TAMPERE UNIVERSITY OF TECHNOLOGY

**Fatemeh Shokrollahi Yancheshmeh**

# UNSUPERVISED ALIGNMENT OF OBJECTS IN IMAGES

Master's Thesis

Examiners: Professor Joni-Kristian Kämäräinen
Professor Mikko Valkama

Examiners and topic approved in the Computing and Electrical Engineering Faculty Council meeting on 4 September 2013

# ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Program in Information Technology
**FATEMEH SHOKROLLAHI YANCHESHMEH : Unsupervised alignment of objects in images**
Master of Science Thesis, 105 pages
December 2013
Major: Communications engineering
Examiners: Professor Joni-kristian Kämäräinen
Professor Mikko Valkama
Keywords: image alignment, image congealing, object detection

With the advent of computer vision, various applications become interested to apply it to interpret the 3D and 2D scenes. The main core of computer vision is visual object detection which deals with detecting and representing objects in the image. Visual object detection requires to learn a model of each class type (e.g. car, cat) to be capable to detect objects belonging to the same class. Class learning benefits from a method which automatically aligns class examples making learning more straightforward.
The objective of this thesis is to further develop the sate-of-the-art feature-based alignment method which rigidly and automatically aligns object class images to a manually selected seed image. We try to compensate the weakness by providing a method to automatically select the best seed from dataset. Our method first extracts features by utilizing dense sampling method and then scale invariant feature transform (SIFT) descriptor is used to find best matches as initial local feature matches. The final alignment is based on spatial scoring procedure where the initial matches are refined to a set of spatially verified matches. The spatial score is used next to calculate similarity scores. We propose an algorithm which operates on spatial and similarity scores and finally selects the best seed. We also investigate the performance of step-wise alignment using minimum spanning tree (MST) and Dijkstra shortest path instead of direct alignment utilizing a single seed. We conduct our experiments using classes of Caltech-101 for which our unsupervised seed selection and step-wise alignment achieve state-of-the-art performance.

# PREFACE

This thesis work owes its progress to all those who without their unfailing support I would not have been able to juggle everything to completion.

First of all, I wish to express my sincere thanks and gratitude to my supervisor Professor Joni-kristian Kämäräinen for his guidance, constant support, endless patience, and objective overview and insightful comments regarding the content of the thesis. It has been a real pleasure and privilege to work with you.

I am specially grateful to Professor Mikko Valkama who gave me the honor of being my second thesis examiner, and for his wise and invaluable advises to improve the thesis content.

My ongoing studies would not have been possible without the funds provided first by Signal Processing department of Tampere University of technology (TUT), and second by My thesis supervisor, Professor Joni-kiristian Kämäräine, and I extend my deepest thanks to them for having the courage to support.

I also wish to thank my dear friends: Pooya Saketi for generously sharing his time and expertise with me to learn LaTex quickly, and to Jukka Lankinen for being helpful, available, and informative whenever I had a question.

I would like to declare my sincere thanks to my dear family for their enduring and supporting both emotionally and financially during this study. To my mum, Zahra who has always sacrificed her wishes to make my dreams come true, and accompanies me with her endless love. To my dad, Mohammad who has tirelessly worked hard to provide a perfect life for his family. He encouraged me to achieve my goal and always tried to show the right way of living, though sometimes being wrong. And to my brother, Ali who has been a very kind, reliable and supportive friend for me in my entire life. I would like to give love and spatial thanks to all of you for keeping my enthusiasm alive when I was drifting.

I would also of course like to express my appreciation to my loving boyfriend, Milad Mostofizadeh for his unwavering help, understanding, listening, patience and changing the atmosphere when the time was tough. Thank and love to you for being there for me, you certainly provided calm in the storm!

Lastly, I would like to thank and acknowledge God as a continuing source of strength and

guidance.

Tampere, December 8th, 2013

*Fatemeh Shokrollahi Yancheshmeh*

# CONTENTS

# ABBREVIATIONS AND SYMBOLS

| | |
|---|---|
| $\theta$ | Angle of rotation |
| $\sigma$ | Sharpness of RBF |
| $\alpha$ | Eigen value |
| $\beta$ | Eigen value |
| $2D$ | Two dimensional |
| $3D$ | Three dimensional |
| $a$ | Arbitrary value |
| $b$ | Arbitrary value |
| $c$ | Arbitrary value |
| $D$ | Distance matrix |
| $E$ | Change |
| $F$ | Transform function |
| $F_s$ | Feature extracted from seed image |
| $F_i$ | Feature extracted from image |
| $G$ | Gaussian kernel(at different scales) |
| $H$ | Entropy estimation |
| $h_x$ | x-shear |
| $h_y$ | y-shear |
| $I$ | Image intensity |
| $I$ | Total number of images |
| $I_i$ | Image from dataset |
| $I_s$ | Seed image |
| $I\left(u,v\right)$ | Intensity of the position with $(u,v)$ coordinates |
| $I_x$ | Intensity in direction of $x$ |
| $I_y$ | Intensity in direction of $y$ |
| $I\left(x,y\right)$ | Image intensity |
| $K$ | Degree of connectivity |
| $K$ | Total number of nearest neighbors |

| | |
|---|---|
| $K$ | Number of best local feature matches |
| $L$ | Scale space |
| $L$ | Best seed's landmarks |
| $M$ | Number if image features |
| $M$ | $2 \times 2$ symmetric matrix |
| $M$ | Rotation matrix |
| $M_i$ | Total number of local features |
| $N$ | the number of seed features |
| $N$ | Number of matched local features |
| $N$ | Total number of images |
| $N_0$ | is the number of $0$ (black) |
| $N_1$ | is the number of $1$ (white) |
| $\vec{P}$ | 2D point |
| $P$ | 2D point |
| $P$ | Number of pixels |
| $P_x$ | Value in the coordinate of $x$ |
| $P_y$ | Value in the coordinate of $y$ |
| $Q$ | 2D point |
| $R$ | Corner region |
| $R$ | Edge point |
| $R$ | Flat region |
| $s_L$ | Score of $L$th seed's best landmark |
| $S_x$ | Scaling factor in the coordinate of $x$ |
| $S_y$ | Scaling factor in the coordinate of $y$ |
| $S_{i,j}$ | Similarity score between seed $i$ and image $j$ |
| $T$ | 2D vector |
| $Tr$ | Trace |

| | |
|---|---|
| $t_x$ | translation in dirction of $x$ |
| $t_y$ | translation in dirction of $y$ |
| $V$ | 2D vector |
| $W$ | Image window |
| $x_i^j$ | Value of $i$th pixel in the $j$th image |
| $x_i^{j'}$ | Pixel value for a transformed image |
| $(x, y)$ | Amount of Shift |
| *BOW* | Bag of words |
| *Det* | Determinant |
| *DPM* | discriminativly trained deformable part-based models) |
| *DLT* | Direct linear transform |
| *EM* | Expectation-maximization |
| *img* | An image from a data set |
| *LBP* | Local binary patterns |
| *MOPS* | Multiscale oriented patches |
| *MST* | Minimum spanning tree |
| *PCA* | Principal components analysis |
| *POEM* | patterns of oriented edge magnitudes |
| *RBF* | Radial basis function |
| *SVM* | Support vector machine |
| *SIFT* | Scale invariant feature transform |
| *simScore* | Similarity matrix |
| *sptScore* | Spatial score matrix |
| *STD* | Standard deviation |

# 1 INTRODUCTION

In recent years, computer vision has become one of the essential part of many applications such as Kinect (motion sensing input devise), digital cameras, Automatic number plate recognition. The aim of computer vision is to make a device capable of analyzing objects (e.g. buildings, tree) around us to generate meaningful and explicit interpretation of contents. An inevitable part of computer vision is visual object detection (see Fig. 1 which handles the techniques of detecting and representing objects included in the image, and has been under very active investigation lately. To detect an object, the visual object detection requires to learn a general model of the class that can be applied for recognizing all the objects having the same class as the model. Learning a class model is quite challenging due to having huge number of class types in the real world, while each class itself consists of objects which vary in shape, poses, locations, and it is not possible to compare all of them together. The most influential approach to achieve the goal of model learning has been visual bag-of-words (BOW) [1, 2] where image content is encoded into a histogram of local feature codes, but the performance of such part-based method drops down when the objects poses vary (i.e. changes in rotation, scaling, and translation) due to ignoring spatial arrangement (constellation) of the local features.

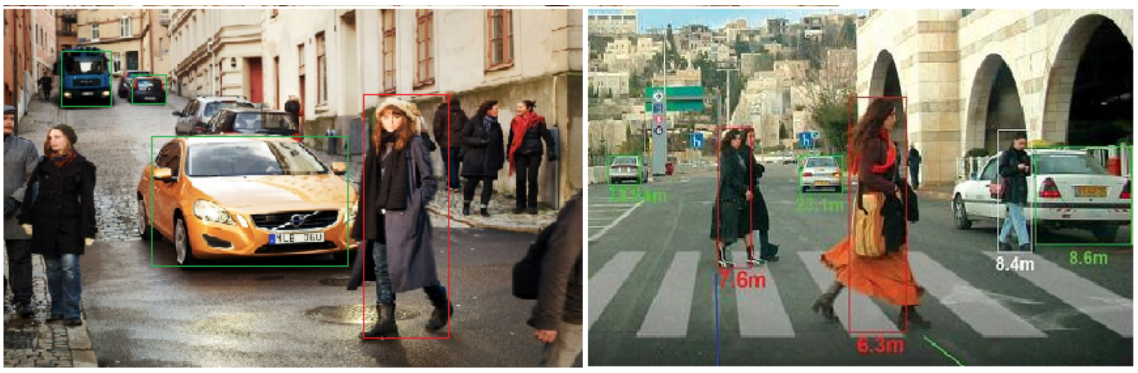


**Figure 1.** Example of visual object detection usage in pedestrian detection and car detection

Efficient learning of the computational model would benefit from automatic alignment of training images that has been recognized first by Miller's congealing method [3]. Congealing is a jointly alignment of set of misaligned images to make them as similar as possible according to some similarity measurement. The term of alignment refers to any kind of allowable transformation such as color transformation, geometry transformation, etc. The main drawback of congealing method is that it requires initial good alignment to converge. The congealing method was later extended and renamed "alignment", and extended by other works [4, 5, 1, 6], but these methods are supervised and rely on manually annotated landmarks, and cannot address the problem of recent dataset (include images with sever geometric variation) where even manually annotated bounding boxes do not guarantee a good alignment. A simple example of image alignment is given in Fig. 2.

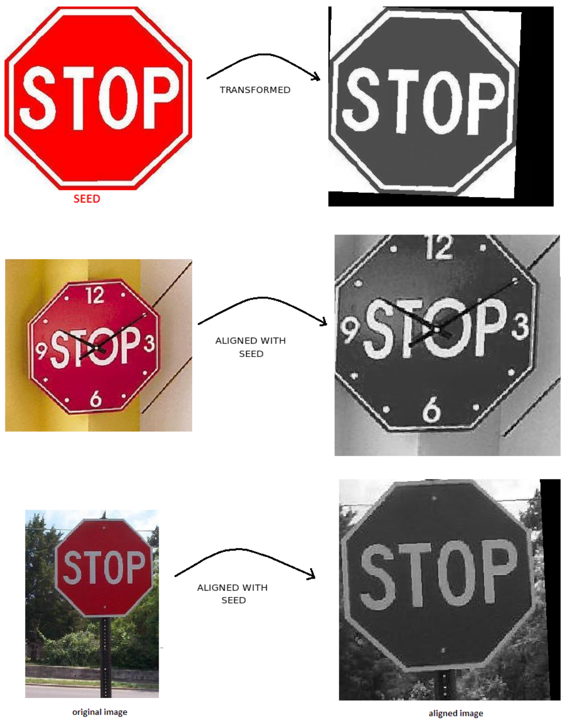


**Figure 2.** Example of aligning 3 images from the class of stop sign to the given seed. The images become more similar together after alignment.

This thesis introduces an improved version of the latest alignment method called feature-based unsupervised alignment [55]. Feature-based alignment method aligns all images from data set to a manually selected seed. The method defined an algorithm to find the best landmarks of the seed. The algorithm extracts local features from all images using dense sampling [5] and SIFT (scale invariant feature transform) descriptor [7], then finds the best matches of seed features and image features (Landmarks) according to randomized spatial scoring procedure used in [8]. The disadvantage of this method is selecting the seed manually which means that someone should browse all images and make a decision of the best seed.

The automatic seed selection is the first contribution of this thesis for which the same process as the one used in feature-based alignment method for creating spatial scores for all pairs of images is implemented. Next, the spatial scores are used to calculate similarity scores used in [8] among all pairs of images. We have defined an algorithm which selects the best seed from the training images based on the spatial and similarity scores.

The second contribution to the original alignment method is that perhaps alignment could be improved by not using only a single global seed and direct transfer, but aligning images through similarity links step by step making alignment more accurate instead of a single long path. The idea is triggered by the work of [9] in which using a tree structure performed very well for object recognition. We investigated step-wise alignment by implementing first a full connected graph of images where similarity scores specifies the weight of the edges. The graph is then given to two different algorithms: minimum spanning tree (MST) [69], and Dijkstra shortest path [69, 70]. MST algorithm constructs a tree structure with the lowest total cost from the graph, and sets the seed as the root of the tree, and the Dijkstra algorithm finds the shortest path from a given source (training image) to the seed. The alignment procedure is done by computing similarity transformation of each visited node along the path to the seed.

We verified our methods using a standard data set, Caltech-101, for its 10 different classes.The optimal seed selection algorithm is explicitly defined and its performance is measured and reported in both qualitative (average images with and without aligning) and quantitative (alignment errors of manually annotated landmarks) performances. The results of unsupervised seed selection are compared to the original method.The result of MST, Dijkstra, and the original alignment with a single seed are evaluated and discussed. In our experiments we show our method of automatically seed selection is effective for most of the classes, and in some cases is even better than the supervised seed selection. The results of step-wise alignment indicated that the alignment performance is significantly improved specifically by utilizing Dijkstra shortest path method.

# 2 OBJECT CLASS DETECTION AND IMAGE ALIGNMENT

This thesis is in the field of computer vision which is a science of simulating human vision, i.e. creating methods to give a machine the ability of how to interpret, process, analyze, and reconstruct a 3D (three dimensional) scene like human's perception of the environment. For instance, by looking at an image including several people, You can easily distinguish each of the people from the background, and tell the shape of their face, skin colors, and even assume their sensation from the facial expression. To achieve the goal, computer vision requires to apply a combination of several fields of knowledge including : computer science, electrical engineering, mathematics, biology, physiology and cognitive science. Therefore, the aim of computer vision is first in analyzing objects in images to generate meaningful and explicit interpretation of contents [10, 11, 12]. Computer vision plays an important role in many applications, such as:detecting unexpected obstacles such as pedestrians on the street, Optical character recognition for reading handwritten postal codes on letters and Fingerprint recognition [13]. Obviously, object detection technology is at the core of many computer vision applications which deals with detecting instances of semantic objects of a certain class (such as humans, buildings, etc) in digital images and videos, and has been studied in many works to improve the performance in recognizing the objects in pictures. That is still a big challenge because of having immense types of object classes in the real world, while each class itself includes objects with extreme variations in shape, appearance, and positions (e.g. see Fig. 3 ). Main issue need to be tackled in recognizing object categories is how to represent and detect objects invariant to distortions, appearance, and geometry variation.

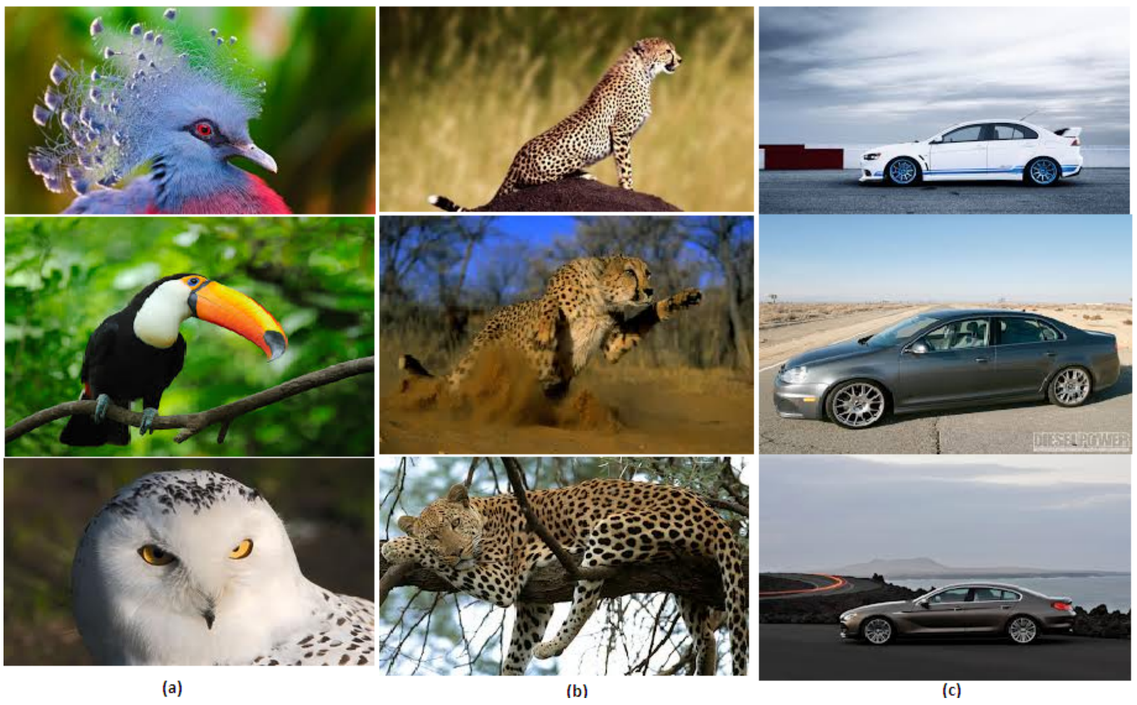


**Figure 3.** Example of object class variation for categorization.(a): appearance variation in class of bird. (b): poses variation in class of spotted cat. (c): scale variation in class of car side [14]

## 2.1 Object class detection

If the class type of an object that we are looking for in the image is known, then it is possible to find the object by rapidly scanning the image to find a match using different algorithms.The object detection system learns a general model of the class so that model is not too selective or else it cannot detect all objects due to variation inside a class type. One example is face detection having broad usage in many applications like digital cameras. One example of face detection is presented in Figure 4.

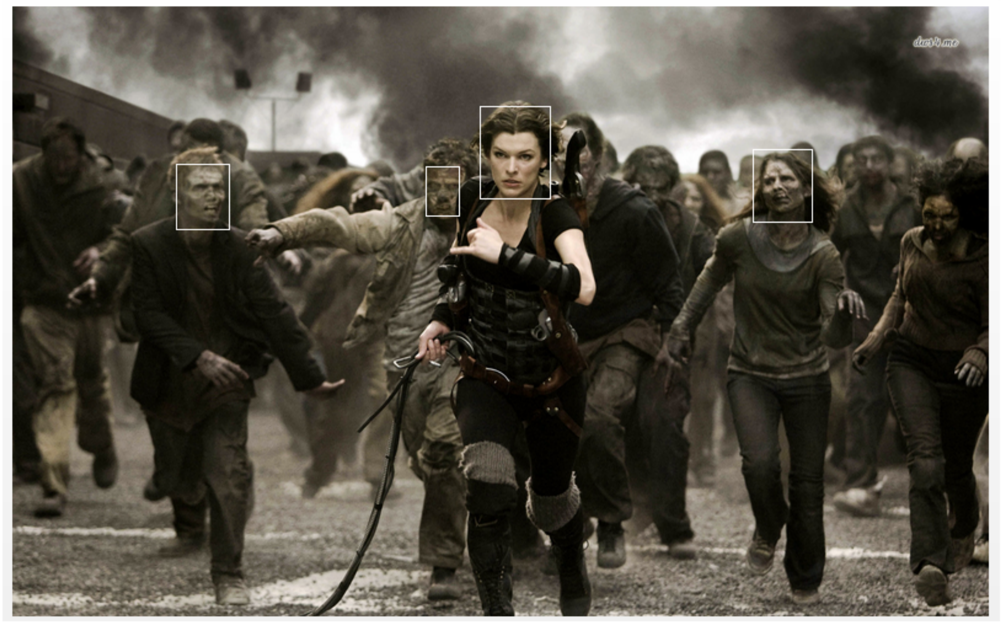


**Figure 4.** Example of face detection [15]

## 2.2 Object classification

The more challenging version of recognition is class or category recognition in which the problem is to identify the specific class of any instance presented in the image such as the general class of car, watch, airplane, face, etc. One common method to object categorization is shifting a search window over a given image and categorizing the object in the window with a classifier [16].

This section covers the most significant object detection approaches having differences with respect to what they are focusing on [17]:

- importance of localization (the location and scale of an object in the image [18] like Fig. 5 ).

  Detecting the presence of the object is the main goal of the methods, and the accurate localization is of second importance. While the exact localization has the highest priority in some other procedures.

- Detecting a single or detecting multiple object classes at a time.

- level of supervision including: supervised (manual), semi-supervised, unsupervised.

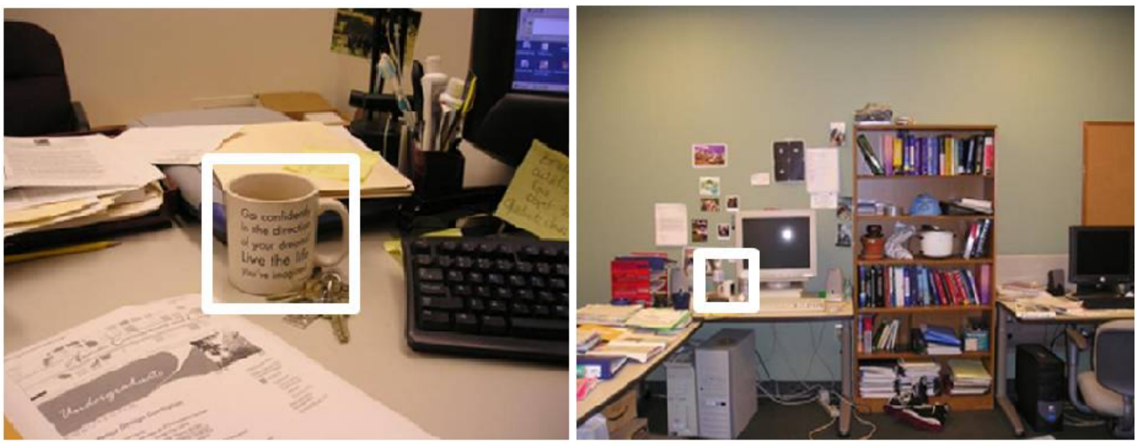


**Figure 5.** Examples of localization of mug [19]

## 2.3 Object detection methods

The main stream approach is to break objects into parts that are easier to detect such as Fig. 6. This is one of the oldest approaches of object recognition which was first introduced by Fischler and Elschlager [20]. In this model, instead of handling with the whole object to create a description, the object is first divided into smaller parts, then the appearance model is defined for each part which represents part features of the object class. Next, the geometric relationships between the parts are matched which is called *spatial structure*. Accordingly, description of local image parts can be less complicated than a description of the whole object. Another advantage is that the system can naturally handle the objects with parts occluded by other elements in the image (see Fig. 8) as well as deformed objects (see fig. 7) [17].

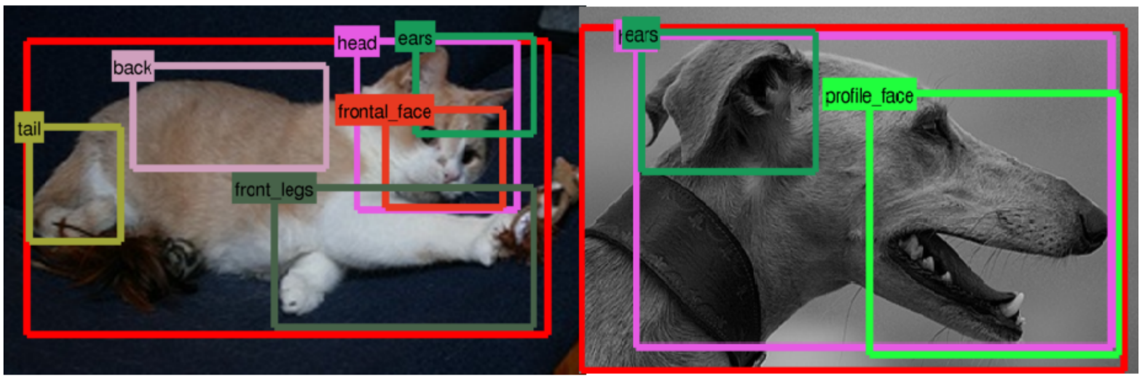


**Figure 6.** Part based recognition [21].

**Figure 7.** Example of deformed object.

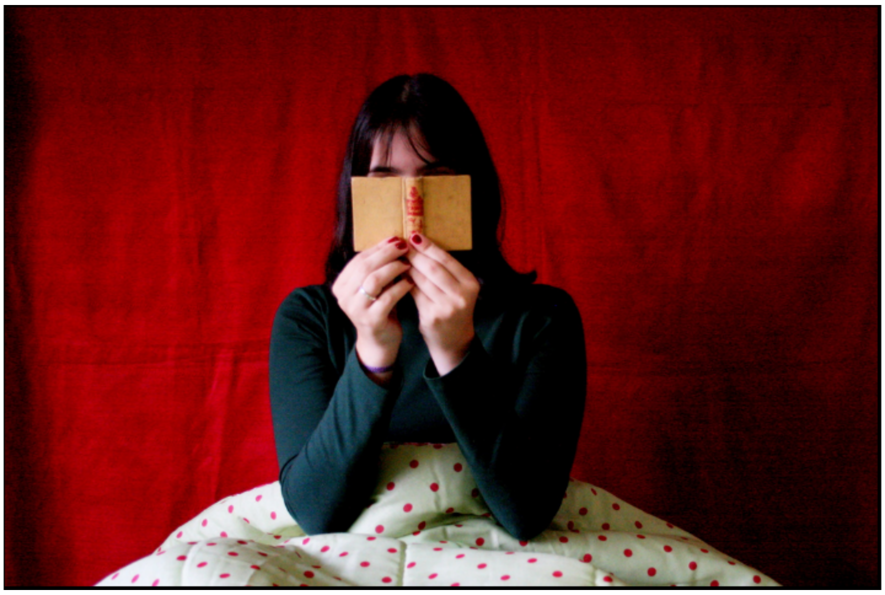


**Figure 8.** Example of occlusion: face is occluded by the book

The main issues in part-based recognition models are the representation of geometric relationships, the representation of divided parts, and algorithms for learning such descriptions and recognizing them at run time. For instance, in the first work by Fischler and Elschlager [20], the pictorial structure containing spring-like links between different feature locations (Fig. 9) was applied to represent geometric relationships [13]. Then they fit the pictorial structure to the image by operating mathematical process (more details in [20].

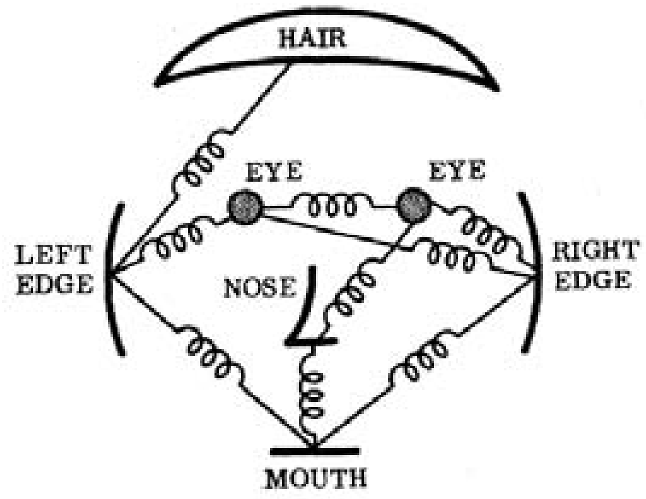


**Figure 9.** Pictorial structure representing the class: human face [20]

The geometric connection between the small parts of an object can be represented using varying topologies (Fig. 10, and Fig. 11). For instance, in the part-based model by Felzenswalb and Huttenlocher (2005) [22], tree topology is applied (Fig. 10(d)). Another topology called *sparse flexible model* proposed by Canoe and Lowe (2006) [23] in which the parts are ordered and represented in a graph shape (Fig. 11(g)), and the geometry of each part is related to the geometry of its $K$ neighboring parts, where $K$ is the degree of connectivity of each part (Fig. 10(c)). The most plain model is visual bag of features (bag of words, BOW) including parts without any geometric relationships (Fig. 10(e)), but they still can be counted as very efficient models [24, 25] . Tree and star topologies are recognized as the most efficient models in terms of inference and learning [22, 26, 27].

