Patch-Duplets for Object Recognition and Pose Estimation

Björn Johansson and Anders Moe

March 17, 2004

Technical report LiTH-ISY-R-2553
ISSN 1400-3902

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
bjorn@isy.liu.se, moe@isy.liu.se

Abstract

This report describes a view-based method for object recognition and estimation of object pose in still images. The method is based on feature vector matching and clustering. A set of interest points, in this case star-patterns, is detected and combined into pairs. A pair of patches, centered around each point in the pair, is extracted from a local orientation image. The patch orientation and size depends on the relative positions of the points, which make them invariant to translation, rotation, and scale. Each pair of patches constitutes a feature vector. The method is demonstrated on a number of real images.

Contents

1 Introduction

The system for view-based object recognition described in this report is related to, or inspired by, a number of methods that have been presented in recent years, see e.g. [11, 12, 14, 13, 10, 7, 6]. They all have in common that they use local descriptors centered around suitably chosen points of interest, making them more robust against occlusions and non-rigidity of objects than global descriptors. The descriptors are in most cases computed in such a way that they are invariant to translation, plane rotation, and scale. They should also be robust against small pose changes and other small image deformations.

Schmid and Mohr [13] compute interest points using the Harris detector [8]. A set of differential invariants, which are invariant to rotation, is computed in the gray-valued image in one scale for the prototype images, and in several scales for the query image.

In Lowe [11, 12] patches are extracted in a local orientation image around interest points which are computed as maxima and minima points of a difference-of-Gaussian function in scale-space. The size of a patch is chosen proportional to the scale of the interest point, and the patch orientation is chosen as the dominant direction in an orientation histogram computed in a region around the interest point.

In Selinger and Nelson [14] a set of segmented contour fragments broken at points of high curvature is detected. The longest curves are selected as key curves, and every one of these provides a seed for a context patch. The size and orientation of the patch depends on the size and orientation of the curve segment. All image curves that intersect the normalized patch are mapped into it with a code specifying their orientation relative to the base segment.

Granlund and Moe [7, 6] uses points of high curvature as as interest points. The feature vectors are based on combinations of three points, where the curvature orientations and the distances and angles between the points are used to compute a description that is invariant to rotation and scale. These feature vectors, or triplets, are after selection further combined to give more selective features.

We see that some methods use local patches extracted directly from the gray-value image or the local orientation, and other methods compute a set of invariants or model parameters of local regions. The majority use information around one single region, but some use information from a combination of regions. The idea of the patches is not to impose a model of what is seen in the image, but rather to use what is really seen. On the other hand, the invariants or model parameters may allow for a more compact description and for additional invariants.

The methods above also differ in the ways they perform the feature matching, clustering, and verification. We will not discuss those here.

1.1 Overview

In training mode we acquire a number of images of an object taken from different views (poses), referred to as prototype images. For each prototype image we compute a number of feature vectors that will serve as a representation of that image. These feature vectors will constitute the training set. In usage mode we find the object pose and location in the query image by comparing the feature vectors in the query image with the feature

vectors in the training set. One or several similar prototype images that resemble the query image are found by a feature matching and clustering procedure. The object pose in the query image is either chosen as the pose of the closest resembling prototype image, or computed as an interpolation between the closest resembling prototype images.

Figure 1 presents an overview of the method used to generate feature vectors from an image or object. Each step will be explained in more detail in the sections to follow. The feature vectors are made up of local patches in a local orientation image. The idea is to have feature vectors (patches) that are invariant to translation, rotation, and scale. This is achieved by a normalization of patch size and orientation that is based on pairs of interest points. For each pair of points we extract two patches. The patches are centered around the points, and their orientation and size are chosen depending on the relative point positions and the distance between the points. Each feature vector consists of such a pair of patches, here referred to as *patch-duplets* or simply duplets. The patches are extracted from the local orientation image. One may also use gray-value or color patches, but we choose local orientation information to become less sensitive to absolute colors, and also because this type of information is more sparse, leading to a more selective matching procedure. We have chosen star-patterns as interest points, because they are fairly scale invariant in themselves.

The main differences between this approach and the ones described above are that we use another type of interest points, that the scale and orientation invariances are found in a different manner, and that the feature vectors are based on two local regions instead of one.

2 Points of Interest

The first step in the process is to find suitable points of interest. They should preferably be invariant to translation, rotation, and scale. One approach, applied by e.g. [11, 12], is to detect features in several scales and compute local maxima in the feature scale-space. Another approach is to find patterns that are scale invariant in themselves, one example is star-patterns. The star-pattern detector described in Section 2.2 is based on local orientation information.

2.1 Local orientation

A simple measure of local orientation is the image gradient, ∇I. We will use the image gradient here, but we will add a post-processing step that performs an orientation selective inhibition and enhancement. The latter step implies that we average along the edges, and subtract in the orthogonal direction. A directional Laplacian operator is used, i.e.

$$\begin{aligned} f(\alpha) &= \left(\sigma^2 - (\cos(\alpha)x + \sin(\alpha)y)^2\right) g(x,y) \\ &= \left(\sigma^2 - \cos^2(\alpha)x^2 - \sin^2(\alpha)y^2 - \sin(2\alpha)xy\right) g(x,y)\,, \end{aligned} \tag{1}$$

where σ is the standard deviation of the Gaussian $g(x,y)$ and α is the angle of the gradient. The last equality in (1) shows that the filter can be constructed as a linear combination

1. Choose an image.

2. Find suitable points of interest, in this case star-patterns.

3. Make point pairs.

4. For each pair of points, extract patches in the local orientation image. The patch orientation is determined by the relative location of the points, and the patch size is proportional to the distance between the points.

5. Use the pair of patches (patch-duplets) as feature vectors.

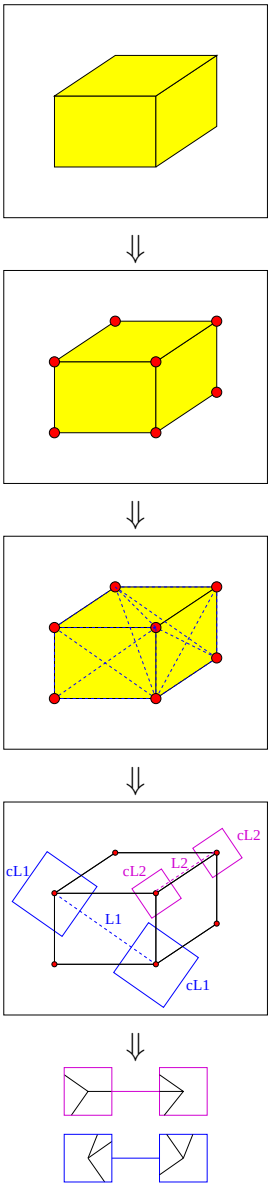


Figure 1: Overview of the feature generation algorithm.

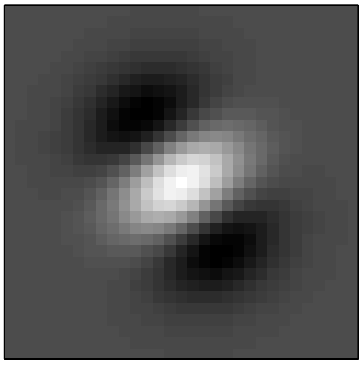


Figure 2: The filter $f(\alpha)$ in (1). Negative values are denoted by black and positive values by white.

of a fixed set of filters, i.e. we can use the idea of steerable filters. Figure 2 shows an example of the filter.

