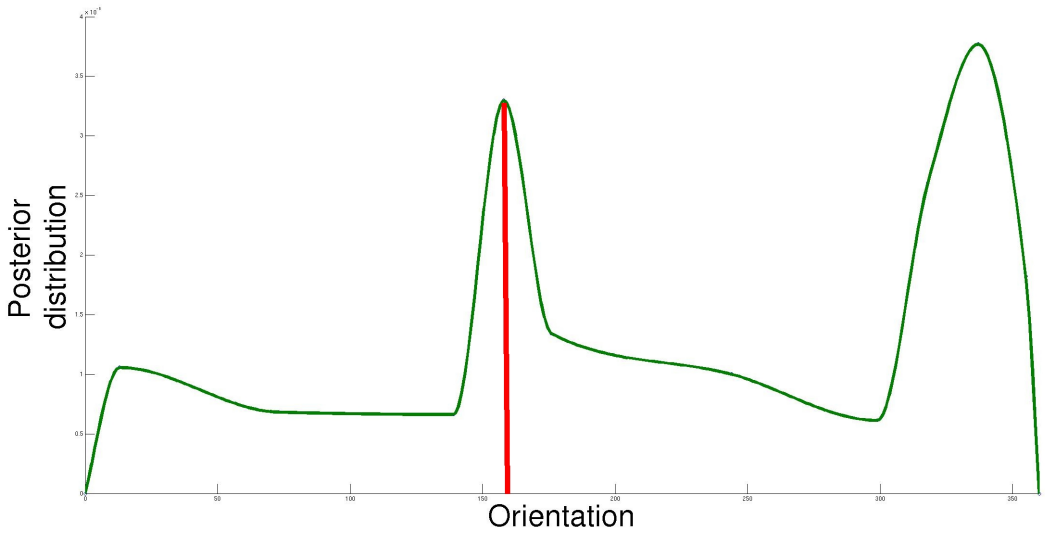


Figure 6.2: Pose posterior distribution estimated by our algorithm on a single image. The ground-truth orientation is indicated by the red segment and it corresponds to the minor mode of this bi-model distribution.

associated to a wrong estimation. Nonetheless, Figure 6.2 clearly shows that the minor mode is associated to the ground-truth viewpoint.

Until now, we have always assumed that the input to our method is a single image. However, algorithms for pose estimation of object categories in video sequences are becoming increasingly popular, as a result of the availability of large annotated video datasets [95, 93, 37, 127]. The interest in video-based approaches is naturally motivated by the fact that for many activities and applications, such as autonomous driving or human-robot-interactions, the target object pose to be estimated is a variable that continuously evolves over time. As a straightforward advantage, video-based approaches can exploit temporal information, which is a very strong cue to enforce consistency upon pose estimation results. With this in mind, we will extend here the algorithm we presented in Chapters 4 and 5 to deal with the problem of pose estimation for object categories in videos. This gives us the possibility to test our generative feature model paradigm on a different domain. Oftentimes, video sequences are recorded "in the wild", video frames are affected by strong motion blur, and the resolution can be low in order to keep the file size reasonably small. Thus, we cannot take for granted that our method will have a similar performance on videos as on still images without any experimental evaluation.

Before showing the last contribution of this thesis, we would like to present a short description of related works of approaches for pose estimation of object categories in video sequences. The categorization we used in Chapter 3 can be extended to video-based approaches in a straightforward manner. That is, some approaches rely only on 2D data [99], like we also do, while others use 3D CAD models for training their algorithms [127, 81, 82].

With regard to 2D-based approaches, the only approach known to the author has been recently presented in [99]. It relies on a modified Hough regression forest that estimates the pose distribution at each frame, and then fuses it with a smoothed average of the past n distributions before determining the maximum a posteriori pose. This approach can be prone to error accumulation in case of consecutively wrong estimations. A broader and very interesting point of view has been taken by [38], where pose estimation of vehicles is inserted within a traffic scene understanding framework. Street scenes are probabilistically analyzed on the basis of a combination of vehicle tracklets, semantic scene labeling, scene flow and egomotion, and vanishing point estimation. Therefore, pose estimation results are temporally consistent not only within themselves, but also with respect to the considered environment.

Regarding 3D-based approaches, [81] learns a statistical manifold of SIFT features from short video sequences of different class instances as a function of appearance and viewpoint. Analogously, a test video is divided in short batches and a discrete

pose is estimated as the closest in terms of KL-divergence to the set of short training sequences. In the follow-up [82], a statistical manifold is now learned for spatio-temporal object parts, whose selection is driven by a twofold criterion of descriptiveness and distinctiveness, and then a discrete pose is estimated by means of KL-divergence. These two works permit to obtain only a very coarse viewpoint, and, more importantly, temporal information is not used to enforce consistency in the estimation of the single video batches. On the contrary, [127] estimates a probabilistic distribution at each frame combining the results of a pose estimator [126] and a motion prior on the viewpoint change. The pose estimator is based on parts learned from training images rendered from CAD models using a structural SVM optimization. The maximum a posteriori pose is prone to error accumulation, as it is obtained using a particle filtering framework that relies only on the past frame information.
Similarly to [127], we also base our strategy to extend our algorithm on a parallelism with tracking. In a nutshell, we create a set of pose "observations" at each frame by sampling from the corresponding posterior distributions estimated by our core algorithm. Then, we find the *best* sequence of pose observations that is supported by the posterior distributions and, at the same time, respects a simple temporal consistency assumption, *i.e.*, large pose changes in nearby frames are not admitted. As a matter of fact, this strategy can be simply viewed as an object tracking problem, where the pose observations represent object detections and the goal of the problem is to find the pose trajectory of the target object in the given video sequence.
Object tracking has been commonly solved by expressing the tracking problem as a linear program (LP), either in monocular sequences [65] or multi-view sequences [64]. Inspired by this, we also express our pose estimation problem using an LP formulation. While the aforementioned related works combine temporal information by using suboptimal strategies [99] or computationally expensive techniques [127], our LP formulation guarantees to achieve a global optimum and has an efficient implementation.
In Section 6.1, we will give a very brief introduction to Linear Programming and algorithmic solutions for linear programs. In Section 6.2, we will show how to formulate our pose estimation problem as a linear program. Finally, we conclude this chapter with an experimental section, where we show that our LP formulation leads to a large improvement in the accuracy of the estimated pose on three publicly available datasets.

6.1 Linear Programming

Linear programming is a technique for the maximization (or minimization) of a linear objective function, subject to linear equality and inequality constraints.

Linear programming has been applied to countless fields of study, ranging from business and economics to engineering and computer science. Since Fourier's work [114], linear programming has been considered one of the most important problems in the optimization field, so that a huge amount of works has been dedicated to it[11].
In formal mathematical terms, a linear program consists of a linear objective function

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{6.1}$$

subject to linear constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\leq b_2 \\ \vdots \qquad \vdots \qquad\qquad \vdots \quad &\leq \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\leq b_m \end{aligned} \tag{6.2}$$

This expression can be more compactly expressed as

$$\max\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in \mathbb{R}^n, \mathbf{A}\mathbf{x} \leq \mathbf{b}\}, \tag{6.3}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the coefficient matrix and $\mathbf{b} \in \mathbb{R}^m$ is the vector that defines the constraints on the LP. The problem constraints can be equalities and/or inequalities, as these can always be converted to a standard form without changing the semantics of the problem.
A point $\mathbf{x} \in \mathbb{R}^n$ is a *feasible* solution if it satisfies all linear constraints, and a feasible solution $\mathbf{x}$ is *optimal* if $\mathbf{c}^T\mathbf{x} > \mathbf{c}^T\mathbf{y}$ for all feasible $\mathbf{y} \in \mathbb{R}^n$. On the other hand, a linear program is unbounded if there exists no constant $M \in \mathbb{R}$ such that $\mathbf{c}^T\mathbf{x} \leq M$.
Let us consider a simple example represented in Figure 6.3 that will help to understand the algorithmic solution to a linear program. In the leftmost plot, we depicted the space of feasible solutions in yellow on the basis of the three linear constraints, while in the rightmost plot we represented the optimal solution as a yellow circle. The fact that the solution lies on a vertex of the polygon representing the space of feasible solutions is not a coincidence. As a matter of fact, the following theorem guarantees that the solution of a linear program always lies on a polygon vertex.