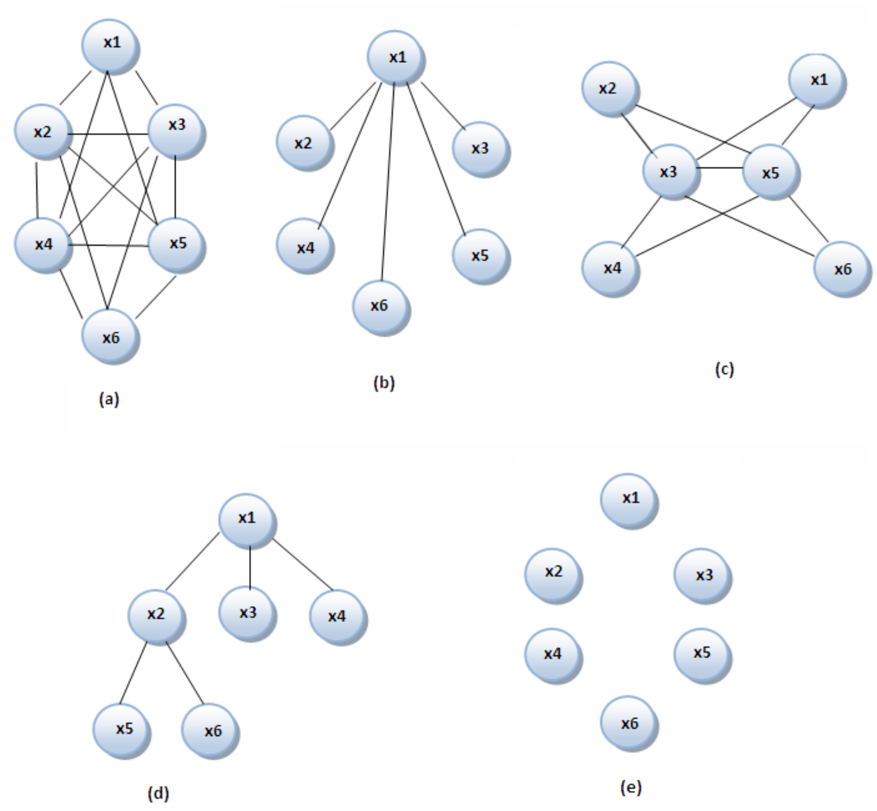
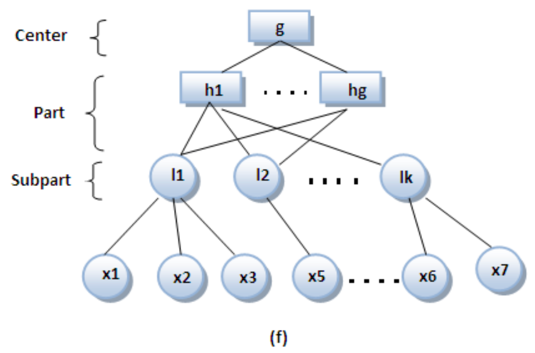


**Figure 10.** Graphical models for geometric spatial priors : (a) constellation [28]; (b) star [29]; (c) k-fan (k = 2) [29]; (d) tree [22]; (e) bag of features [24, 25]



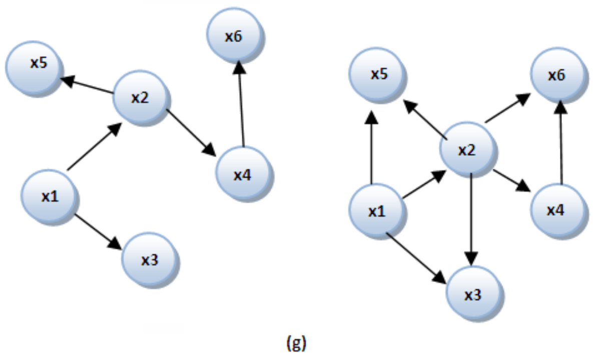


**Figure 11.** Graphical models for geometric spatial priors : (f) hierarchy [30]; (g) Sparse flexible model [23]

Recently, most of the work in the area of computer vision has become interested in applying The pictorial structure to solve the problems such as object recognition [31, 22, 32], facial feature detection [33], human pose estimation [34, 35, 36], and action recognition [37]. Specifically, DPMs [10, 11] ( discriminate trained deformable part-based models) have demonstrated high performance of object detection.

## 2.4 Learning object class model

The popular approach of object detection in part-based system consist of first discovering interest points in the training images including objects. A model of object class is learned by generating a local image description of the interest points, and finally testing all images that consist the objects [17].

The systems of Learning object class model utilize different interest point detection techniques depending on the level of supervision [17]. The level of supervision depends on the manual contribution of selecting interest points. The more supervised, the better quality resulting from accurate interest points leading to less complexity in the process of model learning since the probability of having interest points outside of the object will be quite low. Accordingly , object detection methods can be classified into: supervised, semi-supervised, and unsupervised types, however this classification may not be completely clear since the supervision level does not concern only the part of interest point detection, but also other stages of a system such as labeling, image alignment, and segmentation of training images.

### 2.4.1 Fully Supervised methods

In this type, the interest point selection is done completely manually, and object location in each image is known. Some examples of significant works are poselet-based methods [38, 39]. poselet is a new description of a part with special predefined requirements, and " is trained to respond to a given part of the object at a given viewpoint and pose" [40]. E.g. for the purpose of human detection, lots of poselets are defined such as frontal face, profile face, a head-and-shoulder configuration, etc. Some of the works based on this method are [38, 41, 39] where the appearance of human body parts is learned by manually annotated limb location.

### 2.4.2 Supervised learning methods

In supervised methods, training images are segmented and labeled (i.e. learning in the presence of ground truth).the number of training images and that how much they cover the possible variations in an object class affect the difficulty of learning a model. It is desired to define a model that can be generalized. For instance, with only a single view of a bike, the learning model cannot identify a bike from all directions [17].

Labeling can be done based on different criteria . In approach [16] where the task is divided into two steps: categorization between class object types (such as classification between faces and other classes) and identification within the class object ( face recognition of specific people among others) using Support Vector Machines (SVM), and component-based identification. For categorization part, label is used to distinguishes the class of the object in the image. For identification, label identifies the individual object [16]. e.g. scene categorization (mountains, forests, etc) is performed in [42] by *semantic classifica-*

*tion* process. In semantic classification, images are first segmented in a $10\frac{1}{2}10$ grid, and then each segment is classified to a specific class type (such as road, tree, etc. ) and called *local semantic*. scenes are classified based on the frequency of occurrence of local semantics (object). and then a prototype of each scene category is constructed. one example of the scene categorization but by other similar work [43] is shown in Fig. 12.

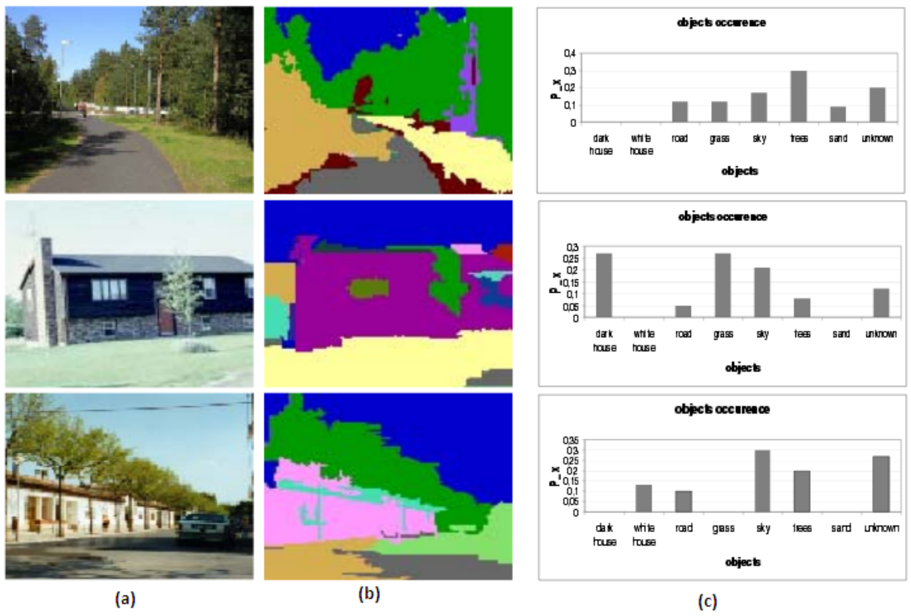


**Figure 12.** Results when classifying scenes by supervised method: (a) original image,(b) object recognition, (c) object occurrence. In this work [43] the goal is to organize images into three different scene categories: *road*, *suburb*, *city*. Images are segmented and labeled manually into 7 objects:*sky*, *grass*, *road*, *vegetation*, *dark house*, *white house*, and *ground*. (c) shows the number of occurrence of each object in the images.

Voila and Jones (2001) [44] proposed a method of real-time face detection where Images are segmented by performing AdaBoost and integral images, Then used for learning a face model. The detection is highly fast, although the training is slow. Also, classifier is learned from labeled data. In another approach [45], kernel methods is utilized for the purpose of object recognition where SVM was selected as kernel machine. This method performed successfully for supervised learning problems like regression and classification.

### 2.4.3 Weakly supervised (semi-supervised) methods

In this approach only a small part of the data is labeled , hens we need to use both labeled and large sets of unlabeled data [17].

Weakly supervised methods can also be divided based on their usage of constellation models. Lots of methods use the structure model. For instance, Agarwal and Roth [46] have used a vocabulary for object's parts, plus their spatial information. Works of Perona's team [47, 48] utilize the Expectation-maximization (EM) algorithm to learn the parts and constellation model. In [49], a hierarchical tree structure of local features Principal components analysis (PCA) SIFT based was defined to align images. Some other methods are not interested in using structure model such as [50] which takes advantage of combining different interest point detectors and local descriptors with AdaBoost, or in [51] Bayesian learning of image features is used.

### 2.4.4 Unsupervised methods

The goal of unsupervised methods is learning from unlabeled data. Some of the most common object detection methods are represented in the following sections, but before that, a brief description of the few methods of interest point detection and local image descriptions is given to have a clear understanding of the rest of the section.

# 3 LOCAL REGION BASED DETECTION

## 3.1 Interest points and detection method

Interest points are particular locations in the images that can be considered as first noticeable character of each object included in the image such as mountain peaks, building corners, etc, and they have clear definition usually in a mathematical way [13] interest points are also called as key point features, distinguished regions, affine regions, and salient regions. To define interest points, methods usually use the appearance of patches of pixels around the location of point, so this is the surrounding area of the point which specifies whether a point is interesting or not. The significant feature that should be considered for interest point detection methods should be stable to change of scale, rotation, noise, illumination. For instance,by changing the viewpoint or imaging conditions, there should be no changes in the discovered interest points [17]Some of the famous interest points detector methods are introduced in the fallowing.

### 3.1.1 Harris corner detector

Harris corner detector is one of the first interest point detectors in which both edge and corner of the image are detected [52] (see Fig. 13). Additionally, the junction is defined as meeting edges at corners. In Harris corner detector, Moravec's corner detector functions [2] are utilized. Such functions work by shifting a local window in the image that captures the intensity of each part of the image. The average variation of image intensity is found using the captured information.The amount of shifting could be adjusted in different directions. Shifting results can be interpreted as follows [52]:

1. The change resulted from all shift are minor if there is no serious intensity variation in the image.
2. In case of having window riding an edge, shift along the edge will result in a minor change. In contrast, a shift perpendicular to the edge will result in a large change.
3. All shifts will result in a large change if the window contains a corner or a single point. Therefore,large changes resulted by any of the shifts should be considered as a corner.

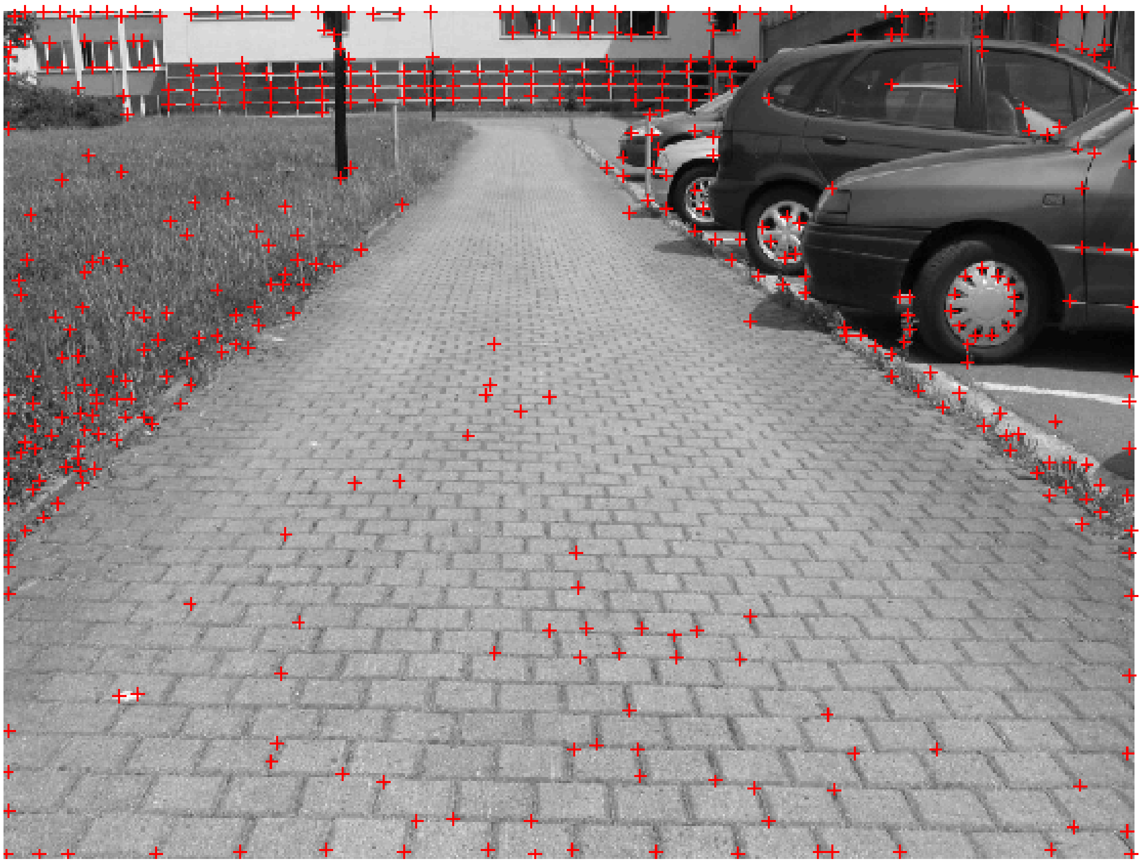


**Figure 13.** Interest points found by Harris corner detector [53]

Harris and Stephens [52] defined a mathematical expression based on the Maravec's idea calculating the change $E$ produced by a shift $(x, y)$, $I$ signifies image intensity and $W$ denotes the image window [52].

$$E_{x,y} = \sum_{u,v} W_{u,v} \left| I_{x+u,y+v} - I_{u,v} \right|^2 \tag{1}$$

The Moravec's method accompanied some difficulties including: an isotropic response due to performing just an uncontinuous set of shifts at every $45$ degrees and noisy response due to applying rectangular binary window. Another drawback is that the operator response too willingly to edges because only the minimum of $E$ is considered. These problems have been solved in Harris and Stephan approach [52], and the final useful equations are:

$$E_{x,y} = (x, y) * M\,(x, y)^T \tag{2}$$

where $M$ is $2 \times 2$ symmetric matrix defined as :

$$M = \sum_{x,y} W\left(x,y\right) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \tag{3}$$

where $I_x$, and $I_y$ are derivatives of $x$ direction, and $y$.

As it was mentioned large variation of $E$ in all directions indicates an interest point. This feature can be represented also by considering $\alpha$ and $\beta$ as eigen values of $M$, then we have: the corner is discovered if both $\alpha$ and $\beta$ are large, if both are too small then there is no interest point at all (i.e. the image is flat), and if one has large value and the other is small, then there is an edge. In [52], the function is defined with some changes in order to get rid of calculating eigenvalues that is computationally expensive [52, 54]:

$$trace\left(M\right) = \alpha + \beta Det\left(M\right) = \alpha\beta \tag{4}$$

therefore,

$$R = Det\left(M\right) - K\left(traceM\right)^2 \tag{5}$$

Where $K$ is a tunable sensitivity parameter in a range of $0.04$ to $0.15$. Positive value for $R$ indicates a corner region, negative value shows having an edge point, and for a flat region the value of $R$ is small. This method is more effective for finding corner points since they have much higher stability than edge-points. This model is robust to rotation and to some degree to intensity variations, but is variant to scale changes. This method has been improved later on by Mikolajczyk and Shmid [55] to compensate its weakness and make it robust to scale variation.The method is known as Harris-Laplace detector which is a dual process performing the Harris corner detector at multiple scales, and automatically selecting the characteristic scale. The system was again extended in the same approach in order to make it affine invariance and named Harris-Affine [55].

### 3.1.2 SIFT (Scale-Invariant Feature Transform) detector

It is an algorithm presented first by David Lowe in 1999 [7] to detect and describe local features in an image by transforming an image to a collection of local feature vectors. The idea is to find minima or maxima of a difference-of-Gaussian function in order to specify significant locations (clue) in scale -space to form feature vectors. The method was improved in 2004 [56]. Next the interest point detector is briefly described,and SIFT Descriptor is explained later since the SIFT detector and SIFT descriptor are two different methods. The algorithm consists of filtering steps, and the overall process is:

1. Scale-space extrema detection: Multiple scales and image locations are investigated to find locations and scales capable to be assigned frequently under various views of same scene using a continues function scale called Gaussian.The scale space of an image is identified by function $L\left(x, y, \sigma\right)$ created from the convolution of a Gaussian kernel(at different scales) for an input image $I\left(x, y\right)$:

$$L_{(x,y,\sigma)} = G_{(x,y,\sigma)} \times I\left(x, y\right) \tag{6}$$

2. Keypoint Localization: Some of the points are ignored by fitting a model to determine the location and scale of interest points, only selecting key points based on a measure of stability. Maximum and minimum of difference-of-Gaussian in scale space is detected in a way that each point is compared to its $8$ neighbors in the current image and $9$ neighbors each in the scales above and below, and the point is chosen if it is the maximum or minimum of all. After detecting a key point candidate, a detailed fit to nearby data is performed to determine location, scale, and ratio of principal curvatures. In the initial work in 1999 key points were found at the location and scale of a central sample point. In latter work, a 3D quadratic function is fitted to improve interpolation accuracy. In addition, the Hessian matrix is used to eliminate edge responses.

3. Orientation assignment: Compute best orientations for each key point region by generating histogram of local gradient directions at selected scale, and assigning a canonical orientation at the peak of smooth histogram where each key specifies stable coordinates: $(x, y, scale, orientation)$.

4. Keypoint description: Calculates local image gradients at the selected scale and rotation to describe each keypoint region. This method is able to remain invariant to image rotation, scaling, translation, and to some degree to illumination variation and affine or 3D projection (describe in more detail in Section 3.2.1).

### 3.1.3 Entropy based detector

the approach is proposed in [57], and is robust to change of rotation, translation and small affine transforms.Here, the detection process takes place by going through the fallowing stages:

1. Finding local image areas entropy (entropy of gray-level or color histogram) over different scales with the form of resizable circles. For example, there is a histogram including a strong peak in a flat image area for which the entropy is low, and a histogram of various peaks or with no peaks (flat) for an image area consisting more alternation which represents higher entropy.
2. Selecting scales which have maximum entropy
3. To weight entropy values, an inter-scale unpredicted measure is used. In this model, the scales with a powerful peak would have higher weight in the image areas, and vice versa [17].

## 3.2 Feature descriptors

Feature descriptors (local image descriptors) are used with interest point detectors. First, the keypoints (features) should be detected. To find which keypoints are matched we need to find which ones belong to corresponding locations in various images. Some famous methods are LBP (local binary patterns), SIFT, steerable pyramid, MOPS (multi-scale oriented patches), multi resolution Gabor filters, etc. But only a few of them are introduced next.

### 3.2.1 SIFT descriptor

Also known as Lowe's keypoint descriptor and is invariant to scale and rotation [56]. The SIFT method includes the following steps (Fig. 14 illustrated the steps):

1. Normalize region around the interest point.
2. Calculating gradient magnitude and orientation at each pixel in the region ( in a $16 \times 16$ window around the keypoint).

3. Applying Gaussian window on the circle to weight them such that as the distance form center point increases the gradient magnitudes become smaller. The advantage of weighting is preventing the descriptor of having large changes resulted by shifting window.

4. Generating an orientation histogram over the $4 \times 4$ sub regions of the window by splitting the weighted gradient in each sub region performing interpolation to $8$ primary directions and then summing.

5. Now, we have $4 \times 4$ descriptors over $16 \times 16$ sample array, and $4 \times 4$ times $8$ primary direction which leads to a vector of $128$ values, so the descriptor length is $128$, and is finally normalized to unit length [13].

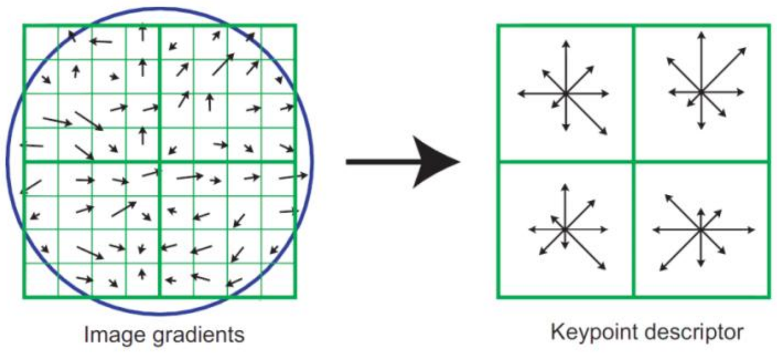


**Figure 14.** example of SIFT descriptor [13]

There are also other approaches such as PCA-SIFT which tries to extend the SIFT method and compensate the disadvantage of high dimensionality $(128)$ of the descriptor which can be difficult for classifiers.

## 3.3 Dense sampling

Dense sampling is an image representation technique which uses a fixed pixel interval to sample points i.e. points are sampled on a regular dense grid (see Fig. 15), so it contains large number of samples at regular intervals. There are two types of dense sampling: multi-scale dense sampling, and dense sampling with a single scale. Dense sampling provides a better coverage of the image such that for each image region it produces a fixed amount of features, and simple spatial relations between the features. Besides, in some cases where the training image has low contrast, then no interest point is detected.

While dense sampling method is capable to provide useful information even if the patches are not matched correctly. In addition, the regular pattern of spatial information between features is easily described in a plane model [5].

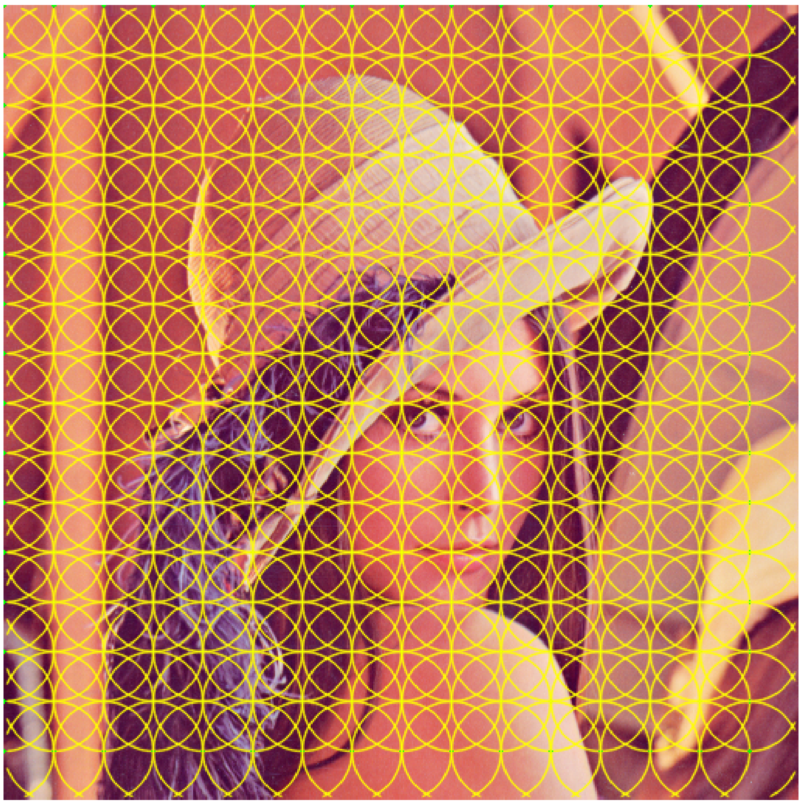


**Figure 15.** example of SIFT dense sampling [14]. A grid (yellow circle) shifts over the image and samples the interest points.

## 3.4 Geometric transformation

Generally, there are five types of linear transformation both in $2D$ and $3D$ spaces [13]. Which are going to be described here in short details in a $2D$ plane (Fig. 16) since the techniques are similar for $3D$ too. Prior to describing the transformation types, geometric primitives including points, lines are required to be introduced:

- 2D points or image's pixels are represented by the values of their coordinates:

$$\vec{P} = \begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

- 2D lines : Considering $\vec{L} = (a, b, c)$, The line equation is:

$$\vec{P}.\vec{L} = aP_x + bP_y + c = 0 \tag{7}$$

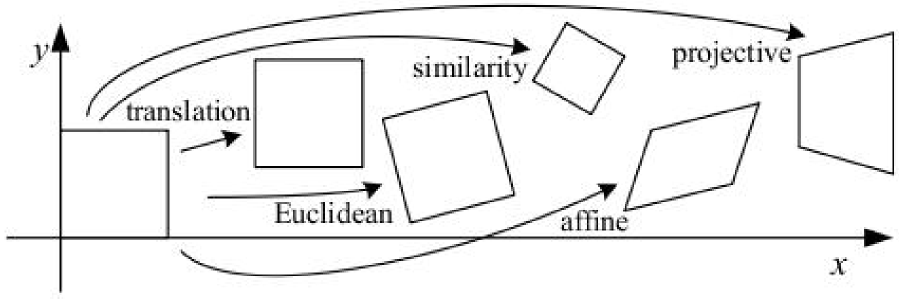


**Figure 16.** Set of 2D planar transformations [13].

1. **Translation**: Translation is described as a function moving the points in a specific direction with a fixed distance. Therefore, it can be considered as a constant vector which is added to all points. $P\,(P_x, P_y)$ is transformed into $Q\,(Q_x, Q_y)$ as follow:

$$\vec{Q} = \vec{M}\vec{P} + \vec{T} \tag{8}$$

where $M$ is defined as:

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \tag{9}$$

$$T = \begin{bmatrix} T_x \\ T_y \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} P_x + T_x \\ P_y + T_y \end{bmatrix} \tag{11}$$

2. Euclidean (also known as $2D$ rigid body motion or the $2D$ Euclidean transformation): Performs both rotations, and translation in the image [13]. Consider $M$ as rotation matrix, the rotation of point $P\,(P_x, P_y)$ is described as:

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$T = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$\vec{Q} = \vec{M}\vec{P} + \vec{T}$$
$$Q_x = P_x \cos\theta - P_y \sin\theta \tag{12}$$
$$Q_y = P_x \sin\theta + P_y \cos\theta$$

The translation was described in previous step, therefore the translation+ rotation equation can be represented as below:

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} P_x \cos\theta - P_y \sin\theta + T_x \\ P_x \sin\theta + P_y \cos\theta + T_y \end{bmatrix} \tag{13}$$

3. Similarity : also known as scaled rotation where scale is defined as following:

$$M = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$
$$T = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} P_x S_x \\ P_y S_y \end{bmatrix} \tag{14}$$

$S_x = S_y \Longrightarrow$ Uniform scaling
$S_x \neq S_y \Longrightarrow$ Differential scaling

The rotation is described before. Thus similarity is expressed as follow:

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} P_x S_x \cos\theta - P_y S_y \sin\theta \\ P_x S_x \cos\theta + P_y S_y \sin\theta \end{bmatrix} \tag{15}$$

4. Projective: also know as perspective transform or homography, and is defined as:

$$\vec{Q} = \vec{M}\vec{P} \tag{16}$$

Where $\bar{M}$ is an arbitrary $3 \times 3$ matrix [13].

5. Affine
In this type, each point such as $P\,(P_x, P_y)$ is transformed into $Q\,(Q_x, Q_y)$ as below:

$$\begin{aligned} Q_x &= aP_x + cP_y + T_x \\ Q_y &= bP_x + dP_y + T_y \end{aligned} \tag{17}$$

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} T_x \\ Ty \end{bmatrix} \tag{18}$$

$$\vec{Q} = \vec{M}\vec{P} + \vec{T} \tag{19}$$

# 4 STATE OF THE ART

## 4.1 Congealing methods

Object position in all training images can be automatically found by congealing instead of manually annotating them. The original congealing method [3] is shortly described here. The notation of ”congealing” first entitled by the seminal work of Learned-Miller [3], and refers to group-wise image alignment. Congealing is an algorithm to jointly align an ensemble of misaligned images (images are represented by a set of array of arbitrary dimension), and make them as similar as possible, according to some similarity measurement [3] (see Fig. 18). The term of alignment refers to any kind of allowable transformations. In [3], pixel values are used directly as features. Every congealing application consists of three essential elements:

- a set of arrays of measurements. e.g. In [3], binary images are selected as set of arrays of measurements.
- a continuous set of allowable transformations.e.g. affine transformation in [3].
- a measure of the joint similarity of the arrays within the set. like a method used in [58]

The general steps of congealing are:

### 4.1.1 Transform parameterization

In [3], The type of transformation considered for congealing is spatial transformation, and composed x-translation, y-translation, rotation, x-scale, y-scale, x-shear, y-shear, and have the transformation of vector $V$ as:

$$V = (t_x, t_y, \theta, s_x, s_y, h_x, h_y) \tag{20}$$

A transformation matrix is generated using the product of the members of $v$ in the same order as they have appeared (constant order):

$$\begin{aligned} U &= F\left(t_x, t_y, \theta, s_x, s_y, h_x, h_y\right) \\ &= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ & \begin{bmatrix} e^{s_x} & 0 & 0 \\ 0 & e^{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & h_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{21}$$

### 4.1.2 Entropy estimation

The image congealing algorithm repeatedly minimizes the entropy across the stack of image's pixels by transforming each image by a small amount with respect to a set of possible affine transforms. Image type is binary. In Binary images, pixels have only the values of $0$ (black) or $1$ (white), and Each image consists of $P$ pixels. The value of $i$th pixel in the $j$th image is expressed by $x_i^j$. The pixel value for a transformed image is specified by $x_i^{j\prime}$. Pixel stack is created by taking a value of a pixel at a particular location from each transformed image which is also shown by Figure 17. It can be expressed as: $x_i^{1\prime}, x_i^{2\prime}, ..., x_i^{N\prime}$ in which N is the total number of images. then empirical entropy is estimated

$$H\left(\hat{x_i}\right) = -\left(\frac{N_0}{N}\log_2\frac{N_0}{N} + \frac{N_1}{N}\log_2\frac{N_1}{N}\right) \tag{22}$$

where $N_0$ is the number of $0$ (black) and $N_1$ is the number of $1$ in the binary-valued pixel stack. The quantity $\sum_{i=1}^{P} H\left(x_i'\right)$ (each image contains P pixels) is minimized iteratively. Fig. 18 demonstrates the result of this method.