The gradient is often too sensitive to contrast, therefore we raise the magnitude of the gradient with a number $a \leq 1$. This can make the estimate sensitive to noise and smooth shadows, but the estimate is improved by the inhibition. The filter in (1) is applied to the mapped gradient magnitude, and negative filter responses are set to zero, i.e.

$$|\nabla I|_{\text{inhib}} = \max\left(0, f(\alpha) \star |\nabla I|^a\right) , \tag{2}$$

where $\max(\cdot, \cdot)$ denotes pixelwise maximum, and $\star$ denotes correlation. The gradient will be inhibited if there exist other large gradients positioned orthogonal to the line, regardless of their orientation. This inhibition will reduce the gradient in for example corners, which may be undesirable. Another approach could be to inhibit only with parallel edges and lines, and ignore orthogonal orientations. Some experiments on this idea have been made and the resulting orientation image sometimes look subjectively better. But the advantage in the application described here is not obvious and the computational cost in the improved implementation is higher. We therefore settle for the simpler version here.

The algorithm to detect local orientation, denoted Algorithm 1, is summarized in Figure 3. The results of running Algorithm 1 on two test images are shown in Figures 4 and 5. We emphasize that this method is not optimal, Figure 6 shows the result on another image of the same car and similar car pose, but with occlusion, different camera focus, and different lightning conditions. We see that the local orientation image differs a great deal from the corresponding one in Figure 5. This is a problem for the following steps in the system, and should be investigated further. But we also emphasize that the car in Figure 6 has a quite different appearance from the car in Figure 5, which will be a problem for any object recognition system. Still, the car is correctly recognized in the experiments below.

2.2 Finding Star-patterns

Let $\mathbf{x} = (x \ y)^T$ be vector in a local Cartesian coordinate system in an image and let $\mathbf{x}_\perp = (-y \ x)^T$ denote the vector orthogonal to $\mathbf{x}$. We define a star-pattern as a pattern

Algorithm 1 *Detection of local orientation*

1. Compute image gradient using Gaussian derivative filters, i.e.

$$\nabla I = \begin{pmatrix} xg(x,y) \star I(x,y) \\ yg(x,y) \star I(x,y) \end{pmatrix} , \tag{3}$$

   where $g(x,y)$ is a Gaussian function. We will use a small standard deviation, typically $\sigma = 1$.

2. Compute orientation selective inhibition and enhancement using steerable filters, i.e.

   (a) Compute the following four correlations:

$$\begin{aligned} v_1 &= g(x,y) \star |\nabla I(x,y)|^a \\ v_2 &= x^2 g(x,y) \star |\nabla I(x,y)|^a \\ v_3 &= y^2 g(x,y) \star |\nabla I(x,y)|^a \\ v_4 &= xyg(x,y) \star |\nabla I(x,y)|^a \end{aligned} \tag{4}$$

   (b) Compute the new gradient magnitude as

$$|\nabla I|_{\text{inhib}} = \max\left(0,\ \sigma^2 v_1 - \cos^2(\alpha) v_2 - \sin^2(\alpha) v_3 - \sin(2\alpha) v_4\right) , \tag{5}$$

   where $\alpha = \angle \nabla I$.

Figure 3: Algorithm for detection of local orientation.

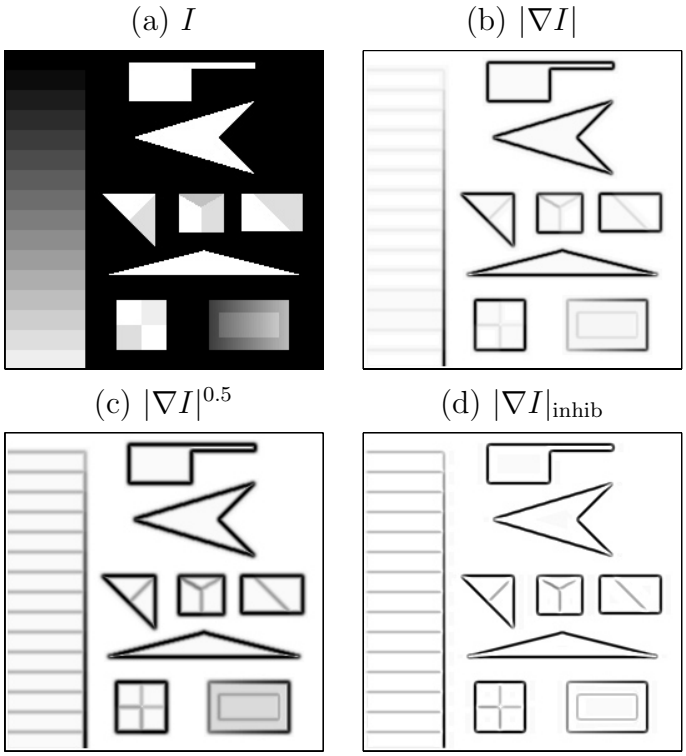


Figure 4: Results from Algorithm 1 on a synthetic test image.

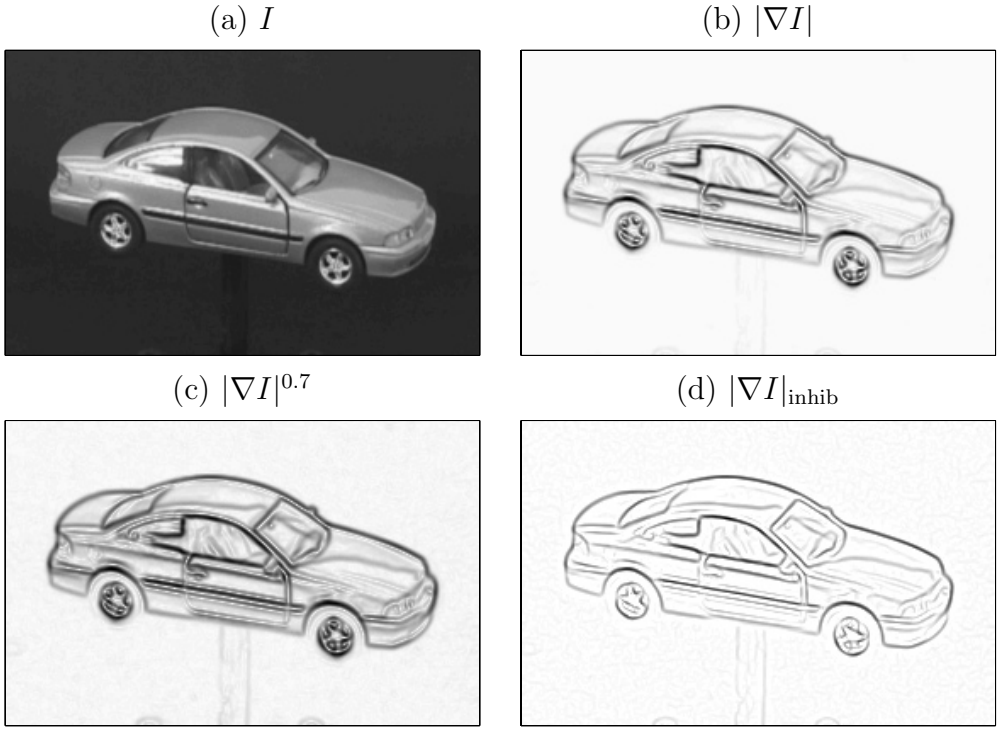


Figure 5: Results from Algorithm 1 on a real test image.