Theorem 3: *If a linear program* $\max\{\mathbf{c}^T\mathbf{x} : \mathbf{x} \in \mathbb{R}^n, \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$ *is feasible and bounded and rank*$(\mathbf{A}) = n$*, the LP has an optimal solution that is a vertex.*

[11] We forward the reader to the following literature on linear programming for a thorough treatment of the topic [114, 110, 19, 20, 88].

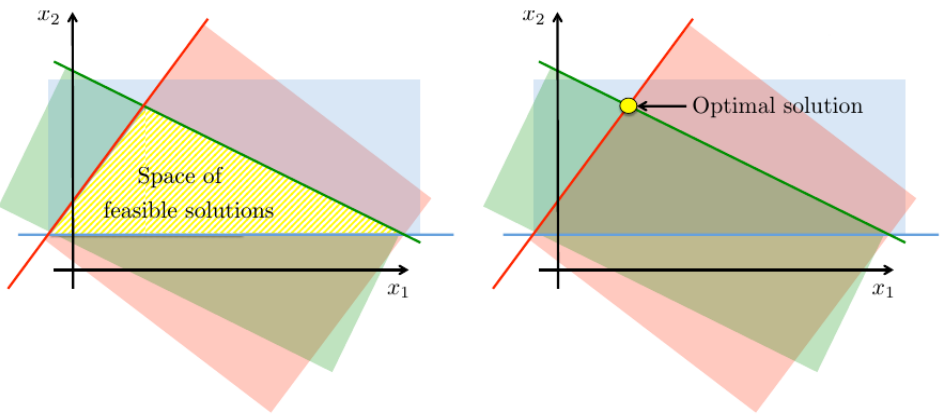


Figure 6.3: (Left) Space of feasible solutions of a linear program. (Right) Optimal solution of the same linear program. (Picture taken from [63])

On the basis of this theorem, an LP could be solved by simply enumerating all vertexes and picking the best one. Unfortunately, this becomes unmanageable as soon as the number of variables and constraints becomes large. Therefore, an algorithmic solution, the Simplex algorithm [18], has been proposed by George B. Dantzig in 1947, which drastically reduces the number of vertexes to be checked.

The Simplex method explores the set of vertexes in an iterative and efficient way in order to find the optimal solution. Starting from a vertex in the feasible region $\mathbf{x}$ that is not optimal, the idea is to move along the edges of the polyhedron to an adjacent vertex $\mathbf{x}'$, where $\mathbf{c}^T\mathbf{x}' > \mathbf{c}^T\mathbf{x}$. Since each move increases the objective function, convergence is guaranteed if the linear program is bounded. The two important aspects of the Simplex algorithm is the optimality criterion, *i.e.*, how to test if a vertex is optimal, and how to move to an adjacent vertex if the tested vertex is not optimal.

With regards to optimality, we first define the concept of basis. Each vertex of the feasible region lies at the intersection of n or more hyperplanes of the constraint set. Let us define the basis $B \subseteq \{1, \ldots, m\}$ with $|B| = n$ for a vertex $\mathbf{x}$, as the set of row indexes of $\mathbf{A}$, whose corresponding hyperplanes intersect at $\mathbf{x}$. If $\mathbf{x} = \mathbf{A}_B^{-1}\mathbf{b}_B$ is a feasible solution, where the subscript B selects only the rows of $\mathbf{A}$ indexed by B, then B is called a *feasible basis*. Furthermore, a basis is *optimal* if it is feasible and the unique $\boldsymbol{\lambda}$ such that $\boldsymbol{\lambda} \in \mathbb{R}^m$ with $\boldsymbol{\lambda}^T\mathbf{A} = \mathbf{c}^T$ and $\lambda_i = 0, \forall i \notin B$, satisfies $\boldsymbol{\lambda} \geq \mathbf{0}$. Now, given a vertex $\mathbf{x}$ with corresponding basis B, we can check if the vertex is optimal by computing the vector $\boldsymbol{\lambda}$ such that $\boldsymbol{\lambda}^T = \mathbf{c}^T\mathbf{A}^{-1}$. If any of the λ_i with $i \in B$ is less than 0, than the basis B, and thus the vertex $\mathbf{x}$ associated to it, is not optimal.

If we know that a vertex is not optimal, than we need to move to an adjacent vertex. If the vertex $\mathbf{x}$ is not optimal than there will be some $\lambda_i < 0$, so let us compute a vector $\mathbf{d} \in \mathbb{R}^n$ such that $\mathbf{A}_{B\setminus\{i\}}\mathbf{d} = \mathbf{0}$ and $\mathbf{a}_i^T\mathbf{d} = -1$, where $\mathbf{a}_i$ is the i-th row of $\mathbf{A}$.
If we move from $\mathbf{x}$ in the direction of $\mathbf{d}$ by a quantity $\varepsilon\mathbf{d}$, with $\varepsilon > 0$, we have

$$\mathbf{c}^T(\mathbf{x} + \varepsilon\mathbf{d}) = \mathbf{c}^T\mathbf{x} + \varepsilon\mathbf{c}^T\mathbf{d} = \mathbf{c}^T\mathbf{x} + \varepsilon\boldsymbol{\lambda}^T\mathbf{A}\mathbf{d} = \mathbf{c}^T\mathbf{x} + \varepsilon\lambda_i\mathbf{a}_i^T\mathbf{d} > \mathbf{c}^T\mathbf{x}, \tag{6.4}$$

as $\varepsilon > 0$, $\lambda_i < 0$ and $\mathbf{a}_i^T\mathbf{d} = -1$. Therefore, we are moving in the correct direction. Now, we have to determine the right amount of movement ε. We denote with $\mathcal{K}$ the set of indexes of the constraints that might be hit by $\mathbf{x} + \varepsilon\mathbf{d}$, that is,

$$\mathcal{K} = \{k : 1 \leq k \leq m, \mathbf{a}_k^T\mathbf{d} > 0\}. \tag{6.5}$$

There are two possible cases. If $\mathcal{K}$ is empty, this means that no vertex can be hit by moving in the $\mathbf{d}$ direction, and thus the LP problem is unbounded. On the contrary, if $\mathcal{K}$ is not empty, we have a set of k's such that $\mathbf{a}_k^T(\mathbf{x} + \varepsilon_k\mathbf{d}) = \mathbf{b}_k$. The optimal ε^* is the smallest of all the ε_k, as with any other ε_k the movement would lead us to a vertex outside the feasible region. Therefore, we move to the vertex $\mathbf{x}' = \mathbf{x} + \varepsilon^*\mathbf{d}$, whose basis is $B' = B \setminus \{i\} \cup \{k^*\}$, where k^* is the index associated to ε^*.
We have summarized the Simplex algorithm in Algorithm 3.

Algorithm 3 Simplex Algorithm

1: Start with a feasible basis B
2: **while** B is not optimal **do**
3: Let $i \in B$ the index with $\lambda_i < 0$
4: Compute $\mathbf{d} \in \mathbb{R}^n$ with $\mathbf{A}_{B\setminus\{i\}}^T\mathbf{d} = \mathbf{0}$ and $\mathbf{a}_i^T\mathbf{d} = -1$
5: Determine $\mathcal{K} = \{k : 1 \leq k \leq m, \mathbf{a}_k^T\mathbf{d} > 0\}$.
6: **if** $\mathcal{K} = \emptyset$ **then**
7: Assert that LP is unbounded
8: **else**
9: Let $k^* \in \mathcal{K}$ be the index where $\min_{k\in\mathcal{K}} \frac{\mathbf{b}_k - \mathbf{a}_k^T\mathbf{x}}{\mathbf{a}_k^T\mathbf{d}}$ is attained
10: Update $B = B \setminus \{i\} \cup \{k^*\}$
11: **end if**
12: **end while**
13: **return** $\mathbf{x}$ identified by B.

In the next section, we will show how the problem of finding the optimal pose trajectory of an unknown object in a video sequence can be interpreted and solved by means of Linear Programming and the Simplex algorithm.