Generally, Image congealing methods working on pixel level require moderate initial alignment in order to converge properly, and are sensitive to intra-class variation and background clutter [59].

### 4.1.3 Extensions

There are also other congealing methods proposed [60, 61, 62, 63], but they are actually extended and improved versions of the original algorithm. In these methods, using original image intensities (as explained in 4.1.2) to represent features resulted in some dis-

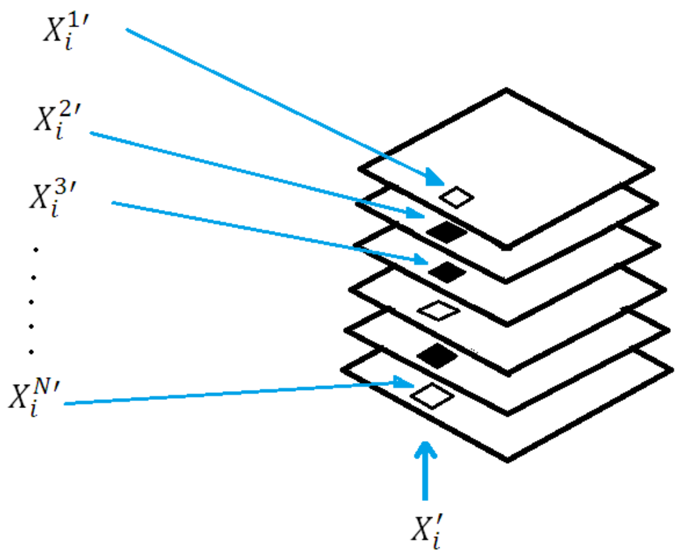


**Figure 17.** A pixel stack is a collection of pixels drawn from the same location in each of a set of N images. Here, the $i$th pixel from each of six images forms a pixel stack. Since half of the pixels are black and half are white, this corresponds to a Bernoulli random variable with parameter $p = 0.5$. The entropy of such a random variable is $(-0.5log_2 0.5 + 0.5log_2 0.5) = 1$ bit [3].

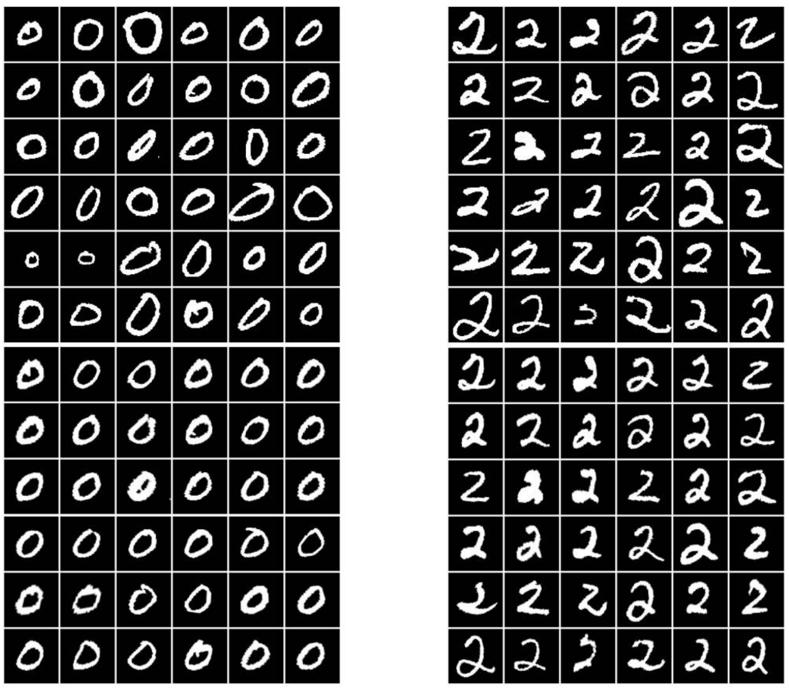


**Figure 18.** top left) Samples of zeros, (top right) samples of twos. (bottom left and right) Results after congealing using the Miller method

advantages. For example, if the image ensemble is large, then a high dimensional space for representing features (pixel intensities) is required, and computing would be time consuming. Another drawback is extracting intensities from all pixels, while each pixel has an approximately close intensity to the neighboring pixels. Thus, there will be extra pixel intensity [64]. A solution to overcome this problem is to create a methods with the task of feature selection to ignore extra information like a method by Xue [64]. To improve the performance and accuracy of congealing, various methods have been proposed [65, 4, 6].

For instance, Cox et al. [4, 66] and Yan et al. [6] create least-squares-based congealing algorithms . In the context of learning paradigm, unsupervised congealing [4, 5, 1], and semi-supervised congealing [2, 55] methods have been proposed. A global affine warp function is defined in [4] to compute pair-wise image distances. A novel technique called Fuzzy-entropy based image congealing is proposed in [66]. This technique uses the original algorithm with fuzzy-entropy to improve the performance.

The thing not considered is a more efficient feature representation, since the cost function is computed in most works by using pixel intensities criteria. However, a recent work by Xueya and Liu [64] tried to refine this problem by utilizing their own method of unsupervised feature selection in order to select only a sub-set of features automatically instead of considering all pixel features. They claimed that just utilizing less the 3 percent of the original feature representation, the accuracy and performance of congealing are considerably improved in comparison to methods with no feature selection. Huang et al. [62] used congealing for images with higher complexity such as faces and cars utilizing SIFT descriptors as the features. In [4] Newton optimization model is applied to improve the estimation of transformation parameters and patterns of oriented edge magnitudes (POEM) instead of the SIFT descriptor. Both edge information and the relation between pixels in a neighboring region is provided by POEM [65] leading to a better performance than congealing with SIFT. In [4] they took the benefit of Leaned-Miller congealing and Least Squares congealing. In this thesis, a different method to congealing works is used since methods based on pixel level processing are not effective when we have an ensemble of object severely varying in position and appearance. The assumption in congealing is that the ensemble contains images of similar objects in almost same poses. They cannot handle the problems of background clutter, occlusion, and noisy images. The problem mainly originates from omitting spatial information about the features. Our approach, the feature-based unsupervised alignment of visual class example gives a solution by defining a local feature-based algorithm to automatically align object class images to a given seed image. The following section describes the original technique.

## 4.2 Feature-based alignment

The recent datasets consist of images with different poses and geometric variation such as changes in translation, rotation, and scaling. Such variations make the learning object class models more challenging. Such challenges can be addressed using feature-based alignment method. Most popular solution is supervised alignment requiring manual annotation of objects. For example, method of Jiang et al. [58] need bounding boxes and Chen et al. [67] and kokkinos and Yuille [68] need manually annotated features in the initial training stage. This thesis uses an unsupervised solution based on spatial scoring in which extracted features are scored if their spatial configuration in other images isthe same. The whole process of the method is presented by the following steps where one of the images in the dataset is considered as a *seed image* which is an image that has good matches in as many other images as possible. Selection of seed will be discussed later.

1. **Detection and description of local features**

   For all images in the dataset, local features are extracted by local feature detector and represented by local feature descriptors.This step produces a vast number of detection failures and false matches (outlier) and thus we need to find out which points are useful (inlieres). There are various detectors and descriptors (see Sections 3.1 and 3.2) that could be used. In current study, SIFT descriptor (vl-SIFT)and dense sampling (vl-densems) have been selected since their performance in the feature-based alignment method was tested previously, and they both performed well.

2. **Finding putative feature matches**

   This step finds putative matches for N local features $F_s$ extracted from a seed image $I_s$. From another image $I_i$total of $M_i$ local features $F_i$ are extracted. Then, distance matrices $D_{N \times M_i}$ between the seed and $i = 1, .., I$ other images are computed using the SIFT descriptor distances. The notable thing is that not only the best matches are selected for each seed features, but also all distances are computed for the reason that local appearance variation makes even the best matches not perfect and thus a small number of the $K$ best matches need to be retained (the correct match is expected to be among the $K$ closest matches). Seed example and background noise in images define the optimal $K$. For instance, Figure 19 shows both wrong and correct feature matches.

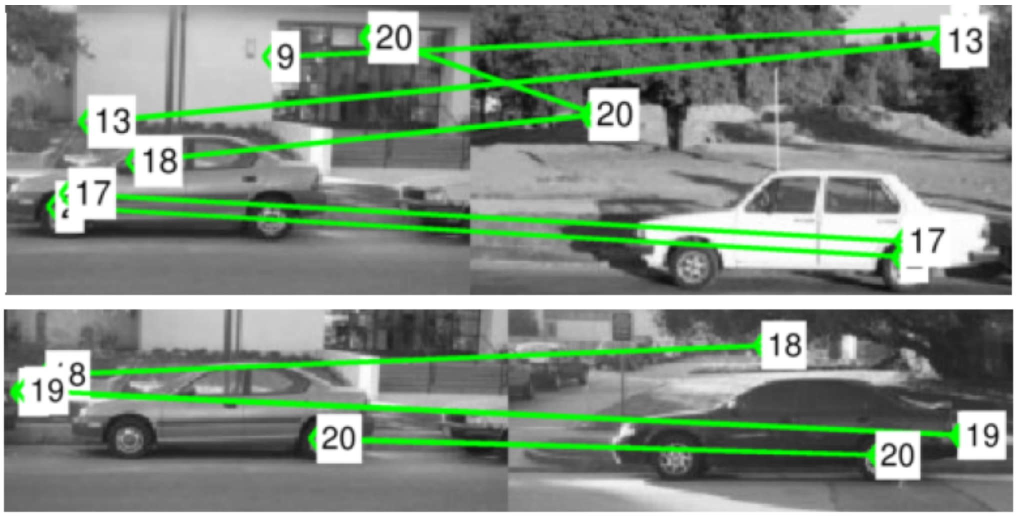


**Figure 19.** Matching local feature (SIFT detector descriptor) including both failed matches and correct ones.

3. **Selecting the best seed features by randomized spatial scoring** At this stage, the spatial location is used for accumulating scores for points which match under some transformation. The rigid linear transformations, $2D$ homography, are preferred for efficiency. The linear method with Umeyama [6] for estimating a similarity transformation (translation, scaling, and rotation) is adopted. Then, the minimum number of correspondences is randomly selected, estimated a homography transformation, transform all image points to the seed image and accumulate scores of the putative features within a preset distance. The pseudo code of this approach is given in Algorithm 1.

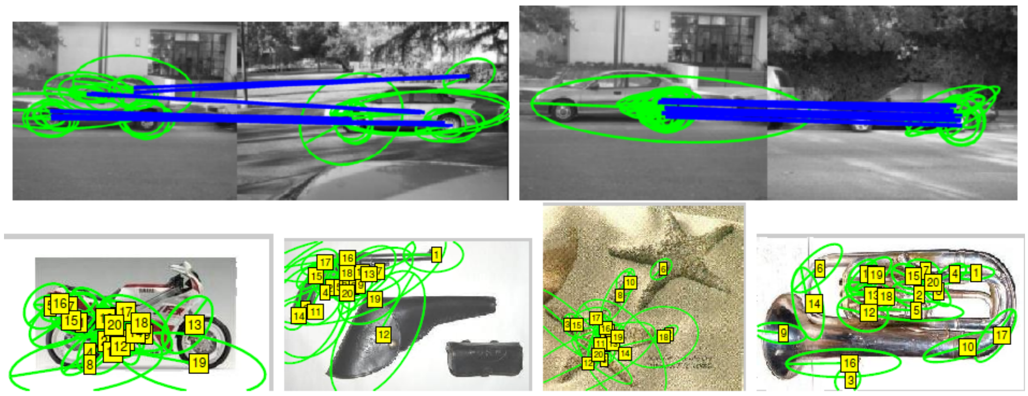


**Figure 20.** Top: examples of the highest scoring spatial matches (v in Alg. 1). Bottom: twenty best overall scoring landmarks (sN).

**Algorithm 1** Class specific landmark selection by spatial scoring.

```
Pick a seed image and remove it from the image ensemble.
Extract seed interest points and form their descriptors$(tot.of N)$
Initials score vectors $s_N$ of $N$ seed feature candidates.
for all images indexed with $i$ do
    Extract interest points and form their descriptors.
    Compute the distance matrix $D_{N \times M_i}$.
    Initials image$-$wise score vector $v \leftarrow 0$.
    for $R$ random iteration do
        Select two random seed features
        Select random correspondences within the $K$ best matches in $D_{N \times M_i}$
        Estimate $2D$ homography from the image $i$ to the seed space (Umeyama method [27])
        Transform all image features to the seed space.
        for all seed feature $j$ (excluding the select two) do
            if the seed feature $j$ has matches closer than $\tau$ within the $K$ best in $D_{N \times M_i}$ then
                Increment the seed feature score: $v(j) \leftarrow v(j) + 1$.
            end if
        end for
    end for
    Sort $v$ and increment the $L$ highest seed scores in $S_N$ .(All images have equal contribution.)
end for
Return coordinates and descriptors of the $L$ best scoring seed interest points.
```

The spatial scoring algorithm outputs the best L seed landmarks based on the top scores.The top scoring seed features represent parts which have been independently verified by other features in a similar configuration in other images.(Bring example image)

4. **Alignment using the selected seed features**

   With the best scoring landmarks $s_1, ..., s_L$ the alignment procedure itself is straightforward. For a number of random iterations, the minimum number of seed landmarks are selected, then homography is estimated to randomly selected Corresponding points within the best matches, and finally, the transformation that produces the highest number of inliers is selected and the transformation re-estimated using all inliers.

The problem with the original method is that, the seed image needs to be selected manually (supervised) so the method is supervised. Besides, when we have a large dataset, it is slow to find the best seed image for each class by testing all as the seed. This method needs to be extended by finding an approach to select a good seed image automatically.

# 5 AUTOMATIC SEED SELECTION IN FEATURE-BASED ALIGNMENT

As mentioned in Section 2, alignment of class examples is a predominant problem in learning object class models for detection, localization and classification. The problem has become more significant with the recent datasets where objects are not centered and poses are not carefully selected. In addition, performing this task manually is expensive and slow that can lead to prolong the progress of business and research laboratories. Thus, it is required to find a technique which has the ability of aligning objects in an unsupervised way which is the topic of this thesis. The disadvantage of feature-based method in Section 4.2 is that it relies on human at the first stage to select the best seed. This is not efficient in the real world where there are enormous number of images and classes. In the worst case, one needs to analyze all images in each class in order to find the one that could be matched best with other images. This section aims to describe a novel method to solve the problem.

## 5.1 Overall description

The goal is to automatically find an appropriate seed image by going through the steps of Algorithm 2. The steps will be explained next.

**Algorithm 2** general steps of best seed selection

```
1: for i = 1 to $I$ do
2:     set the image $i$ from data set as seed
3:     Create distance matrix , $D_{N\times M}$
4:     Compute spatial score matrix, $sptScore_{I\times I}$
5:     Compute similarity matrix, $simScore_{I\times I}$
6: end for
7: Perform optimal seed selection algorithm
```

To clarify the steps, and algorithms explained in rest of the thesis following Assumptions need to be considered.

Assumptions:

- $I$ denotes the total number of images. ($i = 1, 2, , , , , , I$)
- $D$ denotes distance matrix
- $sptScore$ denotes spatial score matrix
- $simScore$ denotes similarity matrix
- $seed$ denotes seed image
- $img$ denotes candidate image (any image from data set excluding seed that is going to be compared with seed)
- $N$ denotes the number of seed features
- $M$ denotes the number of image features
- $K$ denotes the number of best feature matches
- $T$ denotes transform matrix

### 5.1.1 Distance matrix

Distance matrix defines the difference between each seed's feature and candidate image's feature. To create distance matrix, we need to first extract features from images and seed. Thus, feature detection and description methods should be applied. The current work preferred to use dense sampling instead of interest point detection since work of [5] showed that producing more patches in dense sampling outperforms point detectors. The current work used single scale dense sampling version which is appropriate for images invariant in scale and rotation. The multiscale version of dense sampling is not feasible for this work due to performing slowly. Also, it is more suitable for images with scale and rotation variation. Methods which are utilized by this thesis as descriptor and detector are VL-SIFT (VLFEAT [1]), and VL-dense created by Andrea Vedaldi and other researcher. The program selects regions from images and computes their descriptors, and returns them in the form of matrices. For each image, there is a descriptor matrix. Distance Matrices are created by calculating the distance between each seed's feature to each of image's feature. i.e. if we have N number of seed feature, and number of features for $i$th image is M, then we have $N \times M$ calculation. e.g. distance from feature number $1$ of seed should be calculated from all $M$ features of the image. The calculation process is given by a simple example:

[1]http://www.vlfeat.org

For instance, the distance between $x$ and $y$ is calculated by:

$$Distance\,(x, y) = \left|x^2 + y^2 - 2 \times x \times y\right| \tag{23}$$

### 5.1.2 Spatial score matrix

The method to calculate spatial score corresponds to step $8$ to $20$ in algorithm1 (Section 4.2). Similar to alignment procedure all images are aligned using $k$ seed features.

The reason for performing the process of R random iteration is that it is computationally expensive to test all possible combinations to discover the best match, and thus sufficiently large $R = 1000$ is used. To estimate homography the direct linear transform $(DLT)$ [69] is utilized. The selected transformation type to map candidate features to a seed space a similarity transformation because of being capable of detecting objects in various orientations, location, and scales. Besides, its estimation process is faster than affine or projective as it has less free variables. The landmark distance threshold is set to $0.04$, the K best match is set to $5$, and the number of best scoring seed interest points $L = 80$.

The spatial score matrix is constructed by going through steps $8$ to $20$ in algorithm 1.(Section 4.2) and applying above assumptions. The spatial score for each seed is a $1 \times I$ dimensional vector since seed is compared to all images in data set. The spatial score of image $i$ and seed represents the number of matched features between transformed image $i$ and seed. In our case, the $L$ is initialized to $80$ which means we are checking the number of matches only for $80$ number of seed's features. Therefore, the values in spatial score are within $0 - 80$. The following shows one example of the output of the program for spatial score matrix. In this work, the spatial score is a matrix instead of a vector because, we run the algorithm 1.(Section 4.2) for $I$ number of iterations. At each round of this loop, one image from data set is taken as seed, and the spatial scores between the seed and other images are calculated and stored in a row of the spatial score matrix. For instance, if we have a data set containing $10$ images ($i = 1, 2, , , , 10$). In first round, image$1$ is set as seed, and the spatial scores between seed and images in data set ( $i = 1, 2, , , 10$) are calculated. Each score is stored in row $1$ with the same column number as image number in data set. e.g. spatial score between seed (image $1$) and image $3$ is placed in sptScore(1,3). The spatial score is also calculated between seed and image $1$ which are both same images. Obviously, the spatial score between one image and itself generates highest score which is 80 in our model. The process repeats until all images in the data

$$sptScore = \begin{bmatrix} 80 & 67 & 17 & 57 & 42 & 33 & 71 & 17 & 75 & 52 \\ 74 & 80 & 19 & 45 & 58 & 29 & 77 & 18 & 76 & 47 \\ 19 & 18 & 80 & 30 & 16 & 15 & 19 & 16 & 13 & 28 \\ 57 & 44 & 15 & 80 & 54 & 49 & 52 & 16 & 55 & 65 \\ 53 & 57 & 44 & 63 & 80 & 22 & 58 & 28 & 42 & 55 \\ 54 & 50 & 44 & 56 & 41 & 80 & 49 & 27 & 59 & 44 \\ 80 & 77 & 19 & 49 & 57 & 22 & 80 & 21 & 80 & 47 \\ 20 & 22 & 34 & 23 & 21 & 20 & 22 & 80 & 20 & 25 \\ 78 & 59 & 18 & 53 & 42 & 25 & 74 & 26 & 80 & 39 \\ 30 & 34 & 21 & 39 & 36 & 32 & 28 & 18 & 25 & 80 \end{bmatrix} \tag{24}$$

**Figure 21.** Row's number indicates the seed image index, and column number indicates candidate image index in the data set. Scores on diagonal are $80$ that shows the number of matches (landmarks) between the seed and itself. Therefore,it can be interpreted that images with scores closer to $80$ should be similar to seed.

set considered as seed and the spatial scores are calculated. At the end of process, we would have a $I \times I$ sptScore matrix. Following demonstrates the sptScore for the above example:

### 5.1.3 Similarity matrix

Similarity matrix is generated by adapting the technique proposed in [70] for unsupervised visual object categorization with BOF and spatial matching. The reason is that it performed very well for clustering objects in an unsupervised manner by a spatial score which is similar to this work, and in [70] was ' *spatial matching similarity rank*'. In [70], object categorization performance is improved by using similarity. Hence, to select the best seed that requires a reliable factor, the usage of similarity should provide higher accuracy rather than using the plain spatial score.To acquire the similarity matrix, the following steps are used:

1. **Ranking each image according to its spatial score** First, each row of sptScore matrix is sorted in an ascending order, and the sort indeces are stored in another matrix called $reservedIndex$. For instance, consider previous example of sptScore matrix in Section 5.1.2. The sorted sptScore and reservedIndex are :

$$sptScoreSort = \begin{bmatrix} 17 & 17 & 33 & 42 & 52 & 57 & 67 & 71 & 75 & 80 \\ 18 & 19 & 29 & 45 & 47 & 58 & 74 & 76 & 77 & 80 \\ 13 & 15 & 16 & 16 & 18 & 19 & 19 & 28 & 30 & 80 \\ 15 & 16 & 44 & 49 & 52 & 54 & 55 & 57 & 65 & 80 \\ 22 & 28 & 42 & 44 & 53 & 55 & 57 & 58 & 63 & 80 \\ 27 & 41 & 44 & 44 & 49 & 50 & 54 & 56 & 59 & 80 \\ 19 & 21 & 22 & 47 & 49 & 57 & 77 & 80 & 80 & 80 \\ 20 & 20 & 20 & 21 & 22 & 22 & 23 & 25 & 34 & 80 \\ 18 & 25 & 26 & 39 & 42 & 53 & 59 & 74 & 78 & 80 \\ 18 & 21 & 25 & 28 & 30 & 32 & 34 & 36 & 39 & 80 \end{bmatrix} \tag{25}$$

$$reservedIndex = \begin{bmatrix} 3 & 8 & 6 & 5 & 10 & 4 & 2 & 7 & 9 & 1 \\ 8 & 3 & 6 & 4 & 10 & 5 & 1 & 9 & 7 & 2 \\ 9 & 6 & 5 & 8 & 2 & 1 & 7 & 10 & 4 & 3 \\ 3 & 8 & 2 & 6 & 7 & 5 & 9 & 1 & 10 & 4 \\ 6 & 8 & 9 & 3 & 1 & 10 & 2 & 7 & 4 & 5 \\ 8 & 5 & 3 & 10 & 7 & 2 & 1 & 4 & 9 & 6 \\ 3 & 8 & 6 & 10 & 4 & 5 & 2 & 1 & 7 & 9 \\ 1 & 6 & 9 & 5 & 2 & 7 & 4 & 10 & 3 & 8 \\ 3 & 6 & 8 & 10 & 5 & 4 & 2 & 7 & 1 & 9 \\ 8 & 3 & 9 & 7 & 1 & 6 & 2 & 5 & 4 & 10 \end{bmatrix} \tag{26}$$

The value in each row of reservedIndex matrix refers to the column number of the image's place in unsorted sptScore matrix. The highest spatial score occurs in a diagonal of the sptScore matrix which means that the highest score is the one between seed and itself. It is also possible to have the same value as the highest one (80) between seed and other images such as seed 7 in above example. According to sptScore matrix, there are 3 images$(1, 7, 9)$ with the maximum score $80$. It is expected to have the seed's index in the last column of reservedIndex matrix, but row 7 demonstrates that it is not always true. reservedIndex(7,10) = 9 instead of being 7. The presence of this condition should be checked as fallowing

```
**if** reservedIndex $(i, I) \neq i, \forall i \exists \{1, 2, , , 10\}$ **then**
  Swap the values of reservedIndex$(i, I)$ with reservedIndex$(i, I-1)$
**end if**
```

The factor $rank\ (i, j)$ specifies the level of similarity of image $j$ to seed $i$ in compare to all images in a data set. Following is an example according to reservedIndex matrix in (14).

$$\begin{aligned} rank(2,2) &= 10 \\ rank(5,3) &= 4 \end{aligned} \tag{27}$$

2. **Converting rank to similarity**

   Similarity is created by using the simple formula:

$$S\left(i,j\right) = \frac{N}{rank\left(i,j\right)} \tag{28}$$

   To enforce the similarity matrix symmetric the following is applied:

$$S\left(i,j\right) = MAX\left(S\left(i,j\right), S\left(j,i\right)\right) \tag{29}$$

   The similarity matrix for our example would be:

$$simScore = \begin{bmatrix} 1.00 & 1.42 & 10.00 & 1.66 & 2.50 & 3.33 & 1.25 & 5.00 & 1.11 & 2.00 \\ 1.42 & 1.00 & 5.00 & 2.50 & 1.66 & 3.33 & 1.11 & 10.00 & 1.25 & 2.00 \\ 1.66 & 2.00 & 1.00 & 1.11 & 3.33 & 5.00 & 1.42 & 2.50 & 10.00 & 1.25 \\ 1.25 & 3.33 & 10.00 & 1.00 & 1.66 & 2.50 & 2.00 & 5.00 & 1.42 & 1.11 \\ 2.00 & 1.42 & 2.50 & 1.11 & 1.00 & 10.00 & 1.25 & 5.00 & 3.33 & 1.66 \\ 1.42 & 1.66 & 3.33 & 1.25 & 5.00 & 1.00 & 2.00 & 10.00 & 1.11 & 2.50 \\ 1.25 & 1.42 & 10.00 & 2.00 & 1.66 & 3.33 & 1.00 & 5.00 & 1.11 & 2.50 \\ 10.00 & 2.00 & 1.11 & 1.42 & 2.50 & 5.00 & 1.66 & 1.00 & 3.33 & 1.25 \\ 1.11 & 1.42 & 10.00 & 1.66 & 2.00 & 5.00 & 1.25 & 3.33 & 1.00 & 2.50 \\ 2.00 & 1.42 & 5.00 & 1.11 & 1.25 & 1.66 & 2.50 & 10.00 & 3.33 & 1.00 \end{bmatrix} \tag{30}$$

   Obviously, the closer is the value of similarity to $1$, the higher similarity between the image and seed. The similarity values on diagonal are all $1$ which shows the similarity between seed and itself.

$$For N = 50, Highest Rank = 50 \Rightarrow S = \frac{50}{50} = 1; \tag{31}$$

### 5.1.4 Seed selection using optimal seed selection algorithm

The main goal of this part is to create a model to compensate the weakness of mentioned feature-based alignment method where seed image needs to be chosen manually. We utilize two factors: spatial score (sptScore) and similarity matrix (simScore).

Spatial scores and similarity scores for all pairs of images (seed, img) are computed. It was found that seeds which show high scores for all images (img) are suitable candidates for best seed selection. Thus, for each seed, the mean value of all computed scores between the seed and all images are calculated. It is noteworthy that having a high mean value for a seed is not a sufficient condition since it is possible to acquire a high value for mean score by having only a few high spatial scores while other spatial scores are very low. That means that the seed could be matched best only with some of the images, but not for the most of other images.That is why the standard deviation (STD) needs to be taken into account. Therefore, we are looking for images with high ratio of mean value of spatial scores to STD value ($ratio = \frac{meanofspatialscores}{STD}$). To improve the precision of selecting optimal seed process, mean value of similarity scores and the STD value for each image as a seed are calculated. The overall process of selecting optimal seed which is mainly based on comparing mean and STD values of both spatial scores and similarity scores is presented through algotithm. 3. Next, the steps of the algorithm are clarified by performing the algorithm over the mentioned example in Section 5.1.