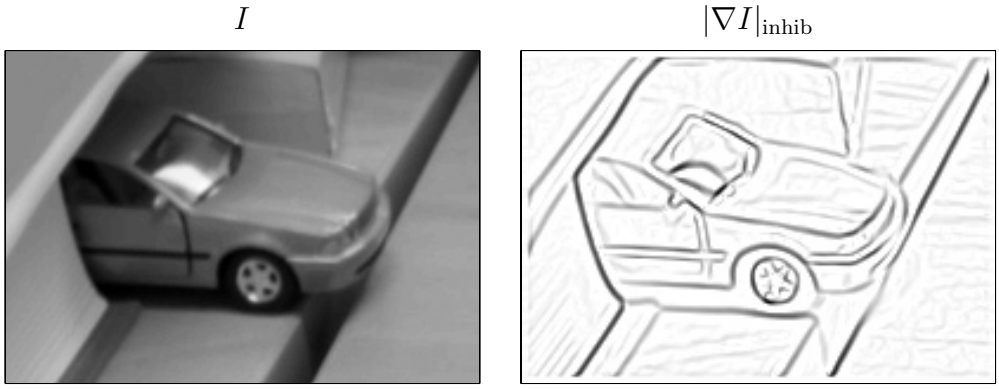


Figure 6: The result of Algorithm 1 on a test image with the same car of similar pose as in Figure 5.

in which the local orientation everywhere is perpendicular to the vector $\mathbf{x}$. Examples of star-patterns are lines, edges, corners, T-junctions, and Y-junctions.

The method we use to find star-patterns is a combination of the ideas described in [3, 9, 1, 2]. All references describe how star-patterns can be detected by correlation with suitable filters on images containing information about the local orientation.

The method in [3] uses the outer product of the image gradient and correlates with a set of filters of the type $xg(x, y)$, $yg(x, y)$, $x^2g(x, y)$ etc., where $g(x, y)$ is a Gaussian function. The location of the star-pattern is improved to sub-pixel accuracy by finding the point around which the pattern is least similar to a circle-pattern.

The references [9, 1, 2] describe star-patterns as a special case of *rotational symmetries*. They are here detected by computing a polynomial expansion on an image that contains the local orientation in double angle representation. The result is in [9] made more selective by an inhibition with a function that is high for simple signals.

The type of filters and orientation description used in [3] and [9] are basically equivalent, and they can be described in a common framework. Then one can see that the main difference, besides the sub-pixel improvement and inhibition with simple signals, is that the method in [9] also inhibits the star-pattern value with a function that is high for circle-patterns. The circle-patterns are in a sense the patterns that are most dissimilar to star-patterns. This circle-inhibition will make the detection more selective, but at the same time the detection may become more sensitive to surrounding features. For example, a bicycle wheel consists of a star-pattern surrounded by a circle-pattern. The circle-inhibition may then give a zero response for this type of pattern. This may also be a problem for the sub-pixel improvement in [3] which minimizes the circle-pattern function.

2.2.1 Theory

We will first describe the theory and algorithm to detect star-patterns, and then we illustrate the method on some test images. The theory is presented using continuous functions and integrals, but in practise we use discrete functions and summations.

For sake of simplicity, let $\nabla I(\mathbf{x}) = \nabla I = (I_x \ I_y)^T$ be a symbol for any local orientation estimate, not necessarily the image gradient. ∇I can for example be the inhibited gradient described in the previous section. Let S_{star} denote the function

$$S_{\text{star}} = \int g(\mathbf{x}) \left\langle \nabla I, \mathbf{x}_\perp \right\rangle^2 d\mathbf{x} = \int g(\mathbf{x}) \left(\mathbf{x}_\perp^T \nabla I \nabla I^T \mathbf{x}_\perp \right) d\mathbf{x} \,, \tag{6}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean scalar product. Furthermore, let S_{circle} denote the function

$$S_{\text{circle}} = \int g(\mathbf{x}) \left\langle \nabla I, \mathbf{x} \right\rangle^2 d\mathbf{x} = \int g(\mathbf{x}) \left(\mathbf{x}^T \nabla I \nabla I^T \mathbf{x} \right) d\mathbf{x} \,. \tag{7}$$

S_{star} will give a high value in regions that contain star-patterns, and low values in regions that contain circle-patterns. The other way round holds for S_{circle}. S_{star} is not as selective as one would desire. One problem is that edges and lines are included in the star-patterns. One way to remove those is to multiply S_{star} with a function that is low for edges and lines and high for every other star-pattern. We choose to use a measure for simple signals,

$$S_{\text{simple}} = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \,, \tag{8}$$

where $\lambda_1 \geq \lambda_2$ are the eigenvalues to the *structure tensor*,

$$\mathbf{T}_{\text{struct}} = \int g(\mathbf{x}) \nabla I \nabla I^T d\mathbf{x} \,. \tag{9}$$

Readers familiar with the double angle representation of local orientation we noticed that (8) is equivalent to

$$S_{\text{simple}} = \frac{|\int g(\mathbf{x}) z(\mathbf{x}) d\mathbf{x}|}{\int g(\mathbf{x}) |z(\mathbf{x})| d\mathbf{x}} \,, \tag{10}$$

where $z = (I_x + iI_y)^2$ is a double angle representation of the local orientation. We have that $0 \leq S_{\text{simple}} \leq 1$, and we compute the inhibited star-pattern detector as

$$\check{S}_{\text{star}} = (1 - S_{\text{simple}}) S_{\text{star}} \,. \tag{11}$$

Moreover, to make S_{star} even more selective we can also inhibit with S_{circle}. The inhibited star-pattern detector then becomes

$$\check{S}_{\text{star}} = (1 - S_{\text{simple}}) \max \left(0, S_{\text{star}} - S_{\text{circle}} \right) \,. \tag{12}$$

Another problem is that the value of S_{star} not only depends on the orientation of ∇I, but also on the magnitude, $|\nabla I|$. This may or may not be a problem, [3] suggests a solution; define

$$S_{\text{circle}}(\mathbf{p}) = \int g(\mathbf{x}) \left\langle \nabla I, \mathbf{x} - \mathbf{p} \right\rangle^2 d\mathbf{x} = \mathbf{p}^T \mathbf{T}_{\text{struct}} \mathbf{p} - 2\mathbf{p}^T \mathbf{v} + S_{\text{circle}}(\mathbf{0}) \,, \tag{13}$$

where

$$\begin{aligned} \mathbf{v} &= \int g(\mathbf{x}) \nabla I \nabla I^T \mathbf{x} d\mathbf{x} \,, \\ S_{\text{circle}}(\mathbf{0}) &= \int g(\mathbf{x}) \, \mathbf{x}^T \nabla I \nabla I^T \mathbf{x} d\mathbf{x} \,. \end{aligned} \tag{14}$$

$S_{\text{circle}}(\mathbf{p})$ in (13) is a generalization of S_{circle} in (7), where the value is now centered around the point $\mathbf{p}$ instead of in the origin ($\mathbf{p}$ is defined in the local coordinate system). By minimizing $S_{\text{circle}}(\mathbf{p})$ with respect to $\mathbf{p}$ we find the center of a star-pattern, i.e.

$$\min_{\mathbf{p}} \; S_{\text{circle}}(\mathbf{p}) \Rightarrow \mathbf{p} = \mathbf{T}_{\text{struct}}^{-1} \mathbf{v} \,. \tag{15}$$

The minimization is only performed in pixels where $\check{S}_{\text{star}}$ has a local maximum, and the pixel is ignored if $\mathbf{p}$ becomes too large compared to the window defined by the Gaussian $g(\mathbf{x})$. We will see in the first example in Section 2.2.2 that the improved position is more invariant to the local energy distribution, but it is still not clear whether the improvement is necessary in the application described here. It is undesirable that the maxima points change locations if the detector is computed in another scale (different σ). One motivation for using the improvement is that the detector will then become more stable in scale than without the improvement.