6.2 LP Interpretation for Our Problem

Previously, we have described a rough sketch of the strategy in order to extend the algorithm presented in Chapters 4 and 5 from still images to video sequences. The goal of this extension is to exploit temporal information in such a way that consistency is enforced in the pose estimation. We first stated that in each frame the pose posterior distribution estimated by our algorithm contains a strong evidence for the correct viewpoint. Then, we proposed to sample from the posterior distribution at each frame in order to create a set of pose observations. On the basis of this set of pose observations, we set our goal as the one to find the best pose sequence over all frames. We can think of the best sequence as the pose which has the best support from the posterior distribution and, at the same time, it meets the constraints we enforce on the pose consistency.
Let $O = \{\mathbf{o}_k^t\}$ be a set of pose observations with $\mathbf{o}_k^t = (\boldsymbol{\alpha}_k^t, s_k^t)$, where $\boldsymbol{\alpha}_k^t$ is a feasible pose value and s_k^t is the score derived by the probability of that pose in frame t obtained from the distribution of Equation (5.14). A path in the graph is defined as a list of ordered pose observations $T = \{\mathbf{o}_{k_1}^{t_1}, \mathbf{o}_{k_2}^{t_2}, \cdots, \mathbf{o}_{k_N}^{t_N}\}$ with $t_1 \leq t_2 \leq \ldots \leq t_N$.
Our goal is thus to find the path T^* that best explains the pose observations. This is equivalent to finding the T that maximizes the posterior probability given the set of pose observations O

$$T^* = \arg\max_T \; P(T|O) \tag{6.6}$$

By further assuming that the observations are conditionally independent, Equation (6.6) can be rewritten as:

$$\begin{aligned} T^* &= \arg\max_T \; P(O|T)P(T) \\ &= \arg\max_T \prod_k P(\mathbf{o}_k|T)P(T), \end{aligned} \tag{6.7}$$

where $P(\mathbf{o}_k|T)$ corresponds to the score of the pose s_k derived from the estimated pose posterior distribution, and $P(T)$ can be represented by a Markov chain:

$$P(T) = P_{\text{in}}(\mathbf{o}_{k_1}^1) \ldots P(\mathbf{o}_{k_t}^t|\mathbf{o}_{k_{t-1}}^{t-1}) \ldots P_{\text{out}}(\mathbf{o}_{k_N}^N). \tag{6.8}$$

This formulation can be directly mapped into a minimum cost network flow problem by constructing a graph, where each node represents a pose observation. Then, our goal becomes that of finding the best trajectory in the graph, or the minimum cost path, which we do in an optimal way by solving a linear problem. We define a directed graph $\mathcal{G} = (V, E)$ with costs $C(i,j)$ and capacities $u(i,j)$ associated with every edge $(i,j) \in E$. An example of such a network is shown in Figure 6.4, which

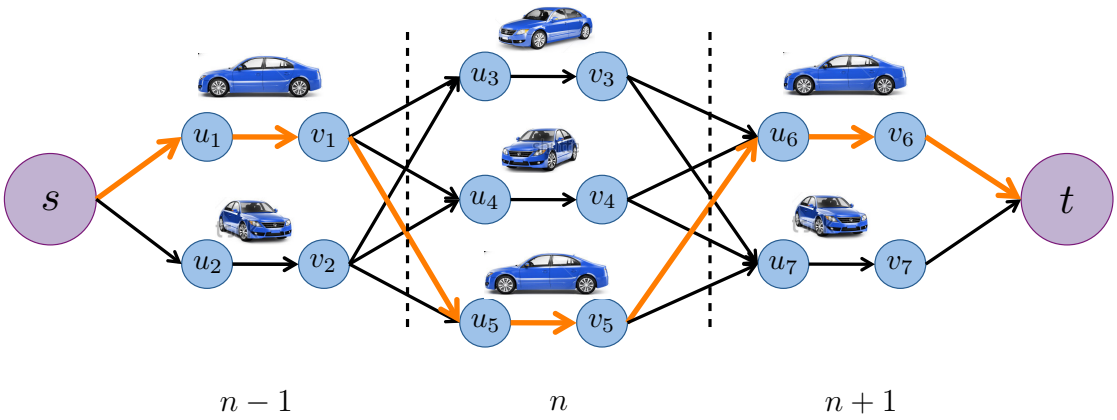


Figure 6.4: Example of a graph spanning 3 frames $(n-1, n, n+1)$, with the special source s and sink t nodes, and a total of 7 pose observations represented by two nodes each: beginning u_i and end v_i nodes. The end node of each pose observation is connected to all beginning nodes of the pose observations in the next Δf frames[12]. The optimal path is marked by orange arrows.

contains two special nodes, the source s and the sink t that have been added to the set of vertexes V, so that all flow that goes through the graph starts at s and ends at t.
More precisely, each pose observation $\mathbf{o}_i$ is represented with two nodes, the beginning node $u_i \in V$ and the end node $v_i \in V$ (see Figure 6.4), so that the pose score can be represented as the edge connecting u_i and v_i.
Finding the best path can be expressed as a Linear Program which is defined by a linear objective function and a set of linear constraints, as we described in Section 6.1. The linearization of the objective function of Equation (6.7) is obtained by defining a set of flow flags $f(i,j) = \{0,1\}$ which indicate if edge (i,j) is part of the optimum path solution or not. By defining the costs as negative log-likelihoods and combining Equations (6.7) and (6.8), the following objective function is obtained

$$\begin{aligned} T^* &= \arg\min_T -\log P(T) - \sum_k \log P(\mathbf{o}_k) \\ &= \arg\min_f \sum_i C_{\text{in}}(i) f_{\text{in}}(i) + \sum_{i,j} C_{\text{t}}(i,j) f_{\text{t}}(i,j) + \sum_i C_{\text{sc}}(i) f_{\text{sc}}(i) + \sum_i C_{\text{out}}(i) f_{\text{out}}(i) \end{aligned} \tag{6.9}$$

subject to the constraint that the flow that enters a node is equal to the flow that

[12]For ease of visualization, we represented in Figure 6.4 only link edges between consecutive frames.

leaves the node:

$$\begin{aligned} f_{\text{in}}(i) + \sum_j f_{\text{t}}(j,i) &= f_{\text{sc}}(i) \\ f_{\text{sc}}(i) &= f_{\text{out}}(i) + \sum_j f_{\text{t}}(i,j). \end{aligned} \tag{6.10}$$

Furthermore, we know that $f(i,j) = \{0,1\}$, a condition that we relax into $0 \leq f(i,j) \leq 1$. This tight relaxation allows us to have a linear program without losing the optimality guarantee. Finally, the overall flow that leaves s has to be 1, as well as the flow that enters t.

As we can see in Equation (6.9), there are four types of costs corresponding to four types of edges as shown in Figure 6.4. The link or transition costs $C_{\text{t}}(i,j)$ corresponding to the link edges connecting a pose observation from frame t to frame $t + \Delta f$. This cost represents the pose difference between the two observations. Assuming that the object pose cannot change a lot from one frame to the next, we define these costs to be an increasing function of the distance between pose observations. Therefore, the cost of a link edge is defined as

$$C_{\text{t}}(i,j) = -\log\left(1 - \frac{\|\alpha_j^{t+\Delta f} - \alpha_i^t\|}{M_\alpha}\right) + C(\Delta f) \tag{6.11}$$

where $C(\Delta f) = -\log\left(B_{\text{jump}}^{\Delta f - 1}\right)$ is the cost depending on the frame difference between pose observations. For all our experiments we allow matches up to two frames apart, which allows us to recover from pose estimation errors that occur on isolated frames. We set the parameter $B_{\text{jump}} = 0.3$. Edges between observations are only created if their pose distance is less than M_α degrees apart. This parameter allows us to adapt the algorithm to different frame rates and object speeds.

The score costs $C_{\text{sc}}(i)$ correspond to the pose observations and express how probable is that particular pose according to the learned distribution of Equation (5.14). The cost is defined as:

$$C_{\text{sc}}(i) = -\log\left(\frac{s_i^t}{M_s}\right), \tag{6.12}$$

where M_s is a normalization factor equivalent to the maximum score of all pose observations. The entrance cost $C_{\text{in}}(i)$ and exit cost $C_{\text{out}}(i)$ are set to 0, since we do not penalize any of the initial or last pose observations. As described in Section 6.1, the solution can be found efficiently using the Simplex algorithm.

In the following section, we provide a large experimental evaluation on three different publicly available datasets. These datasets envisage different difficulties, such as high intra-class variability, occlusions, truncations, and motion blur.

6.3 Experimental Results

In this section, we first show results of a preliminary experiment on the EPFL multi-view car dataset [90], a dataset we have extensively used in Chapter 4 and 5 for experimental evaluation. Later, we test the proposed video-based algorithm on two additional datasets, namely KITTI [37] and YouTube [127], where we will show that the extension we propose improves over the state of the art.