Requirements : spatial-score matrix, and similarity matrix

**Algorithm 3** Optimal seed selection

```
1: Initialize vector named $meanSpt \leftarrow$ NULL
2: Initialize vector named $stdSpt \leftarrow$ NULL
3: Initialize vector named $meanSim \leftarrow$ NULL
4: Initialize vector named $ratio \leftarrow$ NULL
5: for  i = 1 to I images do
6:     Compute mean value from scores stored in row $i$ of $sptScore$ matrix and store to $meanSpt\,(i)$
7:     Compute STD value from row $i$ of $sptScore$ matrix and store to $stdSpt\,(i)$
8:     $ratio\,(i) = \frac{meanStp(i)}{stdSpt(i)}$
9:     Compute mean value from row $i$ of $simScore$ matrix and store to $meanSim\,(i)$
10:    Compute STD value form row $i$ of simScore matrix and store to $stdSim\,(i)$
11: end for
12: Compute mean value of $meanSpt\,(i)$ over $i$ ($i = 1$ to $I$) and store to $MmeanSpt$
13: Compute STD of $meanSpt\,(i)$ over $i$ and store to $range$
14: Select images with $meanSpt$ > $MmeanSpt + \frac{range}{2}$
15: if number of selected images > 1 then
16:     Continue the steps only for selected images
17:     Find mean value over values of $ratio$ of images and store to $meanRatio$
18:     Select images with $ratio$ > $meanRatio$
19:     if number of selected images > 1 then
20:         Continue steps by considering only the selected images
21:         Compute mean value over $meanSim$ of images and store to $MmeanSim$
22:         Select images with $meanSim$ < $MmeanSim$
23:         if number of selected images > 1 then
24:             Continue steps by considering only the selected images
25:             Compute mean value over $stdSim$ of images and store to $range2$
26:             Select images with $stdSim$ < $range2$
27:             if number of selected images > 1 then
28:                 Continue steps by considering only the selected images
29:                 Select the image with minimum value in $meanSim$ vector
30:             end if
31:         end if
32:     end if
33: end if
34: Return the selected image as optimal seed.
```

1. **Mean value of spatial scores**
   Calculate mean value of spatial scores by considering each image as seed that can be achieved by taking mean value from each row of $sptScore$ matrix since each row of the matrix corresponds to seed, and each column indicates an image from a data set. Consider the sptScore from last example:

$$sptScore = \begin{bmatrix} 80 & 67 & 17 & 57 & 42 & 33 & 71 & 17 & 75 & 52 \\ 74 & 80 & 19 & 45 & 58 & 29 & 77 & 18 & 76 & 47 \\ 19 & 18 & 80 & 30 & 16 & 15 & 19 & 16 & 13 & 28 \\ 57 & 44 & 15 & 80 & 54 & 49 & 52 & 16 & 55 & 65 \\ 53 & 57 & 44 & 63 & 80 & 22 & 58 & 28 & 42 & 55 \\ 54 & 50 & 44 & 56 & 41 & 80 & 49 & 27 & 59 & 44 \\ 80 & 77 & 19 & 49 & 57 & 22 & 80 & 21 & 80 & 47 \\ 20 & 22 & 34 & 23 & 21 & 20 & 22 & 80 & 20 & 25 \\ 78 & 59 & 18 & 53 & 42 & 25 & 74 & 26 & 80 & 39 \\ 30 & 34 & 21 & 39 & 36 & 32 & 28 & 18 & 25 & 80 \end{bmatrix} \overset{mean}{\rightarrow} \begin{bmatrix} 51.1 \\ 52.3 \\ 25.4 \\ 48.7 \\ 50.2 \\ 50.4 \\ 53.2 \\ 28.7 \\ 49.4 \\ 34.3 \end{bmatrix} = meanStp \tag{32}$$

2. **Calculate STD values**
   STD values of spatial scores for each image as a seed is calculated by computing STD over each row of sptScore matrix as below:

$$sptScore = \begin{bmatrix} 80 & 67 & 17 & 57 & 42 & 33 & 71 & 17 & 75 & 52 \\ 74 & 80 & 19 & 45 & 58 & 29 & 77 & 18 & 76 & 47 \\ 19 & 18 & 80 & 30 & 16 & 15 & 19 & 16 & 13 & 28 \\ 57 & 44 & 15 & 80 & 54 & 49 & 52 & 16 & 55 & 65 \\ 53 & 57 & 44 & 63 & 80 & 22 & 58 & 28 & 42 & 55 \\ 54 & 50 & 44 & 56 & 41 & 80 & 49 & 27 & 59 & 44 \\ 80 & 77 & 19 & 49 & 57 & 22 & 80 & 21 & 80 & 47 \\ 20 & 22 & 34 & 23 & 21 & 20 & 22 & 80 & 20 & 25 \\ 78 & 59 & 18 & 53 & 42 & 25 & 74 & 26 & 80 & 39 \\ 30 & 34 & 21 & 39 & 36 & 32 & 28 & 18 & 25 & 80 \end{bmatrix} \overset{STD}{\rightarrow} \begin{bmatrix} 23.1 \\ 24.3 \\ 19.9 \\ 20.0 \\ 16.9 \\ 13.0 \\ 25.7 \\ 18.5 \\ 22.9 \\ 17.3 \end{bmatrix} = stdStp \tag{33}$$

3. **Calculating ratio**
   At this step, the output of step 1 ($meanStp$) is simply devided by output of step 2 (stdStp) :

$$ratio = \begin{bmatrix} \frac{51.1}{23.1} \\ \frac{52.3}{24.3} \\ \frac{25.4}{19.9} \\ \frac{48.7}{20.0} \\ \frac{50.2}{16.9} \\ \frac{50.4}{13.0} \\ \frac{53.2}{25.7} \\ \frac{28.7}{18.5} \\ \frac{49.4}{22.9} \\ \frac{34.3}{17.3} \end{bmatrix} = \begin{bmatrix} 2.2 \\ 2.1 \\ 1.2 \\ 2.4 \\ 2.9 \\ 3.6 \\ 2.0 \\ 1.5 \\ 2.1 \\ 1.9 \end{bmatrix} \tag{34}$$

4. **Mean value of similarity scores**
   In this part, mean value of similarity scores for each image is calculated by performing mean over each row of the $simScore$ matrix as below:

$$simScore = \begin{bmatrix} 1.0 & 1.4 & 10.0 & 1.6 & 2.5 & 3.3 & 1.2 & 5.0 & 1.1 & 2.0 \\ 1.4 & 1.0 & 5.0 & 2.5 & 1.6 & 3.3 & 1.1 & 10.0 & 1.2 & 2.0 \\ 1.6 & 2.0 & 1.0 & 1.1 & 3.3 & 5.0 & 1.4 & 2.5 & 10.0 & 1.2 \\ 1.2 & 3.3 & 10.0 & 1.0 & 1.6 & 2.5 & 2.0 & 5.0 & 1.4 & 1.1 \\ 2.0 & 1.4 & 2.5 & 1.1 & 1.0 & 10.0 & 1.2 & 5.0 & 3.3 & 1.6 \\ 1.4 & 1.6 & 3.3 & 1.2 & 5.0 & 1.0 & 2.0 & 10.0 & 1.1 & 2.5 \\ 1.2 & 1.4 & 10.0 & 2.0 & 1.6 & 3.3 & 1.0 & 5.0 & 1.1 & 2.5 \\ 10.0 & 2.0 & 1.1 & 1.4 & 2.5 & 5.0 & 1.6 & 1.0 & 3.3 & 1.2 \\ 1.1 & 1.4 & 10.0 & 1.6 & 2.0 & 5.0 & 1.2 & 3.3 & 1.0 & 2.5 \\ 2.0 & 1.4 & 5.0 & 1.1 & 1.2 & 1.6 & 2.5 & 10.0 & 3.3 & 1.0 \end{bmatrix} \tag{35}$$

$$\xrightarrow{mean} \begin{bmatrix} 3.4 \\ 3.0 \\ 6.1 \\ 2.9 \\ 3.1 \\ 4.6 \\ 2.9 \\ 6.1 \\ 3.1 \\ 3.1 \end{bmatrix} = meanSim \tag{36}$$

5. **Calculate STD for similarity values**

STD over each row of similarity matrix (simScore) is performed:

$$simScore = \begin{bmatrix} 1.0 & 1.4 & 10.0 & 1.6 & 2.5 & 3.3 & 1.2 & 5.0 & 1.1 & 2.0 \\ 1.4 & 1.0 & 5.0 & 2.5 & 1.6 & 3.3 & 1.1 & 10.0 & 1.2 & 2.0 \\ 1.6 & 2.0 & 1.0 & 1.1 & 3.3 & 5.0 & 1.4 & 2.5 & 10.0 & 1.2 \\ 1.2 & 3.3 & 10.0 & 1.0 & 1.6 & 2.5 & 2.0 & 5.0 & 1.4 & 1.1 \\ 2.0 & 1.4 & 2.5 & 1.1 & 1.0 & 10.0 & 1.2 & 5.0 & 3.3 & 1.6 \\ 1.4 & 1.6 & 3.3 & 1.2 & 5.0 & 1.0 & 2.0 & 10.0 & 1.1 & 2.5 \\ 1.2 & 1.4 & 10.0 & 2.0 & 1.6 & 3.3 & 1.0 & 5.0 & 1.1 & 2.5 \\ 10.0 & 2.0 & 1.1 & 1.4 & 2.5 & 5.0 & 1.6 & 1.0 & 3.3 & 1.2 \\ 1.1 & 1.4 & 10.0 & 1.6 & 2.0 & 5.0 & 1.2 & 3.3 & 1.0 & 2.5 \\ 2.0 & 1.4 & 5.0 & 1.1 & 1.2 & 1.6 & 2.5 & 10.0 & 3.3 & 1.0 \end{bmatrix} \overset{STD}{\rightarrow} \tag{37}$$

$$\begin{bmatrix} 3.5 \\ 2.7 \\ 3.5 \\ 2.7 \\ 2.6 \\ 3.0 \\ 2.7 \\ 3.5 \\ 2.7 \\ 2.6 \end{bmatrix} = stdSim \tag{38}$$

6. **Finding** $MmeanSpt$
Corresponds to step 12 of the algorithm. 3 and calculated by finding mean value over all columns of $meanStp$ from step 1 as below:

$$meanStp = \begin{bmatrix} 51.1 \\ 52.3 \\ 25.4 \\ 48.7 \\ 50.2 \\ 50.4 \\ 53.2 \\ 28.7 \\ 49.4 \\ 34.3 \end{bmatrix} \overset{mean}{\rightarrow} 44.3 = MmeanSpt \tag{39}$$

7. **Finding** $range$ Calculated by finding the STD over $meanStp$:

$$meanStp = \begin{bmatrix} 51.1 \\ 52.3 \\ 25.4 \\ 48.7 \\ 50.2 \\ 50.4 \\ 53.2 \\ 28.7 \\ 49.4 \\ 34.3 \end{bmatrix} \overset{STD}{\rightarrow} 10.5 = range \tag{40}$$

8. **Selecting images according to level 14 of algorithm**
In this step, only images with $meanSpt$ > $MmeanSpt + \frac{range}{2}$ are selected for analyzing in rest of the process. The reason for considering the criteria of $MmeanSpt+ \frac{range}{2}$ where images with lower $meanSpt$ are dropped out is that the method tested with several criteria and based on try and error technique. The results of testing showed that the capability of images with $meanSpt$ < $MmeanSpt + \frac{range}{2}$ to become a feasible seed is quite low.

$$\text{meanSpt} = \begin{array}{c c} & c_1 \\ img_1 & 51.1 \\ img_2 & 52.3 \\ img_3 & 25.4 \\ img_4 & 48.7 \\ img_5 & 50.2 \\ img_6 & 50.4 \\ img_7 & 53.2 \\ img_8 & 28.7 \\ img_9 & 49.4 \\ img_10 & 34.3 \end{array} > 44.3 + \frac{10.5}{2} \Rightarrow \begin{bmatrix} 51.1 \\ 52.3 \\ \\ \\ 50.2 \\ 50.4 \\ 53.2 \\ \\ \\ \\ \end{bmatrix} \tag{41}$$

Therefore, selected images are image 1, 2, 5, 6, and 7.

9. **Calculate mean of ratio for remained images** Corresponds to step 17 of algorithm. 3. Ratios are calculated in step 3 , thus:

$$\text{meanSpt} = \begin{array}{c} \\ img_1 \\ img_2 \\ img_3 \\ img_4 \\ img_5 \\ img_6 \\ img_7 \\ img_8 \\ img_9 \\ img_10 \end{array} \begin{array}{c} c_1 \\ \left[\begin{array}{c} 2.2 \\ 2.1 \\ \\ \\ 2.9 \\ 3.6 \\ 2.0 \\ \\ \\ \\ \end{array}\right] \end{array} \overset{mean}{\rightarrow} 2.6 = meanRatio \tag{42}$$

10. **Select images with** $ratio$ **>** $meanRatio$
    This part corresponds to step 18 of algorithm. 3.

$$\text{meanSpt} = \begin{array}{c} \\ img_1 \\ img_2 \\ img_3 \\ img_4 \\ img_5 \\ img_6 \\ img_7 \\ img_8 \\ img_9 \\ img_10 \end{array} \begin{array}{c} c_1 \\ \left[\begin{array}{c} 2.2 \\ 2.1 \\ \\ \\ 2.9 \\ 3.6 \\ 2.0 \\ \\ \\ \\ \end{array}\right] \end{array} > 2.6 \Rightarrow \; = \begin{array}{c} \\ img_1 \\ img_2 \\ img_3 \\ img_4 \\ img_5 \\ img_6 \\ img_7 \\ img_8 \\ img_9 \\ img_10 \end{array} \begin{array}{c} c_1 \\ \left[\begin{array}{c} \\ \\ \\ \\ 2.9 \\ 3.6 \\ \\ \\ \\ \\ \end{array}\right] \end{array} \tag{43}$$

According to (29) only image 5 and 6 are selected.

11. **Find** $Mmeansim$ Similar to step 21 of algorithm, Calculates mean over $meanSim$ columns (from step4) of remained images.

$$\text{meanSim} = \begin{array}{c|c|} & c_1 \\ img_1 & \\ img_2 & \\ img_3 & \\ img_4 & \\ img_5 & 3.1 \\ img_6 & 4.6 \\ img_7 & \\ img_8 & \\ img_9 & \\ img_10 & \end{array} \overset{mean}{\rightarrow} 3.8 = MmeanSim \tag{44}$$

12. **Select images with** $meanSim$ **<** $MmeanSim$

    Same as step 22 of the algorithm.

$$\text{meanSim} = \begin{array}{c|c|} & c_1 \\ img_1 & \\ img_2 & \\ img_3 & \\ img_4 & \\ img_5 & 3.1 \\ img_6 & 4.6 \\ img_7 & \\ img_8 & \\ img_9 & \\ img_10 & \end{array} < 3.8 \Rightarrow \text{meanSim} = \begin{array}{c|c|} & c_1 \\ img_1 & \\ img_2 & \\ img_3 & \\ img_4 & \\ img_5 & 3.1 \\ img_6 & \\ img_7 & \\ img_8 & \\ img_9 & \\ img_10 & \end{array} \tag{45}$$

Now, we have only one image as selected, and according to step 23 of the algorithm the checking process is over and the image $5$ is selected as optimal seed. The accuracy of selection the image $5$ as optimal seed is obvious by comparing the spatial scores, similarity scores, and STDs of images. It is noteworthy that when the algorithm reaches the step where there are only image $5$ and image $6$ with high spatial scores, it selects the one with lower $meanSim$. The reason is that, the experiment demonstrated that when we have images with close spatial scores, and acceptable STDs, then the similarity scores affects the result more significant than spatial scores. That is why at this level we select the image with lower similarity scores ( closer to $1$ means higher similarity between images).

The following table shows the output of the algorithm for 9 types of classes as training data sets in our work. The table shows the spatial scores and similarity scores for both feature-based alignment method where seed is selected manually, and for optimal seeds which are automatically selected by our algorithm. The features for an appropriate seed are having high spatial scores, low STD of spatial scores, and low similarity scores with low STD of similarity scores.

**Table 1.** Comparison of mean and std values between supervised and unsupervised seed selection

| CLASS | Method | spatial score | | similarity | |
|---|---|---|---|---|---|
| | | Mean | $STD$ | Mean | STD |
| STOP−SIGN | Supervised | 58.02 | 20.85 | 4.60 | 7.88 |
| | UNsupervised | 60.82 | 16.45 | 4.52 | 7.88 |
| CAR−SIDE | Supervised | 30.18 | 10.12 | 5.59 | 10.12 |
| | UNsupervised | 30.34 | 10.76 | 5.45 | 8.52 |
| MOTOBIKE | Supervised | 63.54 | 18.05 | 4.68 | 7.84 |
| | UNsupervised | 65.96 | 18.42 | 4.59 | 7.85 |
| FACES−EASY | Supervised | 60.12 | 13.08 | 4.58 | 7.86 |
| | UNsupervised | 62.46 | 13.05 | 4.58 | 7.85 |
| DOLLAR−BILL | Supervised | 57.72 | 17.8726 | 4.4996 | 7.8903 |
| | UNsupervised | 52.94 | 16.08 | 4.60 | 7.85 |
| AIRPLANES | Supervised | 31.2 | 11.11 | 4.64 | 7.85 |
| | UNsupervised | 36.78 | 10.5527 | 4.68 | 7.82 |
| REVOLVER | Supervised | 40.66 | 13.31 | 4.55 | 7.87 |
| | UNsupervised | 40.92 | 10.27 | 4.70 | 7.84 |
| EUPHONIUM | Supervised | 39.92 | 19.00 | 4.59 | 7.87 |
| | UNsupervised | 36.08 | 14.55 | 5.24 | 8.03 |
| STARFISH | Supervised | 22.36 | 11.40 | 5.25 | 7.84 |
| | UNsupervised | 28.80 | 12.14 | 4.79 | 7.86 |

According to table, our method selected seeds with better features for all classes except $euphonium$ and $dollarbill$. For instance, in the class of $revolver$, the spatial score of selected seed with unsupervised method is $40.9$ and STD is $10.27$, while in supervised method the spatial score is $40.66$ with STD of $13.31$. The spatial scores are almost close but the STD value in supervised method is high. If we consider the ratio of spatial score to it's STD, and the ratio of similarity score to it's STD, then we can see that the unsupervised method performed well for all classes, even for $euphonium$ and $dollarbill$. For example, the ratios for the class of $dollarbill$ are calculated in below:

In supervised method:

$$\begin{aligned} sptRatio &= \frac{57.72}{17.87} = 3.23 \\ simRatio &= \frac{4.49}{7.89} = 0.56 \end{aligned} \tag{46}$$

In unsupervised method:

$$\begin{aligned} sptRatio &= \frac{52.94}{16.08} = 5.8 \\ simRatio &= \frac{4.60}{7.85} = 0.58 \end{aligned} \tag{47}$$

The above equations show that the ratio of spatial score to its'STD in unsupervised method is superior ( higher ratio is more desired). The ratio of similarity score to the STD is superior in supervised method ( lower is desired), but we can claim that they are almost close. Consequently, the unsupervised method can compete the supervised one.

## 5.2 Alignment with optimal seed

Once the best seed is selected it is given to the feature-based alignment algorithm (Sec 4.2). Then, each image is transferred to the seed space using the generated transform matrix, then the number of overlapped landmarks from candidate to seed image is calculated same as the algorithm (Sec 4.2). Some example of selected best seed by our method is given in Fig. 22.

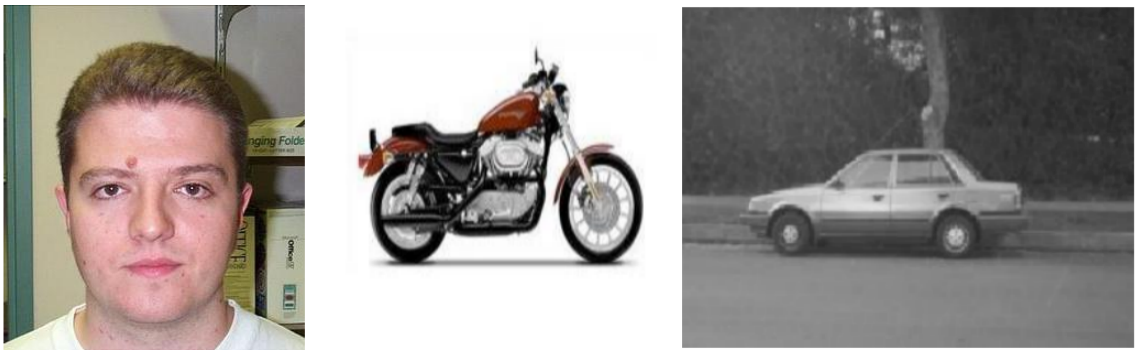


**Figure 22.** Examples of best seed selected by our method for the class of face, motorbike, and car side.

So far In this work, the alignment procedure has been done by transforming each image directly to the seed space like multiplying the image features with its estimated transform matrix $(similaritytransformation)$ :

$$Landmarks \times T \tag{48}$$

A remaining problem is that a single seed does not work well in some cases when the object in the image is failed through the alignment such as Fig.23. One solution could be to use a local approach such as alignment tree instead of applying a single global seed. Similar to landmark selection in $[57]$ . In the next section, we study the following tree-base step-wise alignment paths: Minimum spanning tree, Dijkstra algorithm.

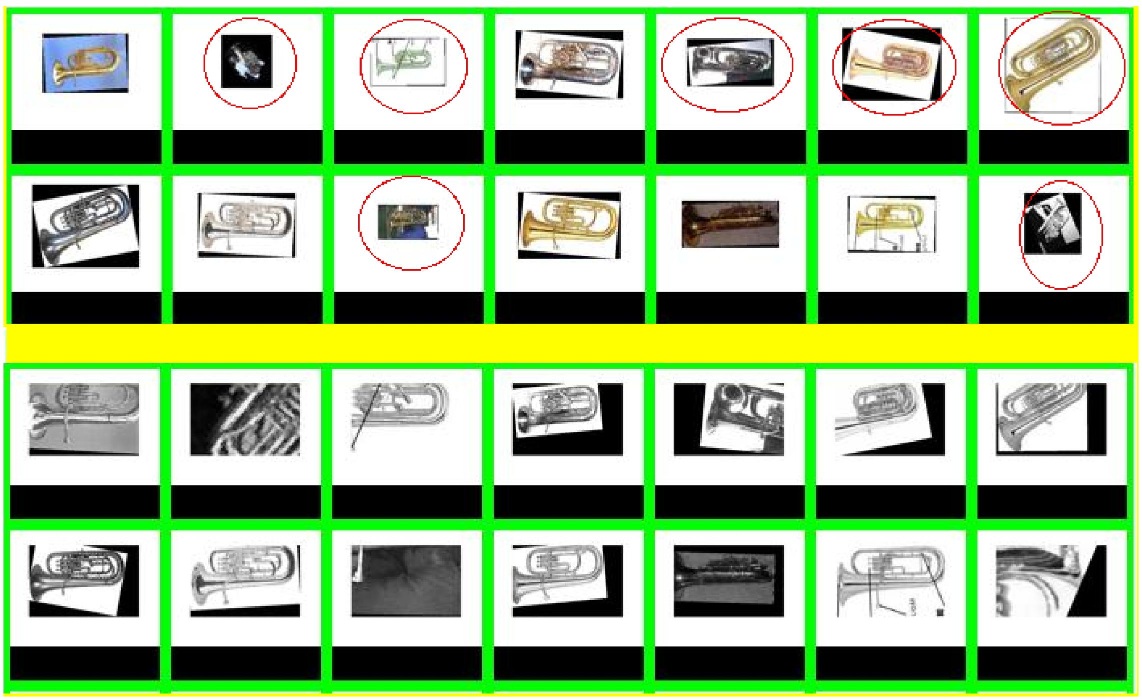


**Figure 23.** Example of failed object from *euphonium* class. Top: original images including euphonium, red circles show objects which are failed via transformation to the seed space. Bottom: transformed images.

# 6 STEP-WISE FEATURE BASED ALIGNMENT

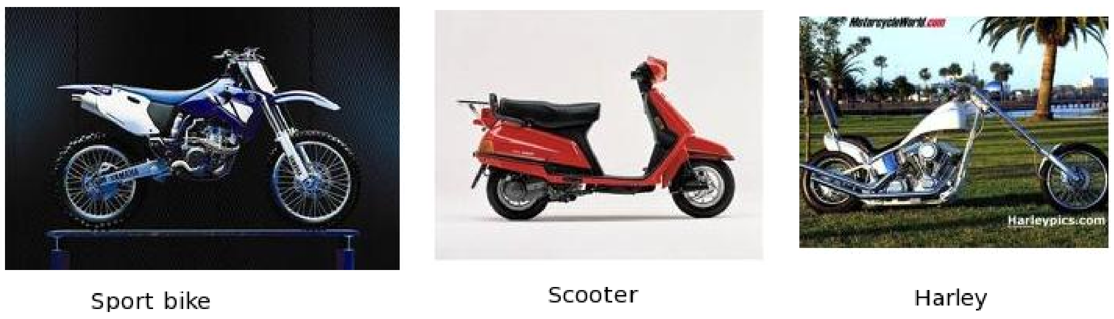


**Figure 24.** Examples of object variation inside a class of motorbike including: scooter, sport bike, Harley davidson.

So far, we attempted to find the best class specific landmarks in a single image (seed) using random spatial scoring and feature similarity. However, it is easy to show that such alignment procedure fails to align classes which are more complicated (e.g. motorbikes' class containing sport bikes, Scooter, and Harley Davidson. Fig. 24 ). A better solution could be a step-wise transfer via more similar examples to the seed. Next, two approaches are proposed, minimum spanning tree and Dijkstra shortest path, both based on a graph of similarity values. The graph is full connected where each node represents an specific image, and similarity values are set as weight of the edges. One example is shown in Fig. 25

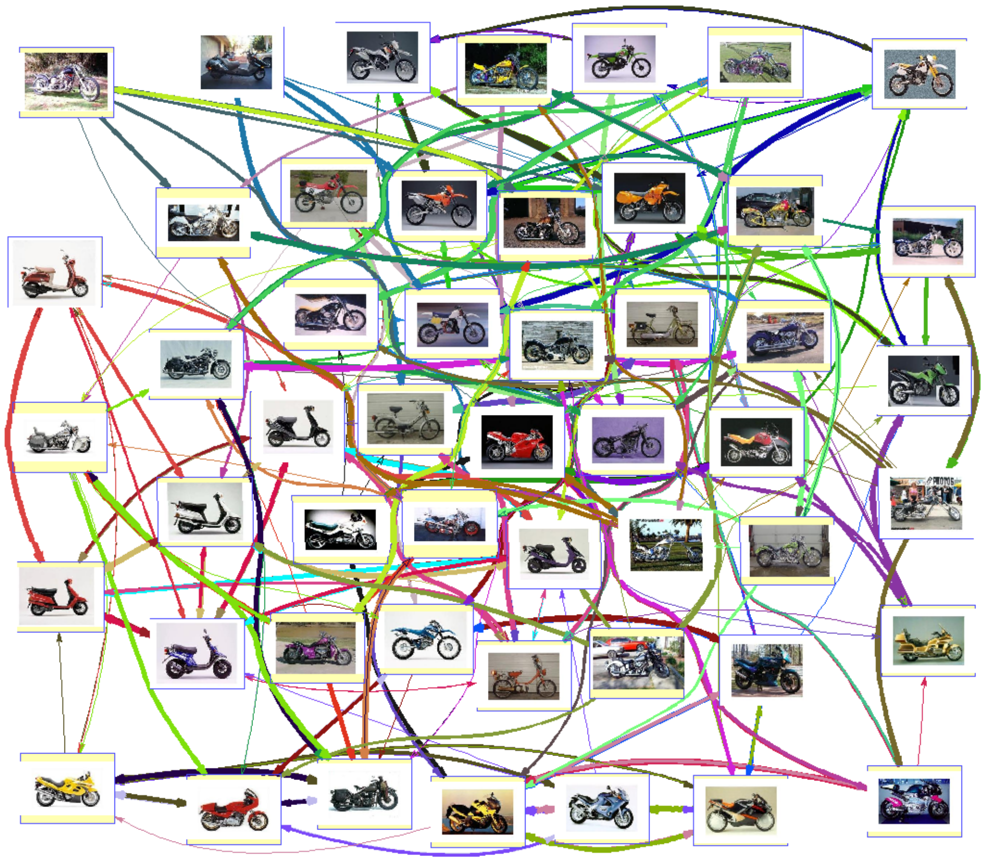


**Figure 25.** Graph of images for the classtype: 3 mixture of scooter, sports, Harley davidson. The thickness of the edges represents the amount of similarity between each node such that the thicker the edge, the higher similarity between the images.

## 6.1 Minimum spanning tree(MST) approach

The step-wise alignment performed to create a more smooth transform of an image to the seed. This is done by concatenating (multiplying) transformations of each visited node (image) along the path to the seed.

A spanning tree is a subgraph of a connected undirected graph that connects all vertices without causing loop (no cycle) [71]. For each graph, there can be several spanning trees. If the edges have weights, then the minimum spanning tree or the minimum weight spanning tree is a spanning tree with the lowest total cost. There are two algorithms commonly used to implement the MST: Prim's algorithm [71], and Kruskal's algorithm [71]. They differ by implementation, time complexity. In addition, if the given graph is unconnected, Prim's algorithm returns only the tree containing the root, but Kruskal's algorithm returns a separate MST for each unconnected part. In this work, only Prim's algorithm is used since we always have a full connected graph.

### 6.1.1 Prim's algorithm

Prim's algorithm initials the MST by setting a node from the graph as the root. The MST is extended by adding the smallest weight edge at time that connects one of the existing MST vertices to any node that does not exist in MST. Minimal edge at a time that connects one of the existing vertices in the growing MST with any other nodes that does not exist in the MST yet [71]. In this thesis, the optimal seed is set as the root. An example is shown in Fig. 26 .

The pseudo code of Prims' algorithm is given below and provided from [71]."In the pseudo code, the connected graph $G$ and the root $r$ of the minimum spanning tree to be grown are inputs to the algorithm. During execution of the algorithm, all vertices that are not in the tree reside in a min-priority queue Q based on a key attribute. For each vertex $v$, the attribute $v.key$ is the minimum weight of any edge connecting $v$ to a vertex in the tree; by convention, $v.key = \infty$ if there is no such edge. the attribute $v.\Pi$names the parent of $v$ in the tree. The algorithm implicitly maintains the set $A$ from GEBERIC-MST as"[76]:
$A = \{(v, v.\Pi) : v \in V - \{r\} - Q\}$. where $V$ is the collection of all vertices in $G$. After terminating algorithm, the min-priority queue $Q$ is empty; the MST $A$ from $G$ is thus
$A = \{(v, v.\pi) : v \in V - \{r\}\}$.