The algorithm to detect star-patterns, denoted Algorithm 2, is summarized in Figure 7. Each step is fairly simple and straightforward. The algorithm needs outputs from a number of filter correlations. S_{star} is computed from the following three correlations:

$$y^2 g(x,y) \star I_x^2, \quad x^2 g(x,y) \star I_y^2, \quad xy g(x,y) \star I_x I_y \,. \tag{17}$$

S_{circle} is computed from

$$x^2 g(x,y) \star I_x^2, \quad y^2 g(x,y) \star I_y^2, \quad xy g(x,y) \star I_x I_y \,. \tag{18}$$

S_{simple} and $\mathbf{T}_{\text{struct}}$ are computed from

$$g(x,y) \star I_x^2, \quad g(x,y) \star I_y^2, \quad g(x,y) \star I_x I_y \,. \tag{19}$$

Finally, $\mathbf{v}$ that is used for the improvement is computed from

$$x g(x,y) \star I_x^2, \quad y g(x,y) \star I_y^2, \quad x g(x,y) \star I_x I_y, \quad y g(x,y) \star I_x I_y \,. \tag{20}$$

In other words, we need to compute a subset of monomes (or derivatives) up to the second order on the three images I_x^2, I_y^2, and $I_x I_y$. All filters can be made separable, and furthermore they can be approximately implemented by Gaussian filters followed by small differential filters, see e.g. [9].

We have here described a set of tools for detection of star-patterns. We can increase the selectivity by inhibition with a measure for simple signals and also with a measure for circle-patterns. We can also improve the estimated position to subpixel accuracy. It is not clear which combination of tools is optimal, and the choice probably depends on the application.

Algorithm 2 *Detection of star-patterns*

1. Compute local orientation, e.g. using Algorithm 1.
2. Compute the star-pattern detector, S_{star}, in (6).
3. Compute the measure for simple signals, S_{simple}, in (8).
   Also compute S_{circle} if needed in step 4.
4. Compute the inhibited star-pattern detector, $\check{S}_{\text{star}}$, using either (11) or (12).
5. Find local maxima points, $\{\mathbf{x}_k\}$, in $\check{S}_{\text{star}}$. Local maxima points are computed in two steps:
   (a) Correlate $\check{S}_{\text{star}}$ with a Laplacian filter,
   $$(2\sigma^2 - x^2 - y^2)g(x, y)\,, \tag{16}$$
   where the Gaussian $g(x, y)$ has the same σ as before. This step may also be seen as an inhibition/enhancement, similar to the one used for the local orientation (step 2 in Algorithm 1).
   (b) Compute non-max-suppression in a 3×3 window and threshold.
6. In each maxima point, $\mathbf{x}_k$, compute improvement $\mathbf{p}_k$ from (15). The resulting star-pattern positions are given by $\mathbf{x}_k + \mathbf{p}_k$.

Figure 7: Algorithm for detection of star-patterns.

2.2.2 Examples

We will now illustrate Algorithm 2 on the test images in Figures 4 and 5. The local orientation in step 1 is computed from Algorithm 1 (also shown in Figures 4 and 5).

The result of Algorithm 2 on the first image is shown in Figure 8(a). This image is a fairly well known test image for point of interest detectors, in particular corner detectors, see e.g. [15]. The inhibition in (11) is used, since the inhibition in (12) performed slightly worse. It is difficult to see the improvement vectors $\mathbf{p}_k$ in Figure 8(a). Therefore, Figure 8(b) shows a magnification of a region in the lower left part of Figure 8(a). We can for example see in the left part of the images in Figures 8(a,b) that the distance between the maximum point and the star-pattern center is larger in the bottom part than in the upper part. This means that the maxima depend on the local energy distribution, but we can also see that step 6 in Algorithm 2 is improving the estimated positions. Figures 8(c,d) shows some intermediate results, and the alternative inhibition (12) is shown in Figure 8(e) for comparison.

The result of Algorithm 2 on the second image is shown in Figure 9(a). The inhibition in (12) is used because the result looked somewhat more correct, but the inhibition in (11) may performed equally well if a comparative evaluation were to be performed. Figures 8(b,d) shows some intermediate results, and the alternative inhibition (11) is shown in Figure 8(c) for comparison.

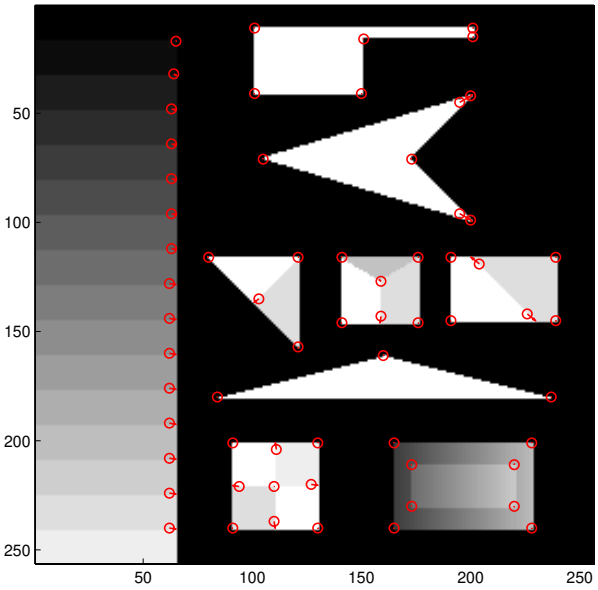
To further refine the method, we can for example compute the detector in several scales and take the sum or the product between the scales. This is done in the experiments, Section 6, where the product between two scales is used.

3 Point Pairs

If all possible point pairs are constructed in an image with N interest points we would get N^2 pairs, which would result in about 6400 pairs for the image in Figure 8(a). This is unnecessarily many, so some rules for allowed combinations are needed. The rules used in the experiments are to only combine each point with the M nearest neighbors which satisfies $d_{\min} < d < d_{\max}$, where d is the distance between the points.

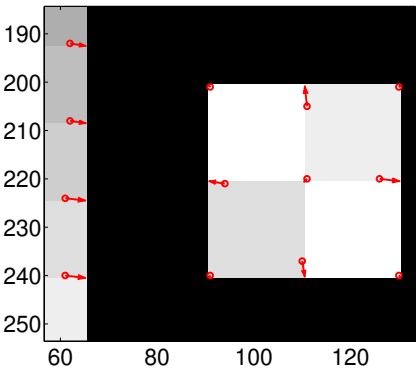
An example of the resulting combinations with $d_{\min} = 10$, $d_{\min} = 100$, and $M = 2$ is shown in Figure 10. Notice that the pairs are directed, i.e. the points in the pair are ordered, and that the opposite directed pair does not have to be chosen. Other rules can be used, for example to combine a point with all neighboring points within a fixed distance. Perceptual grouping can also be used to decrease the number of combinations and at the same time make the combinations more robust. An example would be to only combine points which are connected with a contour in the image, as in [14]. This should increase the probability that the two points belong to the same object. The drawback is that the regions around the pairs get more similar to each other, and the information in the pair gets lower.