6.3.1 Preliminary experiment

The EPFL multi-view car dataset has been used for experimental evaluation in the previous chapters of this thesis, as it is one of the few publicly available datasets at the time of writing with ground-truth pose annotations. As the cars are just rotating on a pedestal, the change in orientation is smooth and predictable. Therefore, we can only consider this dataset for a preliminary experiment in which to test our method.
We compare our method to four single-frame state-of-the-art pose estimators [50, 47, 118, 90], the two methods we proposed in Chapter 4 and 5 indicated as "Fenzi *et al.* (2013)" and "Fenzi *et al.* (2014)", and one video-based approach that has been recently proposed [99]. We used the same testing framework as in the previous chapters, *i.e.*, two different splits for training and testing:

50% Split: training the model on the first 10 sequences and testing it on the second 10;

Leave-One-Out split: training the model on 19 sequences and testing it on the remaining one.

We build our model according to Chapter 4 and we estimate the pose posterior distribution in each frame as explained in Chapter 5. For each frame, we extract N pose values from the posterior distribution and construct the graph over all the frames of the sequence. The best path is found using the Gurobi library [45].
In Table 6.1, we show that our method not only outperforms all the other single-frame pose estimators, but it also obtains more accurate results than the other video-based approach [99]. In particular, our Mean Absolute Error (MAE) is approximately 80% smaller with respect to the results published in [50], which is the most accurate method on this dataset so far. As the results in the leftmost two columns show, the overall mean is affected by very few flipping errors. Since our method does not take a hard decision in each frame, but finds the path of orientations that best explains the whole sequence of pose observations, we can reduce the effect of these wrong estimations to a minimum.

Table 6.1: EPFL multi-view car dataset. Mean Absolute Error (MAE) and its 90^{th} and 95^{th} percentiles. The two leftmost columns show a clearer insight into the algorithms performance by removing the influence of large errors from the overall mean.

Method	MAE [°] 90^{th} perc.	MAE [°] 95^{th} perc.	MAE [°]
50% split			
Ozuysal *et al.* [90]	-	-	46.48
Torki *et al.* [118]	19.4	26.7	33.98
Fenzi *et al.* (2013) [29]	14.51	22.83	31.27
Redondo-Cabrera *et al.* [99]	-	-	29.7
Hara *et al.* [47]	7.73	16.18	24.24
Fenzi *et al.* (2014) [28]	12.67	17.77	23.38
He *et al.* [50]	-	-	15.8
Proposed [30]	**3.91**	**4.30**	**4.89**
Leave-One-Out split			
Torki *et al.* [118]	23.13	26.85	34.90
Fenzi *et al.* (2013) [29]	14.41	22.72	31.16
Fenzi *et al.* (2014) [28]	15.53	19.27	24.53
Proposed [30]	**5.6**	**6.26**	**7.10**

In the following sections, we test the performance of our method in real-world scenarios on two publicly available datasets, which present occlusions, truncations, and high motion blur.

6.3.2 KITTI dataset

The first test dataset consists of 11 sequences from the KITTI benchmark dataset [37], which were annotated by [127]. They show real-world scenarios of busy streets, where target cars undergo significant viewpoint changes. In some sequences, the car is partially or even totally occluded by other cars, pedestrians or road signs, making the pose estimation problem very challenging. An important difficulty of this dataset is that the car size varies significantly in each sequence, ranging from 500 pixels down to 40 pixels, as a result of the relative motion between the moving camera and the car.
We followed the same paradigm as in the preliminary examples. We extract features from each test image, we matched them against the model, which is learned

Table 6.2: KITTI dataset. Viewpoint accuracy/mean absolute error (MAE) in degrees

	Proposed 1st GT [30]	Proposed [30]	Xiang *et al.* 1st GT [127]	Xiang *et al.* [126]	Fenzi *et al.* (2014) [28]
KITTI01	**1.00/4.04°**	0.96/5.54°	0.95/6.5°	0.57/44.46°	0.35/64.87°
KITTI02	0.81/9.74°	0.67/12.81°	**1.00/5.40°**	0.33/119.54°	0.22/72.45°
KITTI03	**0.81/9.75°**	0.46/16.33°	0.42/15.64°	0.50/15.99°	0.13/62.81°
KITTI04	**0.72/10.55°**	0.65/12.10°	0.22/27.05°	0.17/58.42°	0.24/63.11°
KITTI05	**0.93/4.35°**	**0.93**/5.93°	0.36/23.59°	0.64/23.65°	0.76/12.82°
KITTI06	**1.00/5.02°**	**1.00**/5.08°	0.31/21.63°	0.59/20.29°	0.71/12.72°
KITTI07	0.78/12.78°	0.21/24.8°	**0.96/6.86°**	0.70/24.50°	0.09/56.80°
KITTI08	0.70/10.74°	**0.81/10.00°**	0.57/15.61°	0.67/23.26°	0.52/42.00°
KITTI09	**0.90/8.23°**	**0.90**/8.33°	0.50/21.63°	0.50/17.60°	0.33/32.85°
KITTI10	**0.92/7.09°**	**0.92**/7.18°	0.81/7.99°	0.44/56.78°	0.47/49.20°
KITTI11	0.54/16.00°	0.48/29.7°	**0.88/9.33°**	0.68/12.29°	0.32/76.17°
Mean	**0.83/8.93°**	0.73/12.53°	0.63/14.66°	0.53/37.89°	0.38/49.62°

from the first 10 sequences of the EPFL dataset, and then we obtain a pose posterior distribution. Finally, we sample N pose values from the posterior distribution, we construct the graph over all frames, and we find the pose path that best explains the pose observations over the sequence. To cope with the small size of the targets, we up-sample the images by 2, since our approach is based on local sparse features and we need a minimum amount of features to estimate a reliable posterior distribution.
In order to evaluate viewpoint estimation, we report two metrics:

Viewpoint Accuracy: the ratio of estimated viewpoints whose deviation from the ground truth viewpoint is less than 15°

Mean Absolute Error (MAE): the mean absolute difference in degrees between the estimated and the ground truth viewpoints for the entire sequence

In Table 6.2, we compare our results on each KITTI sequence and the overall mean with [127]. The notation "1st GT" indicates that the ground truth orientation of the first frame is used to initialize the methods. Our method outperforms [127] reducing the error by approximately 40% and increasing accuracy by 20 percentage points when the ground truth of the first frame is used. Furthermore, we obtain more accurate results even when the pose of the first frame is not given.
Additionally, we provide a visual example of our algorithm in Figure 6.5, where we estimate the pose of the car pointed by the red arrow. The pose estimation is

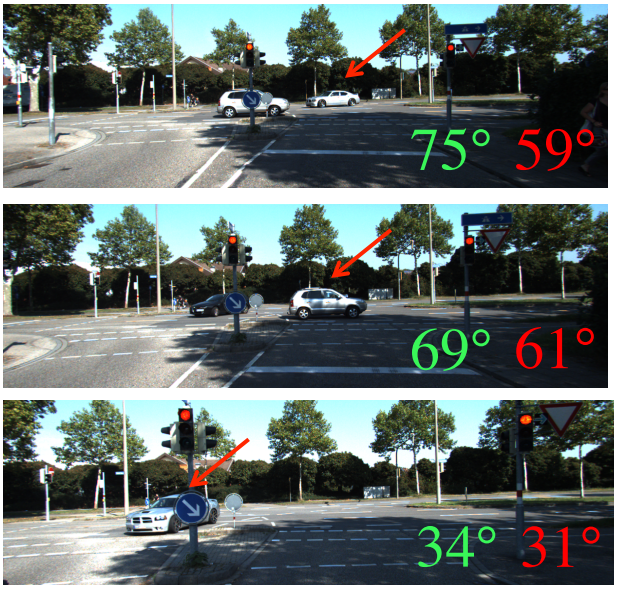


Figure 6.5: Visual results of our algorithm when estimating the pose of the car pointed by the red arrow. Ground truth in red, algorithm estimation in green. Sequence: KITTI01.

accurate even when the car is fully occluded. As a further insight into our algorithm we provide two frame-by-frame graphs from the KITTI dataset in Figure 6.6, where we plot the results of the single-frame pose estimator, the pose value selected by the video-based extension presented in this Chapter, and the ground truth. In the first graph, thanks to the Linear Programming formulation, our algorithm can completely recover from the spurious errors of the single-frame pose estimator, as a correct evidence is provided in most frames. However, when correct evidence is more counterbalanced by opposite, wrong evidence, as in the second plot around frame 23, the LP formulation leads to a conservative solution and the pose in the middle is just output.