**Algorithm 4** MST-PRIM(G,W,r)

```
for each $u \in G.V$ do
    $u.key = \infty$
    $u.\pi$ = NIL
end for
$r.key = 0$
$Q = G.V$
while $Q \neq \phi$ do
    $u$ = EXTRACT-MIN($Q$)
    for each $v \in G.Adj\,[u]$ do
        if $v \in Q$ and $w\,(u, v) < v.key$ then
            $v.\pi = u$
            $v.key = w\,(u, v)$
        end if
    end for
end while
```

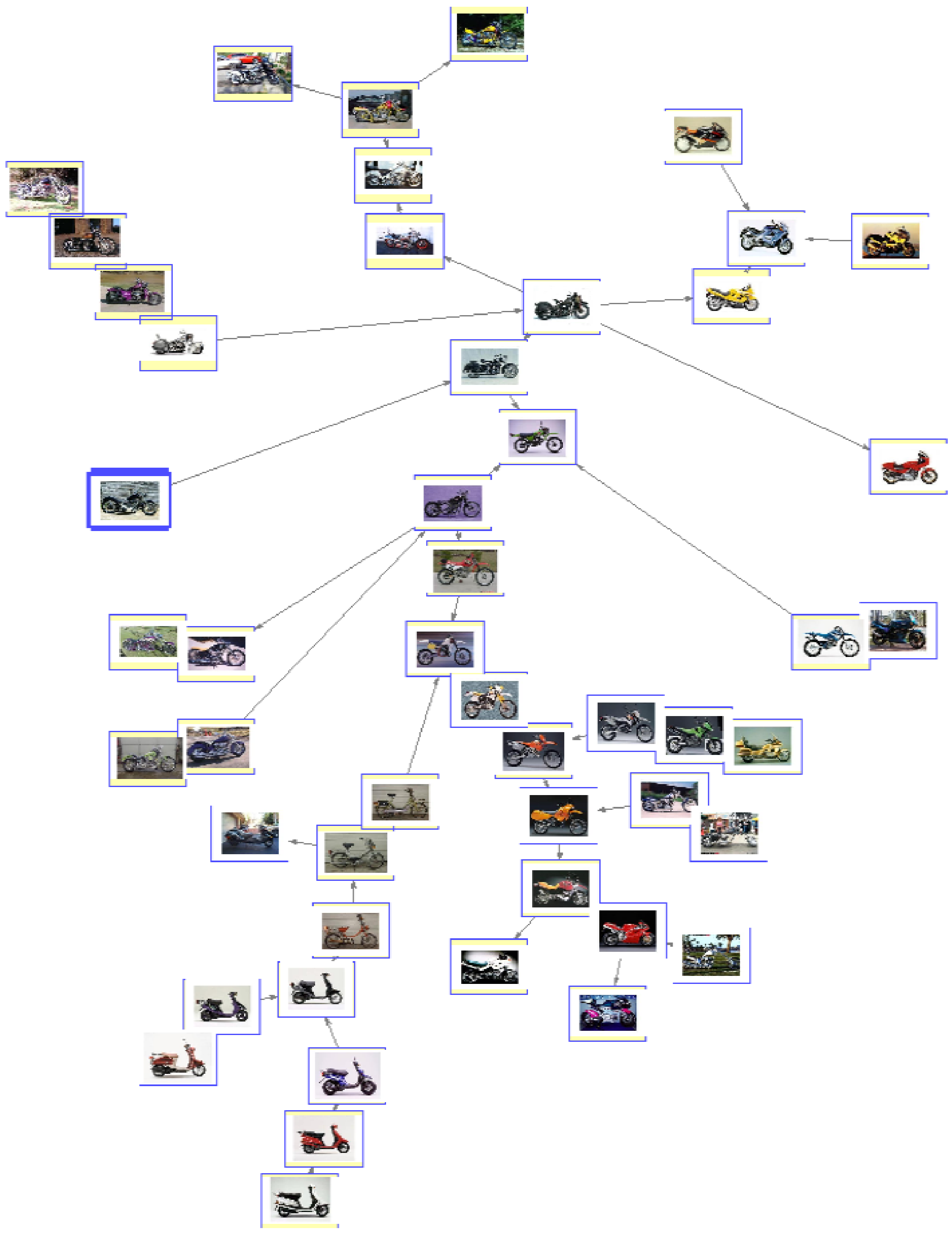


**Figure 26.** Minimum spanning tree created for the graph in Fig. 25. Root (seed) is specified with dark blue border. Note that images with less similarity to the seed are located far from the root. Also, images with the same intra-class type such as scooters are arranged in a same branch.

### 6.1.2 Step-wise alignment through MST

The goal is to align each image step by step via the visited links to the root. First a vector consisting of the visited nodes along the path is generated, and then using this vector transform matrices which are pre-calculated are concatenated by multiplying to form a new transform matrix which transfers an image to the root (seed). See Figure 27.

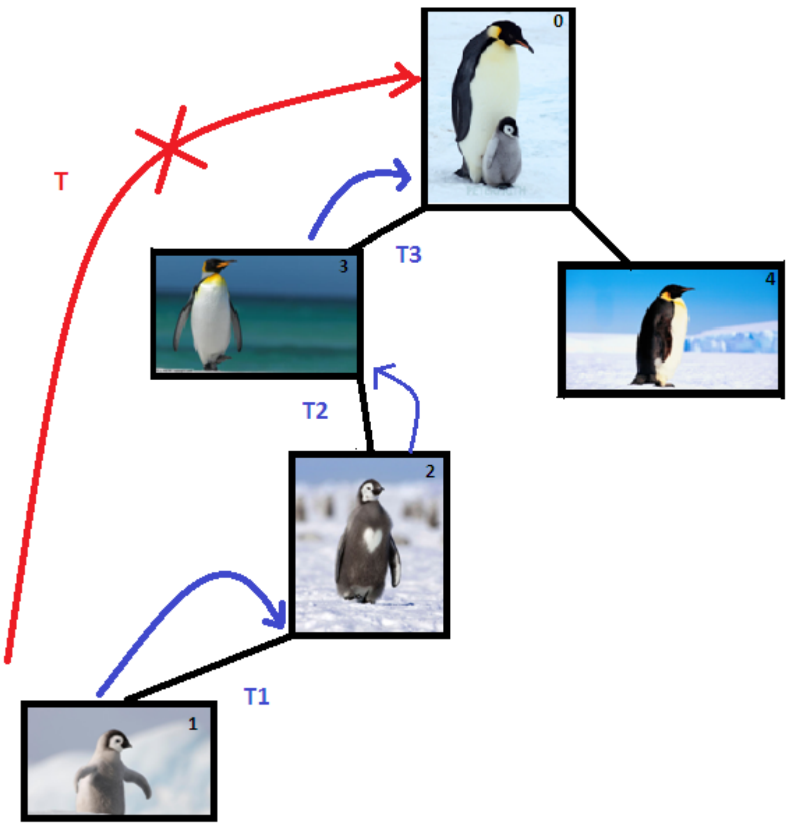


**Figure 27.** Example of step-wise MST alignment. Instead of directly transform node 1 to the root(node 0), the transformation is done step by step thorough the more similar images to the image 1.Transform function from node1 to root equals: $T = T_1 \times T_2 \times T_3$

The experimental part (Sec 7.4) outperforms the direct alignment using a single seed for same classes, but for some it is inferior. One reason could be that the MST algorithm at each stage picks the closest neighbor that does not cause a loop (cycle). Thus, it is possible that a node is not connected to its nearest neighbor if it already exists. For example in Fig. 28 , we have a full connected graph. Prim's algorithm can construct several MST from if the graph includes links having same weight such as Fig. 30. In this example, node $A$ is considered as root. The algorithm grows MST until it reaches an state where there are two edges with the same cost 3 (state $c$ in Fig. 29 ). The algorithm randomly picks on of them, and thus two MSTs can be generated such as Fig. 30(d.1), and (d.2). The point in these two MSTs is having different cost for node $D$ if it is going to be transformed step by step towards root ($A$). The cost of transformation for node $D$ in d.1 is higher than d.2, and even higher than direct transformation via the graph in Fig. 28.

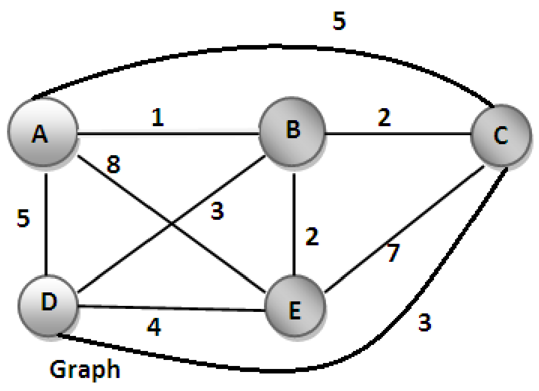


**Figure 28.** Full connected graph

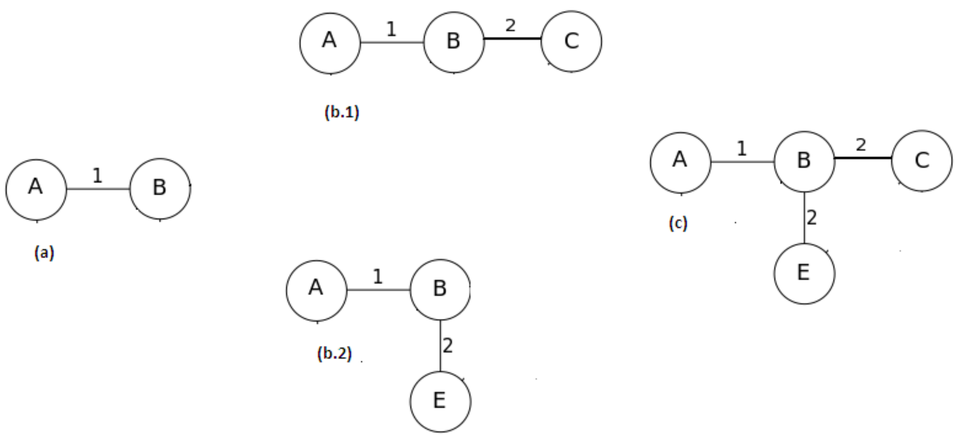


**Figure 29.** The steps of growing MST from root A.

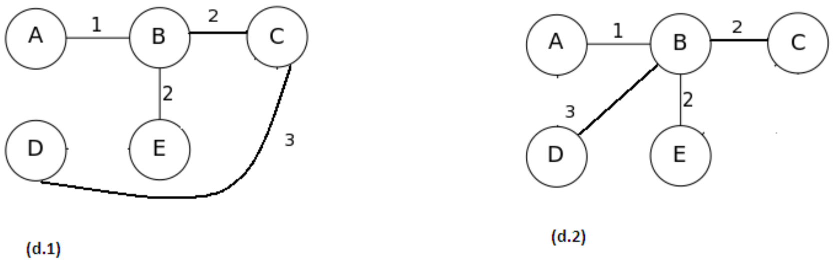


**Figure 30.** Two different structures of MST is created. Cost for transforming node $D$ towards $A$ in (d.1) is: 6, and in (d.2) is: 4.

## 6.2 Dijkstra approach

Another approach is to utilize the full graph and find the shortest path from each node to the seed by Dijkstra algorithm [71]. The implemented algorithm takes the optimal seed as the goal node and finds the shortest path from each candidate image to the seed by summing the weights of links between source (image) and the sink (seed) nodes.

The algorithm utilizes a data structure $Q$, which is a priority queue (node with lower weight has higher priority). For each node, fields named $w$, $Color$,$d$, and $\pi$ are considered and described as following:

- $w$: contains the weights of all the edges.
- $Color$: have one of these three possible values: WHITE, GRAY, BLACK
  - WHITE: indicates that the node is not found or visited.
  - GRAY: indicates the node is found, but is not processed completely.
  - BLACK: indicates the node is processed completely (i.e. all of the node's neighbors are investigated).
- $d$: indicates the distance from the $source$.
- $\pi$: indicates the node's neighbor from which the node is discovered.

. Dijkstra algorithm uses the algorithm RELAX. The relaxation of the edge ($u$, $v$) tests could the shortest path found to vertex $v$ be improved by routing its end through $u$ and does so if necessary [72]. The pseudo code algorithm [72] is given below:

**Algorithm 5** Dijkstra($source$, $w$, $seed$)

```
1:  In the beginning the fields of each vertex are $colour = WHITE$, $d = \infty$, $\pi$ = NULL
2:  Initialize vector $path \leftarrow$ NULL                                  #
    used to store visited node's ID along the shortes path
3:  $source \rightarrow colour = GRAY$
4:  $source \rightarrow d = 0$
5:  PUSH($Q$,$source$)
6:  while $Q \neq 0$ do
7:      $u$ = EXTRACT-MIN ($Q$)
8:      if $u = seed$ then
9:          $temp = source$
10:         while $temp \rightarrow \pi \neq$ NULL do
11:             include $temp \rightarrow \pi$ to vector $path$
12:             $temp = temp \rightarrow \pi$
13:         end while
14:         return $path$
15:         exit
16:     end if
17:     for each $v \in u \rightarrow Adj$ do
18:         if $v \rightarrow colour = WHITE$ then
19:             $v \rightarrow colour = GRAY$
20:             PUSH($Q$, $v$)
21:         end if
22:         RELAX ($u$, $v$, $w$)
23:     end for
24:     $u \rightarrow colour = BLACK$
25: end while
```

**Algorithm 6** RELAX($u$, $v$, $w$)

```
1: if $v \rightarrow d > u \rightarrow d + w\,(u, v)$ then
2:     $v \rightarrow d = u \rightarrow d + w\,(u, v)$
3:     $v \rightarrow \pi = u$
4: end if
```

### 6.2.1 Step-wise alignment via the shortest path

Each image is aligned to the seed by the found of Dijkstra similar to the MST. To do step-wise alignment, we need to know the found links between source(image) and sink(seed) that can be achieved by using vector $path$ which is the output of above algorithm. The $path$ vector contains the ID (e.g. image index, name) of visited nodes from source to seed. For example:

$$\begin{aligned} SourceID &= 8 \\ SeedID &= 17 \\ path &= \begin{bmatrix} 8 & 4 & 26 & 11 & 17 \end{bmatrix} \end{aligned} \tag{49}$$

$path$ vector shows that for transforming image8 to seed17, we need to find transformation between each pairs of visited nodes such as transformation from image 8 to image $4(T_{8-4})$. Therefore:

$$NewTransformmatrix = T_{8-4} \times T_{4-26} \times T_{26-11} \times T_{11-27} \tag{50}$$

# 7 EXPERIMENTS

We verified our seed selection procedure (Sec 5) and step-wise alignment(Sec 6) with images from in the caltech-101 [73]. As the results we provide average images and mean squared errors of annotated landmarks.

## 7.1 Caltech-101 data

We selected the following classes from Caltech-101: stop-sign, faces easy, airplane, starfish, car side, revolver, euphonium, motorbike, dollar bill, and another of motorbikes containing: scooters, harley davidson, and sport bikes. The reason for applying the special motorbike class wast to test the algorithm performance in case of having objects from the same class but with severe appearance variation.

## 7.2 Performance evaluation

We adopt the following measures: average images and landmark mean squared error(MSE) graph which have been used in [61].

### 7.2.1 Average images

We report average images with and without alignment. This provides qualitative performance of alignment. The average image is generated by first computing the mean size of all images and set as the standard position to make images centered. Finally, the average image is computed (see Fig. 31). For aligned images we first transform them to the seed space and the process is similar to previous (see Fig. 32).

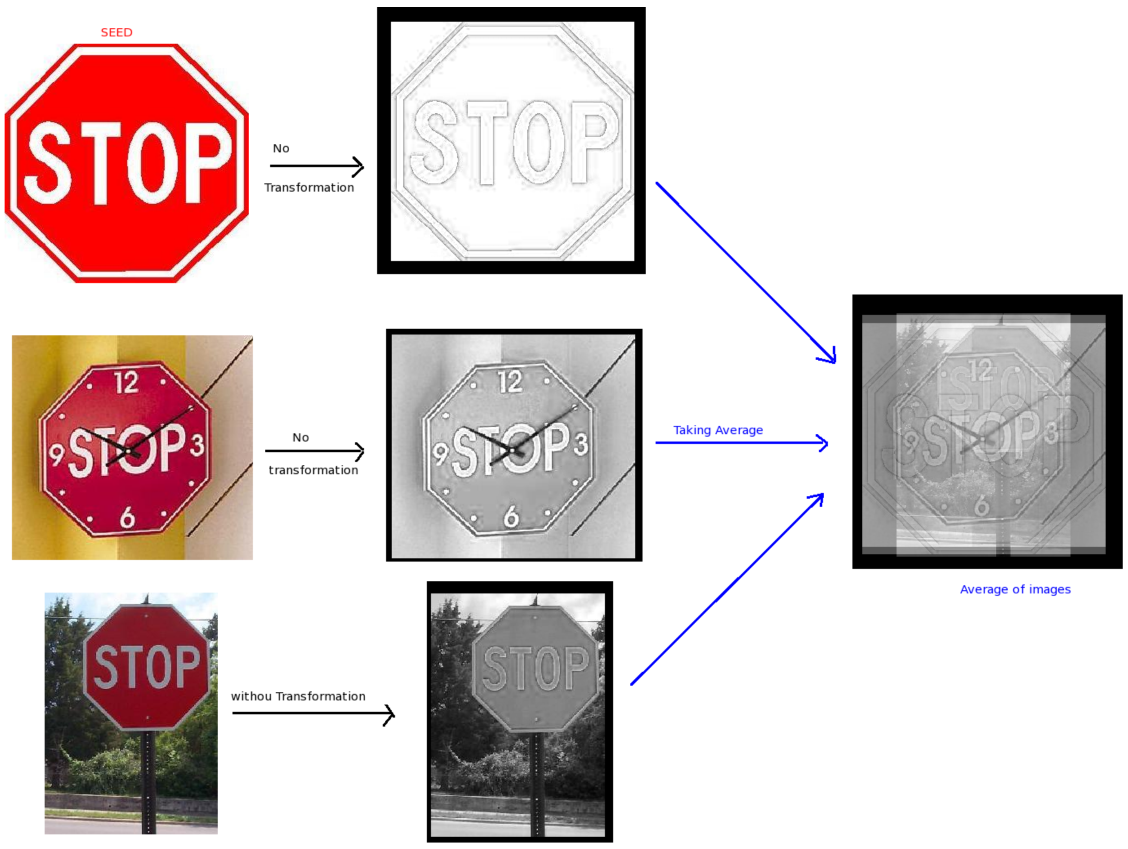


**Figure 31.** Example of average of three images from stop sign class without doing alignment or transferring images to the seed space. The result of averaging shows an obscure image of stop sign

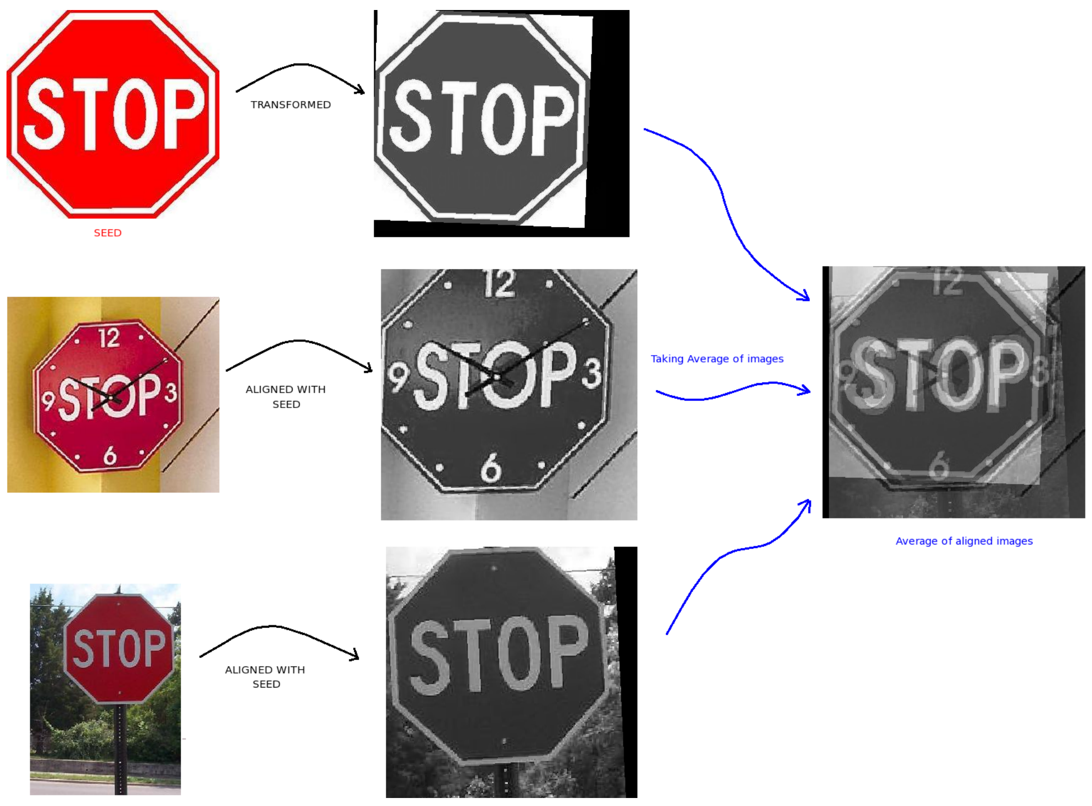


**Figure 32.** Example of average of three images from stop sign class with alignment. Images are first transformed to the seed (top image) space according to estimated similarity transformation. Then the average of transformed images is created. The average image shows a more clear image of stop sign.

### 7.2.2 Landmark mean squared error (MSE) graph

To quantitatively evaluate alignment we annotated landmarks to each category like Fig. 33, and computed their distances (according to Euclidean distance) after alignment. The alginment procedures is performed using the estimated similarity transformation in Alg.1. Then the generated landmarks MSE is compared to ground truth error. The ground truth point truth tells the best results that can be possibly achieved with the data set since ground truth points are selected manually, While landmarks are automatically chosen. Theoretically it is possible to get results which are more accurate than the ground truth, but then there must be some errors in the ground truth.

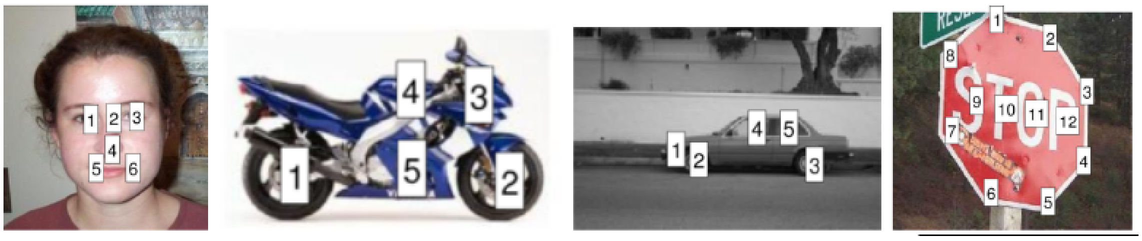


**Figure 33.** Caltech-101 images with annotated landmarks

Ground truth MSE is computed via below steps:

1. Extract ground truth point of the seed.
2. Repeat for each image from dataset:
   - Extract ground truth point in the image.
   - Calculate similarity transformation from ground truth point of the image to the seed's ground truth points.
   - Transfer seed's ground truth point using the calculated transformation.
   - Compute the Euclidean distance of seed's ground truth points before and after transformation.

## 7.3 Automatic seed selection result

We compare our method to supervised one (Sec 4.2) according to generated average image and landmark MSE for each method. The results are demonstrated in following.

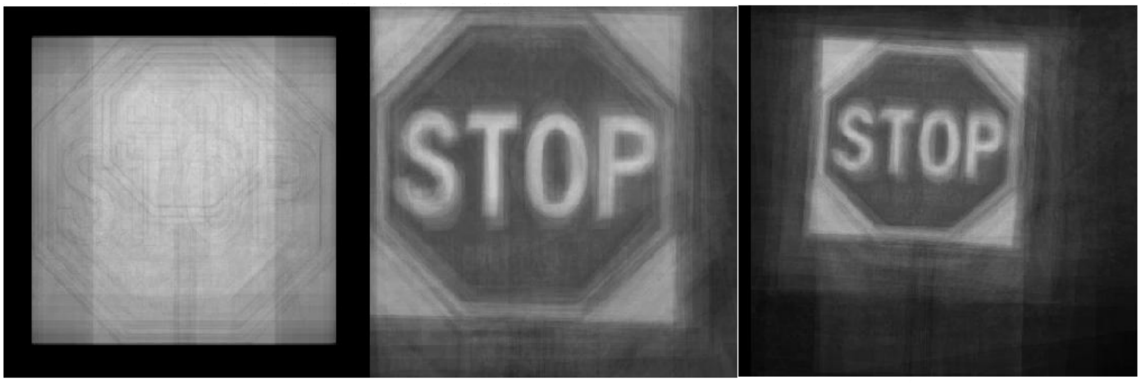


**Figure 34.** Results of average images for the class of stop sign.left: without alignment, middle: supervised method, right: unsupervised seed selection

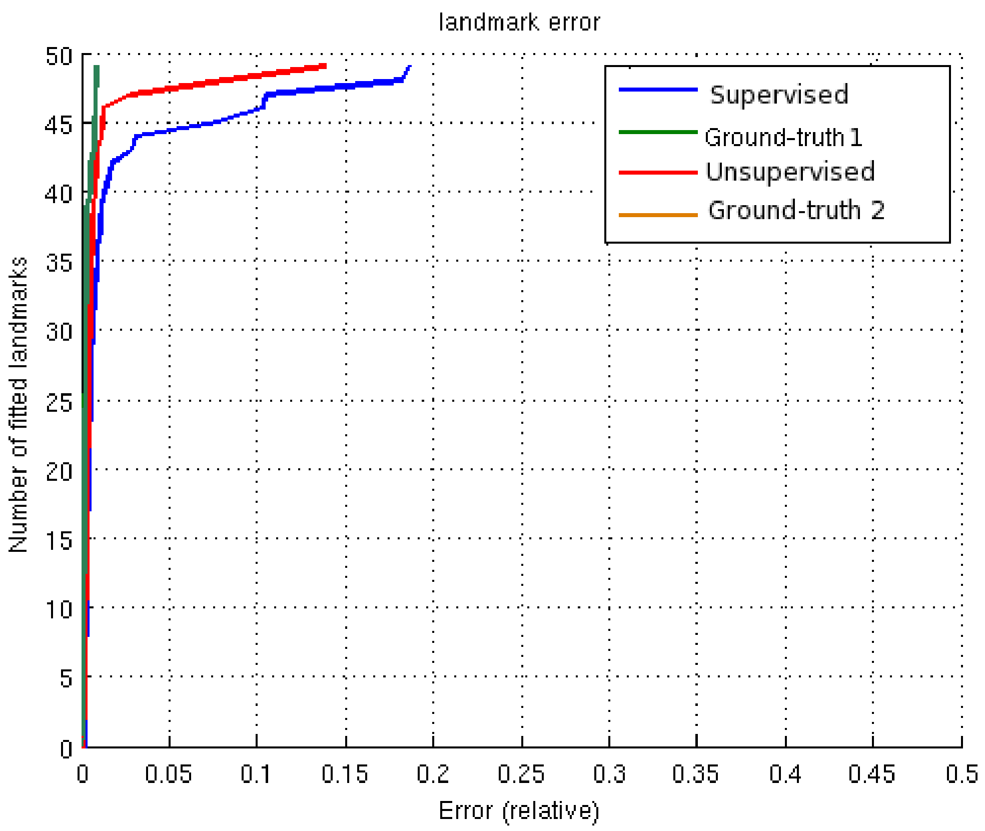


**Figure 35.** Quantitative results for the alignments in Fig. 34. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(red curve, light green curve), our method (blue curve, and green curve). In this figure, the ground truth 1 and 2 are overlapped that is why the ground truth 2 (orange curve) can not be seen.

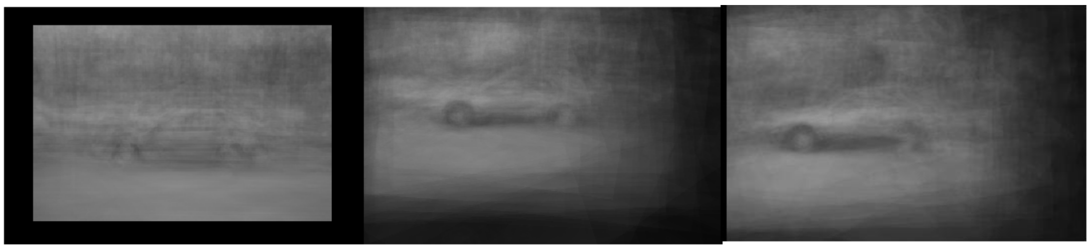


**Figure 36.** Results of average images for the class of car side.left: without alignment, middle: supervised method, right: unsupervised seed selection

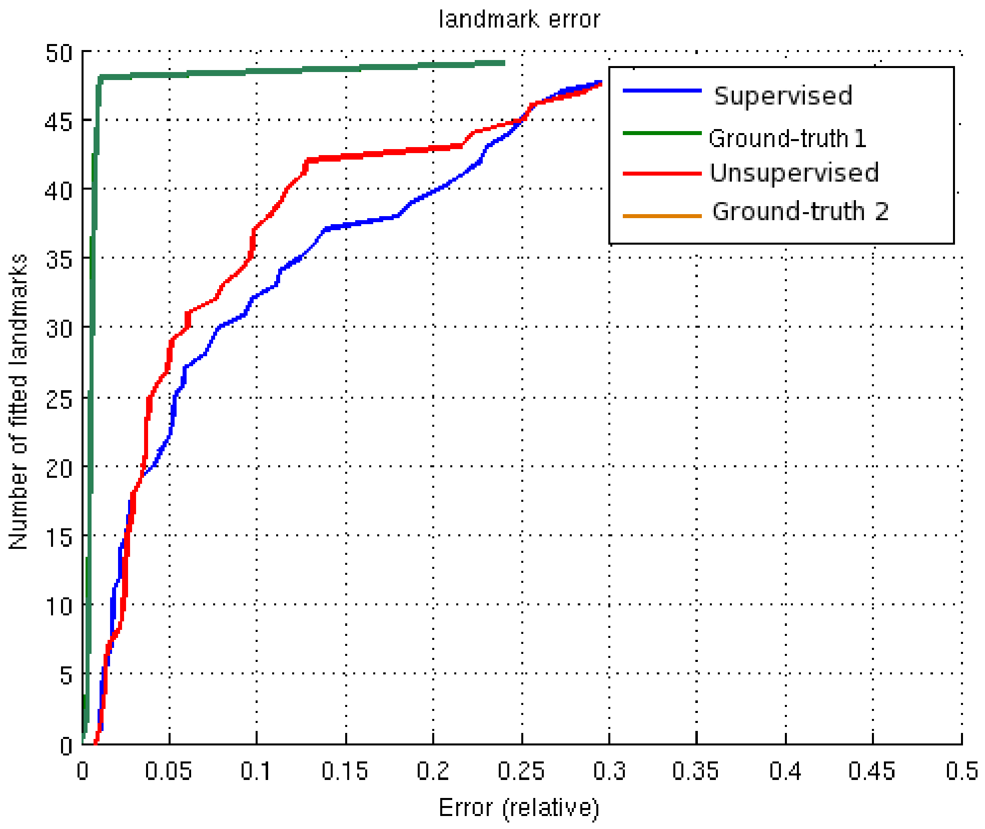


**Figure 37.** Quantitative results for the alignments in Fig. 36. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(red curve, light green curve), our method (blue curve, and green curve). In this figure, the ground truth 1 and 2 are overlapped that is why the ground truth 2 (orange) can not be seen.