(a) Maxima points $\{\mathbf{x}_k\}$ (circles), Improvements $\{\mathbf{p}_k\}$ (vectors)

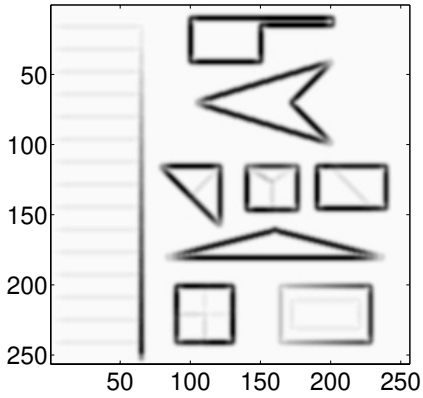


(b) Zoomed region in (a)

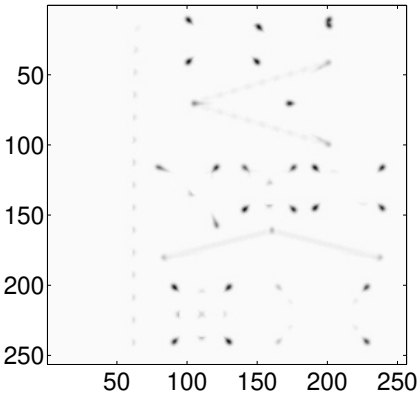
(c) S_{star}



(d) $\check{S}_{\text{star}}$ in (11)



(e) $\check{S}_{\text{star}}$ in (12)



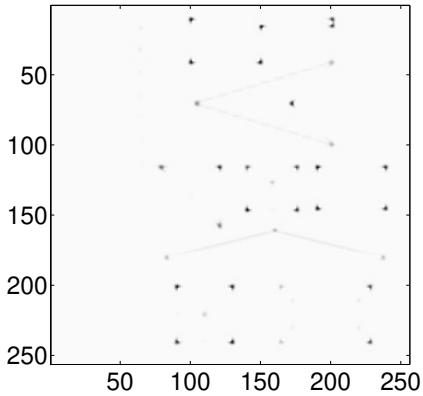
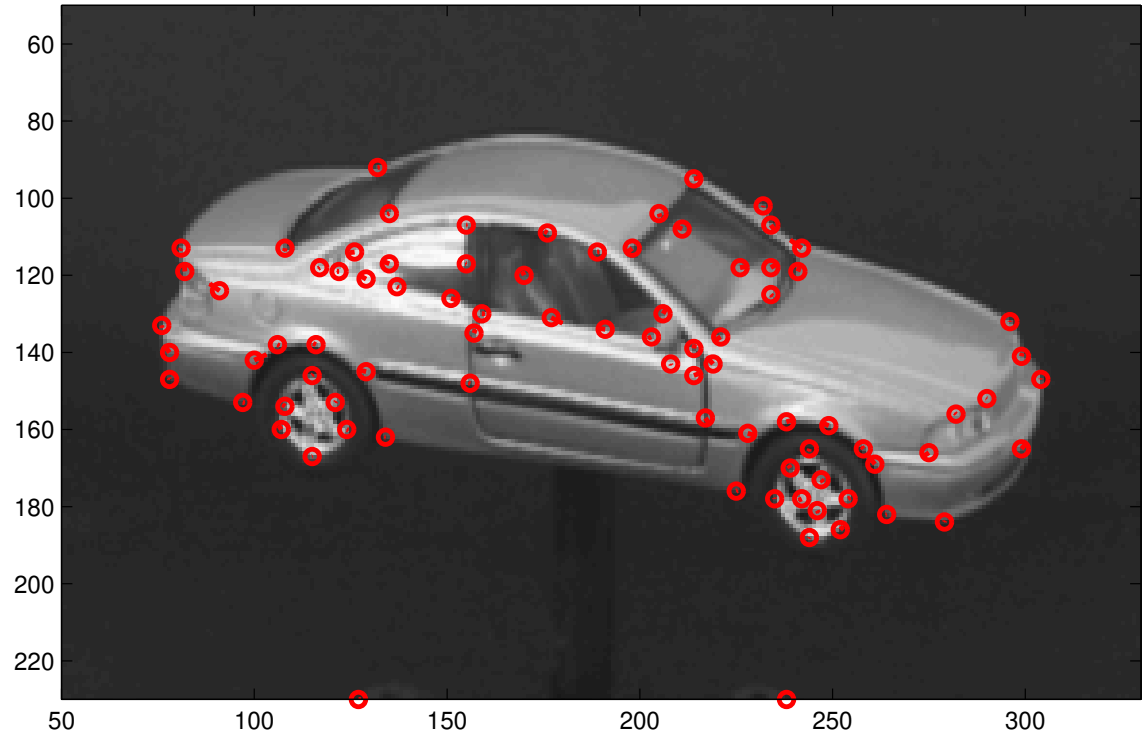


Figure 8: Detection of star-patterns, results of Algorithm 2 on a synthetic test image. $\check{S}_{\text{star}}$ in (11) was used for inhibition. (a) Maxima points $\{\mathbf{x}_k\}$ (circles) and improvements $\{\mathbf{p}_k\}$ (vectors). (b) Zoomed region in (a) to better show some of the improvement vectors. (c,d) Intermediate results. (e) Alternative inhibition in (12) for comparison.

(a) Maxima points $\{\mathbf{x}_k\}$ (circles), Improvements $\{\mathbf{p}_k\}$ (vectors)



(b) S_{star} (c) $\check{S}_{\text{star}}$ in (11) (d) $\check{S}_{\text{star}}$ in (12)

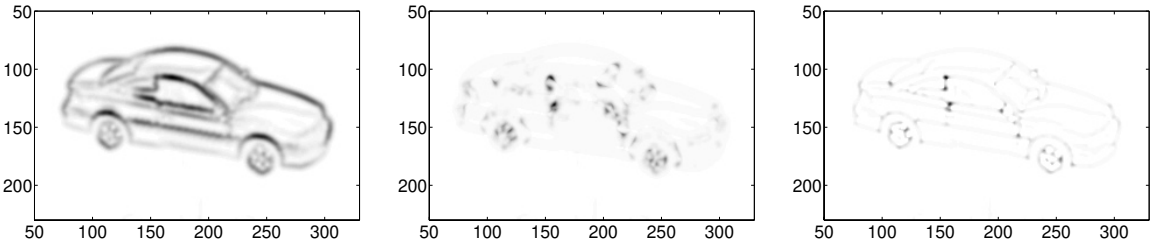


Figure 9: Detection of star-patterns, results of Algorithm 2 on a real test image. $\check{S}_{\text{star}}$ in (12) was used for inhibition. (a) Maxima points $\{\mathbf{x}_k\}$ (circles), Improvements $\{\mathbf{p}_k\}$ (vectors) (b,d) Intermediate results. (c) Alternative inhibition in (11) for comparison.

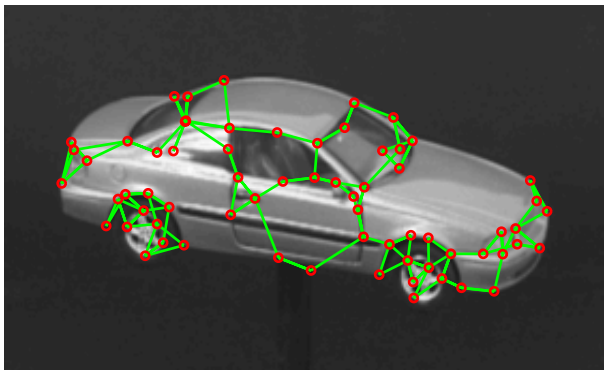


Figure 10: The constructed point pairs are marked with lines.

4 Patch-Duplets

The point pairs described in the previous section will now be used to extract local regions (patches) in the local orientation image. The local orientation is described in double angle representation (see e.g. [4]), which makes the description invariant to the sign of the orientation vectors. For each pair of points we compute two patches centered around the points, again see Figure 1 for an illustration. The patches are oriented along a line going through the points, and the size of each patch is proportional to the distance between the points. Each pair of local orientation patches will serve as a feature vector, referred to as *patch-duplets*, or simply duplets. The feature vectors will be invariant to translation, rotation, and scale assuming that the interest points are invariant to rotation and scale.