6.3.3 YouTube dataset

We also perform experiments on the YouTube dataset [127], that contains 9 sequences where a racing car undergoes significant orientation changes. In many sequences, pictures are strongly blurred as a result of the high speed, and often the car is surrounded by smoke, due to the sliding of tires on the ground, making the pose estimation problem extremely challenging.
In Table 6.3, we report the results of our experiment on this dataset in terms of Viewpoint Accuracy and MAE. As we can see, we outperform [127] also on this

Table 6.3: YouTube dataset. Viewpoint accuracy/mean absolute error (MAE) in degrees.

	Proposed 1st GT [30]	Proposed [30]	Xiang *et al.* 1st GT [127]	Xiang *et al.* [126]	Fenzi *et al.* (2014) [28]
Race1	0.58/**16.64°**	0.54/18.47°	**0.67**/18.73°	0.52/42.62°	0.09/79.88°
Race2	**0.80/9.51°**	0.74/10.35°	0.77/10.83°	0.53/44.30°	0.36/54.26°
Race3	0.55/16.20°	0.55/16.29°	**0.83/9.28°**	0.64/46.08°	0.22/66.63°
Race4	**0.69/10.87°**	0.56/17.85°	**0.69**/15.83°	0.79/13.37°	0.20/62.47°
Race5	**0.73**/11.39°	0.55/18.01°	0.71/**10.75°**	0.54/57.79°	0.23/45.92°
Race6	**0.68/15.54°**	0.47/16.93°	0.43/18.47°	0.31/37.08°	0.43/44.90°
SUV1	**0.94/4.88°**	0.93/5.29°	0.82/7.81°	0.47/78.38°	0.14/78.65°
SUV2	**0.89/6.42°**	0.61/15.14°	0.57/19.56°	0.39/63.41°	0.44/36.07°
Sedan	0.72/12.20°	0.71/12.32°	**0.76/9.87°**	0.79/20.84°	0.40/23.05°
Mean	**0.71/12.18°**	0.63/13.86°	0.69/13.46°	0.54/47.24°	0.28/54.67°

dataset, and more interestingly, we observe that the two methods are complementary. In sequences SUV2, KITTI04, KITTI05, and KITTI06, [127] performs poorly, while our method achieves a very high accuracy in viewpoint estimation. On the contrary, in sequences KITTI11 or Race3, [127] outperforms our method.

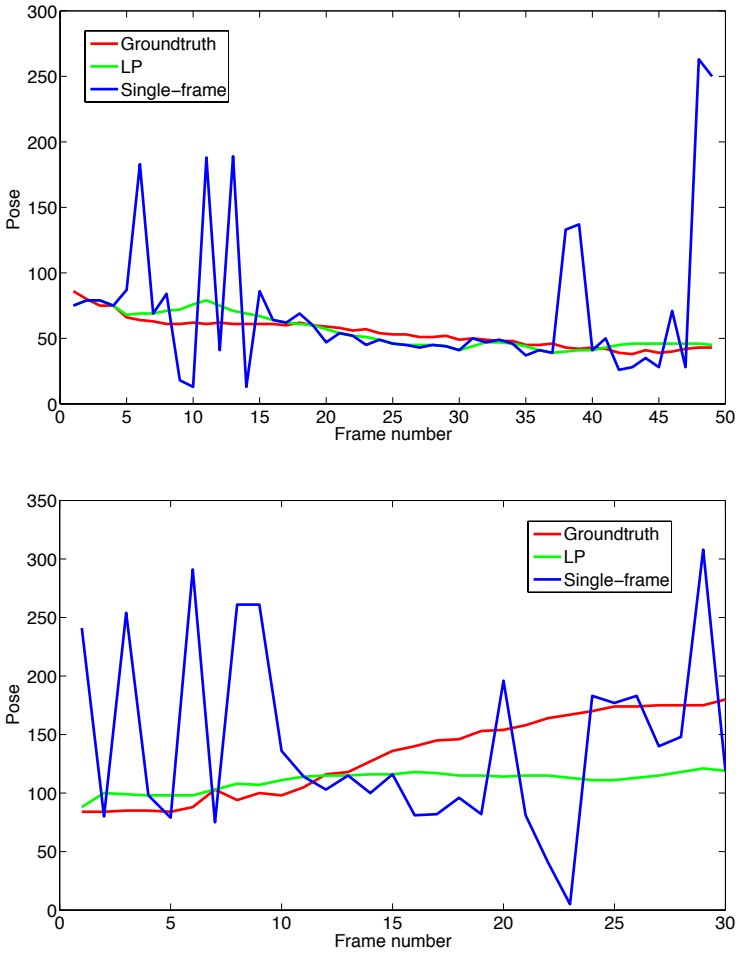


Figure 6.6: Ground truth pose (red), pose output by choosing the largest mode of the pose distribution (blue), optimal solution provided by the proposed method (green).

Chapter 7

Conclusions

In this thesis, we proposed a novel method for pose estimation of object categories, *i.e.*, for recovering the pose of an unknown object when the class membership is the only available information. This problem has recently become very popular in research as well as in the industry thanks to its application in fields such as autonomous driving and robotics. In fact, the knowledge of the orientation of other vehicles in the scene permits to make correct driving decisions, just as the knowledge of the spatial orientations of objects permits to perform precise manipulation. When the number of objects is small and all objects are known in advance, the recovery of a correct pose can be performed by single object pose estimators. When these two assumptions are not met, it becomes necessary to develop methods to solve the pose estimation problem for a generic object of a given class. However, the generality of the problem makes the task much more challenging, as difficulties such as definition and construction of a class model as well as intra-class variability appear.

Many works in the literature have tried to solve the problem by relying on 3D models based on CAD design or Structure from Motion techniques. The underlying claim behind this choice is that the knowledge of the 3D geometry of the class permits to provide a continuous estimation of the pose. However, 3D-based methods depend on the availability of 3D training data or the feasibility of 3D reconstruction, which cannot always be guaranteed. Therefore, several authors have proposed methods based only on 2D training information, counting on the ubiquitous availability of images as well as the greater simplicity of the resulting models. However, 2D-based methods have the strong disadvantage of providing only a discrete, very often coarse, value for the pose. In this thesis, we proposed a solution that bridges this gap by devising an innovative method that provides a real-valued pose by relying only on a 2D-based class model learned from image data.

We had the intuition for the method presented in this thesis by observing a simple experimental fact: while features are not invariant to out-of-plane rotations, the variation in the components of the feature descriptor is smooth as a function of

the viewpoint change. This suggests that the change in the feature descriptor is informative with respect to the viewpoint and, thus, we developed a prediction function that could capture and reproduce this informativeness. This function, which we called the *generative feature model*, is implemented as a Radial Basis Function network and trained with descriptors of the same patch extracted from different viewpoints. The generative feature model represents the key ingredient of the pose estimation method for object categories presented in this thesis.

Before building our full method, we first investigated the applicability of generative feature models for the pose estimation of a single specific object. This is an easier situation with respect to the case of object categories, as the object identity is known in advance. Given a set of training images depicting the target object under different viewpoints, we showed how to learn a set of generative feature models and how to embed it in a probabilistic framework in order to provide a posterior distribution for the pose. By experimenting on a large car dataset, we investigated different combinations of descriptors and detectors, and we noticed that the performance of our method is strongly dependent on the feature detector. The higher number of features produced by the SURF detector permits to obtain a much higher accuracy with respect to SIFT. On the contrary, the higher dimensional SIFT descriptor provides more accuracy than the SURF counterpart. With the best combination of detector and descriptor, we outperformed the state of the art in pose estimation accuracy by 45% in this experiment.

After this preliminary investigation, we turned to the actual problem addressed in this thesis: pose estimation for object categories. In order to have a representative class model, we first aggregated the generative feature models deriving from many different training instances. As the number of models becomes unmanageable, we proposed to cluster the generative feature models by means of spectral clustering. As similarity measure, we used dynamic time warping to evaluate the alignment of pairs of feature tracks in terms descriptor distance and orientation overlapping. As each cluster is a collection of generative feature models, we set its regression function, the so-called *generative cluster model*, as the weighted linear combination of the corresponding generative feature models. We evaluated the performance of our class representation on two publicly available datasets of cars and faces. Our experiments confirmed that clustering the generative feature models into generative cluster models is beneficial in terms of pose accuracy with respect to a more naïve application of our method, as the improvement in pose accuracy is approximately 15%. These experiments permitted to highlight one important weakness of our initial algorithm: it strongly depends on the correctness of the matches established between test features and generative cluster models. This often led to what is known as "flipping errors", *i.e.*, errors due to a strong similarity between different views of the object. We identified the first cause of flipping errors as the

pose classifier that we used to reduce the pose search space. It turned out that the hard decision taken by the pose classifier is often wrong, and it is detrimental because the remaining part of the algorithm cannot revert it. The second and more important cause is due to the total lack of geometry in our initial algorithm. As a matter of fact, generative cluster models are only based on feature descriptors and also the matching step does not take feature arrangement into account.