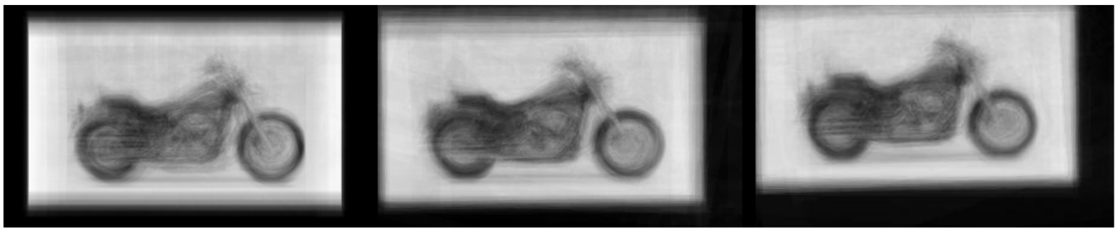


**Figure 38.** Results of average images for the class of Motorbike. left: without alignment, middle: supervised method, right: unsupervised seed selection

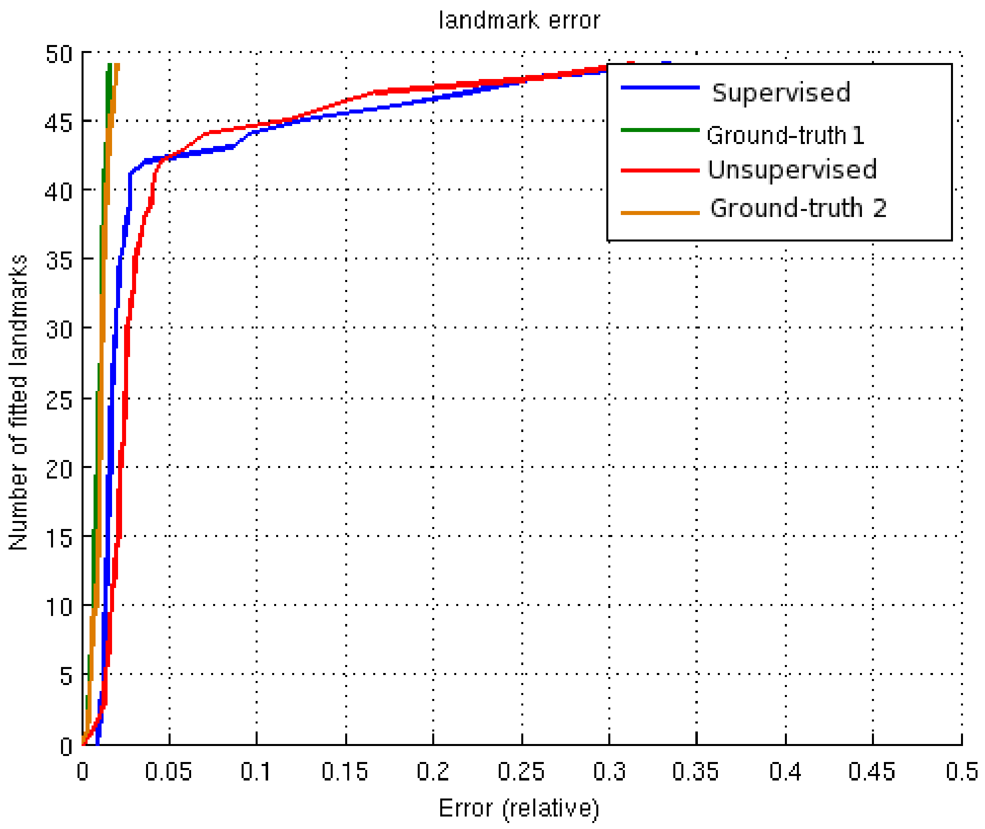


**Figure 39.** Quantitative results for the alignments in Fig. 38. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(blue curve, 1 green curve), our method (red, and orange curves)

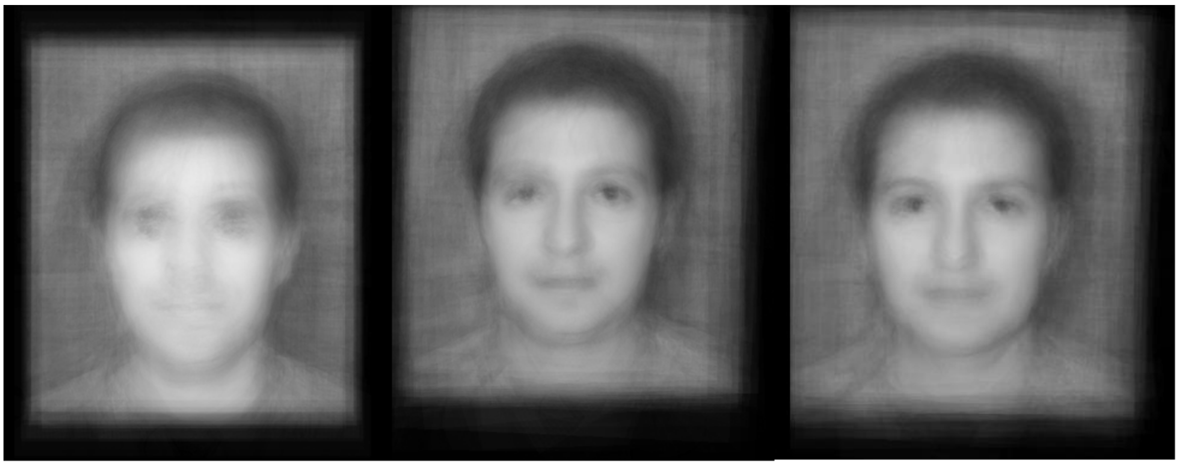


**Figure 40.** Results of average images for the class of face. left: without alignment, middle: supervised method, right: unsupervised seed selection

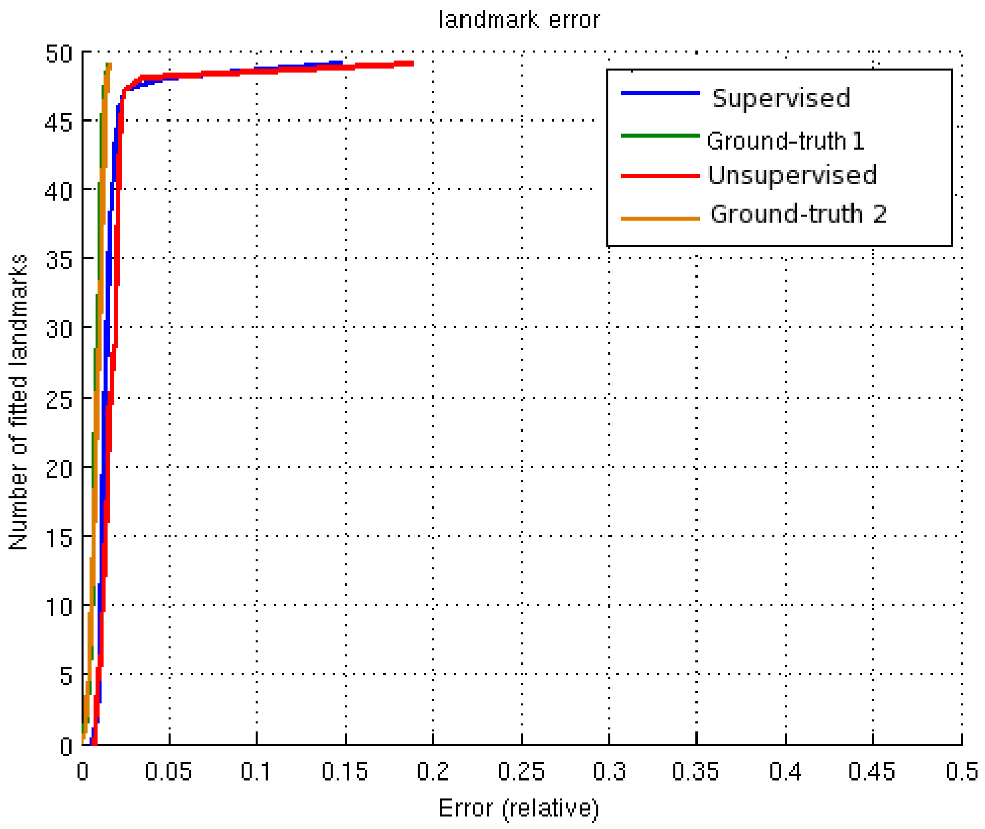


**Figure 41.** Quantitative results for the alignments in Fig. 40. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(blue curve, 1 green curve), our method (red, and orange curves)

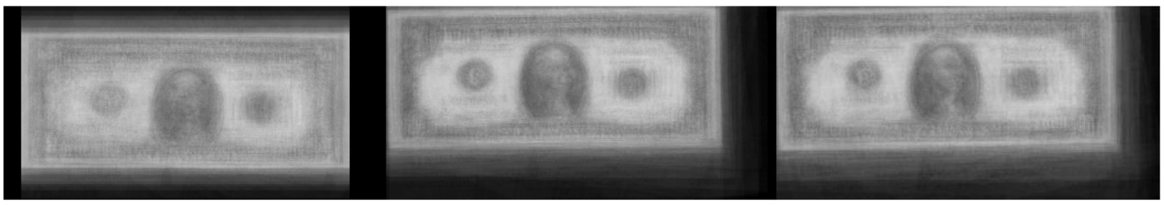


**Figure 42.** Results of average images for the class of dollar. left: without alignment, middle: supervised method, right: unsupervised seed selection

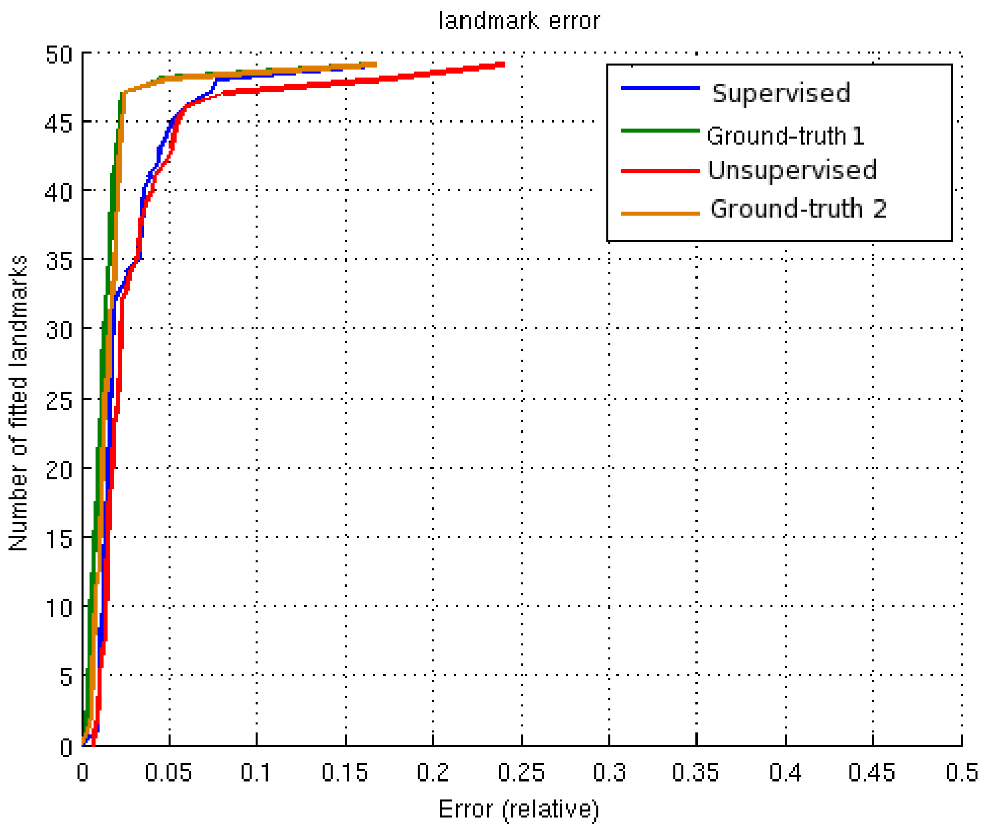


**Figure 43.** Quantitative results for the alignments in Fig. 42. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(blue curve, 1 green curve), our method (red, and orange curves)

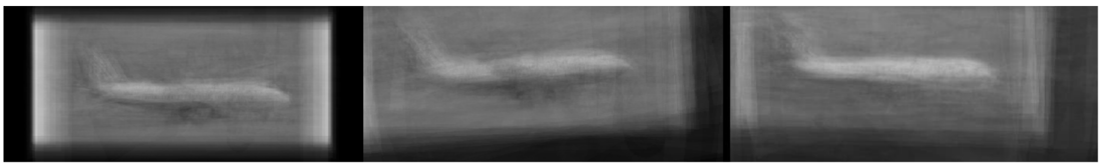


**Figure 44.** Results of average images for the class of airplane. left: without alignment, middle: supervised method, right: unsupervised seed selection

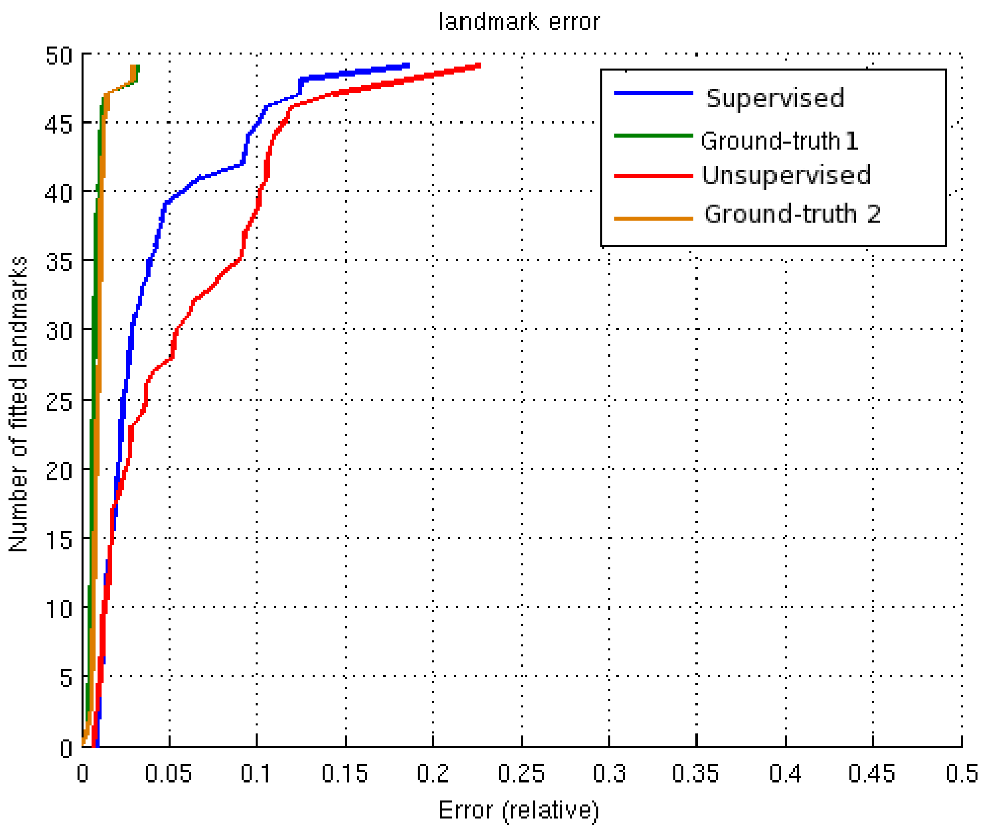


**Figure 45.** Quantitative results for the alignments in Fig. 44. Graphs represent cumulative error curves for feature-based alignment method with supervised seed (blue curve, 1 green curve), our method (red, and orange curves)

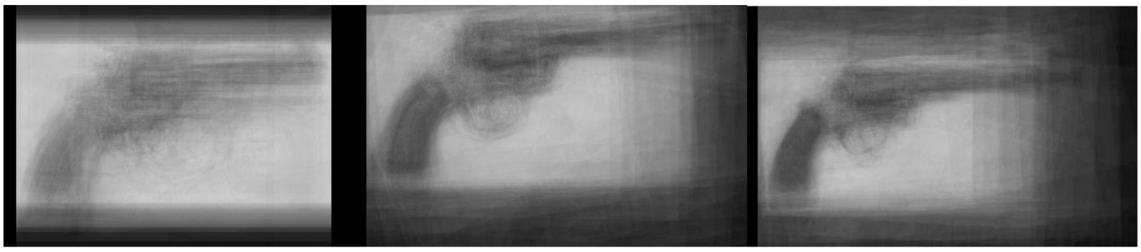


**Figure 46.** Results of average images for the class of revolver. left: without alignment, middle: supervised method, right: unsupervised seed selection

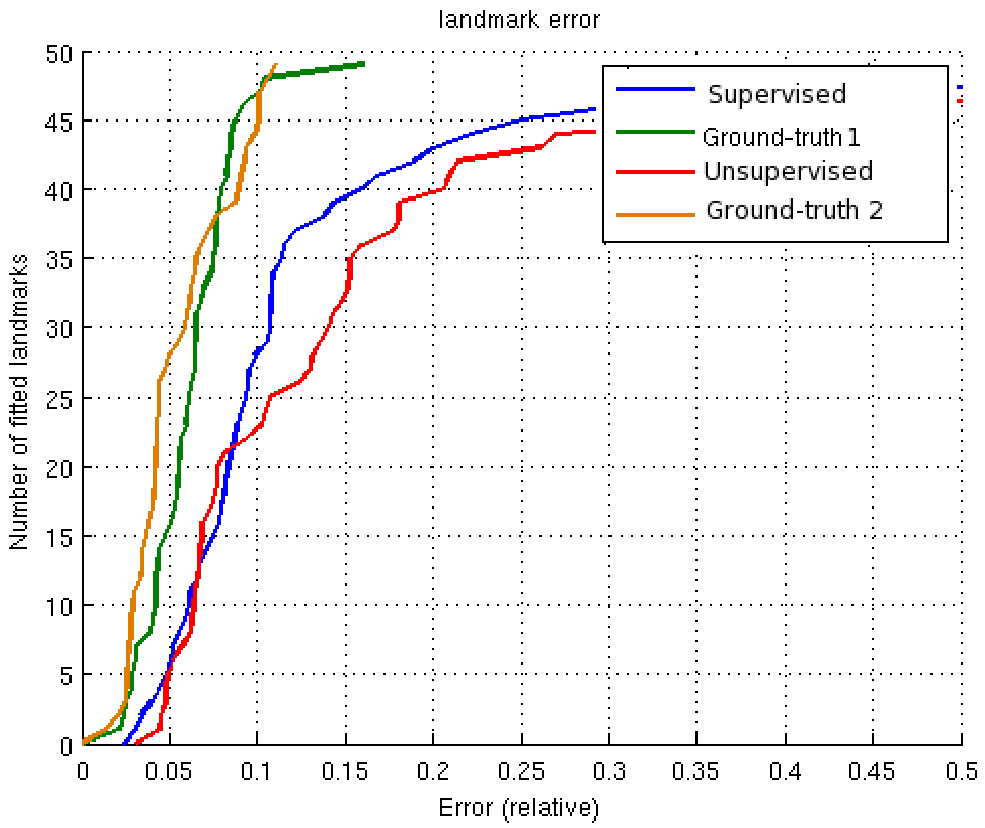


**Figure 47.** Quantitative results for the alignments in Fig. 46. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(blue curve, 1 green curve), our method (red, and orange curves)

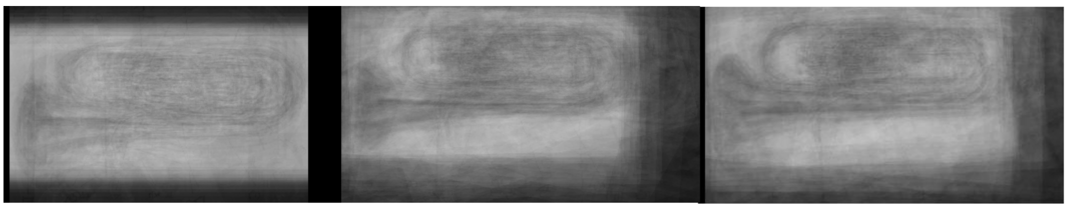


**Figure 48.** Results of average images for the class of euphoniom. left: without alignment, middle: supervised method, right: unsupervised seed selection

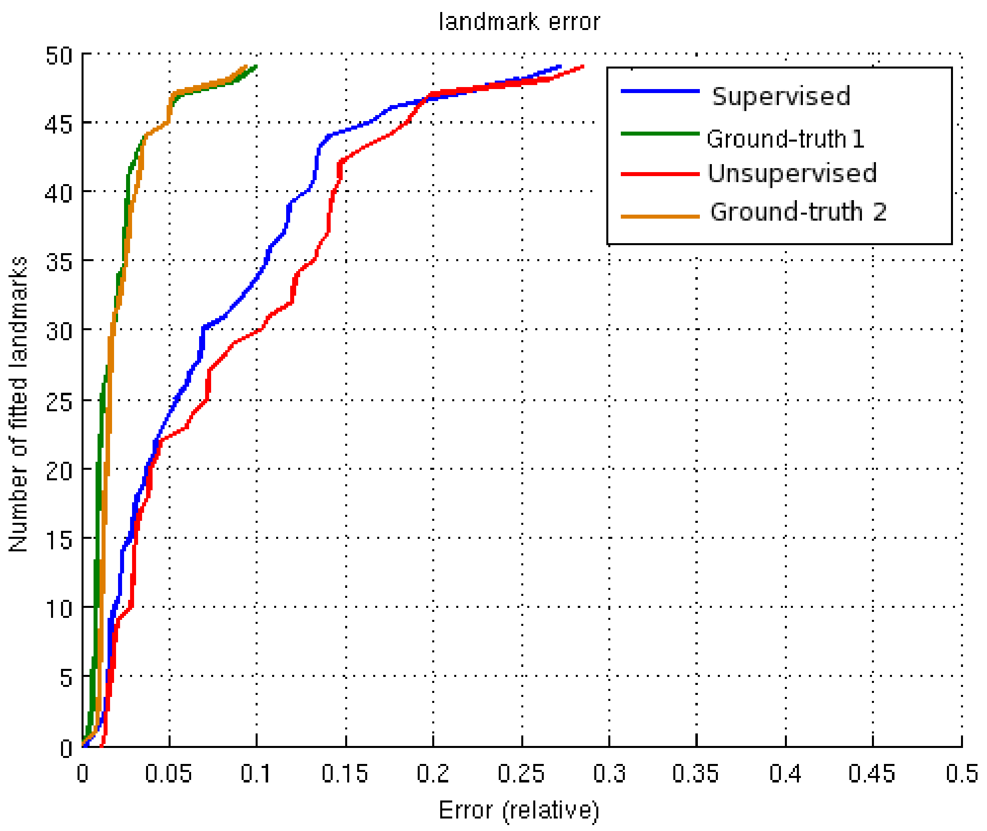


**Figure 49.** Quantitative results for the alignments in Fig. 48. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(blue curve, 1 green curve), our method (red, and orange curves)

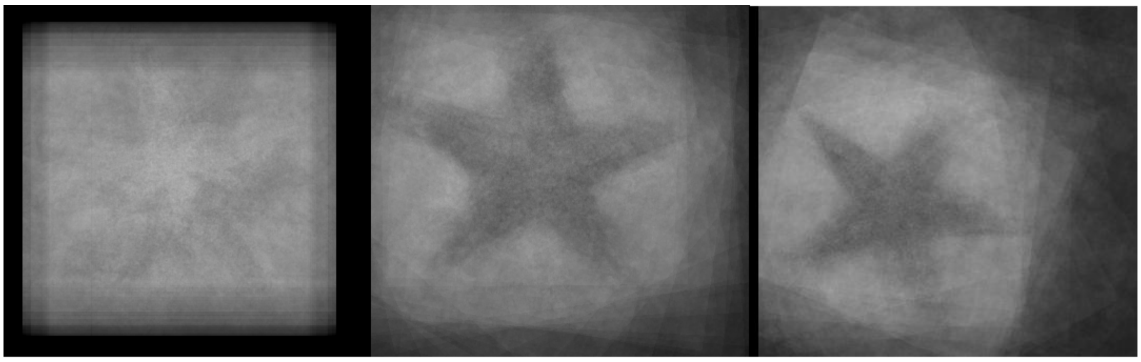


**Figure 50.** Results of average images for the class of star fish. left: without alignment, middle: supervised method, right: unsupervised seed selection

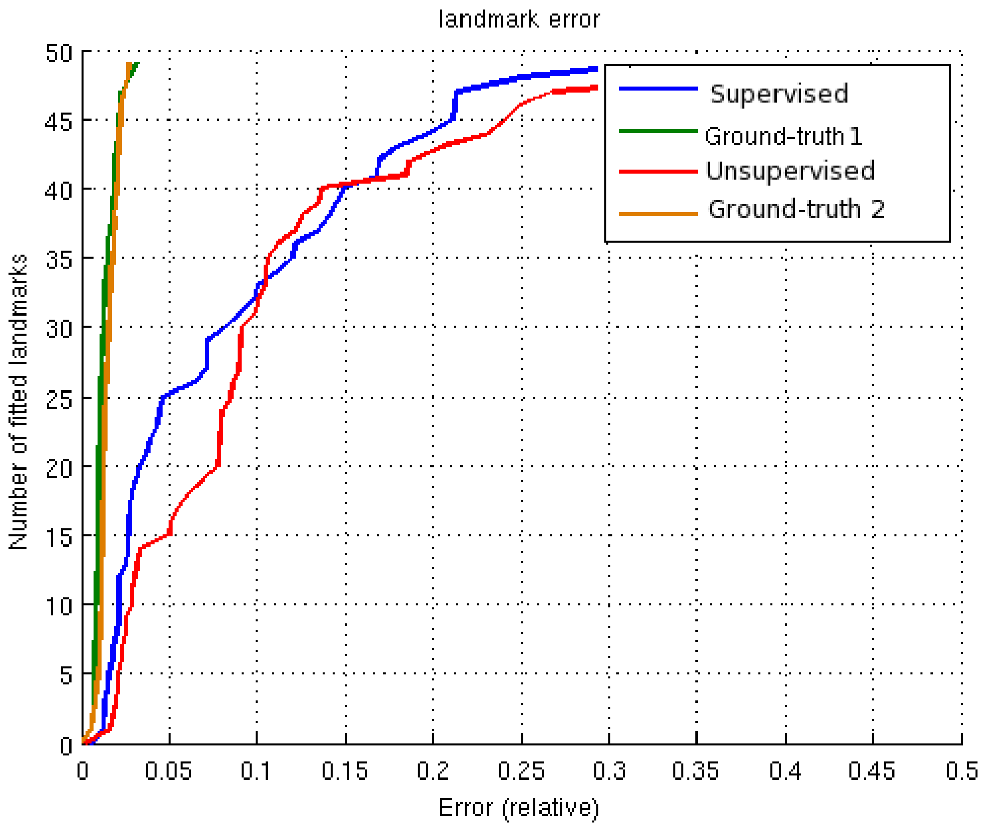


**Figure 51.** Quantitative results for the alignments in Fig. 50. Graphs represent cumulative error curves for feature-based alignment method with supervised seed selection(blue curve, 1 green curve), our method (red, and orange curves)

Considering all of the results, the unsupervised seed selection outperforms the supervised method for the classes of stop sign, car side, motorbike, face. For instance, the number of fitted landmarks in car side class is 28 and 21 for unsupervised and supervised methods respectively. Both methods have close performance for euphonium and dollar bill. It seems that The performance of our method drops down by increasing the difficulty of the class type since the landmarks MSE of the classes of airplane, revolver, and star fish in supervised method are much closer to the ground truth curve. The performance degradation for these classes is unexpected because the scores presented in Table 1 demonstrated that the selected seed for these classes have higher scores than the manually selected seed.

## 7.4 Step-wise alignment result

This section presents average images and landmark MSE resulted from both using MST and Dijkstra shortest path. The average images and landmark MSE of directed alignment and step wise alignment methods are compared to each other.