The patches are sampled in a 4×4 grid using linear interpolation, giving 16 samples. To avoid aliasing the patches should be low-pass filtered before the sampling, with filters having cut-off frequencies proportional to the patch sizes. However, aliasing is not considered to be any big problem since we are not going to reconstruct anything. Image deformations are on the other hand considered to be a problem, so the orient image is blurred before the extraction of the patches, to reduce the sensitivity to those. This will of course reduce the aliasing as well. The feature vectors are then constructed from the samples of the two patches in the duplets, giving feature vectors with 32 complex valued elements. Together with each feature vector we store the positions of the two interest points in the duplet. This makes it possible to estimate the translation, rotation, and scaling of the object from a correspondence between two duplets.

Figure 11 contains some examples of patches extracted from an image, just to give an idea of what the patches may look like. The number of points has been reduced by choosing a higher threshold, each point is connected only to one other point, and only one patch is shown for each pair of points. This reduces the number of patches which makes the illustration easier to comprehend. But note that these patches are not a subset of the true patches, since additional points will change the selected pairs. Figure 11(c) shows the patches extracted from Figure 11(b) in high resolution (20×20). However, high resolution patches give high-dimensional feature vectors and a computationally complex matching procedure. Therefore, the low-resolution version (4×4) is used in the experiments, see

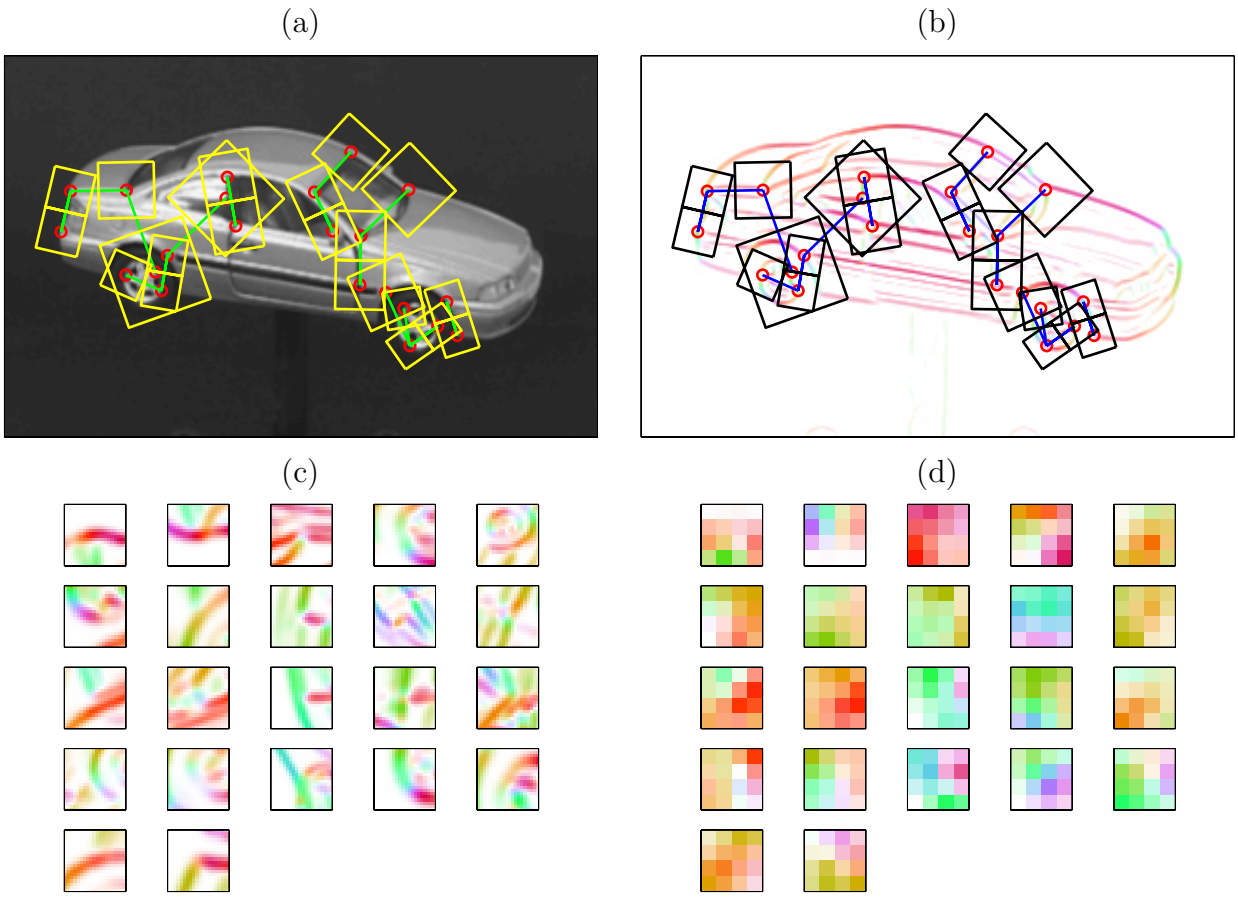


Figure 11: Example of orientation patches extracted from pairs of points. (a) The gray-valued image and the patches. (b) The same patches on the local orientation image. The color represents the local orientation. (c) The patches in high resolution (for illustration purposes). For comparison with (b), note that the normalization changes the colors. (d) The patches in low resolution (used in the experiments).

Figure 11(d). The sizes of the patches used in the experiments are chosen as half the distance between the points in the pair.

5 Feature Matching and Clustering

5.1 Matching

The matching of feature vectors is done by first normalizing them to get more robust to illumination variations, and then computing the distance between them. The L1-norm is used instead of the L2-norm to make the matching not as sensitive to outliers in the feature vectors. A way to get even less sensitive is to use a sigmoid function on the error. This would also change the error metric so that a feature component which is "classified" as wrong contributes with the same error independently of how wrong it is. Another

way to get the same effects as with the sigmoid function is to channel code each feature component in the feature vectors before matching them, see e.g. [16].

For each duplet in the query image, the k closest prototype duplets in the training set are kept as possible matches, if the distance is below a threshold.

5.2 Clustering

For each duplet in the query image at least one estimate of the pose angles, scale, orientation, and position of the object is obtained. Hence, one way to get an estimate of these parameters would be to cluster in this 6 dimensional space and if there is a significant cluster in this space we say that the object is present in the image with the parameters corresponding to the cluster position. Each vote in the clustering should be weighted according to its significance. This method has not been tested yet, instead a hypothesis and verification process described below is used. This method should give about the same results, but is easier to implement and can in some cases be faster. The methods can also be combined, as in [11, 12] where the clustering method is combined with a verification method similar to the one used here.

Before the hypothesis and verification process a 2D weighted histogram is computed on the pose angle votes. For each bin larger than some threshold, starting with the largest, hypotheses of the translation, rotation and scale transformation between the image and the prototype image is made. A hypothesis is obtained by computing the transformation between a duplet in the query image and a matching duplet in the prototype image, i.e.

$$\mathbf{p}_q = \mathbf{t} + s\mathbf{R}\mathbf{p}_p, \tag{21}$$

where $\mathbf{p}_q, \mathbf{p}_p$ is the position of one of the points in the duplet in the query image and prototype image respectively, $\mathbf{t}$ is a translation vector, s is a scaling, and $\mathbf{R}$ is a rotation matrix. The transformation has 4 degrees of freedom, so the two points in a duplet is enough to compute the transformation. If $\mathbf{p}_p$ is written in homogeneous coordinates $\tilde{\mathbf{p}}_p$, this transformation can be written as

$$\mathbf{p}_q = \begin{pmatrix} s\mathbf{R} + \mathbf{t} \end{pmatrix} \begin{pmatrix} \mathbf{p}_p \\ 1 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 \\ -a_2 & a_1 & a_4 \end{pmatrix} \begin{pmatrix} \mathbf{p}_p \\ 1 \end{pmatrix} = \mathbf{T}\tilde{\mathbf{p}}_p. \tag{22}$$