Therefore, we studied how to enrich our approach with geometrical context, as spatial information is an extremely important cue for disambiguation. More specifically, we reformulate the matching step as a graph matching problem, where the spatial arrangement between matching features is taking into account. We enforce that pairs of matching features share geometrical consistency in terms of distance and orientation in their spatial arrangement. We show that the introduction of geometrical cues in the matching step is beneficial for the accuracy of the proposed method, as the mean absolute error is reduced by approximately 25%. More interestingly, we investigated the real strength of the graph matching step with respect to the pose classifier that we initially used. It turned out that graph matching permits to totally replace the pose classifier. Furthermore, this experiment confirmed that the class representation we learned on one car dataset is general enough to work on other car datasets, thus showing that our approach is robust against domain changes.

In spite of the important results we obtained by adding graph matching to our baseline algorithm, we still experienced a not negligible number of flipping errors. As a matter of fact, the performance of our algorithm mainly alternates between long sequences of correct estimations and short bursts of wrong estimations. Nonetheless, we noticed that even in the case of wrong estimations the posterior distribution that our method deliver contains a strong minor mode located at the correct pose. Therefore, we extended our algorithm to videos in order to apply temporal constraints on the sequence of estimated poses. We assumed that the pose of an object can only have a small and smooth change over consecutive frames. On this basis, we built a graph by sampling from the posterior distributions estimated in each frame and we found the pose trajectory that best explains the pose observations by using a Linear Programming formulation. This formulation permits to remove the spurious wrong observations and to deliver a globally optimum pose trajectory. Thanks to a set of experiments on two video datasets, we showed that the embedding of spatial and temporal constraints permits to provide a much more accurate pose when video sequences are available. More specifically, we increased the pose accuracy by 40% with respect to 3D-based state-of-the-art methods on videos featuring low resolution, object truncations and a large amount of motion blur.

To conclude, we provided a novel algorithm for pose estimation of object cate-

gories that is based on the innovative concept of generative feature models. We showed how to efficiently cluster generative feature models into generative cluster models and how to estimate the pose of an unknown object of a given class in a probabilistic fashion. Furthermore, we illustrated how to introduce geometrical as well as temporal constraints in the formulation that are extremely helpful in improving the pose accuracy.

In the short term, it would be interesting to investigate if the spatial aggregation of features, *e.g.*, by means of Fisher encoding, could be beneficial to create a more robust regression-based method. Since our algorithm seems to be sensitive to matching and geometry, a feature encoding into larger, spatially fixed "containers" would remove the matching step and simultaneously permit to keep geometry into account. From a larger perspective, the work we presented in this thesis is not conclusive, but it can lead to a set of possible extensions. By large, the most important direction that can be pursued is to combine pose estimation and object detection. Whereas here we focused only on pose estimation by assuming that the location of the object is given, it would be interesting to extend our method so that the estimation of the pose becomes informative about the location of the object and vice versa.

Bibliography

[1] URL `https://3dwarehouse.sketchup.com/`.

[2] O. Aghazadeh, J. Sullivan, and S. Carlsson. Novelty Detection from an Ego-Centric Perspective. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3297–3304, 2011.

[3] G.J. Agin and T.O. Binford. Computer Description of Curved Objects. *IEEE Transactions on Computers*, 25(4):439–449, 1976.

[4] S.Y. Bao, Y. Xiang, and S. Savarese. Object Co-detection. In *IEEE European Conference on Computer Vision (ECCV)*, pages 86–101, 2012.

[5] H. Barrow and R. Popplestone. Relational Descriptions in Picture Processing. *Machine Intelligence*, 6:377–396, 1971.

[6] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *IEEE European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[8] A. Berg, T. Berg, and J. Malik. Shape Matching and Object Recognition using Low Distortion Correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26–33, 2005.

[9] D. S. Broomhead and D. Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321–355, 1988.

[10] T. Caelli and S. Kosinov. An Eigenspace Projection Clustering Method for Inexact Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(4):515–519, 2004.

[11] T. S. Caetano, T. Caelli, and D. A. C. Barone. Graphical Models for Graph Matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 466–473, 2004.

[12] M. Carcassoni and E. R. Hancock. Spectral Correspondence for Point Pattern Matching. *Pattern Recognition (PR)*, 36(1):193–204, 2003.

[13] W. J. Christmas, J. Kittler, and M. Petrou. Structural Matching in Computer Vision Using Probabilistic Relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 17(8):749–764, 1995.

[14] A. Collet, D. Berenson, S.S. Srinivasa, and D. Ferguson. Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 48–55, 2009.

[15] T. Cour, P. Srinivasan, and J. Shi. Balanced Graph Matching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 313–320, 2006.

[16] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. In *IEEE European Conference on Computer Vision Workshops (ECCVW)*, pages 1–22, 2004.

[17] F.C. Kirwan D. Mumford, J. Fogarty. *Geometric Invariant Theory*. Springer Science & Business Media, 1994.

[18] G.B. Dantzig. Origins of the Simplex Method. Technical report, Department of Operations Research, Stanford University, 1987.

[19] G.B. Dantzig and M.N. Thapa. *Linear Programming 1: Introduction*. Springer-Verlag, Secaucus, NJ, USA, 1997.

[20] G.B. Dantzig and M.N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer-Verlag, Secaucus, NJ, USA, 2003.

[21] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A Tensor-Based Algorithm for High-Order Graph Matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1980–1987, 2009.

[22] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A Tensor-Based Algorithm for High-Order Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(12):2383–2395, 2011.

[23] D. Eggert, L. Stark, and K. Bowyer. Aspect Graphs and Their Use in Object Recognition. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 13 (3-4):347–375, 1995.

[24] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC 2006) Results. Technical report, University of Oxford, 2006.

[25] L. Fei-Fei and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 524–531, 2005.

[26] P. Felzenszwalb, D. McAllester, and D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[27] P. Felzenszwalb, R. Girschick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(9):1627–1645, 2010.

[28] M. Fenzi and J. Ostermann. Embedding Geometry in Generative Models for Pose Estimation of Object Categories. In *British Machine Vision Conference (BMVC)*, 2014.

[29] M. Fenzi, L. Leal-Taixé, B. Rosenhahn, and J. Ostermann. Class Generative Models based on Feature Regression for Pose Estimation of Object Categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 755–762, 2013.

[30] M. Fenzi, L. Leal-Taixé, K. Schindler, and J. Ostermann. Pose Estimation of Object Categories in Videos Using Linear Programming. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 821–828, 2015.

[31] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[32] M. A. Fischler and R. A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973.

[33] Y. Freund and R.E.Schapire. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences (JCSS)*, 55(1):119–139, 1997.

[34] J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics (AS)*, 38(2):337–407, 2001.

[35] J. Gall and V. Lempitsky. Class-Specific Hough Forests for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1022–1029, 2010.

[36] J. Gall, N. Razavi, and L. Van Gool. An Introduction to Random Forests for Multi-class Object Detection. In F. Dellaert, J.-M. Frahm, M. Pollefeys, L. Leal-Taixé, and B. Rosenhahn, editors, *Theoretic Foundations of Computer Vision: Outdoor and Large-Scale Real-World Scene Analysis*, pages 243–263. Springer, 2012.

[37] A. Geiger, P. Lenz, and R. Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[38] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3D Traffic Scene Understanding from Movable Platforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(5):1012–1025, 2014.

[39] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-Aware Object Detection and Pose Estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1275–1282, 2011.

[40] I. Gordon and D.G. Lowe. Scene Modelling, Recognition and Tracking with Invariant Image Features. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 110–119, 2004.

[41] I. Gordon and D.G. Lowe. What and Where: 3D Object Recognition with Accurate Pose. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, pages 67–82. Springer-Verlag, 2006.

[42] N. Gourier, D. Hall, and J. L. Crowley. Estimating Face Orientation from Robust Detection of Salient Facial Features. *Proceedings of Pointing 2004, International Conference on Pattern Recognition (ICPR), International Workshop on Visual Observation of Deictic Gestures*, pages 17–25, 2004.

[43] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1458–1465, 2005.