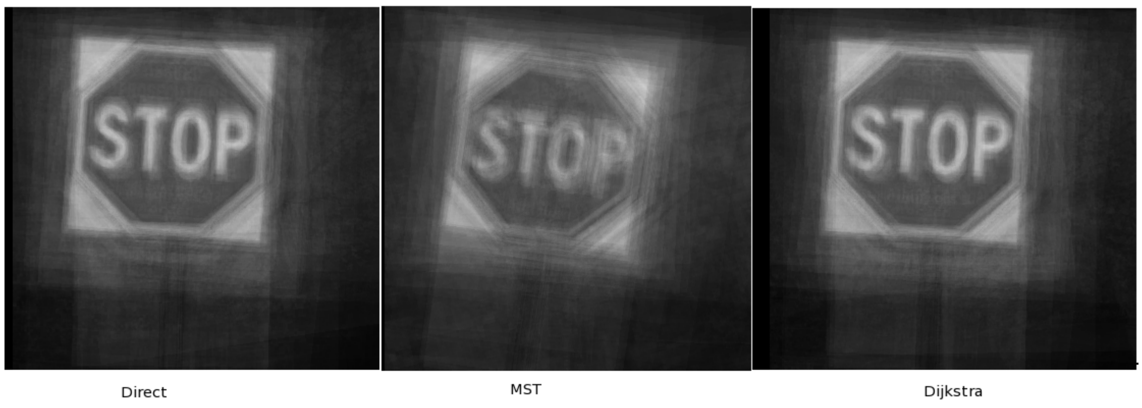


**Figure 52.** Results of average images for the class of stop sign. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

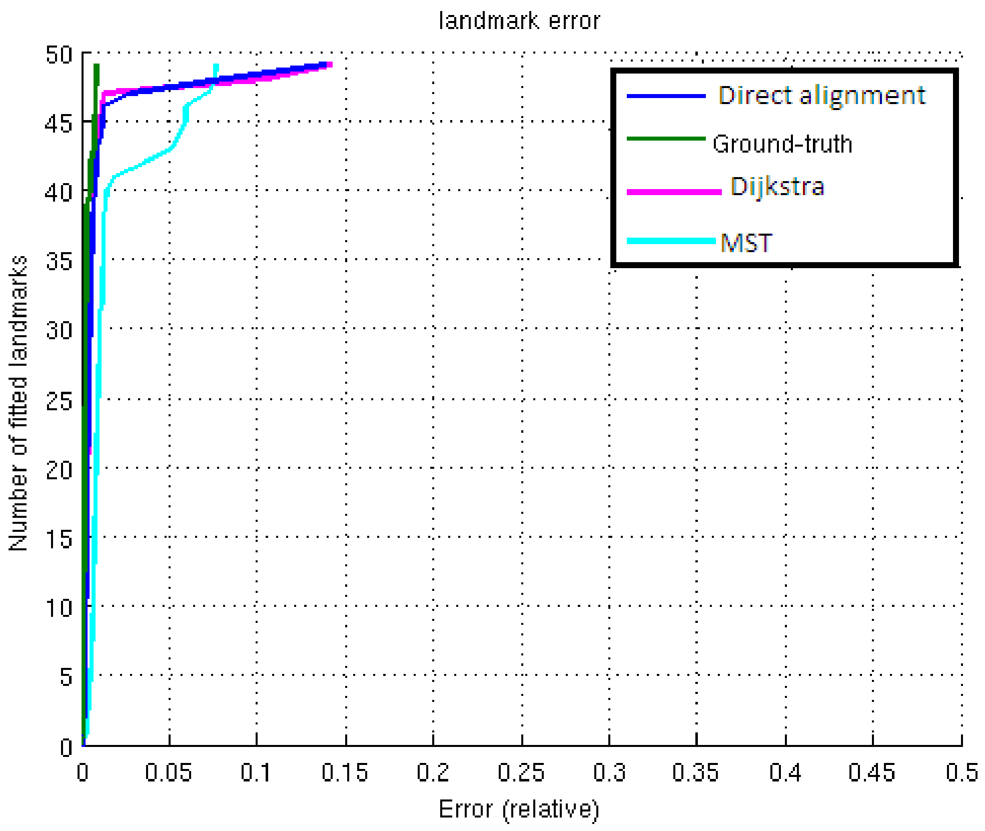


**Figure 53.** Quantitative results for the alignments in Fig. 52. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

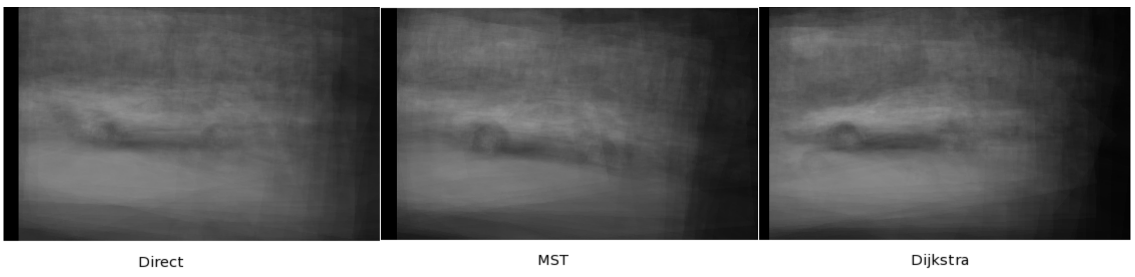


**Figure 54.** Results of average images for the class of car side. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

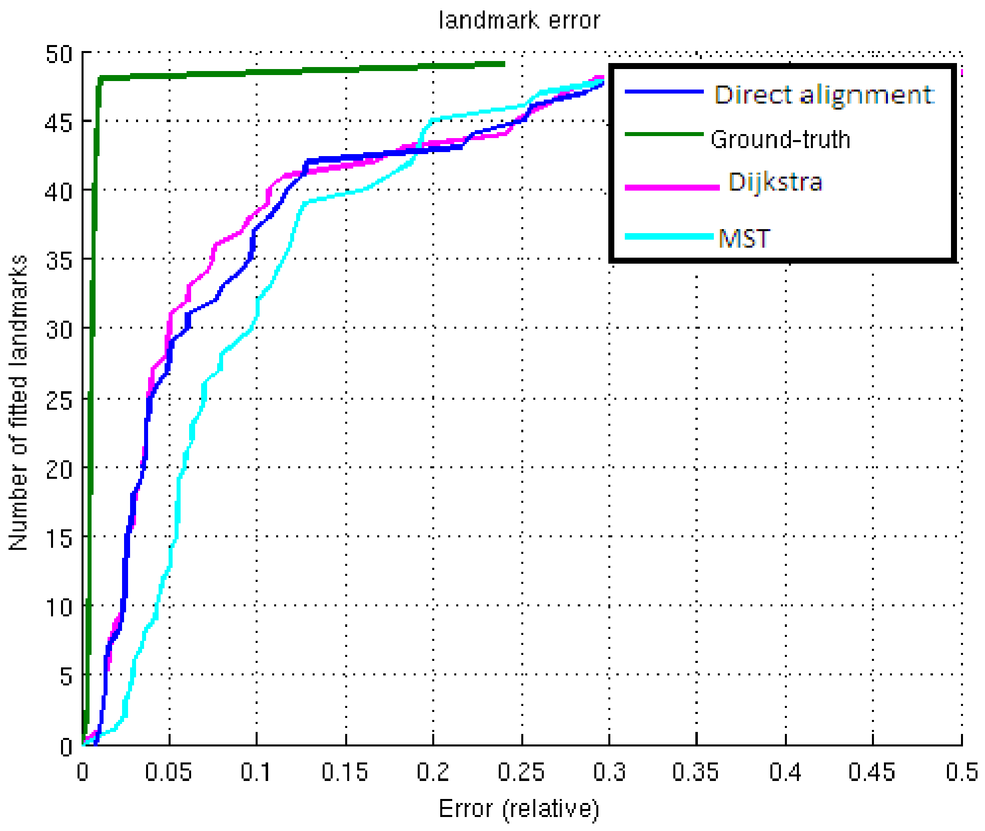


**Figure 55.** Quantitative results for the alignments in Fig. 54. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

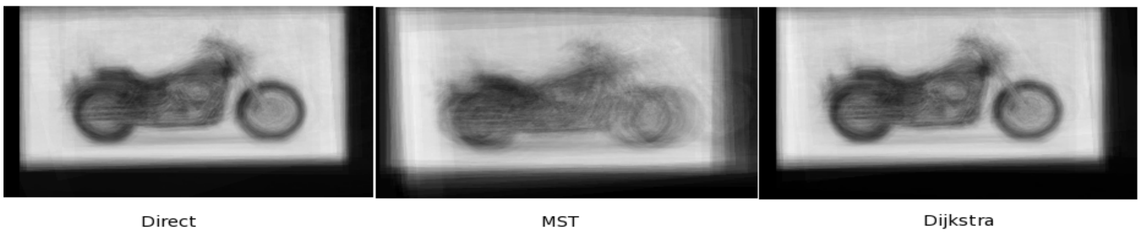


**Figure 56.** Results of average images for the class of motorbike. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

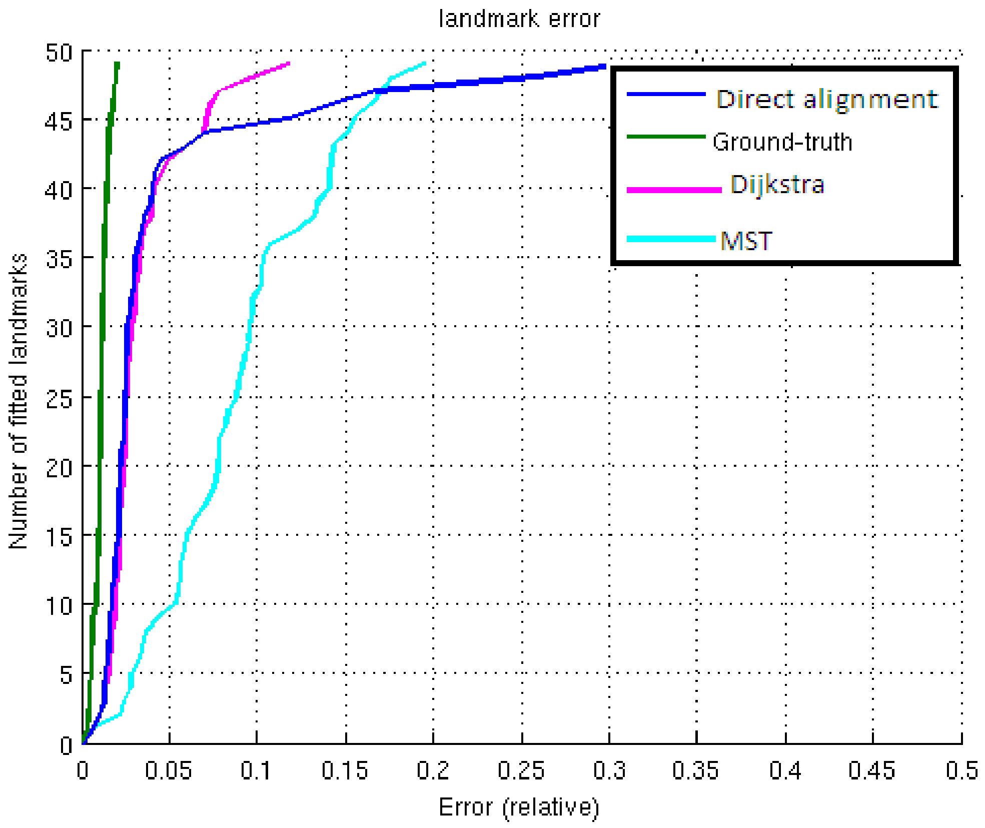


**Figure 57.** Quantitative results for the alignments in Fig. 56. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

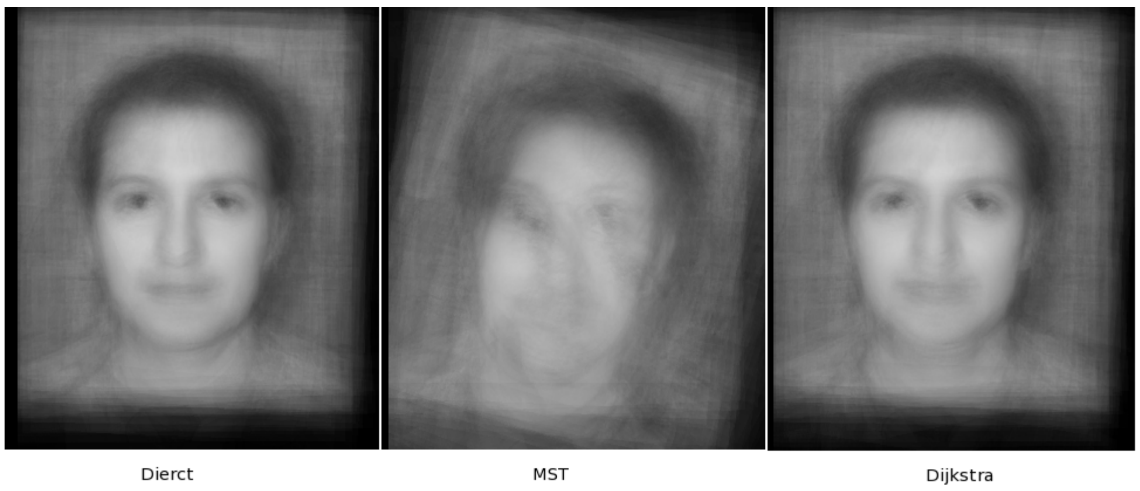


**Figure 58.** Results of average images for the class of face. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

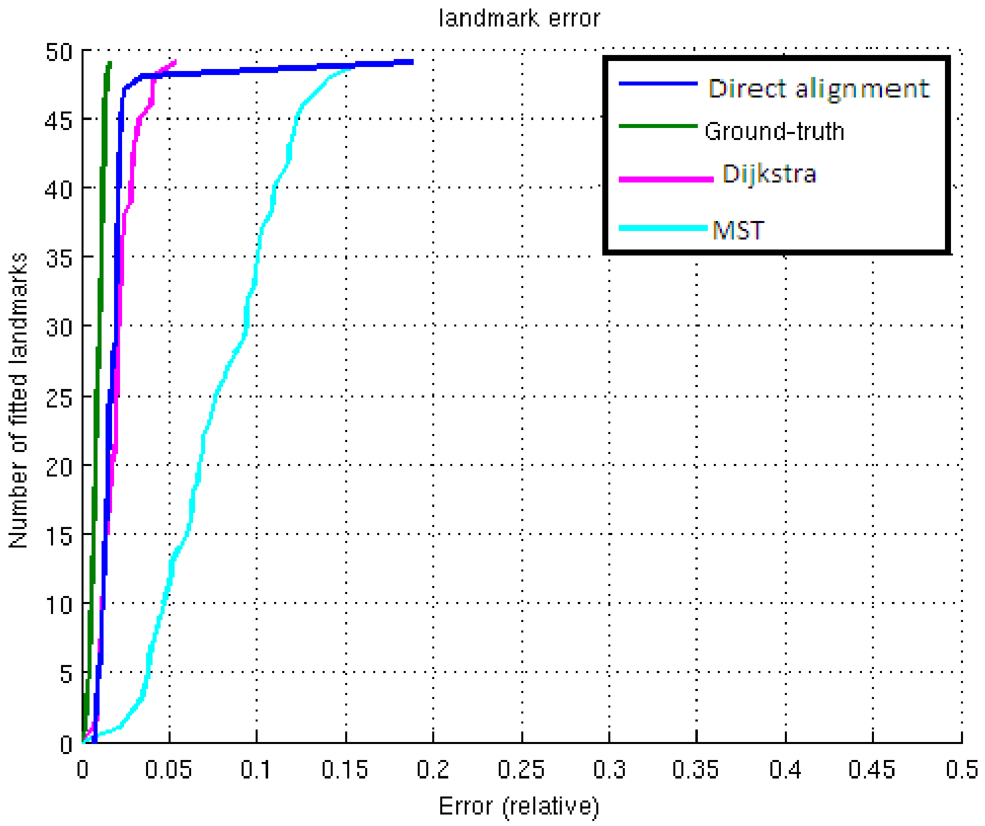


**Figure 59.** Quantitative results for the alignments in Fig. 58. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

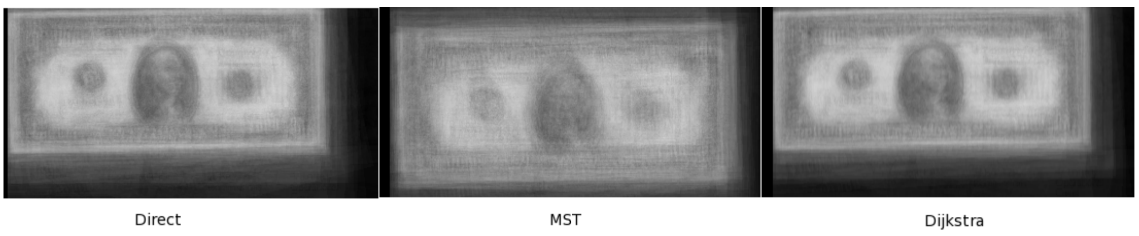


**Figure 60.** Results of average images for the class of dollar. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

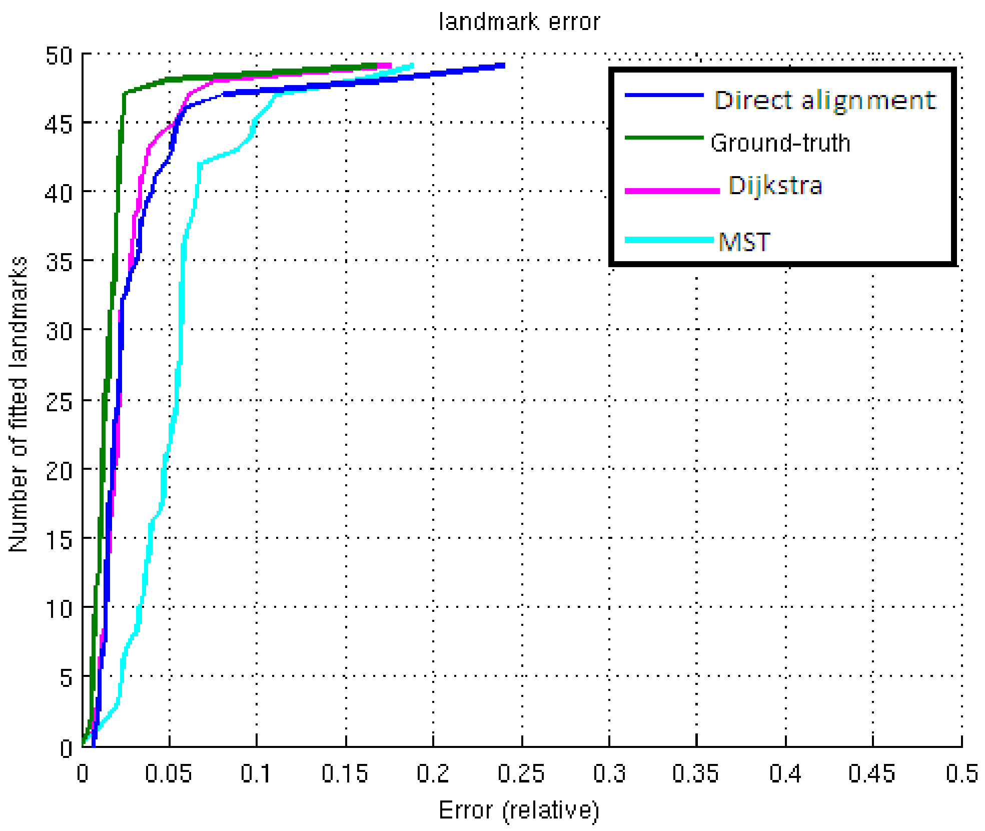


**Figure 61.** Quantitative results for the alignments in Fig. 60. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

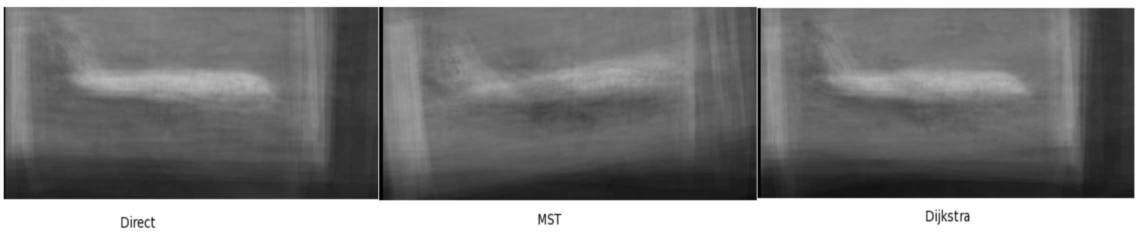


**Figure 62.** Results of average images for the class of airplane. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

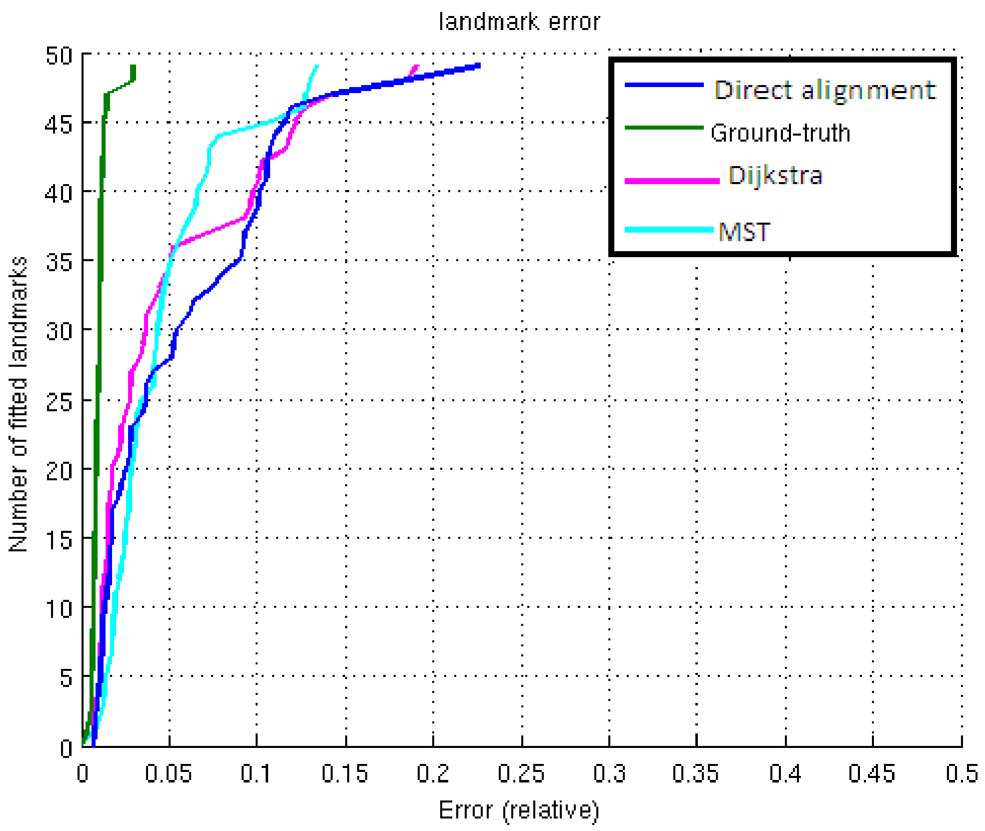


**Figure 63.** Quantitative results for the alignments in Fig. 62. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

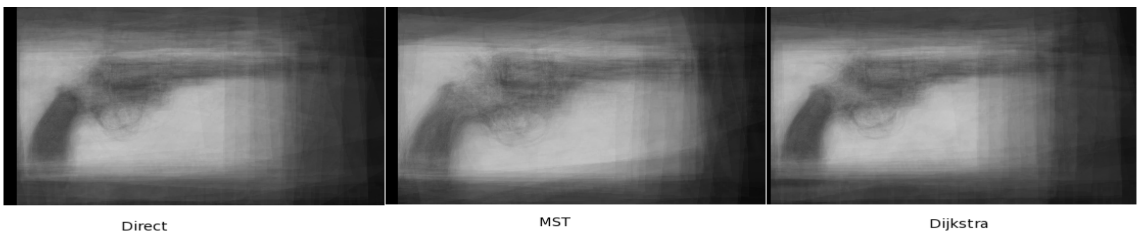


**Figure 64.** Results of average images for the class of revolver. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

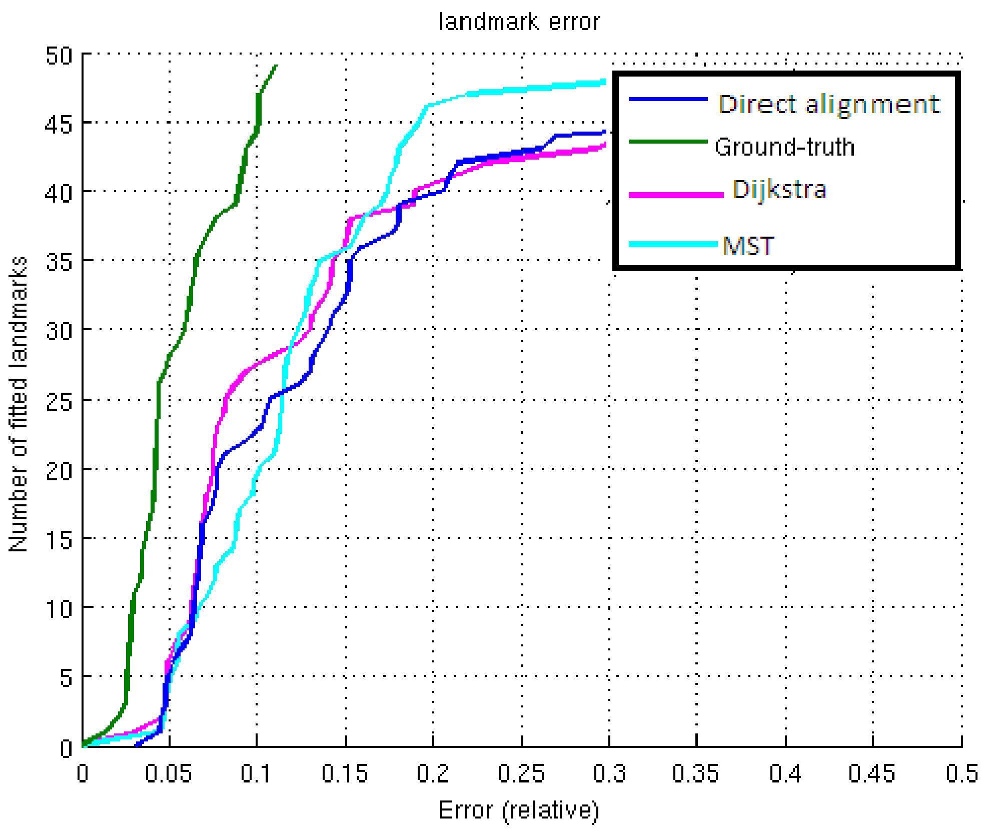


**Figure 65.** Quantitative results for the alignments in Fig. 64. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

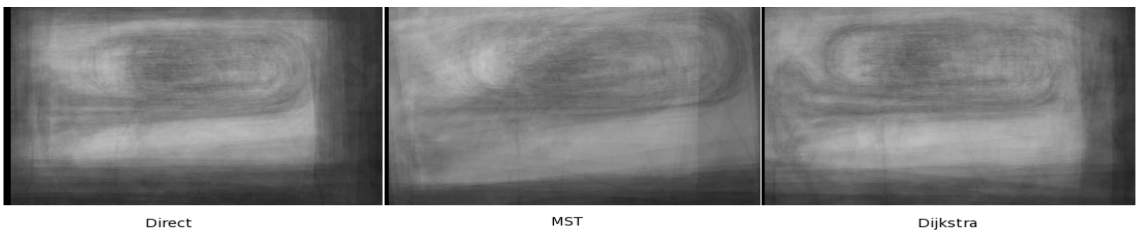


**Figure 66.** Results of average images for the class of euphonium. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

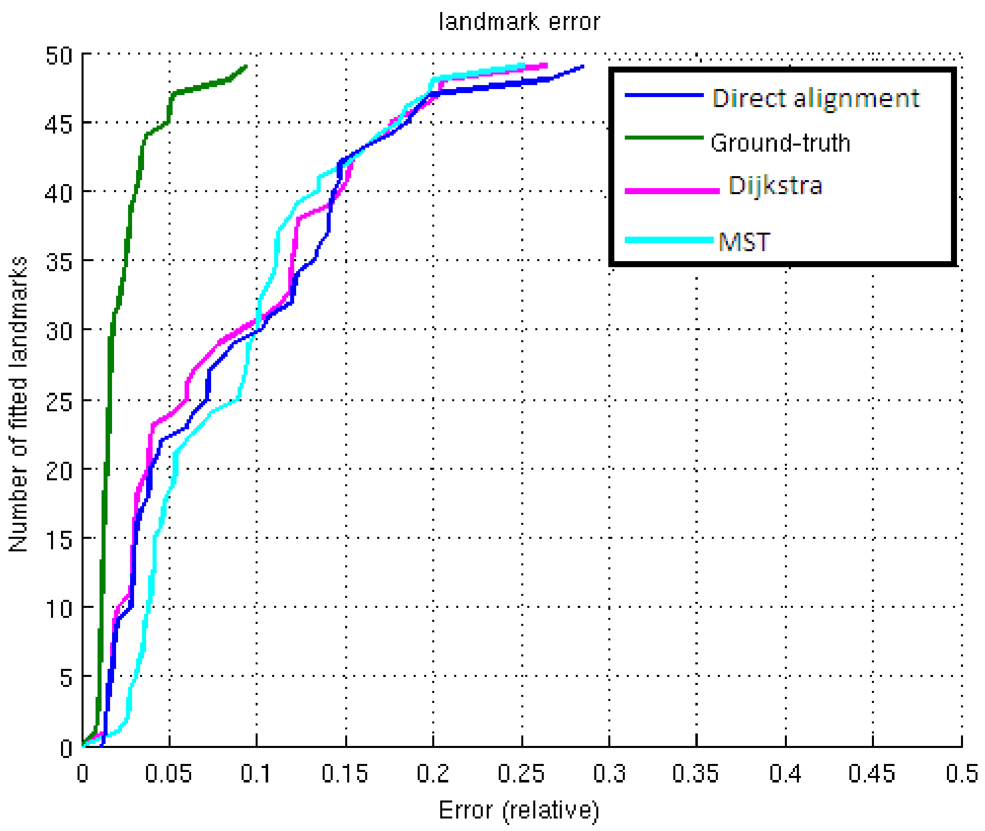


**Figure 67.** Quantitative results for the alignments in Fig. 66. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

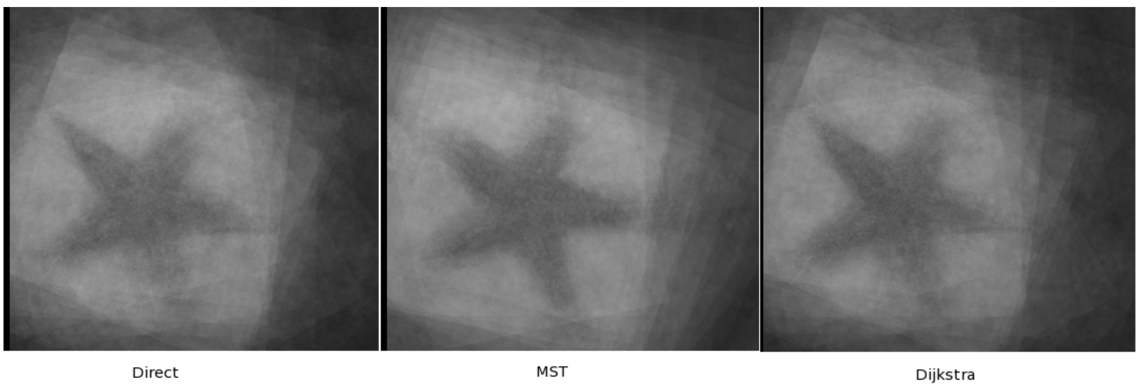


**Figure 68.** Results of average images for the class of star fish. left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