This hypothesis is then verified by transforming the other correspondences found for that prototype image to the query image. The error is weighted with the distance between the hypothesis duplet and the verification duplet, to account for errors in the scaling and rotation estimates. A constant term a is then added to the weighting to account for the error in the translation estimate, i.e.

$$e = \frac{||\mathbf{p}_{q1} - \mathbf{T}\tilde{\mathbf{p}}_{p1}|| + ||\mathbf{p}_{q2} - \mathbf{T}\tilde{\mathbf{p}}_{p2}||}{||(\mathbf{p}_{q1} + \mathbf{p}_{q2})/2 - (\mathbf{p}_{qh1} + \mathbf{p}_{qh2})/2|| + a}, \tag{23}$$

where $\mathbf{p}_1, \mathbf{p}_2$ are the positions of the two points in the duplet and $\mathbf{p}_{qh}$ is the position of the hypothesis duplet in the query image. The hypothesis with the largest number of

	ϕ	θ
Training	$0°, 10°, 20°, ..., 180°$	$0°, 10°, ..., 40°$
Evaluation	$5°, 15°, 25°, ..., 175°$	$5°, 15°, ..., 35°$

Table 1: The used pose angles for the training and the evaluation.

correspondences with an error below a threshold is said to be correct and these correspondences are kept. At least one correspondence other than the hypothesis must be accepted to keep any correspondence. A faster method would be to stop when a hypothesis which is good enough according to some criterion is found. This hypothesis verification process starts with the correspondence that has the highest confidence. If only a very approximate estimate of the 6 object parameters is enough, this process can be stopped as soon as a hypothesis is found to be correct. However, if a more exact estimate of the parameters is wanted this needs to be done for a number of prototype images to allow interpolation, or just to find the prototype view with the highest weighted sum of correspondences which hopefully corresponds to the closest prototype view. The weighting of a vote is done with the confidence of the correspondence, but also with the number of prototype duplets belonging to the same prototype view as the correspondence. This removes the bias towards prototype views containing many duplets. The interpolated pose estimate is obtained by channel coding the votes for the two pose angles, weighting them as above, takeing the outer product between the two channel coded pose angles and then summing them. This gives a 2D histogram with overlapping bins, which is then decoded to obtain the estimate, see e.g. [16] for details.

6 Experiments

The method has been tested on the toy car in Figure 12. Since the duplet feature vectors are assumed to be invariant to translation, scale and rotation, we only need to collect data with views from different pose angles, see Figure 12. The pose angles are varied with $10°$ increments between $0° - 180°$ for the ϕ angle and between $0° - 40°$ for the θ angle, see Table 1. This gives in total 95 prototype images. The duplets are constructed by connecting each point of interest to the two closest neighbors, which gives about 50-150 duplets in each image and a total of 11028 feature vectors. We use a 4×4 sample grid for each of the two patches in the duplet, giving complex valued feature vectors of length 32.

The evaluation is made on images of the car having the pose angles specified in table 1. Some other images with unknown pose angles, background, and occlusions are also tried. One evaluation image and the votes from the duplets are shown in Figure 13.

For each evaluation image a hypothesis of the object pose is made by finding the prototype view obtaining the highest number of votes weighted with the confidence for the votes. The results are plotted in Figure 14 for the two pose angles. Since one of the closest views are always found the mean absolute error is $5°$ for both pose angles. An improvement of the estimates is obtained by using the channel-coding based interpolation method, the results are plotted in Figure 15. The mean absolute error is now $1.25°$ and

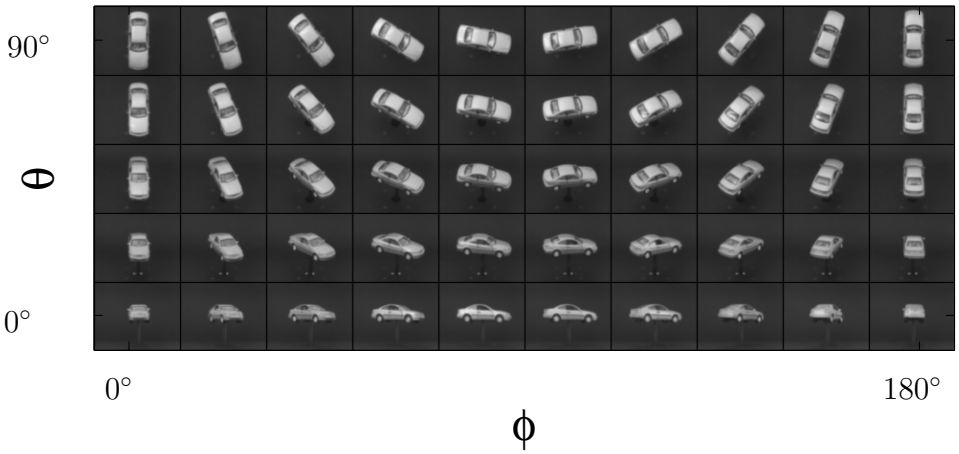


Figure 12: Example of images of a toy car from different poses. Both pose angles are varied with 20° increments.

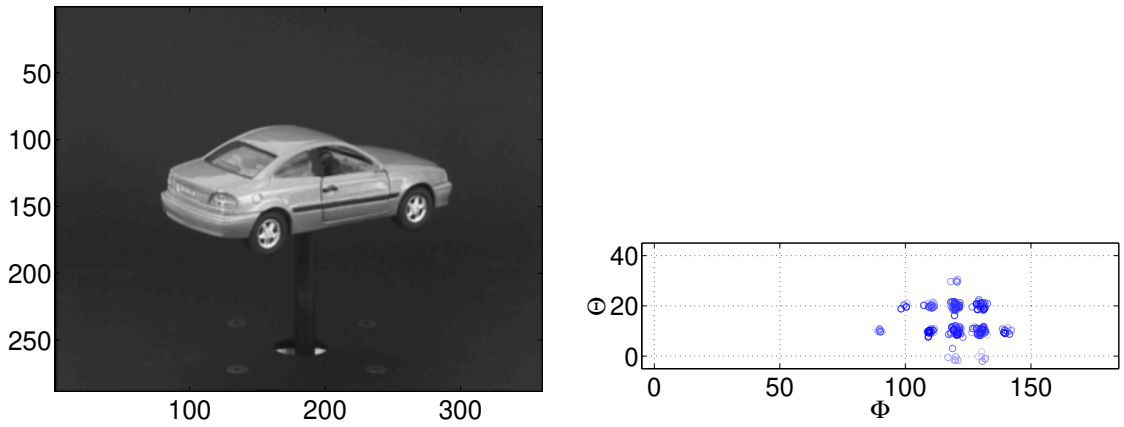


Figure 13: Evaluation image with $\phi = 125$ and $\theta = 15$, and the obtained votes. Some noise have been added to the estimates to make it possible to see the number of votes, and the color intensity represents the confidence for the votes. The votes are plotted after the verification step.