[44] C. Gu and X. Ren. Discriminative Mixture-of-Templates for Viewpoint Classification. In *IEEE European Conference on Computer Vision (ECCV)*, pages 408–421, 2010.

[45] Inc. Gurobi Optimization. Gurobi Optimizer Reference Manual, 2015. URL `http://www.gurobi.com`.

[46] M. Al Haj and L.S. Davis J. Gonzàlez. On Partial Least Squares in Head Pose Estimation: How to Simultaneously Deal with Misalignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2602–2609, 2012.

[47] K. Hara and R. Chellappa. Growing Regression Forests by Classification: Application to Object Pose Estimation. In *IEEE European Conference on Computer Vision (ECCV)*, pages 552–567, 2014.

[48] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 23.1–23.6, 1988.

[49] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[50] K. He, L. Sigal, and S. Sclaroff. Parameterizing Object Detectors in the Continuous Pose Space. In *IEEE European Conference on Computer Vision (ECCV)*, pages 450–465, 2014.

[51] S. Hinterstoisser, S. Benhimane, and N. Navab. N3M: Natural 3D Markers for Real-Time Object Detection and Pose Estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–7, 2007.

[52] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Simultaneous Recognition and Homography Extraction of Local Patches with a Simple Linear Classifier. In *British Machine Vision Conference (BMVC)*, pages 10.1–10.10, 2008.

[53] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Online Learning of Patch Perspective Rectification for Efficient Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[54] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua, and V. Lepetit. Real-Time Learning of Accurate Patch Rectification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2945–2952, 2009.

[55] E. Hsiao, A. Collet, and M. Hebert. Making Specific Features Less Discriminative to Improve Point-Based 3D Object Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2653–2660, 2010.

[56] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From Structure-from-Motion Point Clouds to Fast Location Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2599–2606, 2009.

[57] H. Ishikawa. Exact Optimization for Markov Random Fields with Convex Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(10):1333–1336, 2003.

[58] K. Kim, V. Lepetit, and W. Woo. Scalable Real-Time Planar Targets Tracking for Digilog Book. In *Computer Graphics International (CGI)*, pages 1145–1154, 2010.

[59] K. Kim, V. Lepetit, and W. Woo. Keyframe-based Modeling and Tracking of Multiple 3D Objects. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 193–198, 2010.

[60] J.J. Koenderink. The Structure of Images. *Biological Cybernetics (BC)*, 50 (5):363–370, 1984.

[61] J.J. Koenderink and A.J. van Doorn. Representation of Local Geometry in the Visual System. *Biological Cybernetics*, 55(6):367–375, 1987.

[62] K. Köser and R. Koch. Perspectively Invariant Normal Features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[63] L. Leal-Taixé. *Multiple Object Tracking with Context Awareness*. PhD thesis, Leibniz Universität Hannover, 2014.

[64] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn. Branch-and-Price Global Optimization for Multi-view Multi-target Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1987–1994, 2012.

[65] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese. Learning an Image-Based Motion Context for Multiple People Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3542–3549, 2014.

[66] B. Leibe, A. Leonardis, and B. Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. In *IEEE European Conference on Computer Vision Workshops (ECCVW)*, pages 17–32, 2004.

[67] M. Leordeanu and M. Hebert. A Spectral Technique for Correspondence ProblemsUsing Pairwise Constraints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1482–1489, 2005.

[68] V. Lepetit, J. Pilet, and P. Fua. Point Matching as a Classification Problem for Fast and Robust Object Pose Estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–250, 2004.

[69] V. Lepetit, P. Lagger, and P. Fua. Randomized Trees for Real-Time Keypoint Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 775–781, 2005.

[70] S. Z. Li. A Markov Random Field Model for Object Matching Under Contextual Constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 866–869, 1994.

[71] Y. Li, N. Snavely, and D. Huttenlocher. Location Recognition using Prioritized Feature Matching. In *IEEE European Conference on Computer Vision (ECCV)*, pages 791–804, 2010.

[72] J. Liebelt and C. Schmid. Multi-View Object Class Detection with a 3D Geometric Model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1688–1695, 2010.

[73] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-Independent Object Class Detection using 3D Feature Maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[74] T. Lindeberg. Scale-space Theory: A Basic Tool for Analysing Structures at Different Scales. *Journal of Applied Statistics (JAS)*, 21(2):224–270, 1994.

[75] T. Lindeberg. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision (IJCV)*, 30(2):79–116, 1998.

[76] R. J. López-Sastre, T. Tuytelaars, and S. Savarese. Deformable Part Models Revisited: A Performance Evaluation for Object Category Pose Estimation. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1052–1059, 2011.

[77] D.G. Lowe. Local Feature View Clustering for 3D Object Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 682–688, 2001.

[78] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[79] J. Maciel and J. Costeira. A Global Solution to Sparse Correspondence Problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(2):187–199, 2003.

[80] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982.

[81] L. Mei, M. Sun, K. Carter, A. Hero, and S. Savarese. Unsupervised Object Pose Classification from Short Video Sequences. In *British Machine Vision Conference (BMVC)*, pages 1–12, 2009.

[82] L. Mei, J. Liu, A.O. Hero, and S. Savarese. Robust Object Pose Estimation via Statistical Manifold Modeling. In *IEEE International Conference on Computer Vision (ICCV)*, pages 967–974, 2011.

[83] C. A. Micchelli. Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions. *Constructive Approximation*, 2 (1):11–22, 1986.

[84] K. Mikolajczyk. *Detection of Local Features Invariant to Affine Transformation*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[85] E. Molla and V. Lepetit. Augmented Reality for Board Games. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 253–254, 2010.

[86] H. Moravec. Rover Visual Obstacle Avoidance. In *International Joint Conference on Artificial Intelligence*, pages 785–790, 1981.

[87] M. Müller. *Information Retrieval for Music and Motion*. Springer-Verlag New York, 2007.

[88] K.G. Murty. *Linear Programming*. John Wiley & Sons, New York, NY, USA, 1983.

[89] A. Y. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 849–856, 2001.

[90] M. Özuysal, V. Lepetit, and P. Fua. Pose Estimation for Category Specific Multiview Object Localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 778–785, 2009.

[91] Y. Panagakis, M. A. Nicolaou, S. Zafeiriou, and M. Pantic. Robust Canonical Time Warping for the Alignment of Grossly Corrupted Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 540–547, 2013.

[92] Y. Park, V. Lepetit, and W. Woo. Multiple 3D Object Tracking for Augmented Reality. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 117–120, 2008.

[93] H. Pirsiavash and D. Ramanan. Detecting Activities of Daily Living in First-person Camera Views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2847–2854, 2012.

[94] T. Poggio and F. Girosi. Networks for Approximation and Learning. *IEEE Proceedings*, 78(9):1481–1497, 1990.

[95] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning Object Class Detectors from Weakly Annotated Video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3282–3289, 2012.

[96] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series, 1993.

[97] T. M. Rath and R. Manmatha. Word Image Matching Using Dynamic Time Warping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 521–527, 2003.

[98] N. Razavi, J. Gall, and L. Van Gool. Backprojection Revisited: Scalable Multi-view Object Detection and Similarity Metrics for Detections. In *IEEE European Conference on Computer Vision (ECCV)*, pages 620–633, 2010.

[99] C. Redondo-Cabrera, R. L'opez-Sastre, and T. Tuytelaars. All Together Now: Simultaneous Object Detection and Continuous Pose Estimation using a Hough Forest with Probabilistic Locally Enhanced Voting. In *British Machine Vision Conference (BMVC)*, 2014.

[100] L.G. Roberts. *Machine Perception of Three-dimensional Solids*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering, 1963.

[101] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D Object Modeling and Recognition Using Affine-Invariant Patches and Multi-View Spatial Constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 272–280, 2003.

[102] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. *International Journal of Computer Vision (IJCV)*, 66(3):231–259, 2006.

[103] H. Sakoe and S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[104] S. Satkin, J. Lin, and M. Hebert. Data-Driven Scene Understanding from 3D Models. In *British Machine Vision Conference (BMVC)*, pages 1–11, 2012.

[105] T. Sattler, B. Leibe, and L. Kobbelt. Fast Image-based Localization using Direct 2D-to-3D Matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 667–674, 2011.

[106] S. Savarese and L. Fei-Fei. 3D Generic Object Categorization, Localization and Pose Estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[107] S. Savarese and L. Fei-Fei. View Synthesis for Recognizing Unseen Poses of Object Classes. In *IEEE European Conference on Computer Vision (ECCV)*, pages 602–615, 2008.