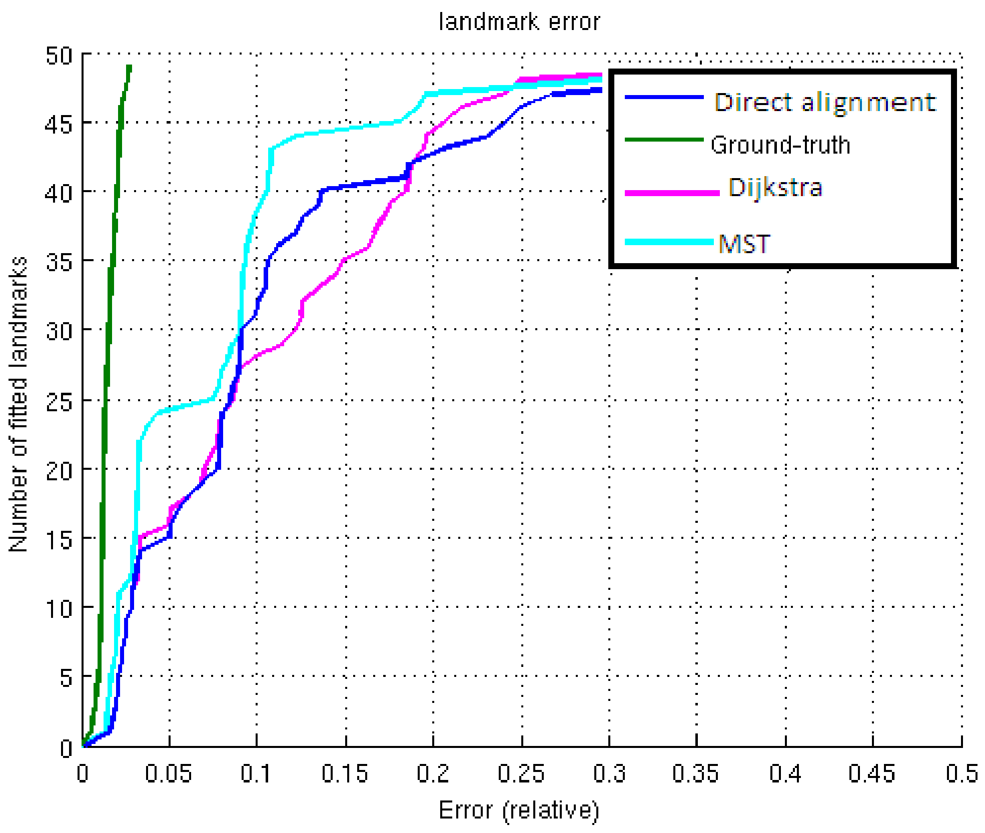


**Figure 69.** Quantitative results for the alignments in Fig. 68. Graphs represent cumulative error curves for direct alignment of images (blue curve), MST method (cyan), and Dijkstra method (magenta)

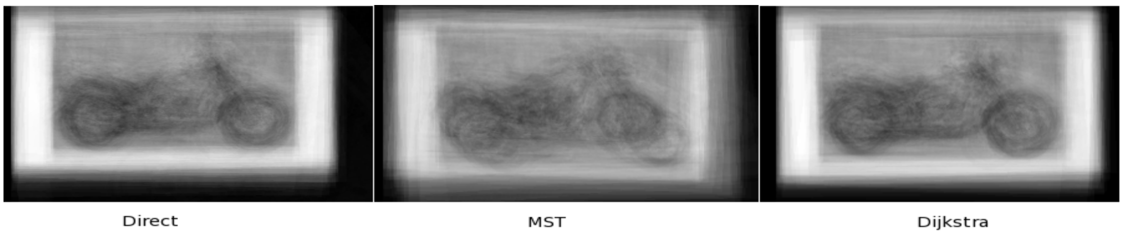


**Figure 70.** Results of average images for the class of 3 types of motorbikes(sport bike, scooter, Harley davidson). left: Direct alignment based on unsupervised seed selection, middle: MST, right: Dijkstra

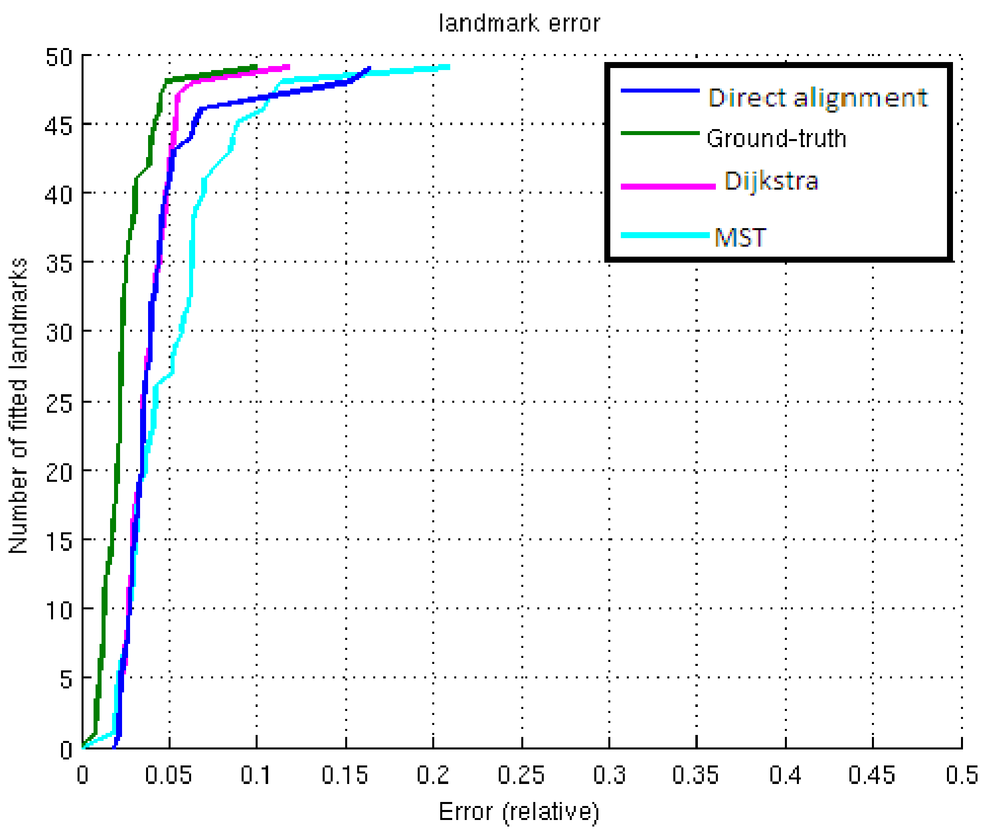


**Figure 71**

It is interpreted from the results that the alignment is improved through step-wise method specifically by utilizing Dijkstra shortest path method since the number of fitted landmarks for all classes except star fish is higher that the direct alignment and MST method. According to the landmark MSE curves, the MST method only shows a high performance for the class of star fish and airplane, while for other classes it is even lower than the direct alignment (the reason could be the one mentioned in Sec 6.1.2).

We also analyzed the accuracy of our method by transforming each image to the optimal seed under direct alignment, MST and Dijkstra methods. The results showed that the number of failed images via step-wise alignment is less than direct alignment. In addition, some images which have been failed through MST were corrected through Dijkstra and vice versa. One example is shown in following figures:

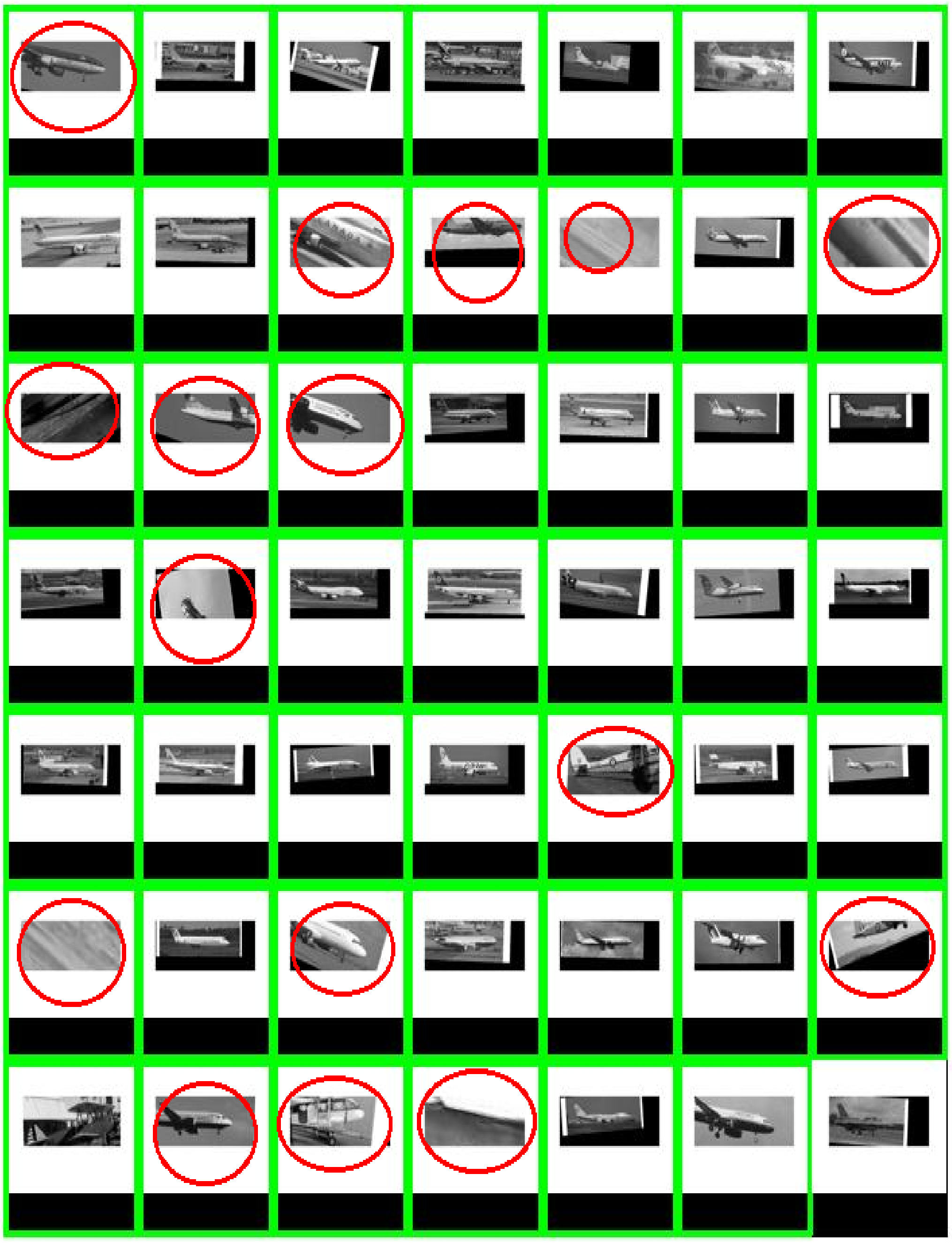


**Figure 72.** Class of airplane, Transformed images via direct alignment. Red circles show failed images after alignment

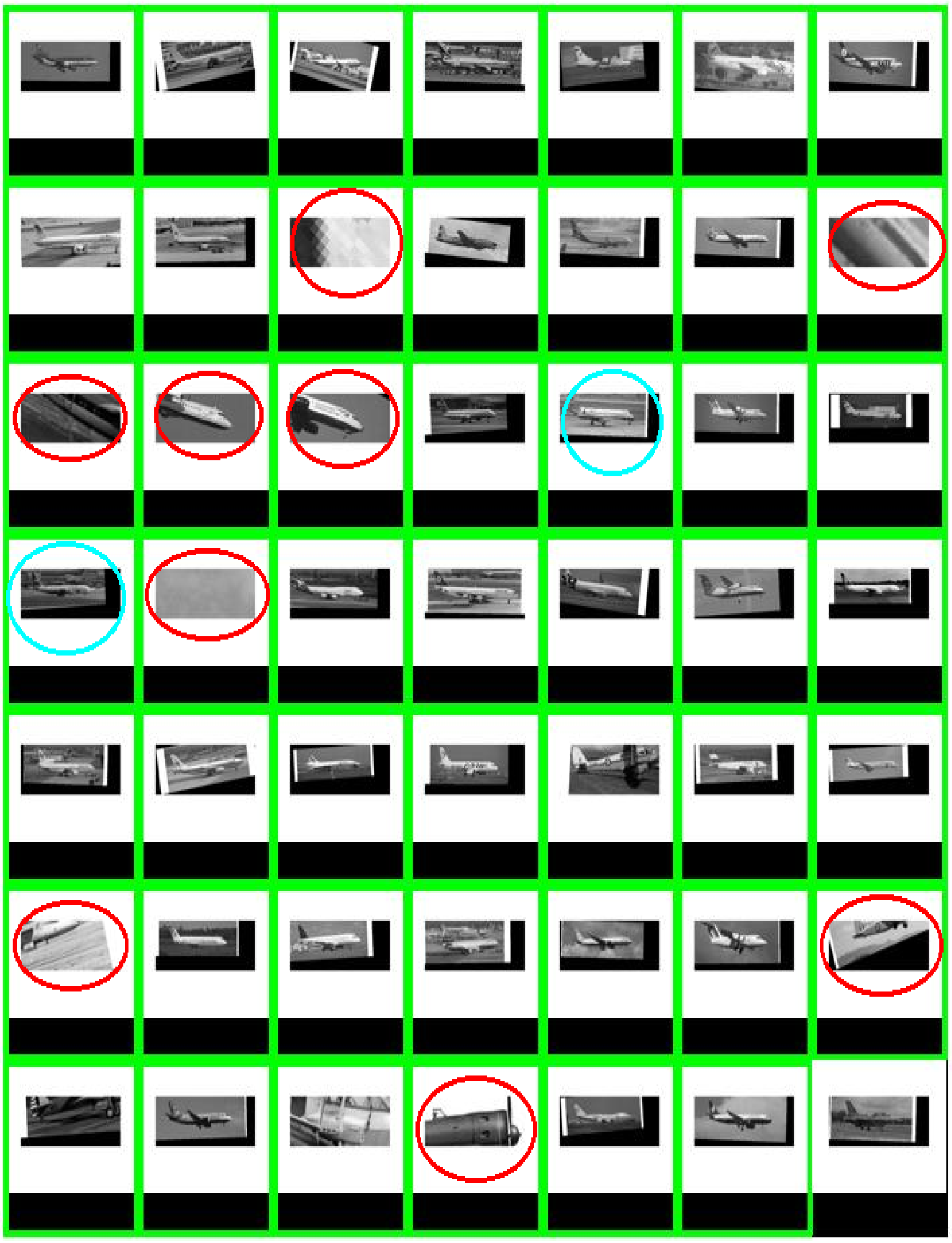


**Figure 73.** Class of airplane, Transformed images via alignment using Dijkstra shortest path method. Red circles show failed images after alignment and Cyan circles show images failed via MST but corrected through Dijkstra.

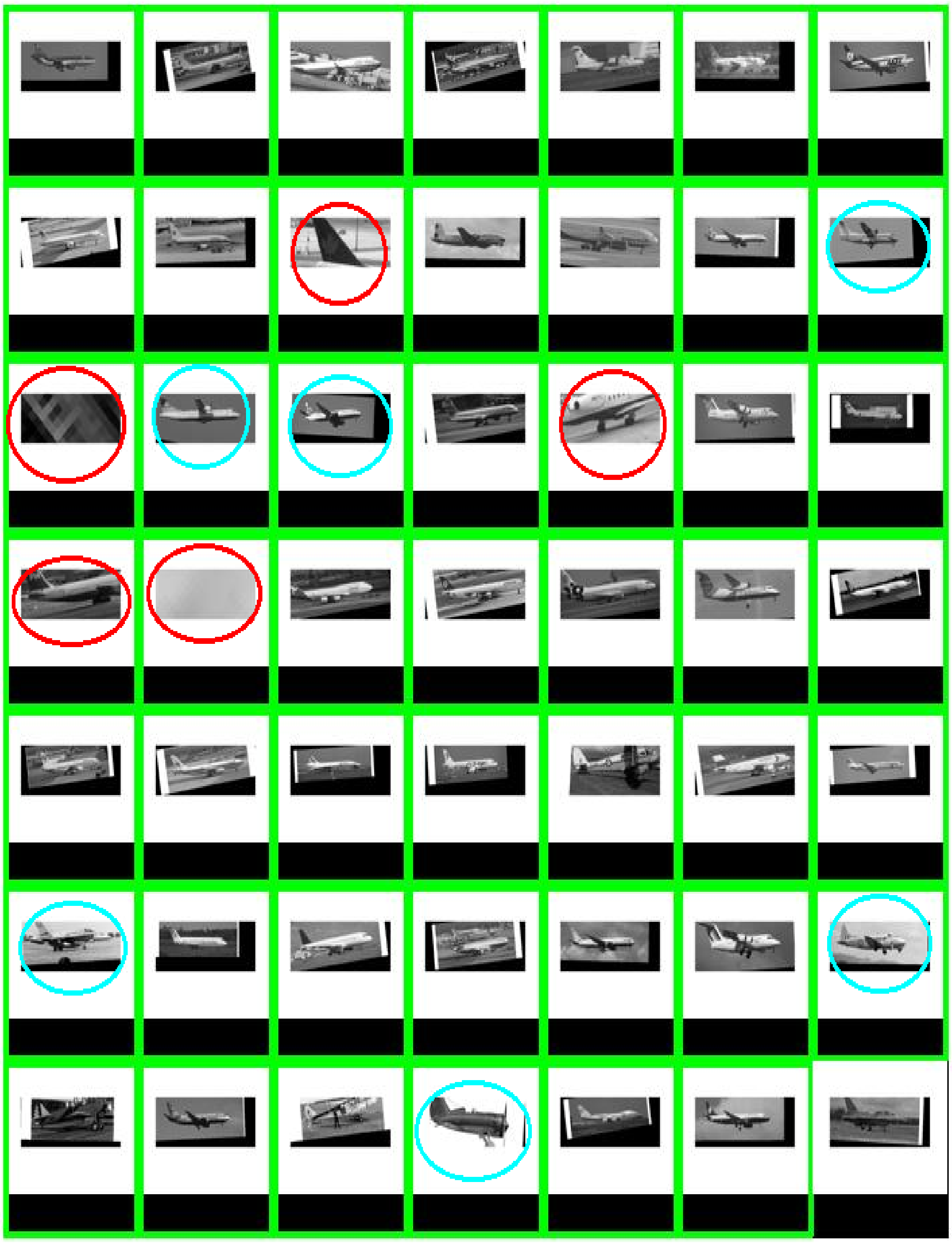


**Figure 74.** Class of airplane, Transformed images via alignment using MST method. Red circles show failed images after alignment and Cyan circles show images failed via Dijkstra but corrected through MST.

# 8 CONCLUSIONS

In this thesis, the problem of feature-based alignment was investigated. In particular, we investigated the part-based method by selecting an optimal seed automatically. The original method has the following difficulties: selecting seed manually, and using a single global seed for alignment that does not work perfectly in sub-classes. We proposed unsupervised optimal seed selection method to solve the problem of seed selection and to improve alignment, step-wise alignment via minimum spanning tree and Dijkstra shortest path are utilized . Based on our experiment we conclude that the average performance of our unsupervised seed selection technique is almost equal to the manual seed selection method, only in a few cases the manually selected seed produced better average images and lower MSE error, but in most of cases the results of unsupervised method were similar or even better than the original. According to the results of step-wise alignment, minimum spanning tree and Dijkstra shortest path outperform the direct alignment using a single seed. Moreover, overall efficiency of Dijkstra overcomes both single seed alignment and MST. It can be claimed that visual object categorization can benefit from our method since the step-wise alignment makes the class learning more straightforward.

Experiments also indicated the drawback of the our unsupervised method. The main difficulty is the high computation time when the data set consists of a large number of images, since it needs to compute spatial scores and similarities for each image as a seed. For instance, for a data set of $50$ images, we need $50 \times 50$ iterations. The other thing is that although the algorithm is capable to find a seed with suitable scores, it still needs some refinement or other factors to improve the precision when images have an equally good average and STD score values. Surprising aspect of the result is that in some classes the images that failed with MST alignment were corrected with Dijkstra and vice versa. It seems that the two methods complement each other. Consequently, the method would be more efficient if we can implement an algorithm with a combination of both, and recognize when to use Dijkstra and MST.

Regarding the utilized detection and description methods in the experiments, the single scale dense sampling and SIFT descriptor worked well for this work since the used training images contain only minor variation in scale and rotation. In this case, multiscale dense sampling, or a method of combination of interest points and dense sampling called "dense interest points"' [59] can be considered.

# REFERENCES

[1] Tinne Tuytelaars. Dense interest points. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2281–2288. IEEE, 2010.

[2] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, DTIC Document, 1980.

[3] Erik G Learned-Miller. Data driven image models through continuous joint alignment. *Journal IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2006.

[4] Weiyuan Ni and Alice Caplier. Newton optimization based congealing for facial image alignment. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 577–580. IEEE, 2011.

[5] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *Computer Vision–ECCV 2006*, pages 490–503. Springer, 2006.

[6] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(4):376–380, 1991.

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

[8] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[9] Brent C Munsell, Andrew Temlyakov, Martin Styner, and Song Wang. Pre-organizing shape instances for landmark-based shape correspondence. *International journal of computer vision*, 97(2):210–228, 2012.

[10] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice Hall, 1st edition, 2001. ISBN 0-13-030796-3.

[11] T. Morris. *Computer Vision and Image Processing*. Cornerstones of computing. Palgrave Macmillan Limited, 2004.

[12] Bernd Jahne and Horst HauBecker. *Computer Vision and Applications, A Guide for Students and Practitioners*. Academic Press, 1st edition, 2000. ISBN 0-13-085198-1.

[13] R Szeliski. *Computer vision: algorithms and applications*. Springer, 2011.

[14] Atsushi Tatsuma. Opencv sift dense sampling.

[15] http://www.wallsave.com/wallpaper/1440x900/umbrella-croporation-alice-resident-evil-afterlife-movie 379677.html. Wallsave.

[16] Bernd Heisele. Visual object recognition with supervised learning. *Intelligent Systems, IEEE*, 18(3):38–42, 2003.

[17] Ilonen Jarmo. *Supervised Local Image Feature Detection*. PhD thesis, LUT, 2007.

[18] Kevin Murphy, Antonio Torralba, Daniel Eaton, and William Freeman. Object detection and localization using local and global features. In *Toward Category-Level Object Recognition*, pages 382–400. Springer, 2006.

[19] Sung Ju Hwang and Kristen Grauman. Reading between the lines: Object localization using implicit cues from image tags. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(6):1145–1158, 2012.

[20] Martin A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, C-22(1):67–92, 1973.

[21] Ivan Laptev Stefan Carlsson, Hossein Azizpour. part based modles.

[22] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[23] Gustavo Carneiro and David Lowe. Sparse flexible models of local features. In *Computer Vision–ECCV 2006*, pages 29–43. Springer, 2006.

[24] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, page 22, 2004.

[25] Nuno Miguel Borges de Pinho Cruz. *Bayesian Models for Visual Information Retrieval*. PhD thesis, Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2000.

[26] Robert Fergus, Pietro Perona, and Andrew Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 380–387. IEEE, 2005.

[27] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[28] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–264. IEEE, 2003.

[29] David Crandall, Pedro Felzenszwalb, and Daniel Huttenlocher. Spatial priors for part-based recognition using statistical models. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 10–17. IEEE, 2005.

[30] Guillaume Bouchard and Bill Triggs. Hierarchical part-based visual object categorization. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 710–715. IEEE, 2005.

[31] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[32] Long Zhu, Yuanhao Chen, Alan Yuille, and William Freeman. Latent hierarchical structural learning for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1062–1069. IEEE, 2010.

[33] Cootes T Cristinacce, D. Feature detection and tracking with constrained local models. BMVC, 2006.

[34] Deva Ramanan. Learning to parse images of articulated bodies. In *Advances in neural information processing systems*, pages 1129–1136, 2006.

[35] Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. Pose search: retrieving people using their pose. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1–8. IEEE, 2009.

[36] Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1385–1392. IEEE, 2011.

[37] Weilong Yang, Yang Wang, and Greg Mori. Recognizing human actions from still images with latent poses. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2030–2037. IEEE, 2010.

[38] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1365–1372. IEEE, 2009.

[39] Yang Wang, Duan Tran, and Zicheng Liao. Learning hierarchical poselets for human parsing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1705–1712. IEEE, 2011.

[40] Lubomir Bourdev. *Poselets and their applications in high-level computer vision.* PhD thesis, University of California, 2011.

[41] Patrick Ott and Mark Everingham. Shared parts for deformable part-based models. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1513–1520. IEEE, 2011.

[42] Julia Vogel. *Semantic scene modeling and retrieval.* PhD thesis, Citeseer, 2004.

[43] Anna Bosch, Xavier Munoz, Arnau Oliver, and Robert Marti. Object and scene classification: what does a supervised approach provide us? In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 773–777. IEEE, 2006.

[44] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[45] Alex D Holub, Max Welling, and Pietro Perona. Combining generative models and fisher kernels for object recognition. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 136–143. IEEE, 2005.

[46] Shivani Agarwal and Dan Roth. Learning a sparse representation for object detection. *Computer Vision-ECCV 2002*, pages 97–101, 2006.

[47] Markus Weber, Max Welling, and Pietro Perona. *Unsupervised learning of models for recognition.* Springer, 2000.

[48] Robert Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition,*

*2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–264. IEEE, 2003.

[49] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Multiple object class detection with a generative model. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 26–36. IEEE, 2006.

[50] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. Generic object recognition with boosting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):416–431, 2006.

[51] Peter Carbonetto, Gyuri Dorko, Cordelia Schmid, and N de Freitas. Bayesian learning for weakly supervised object classification. Technical report, Citeseer, 2004.

[52] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[53] MultiMedia LLC. Visual odometry.

[54] Wikipedia. Corner detection.

[55] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[56] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[57] Timor Kadir, Andrew Zisserman, and Michael Brady. An affine invariant salient region detector. In *Computer Vision-ECCV 2004*, pages 228–241. Springer, 2004.

[58] Tingting Jiang, Frederic Jurie, and Cordelia Schmid. Learning shape prior models for object matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 848–855. IEEE, 2009.

[59] Jukka Lankinen and Joni-Kristian Kämäräinen. Local feature based unsupervised alignment of object class images. In *BMVC*, pages 1–11, 2011.

[60] Brendan J Frey and Nebojsa Jojic. Transformation-invariant clustering and dimensionality reduction using em. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1000–1014, 2000.

[61] Andrea Vedaldi and Stefano Soatto. A complexity-distortion approach to joint pattern alignment. In *Advances in Neural Information Processing Systems*, pages 1425–1432, 2006.

[62] Gary B Huang, Vidit Jain, and Erik Learned-Miller. Unsupervised joint alignment of complex images. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[63] Mark Cox, Sridha Sridharan, Simon Lucey, and Jeffrey Cohn. Least squares congealing for unsupervised alignment of images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[64] Ya Xue and Xiaoming Liu. Image congealing via efficient feature selection. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 185–192. IEEE, 2012.

[65] Ngoc-Son Vu and Alice Caplier. Face recognition with patterns of oriented edge magnitudes. In *Computer Vision–ECCV 2010*, pages 313–326. Springer, 2010.

[66] Neil Mac Parthalain and Harry Strange. Fuzzy-entropy based image congealing. In *Fuzzy Systems (FUZZ), 2013 IEEE International Conference on*, pages 1–8. IEEE, 2013.

[67] Yuanhao Chen, Long Zhu, Alan Yuille, and Hongjiang Zhang. Unsupervised learning of probabilistic object models (poms) for object classification, segmentation and recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[68] Iasonas Kokkinos and Alan Yuille. Unsupervised learning of object deformation models. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[69] Philippe Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *arXiv preprint math/0604233*, 2006.

[70] Teemu Kinnunen, Jukka Lankinen, Joni-Kristian Kämäräinen, Lasse Lensu, and Heikki Kälviäinen. Unsupervised visual object categorisation with bof and spatial matching. In *Image Analysis*, pages 384–395. Springer, 2013.

[71] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson and Clifford Stein. *Introduction to algorithm*. MIT Press, 1st edition, 1990. 978-0-262-03384-8.

[72] Terhi Kilamo. Lecture slides of the course of utilization of data structure. In *Lecuter slides of the course of Utilization of data structure*, volume 1, pages 282–285. TUT, 2012.

[73] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.