[108] S. Savarese and L. Fei-Fei. Multi-view Object Categorization and Pose Estimation. In R. Cipolla, S. Battiato, and G.M. Farinella, editors, *Computer Vision*, volume 285 of *Studies in Computational Intelligence*, pages 205–231. Springer Berlin Heidelberg, 2010.

[109] C. Schmid and R. Mohr. Local Greyvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(5):530–535, May 1997.

[110] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, New York, NY, USA, 1998.

[111] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On Aligning Curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25 (1):116–125, 2003.

[112] T. Serre, L. Wolf, and T. Poggio. Object Recognition with Features Inspired by Visual Cortex. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 994–1000, 2005.

[113] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22 (8):888–905, 2000.

[114] G. Sierksma. *Linear and Integer Programming: Theory and Practice, Second Edition.* Mercel Dekker, Inc., 2001.

[115] Robert Sim and Gregory Dudek. Learning Generative Models of Scene Features. *International Journal of Computer Vision (IJCV)*, 60(1):45–61, 2004.

[116] M. Torki and A. Elgammal. Putting Local Features on a Manifold. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1743–1750, 2010.

[117] M. Torki and A. Elgammal. One-Shot Multi-Set Non-Rigid Feature-Spatial Matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3058–3065, 2010.

[118] M. Torki and A. Elgammal. Regression from Local Features for Viewpoint and Pose Estimation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2603–2610, 2011.

[119] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing Visual Features for Multiclass and Multiview Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(5):854–869, 2007.

[120] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *International Conference on Machine Learning (ICML)*, pages 104–112, 2004.

[121] S.A. Underwood and C.L. Coates. Visual Learning from Multiple Views. *IEEE Transactions on Computers*, 24(6):651–661, 1975.

[122] L. Vacchetti, V. Lepetit, and P. Fua. Fusing Online and Offline Information for Stable 3D Tracking in Real-Time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 241–248, 2003.

[123] L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking using Online and Offline Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(10):1385–1391, 2004.

[124] P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 511–518, 2001.

[125] H. Wang and E. R. Hancock. A Kernel View of Spectral Point Pattern Matching. In *Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition (SSPR)*, pages 361–369, 2004.

[126] Y. Xiang and S. Savarese. Estimating the Aspect Layout of Object Categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3410–3417, 2012.

[127] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese. Monocular Multiview Object Tracking with 3D Aspect Parts. In *IEEE European Conference on Computer Vision (ECCV)*, pages 220–235, 2014.

[128] J. Xiao, J. Chen, D.-Y. Yeung, and L. Quan. Structuring Visual Words in 3D for Arbitrary-View Object Localization. In *IEEE European Conference on Computer Vision (ECCV)*, pages 725–737, 2010.

[129] R. Zass and A. Shashua. Probabilistic Graph and Hypergraph Matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[130] F. Zhou and F. De la Torre. Canonical Time Warping for Alignment of Human Behavior. In *Advances in Neural Information Processing Systems Conference (NIPS)*, pages 2286–2294, 2009.

[131] F. Zhou and F. De la Torre. Generalized Time Warping for Multi-modal Alignment of Human Motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1282–1289, 2012.

[132] A. Zisserman, D. Forsyth, J. Mundy, C. Rothwell, J. Liu, and N. Pillow. 3D Object Recognition using Invariance. Technical report, Oxford University, 1994.

Michele Fenzi

Ph.D. in Computer Vision

Institut für Informationsverarbeitung
Leibniz Universität Hannover
Appelstr. 9A, 30167, Hannover, Germany
http://www.tnt.uni-hannover.de/staff/fenzi/
fenzi@tnt.uni-hannover.de

Education

Sep. 2010 - Nov. 2015	Ph.D. in Computer Vision, Leibniz Universität Hannover, Hannover, Germany
Advisor	Prof. Dr.-Ing. Jörn Ostermann
Topics	Pose estimation, object recognition, object classification, 3D reconstruction, object tracking
Apr. - Oct. 2009	Master's Thesis, SIT Fraunhofer Institute, Darmstadt, Germany
Advisor	Dr.-Ing. Martin Steinbach
Topics	Digital watermarking
Sep. 2001 - Dec. 2009	B.S. and M.Sc. in Telecommunications Engineering, University of Florence, Florence, Italy
Selected courses	Communication Networks, Mobile Communications, Optical Communications, Signal Processing, Image Processing

Work Experience

Feb. 2015 - Apr. 2015	Visiting Researcher at ESAT/VISICS lab, K.U. Leuven, Belgium
Advisor	Prof. Tinne Tuytelaars
Sep. 2010 - May 2015	Research Assistant at Institut für Informationsverarbeitung, Leibniz Universität Hannover

- Pose Estimation for Object Categories and Single Objects, Object Detection and Classification, Object Tracking, 3D Reconstruction
- Developer and Project coordinator for BMBF-funded project ASEV
  The ASEV project involved the development of a fully automatic video surveillance system for airport aprons. The system has been succesfully deployed for 3 months at the Braunschweig-Wolfsburg airport.
- Teaching Assistant for Source Coding
  Contents: Information theory, lossless coding, lossy coding, audio/video coding standards
- Teaching Assistant for Laboratory for Information Processing
  Contents: Implementation and application of lossless and lossy coding methods

Jan. 2007 - May 2007	Software developer at Sicuring srl, Florence, Italy
Description	E-learning applications for safety management at work

Scientific Profile

Author	(Co-)author of 9 peer-reviewed publications
Reviewer	Reviewer for ICCV, ECCV, CVPR, DAGM, ICIP, IMAVIS, IET-CV
Achievements	Selected for the Doctoral Consortium at ICCV 2015
	Best Paper Award for "Markov Random Fields Pre-Warping to Prevent Collusion in Image Transaction Watermarking", SPPRA 2010

Technical Knowledge & Skills

Programming Languages	C, C++, Matlab
Tools	OpenCV, Qt, CMake, Git, LaTeX, HTML, VBA, SQL, MS Office, Libreoffice
Operating Systems	Linux, Windows
Languages	Italian (native), English (professional), German (intermediate)

Publications

Book Chapters

R. Dragon, **M. Fenzi**, W. Siberski, B. Rosenhahn, J. Ostermann
Towards Feature-Based Situation Assessment for Airport Apron Video Surveillance
Theoretic Foundations of Computer Vision: Outdoor and Large-Scale Real-World Scene Analysis, 2012
edited by F. Dellaert, J.-M. Frahm, M. Pollefeys, L. Leal-Taixé, B. Rosenhahn

Conference Contributions

M. Fenzi, L. Leal-Taixé, J. Ostermann, T. Tuytelaars
Continuous Pose Estimation with a Spatial Ensemble of Fisher Regressors
IEEE International Conference on Computer Vision (ICCV) 2015, Santiago, Chile

M. Fenzi, L. Leal-Taixé, K. Schindler, J. Ostermann
Pose Estimation of Object Categories in Videos Using Linear Programming
IEEE Winter Conference on Applications of Computer Vision (WACV) 2015, HI, USA

M. Fenzi, J. Ostermann
Embedding Geometry in Generative Models for Pose Estimation of Object Categories
British Machine Vision Conference (BMVC) 2014, United Kingdom, **oral**

M. Fenzi, N. Mentzer, G. Payà-Vayà, H. Blume, T.N. Nguyen, T. Risse, J. Ostermann
ASEV - Automatic Situation Assessment for Event-driven Video Analysis
IEEE Conference on Advanced Video and Signal-Based Surveillance (AVSS) 2014, South Korea, **oral**

L. Leal-Taixé, **M. Fenzi**, A. Kuznetsova, B. Rosenhahn, S. Savarese
Learning an image-based motion context for multiple people tracking
IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2014, OH, USA

M. Fenzi, L. Leal-Taixé, B. Rosenhahn, J. Ostermann
Class Generative Models based on Feature Regression for Pose Estimation of Object Categories
IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2013, OR, USA

M. Fenzi, R. Dragon, L. Leal-Taixé, B. Rosenhahn, J. Ostermann
3D Object Recognition and Pose Estimation for Multiple Objects using Multi-Prioritized RANSAC and Model Updating
Annual Symposium of the German Association for Pattern Recognition (DAGM) 2012, Austria, **oral**

M. Fenzi, H. Liu, M. Steinbach, R. Caldelli
Markov Random Fields Pre-Warping to Prevent Collusion in Image Transaction Watermarking
International Conference on Signal Processing, Pattern Recognition and Applications (SPPRA) 2010, Austria
Best Paper Award