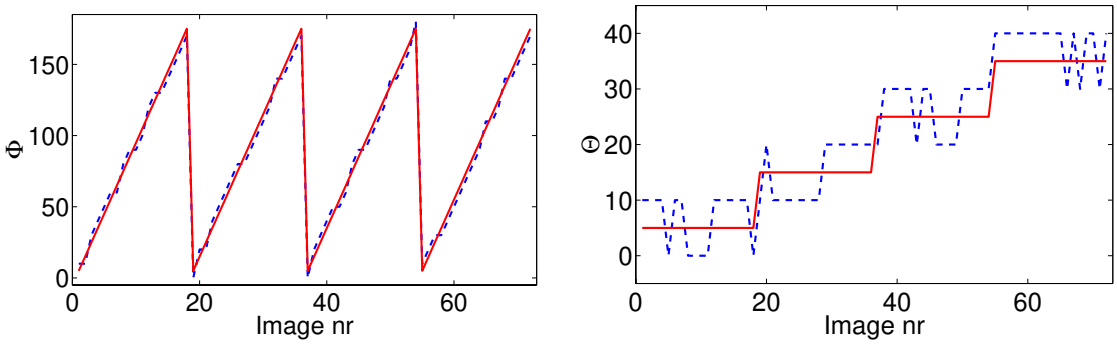


Figure 14: Estimated pose angles. Red solid: correct angle. Blue dashed: estimated angle.

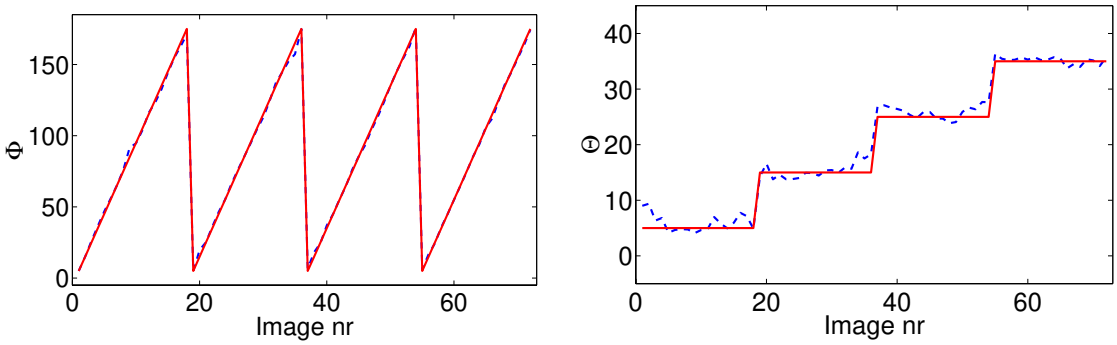


Figure 15: Estimated pose angles with interpolation. Red solid: correct angle. Blue dashed: estimated angle.

1.06° for the ϕ and θ angles respectively.

The results for seven images of the toy car with various backgrounds and occlusions are shown in Figures 16 - 22. The pose angles are unknown, so the view corresponding to the largest cluster is also displayed in the cases when significant clusters are obtained.

The method gives good estimates for all images, except for the image in Figure 22, where no significant cluster is obtained. Depending on the background the number of obtained duplets varies a great deal. For example, the image in Figure 16 gives 480 duplets while the image in Figure 18 gives 1166 duplets.

Tests are also made on how the performance is affected when increasing the angular distance between the prototype views. If the increment steps for the pose angles are increased from 10° to 20° and the evaluation is made on the poses specified in Table 2, one of the closest views is still always found, thus a mean absolute error of 10°, and if the interpolation is used the obtained mean absolute error is 2.66° in θ and 4.21° in ϕ. On the images with background the performance seems to be as good as for 10° increments, not shown here. With 40° increment the stability is reduced to the point that it fails for the images in Figure 20, 21 and 22. It still works for all the other images, but the obtained votes for the image in Figure 17 are very few.

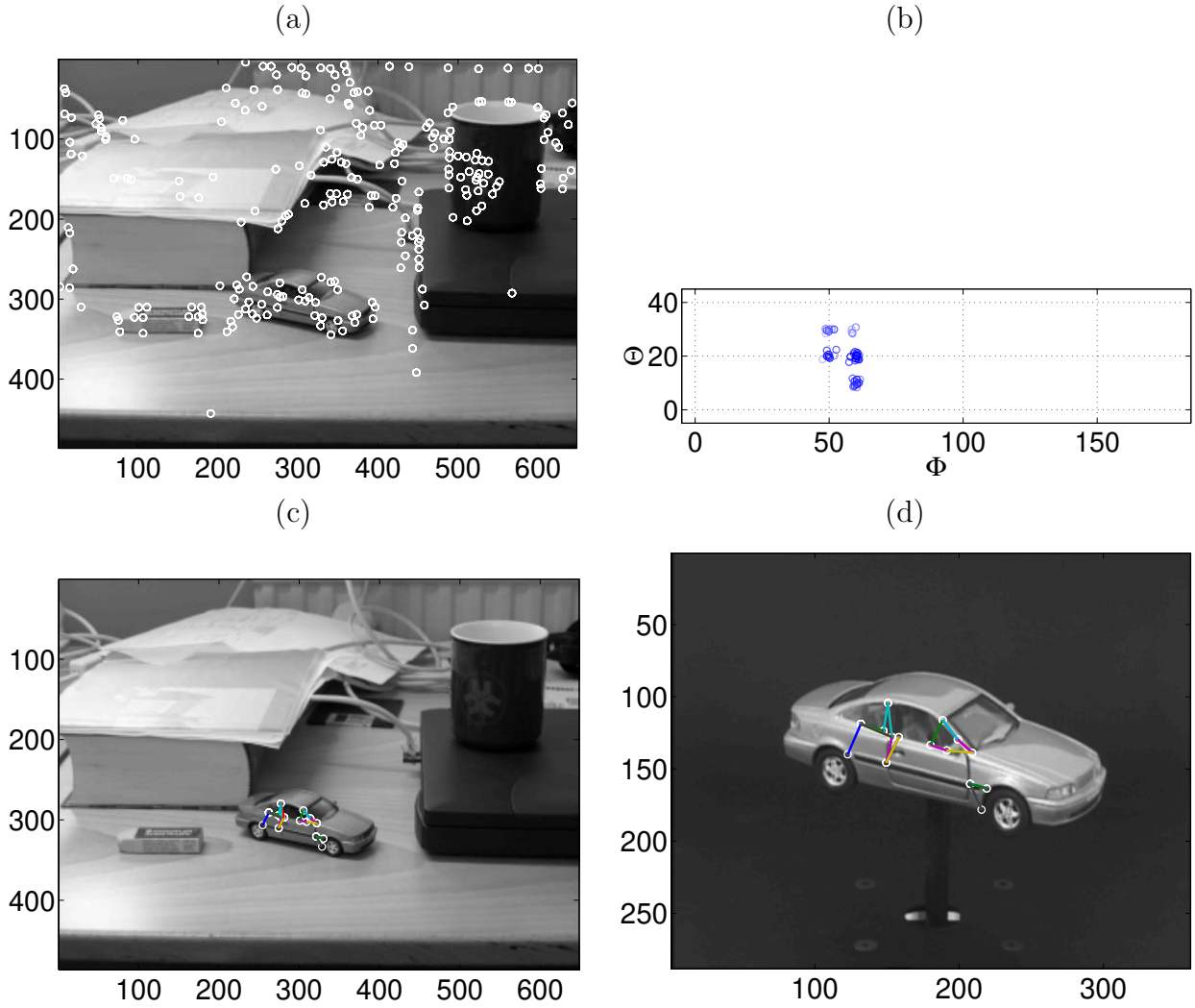


Figure 16: (a) Input image with feature points. (b) Estimates from the duplets. (c) Input image with the duplets corresponding to the votes in the largest cluster. (d) Image corresponding to the largest cluster and the duplets corresponding to the votes in that cluster.

	ϕ	θ
Training	$0°, 20°, 40°, ..., 180°$	$0°, 20°, 40°$
Evaluation	$10°, 30°, 50°, ..., 170°$	$10°, 30°$

Table 2: The used pose angles for the training and the evaluation for the second case.

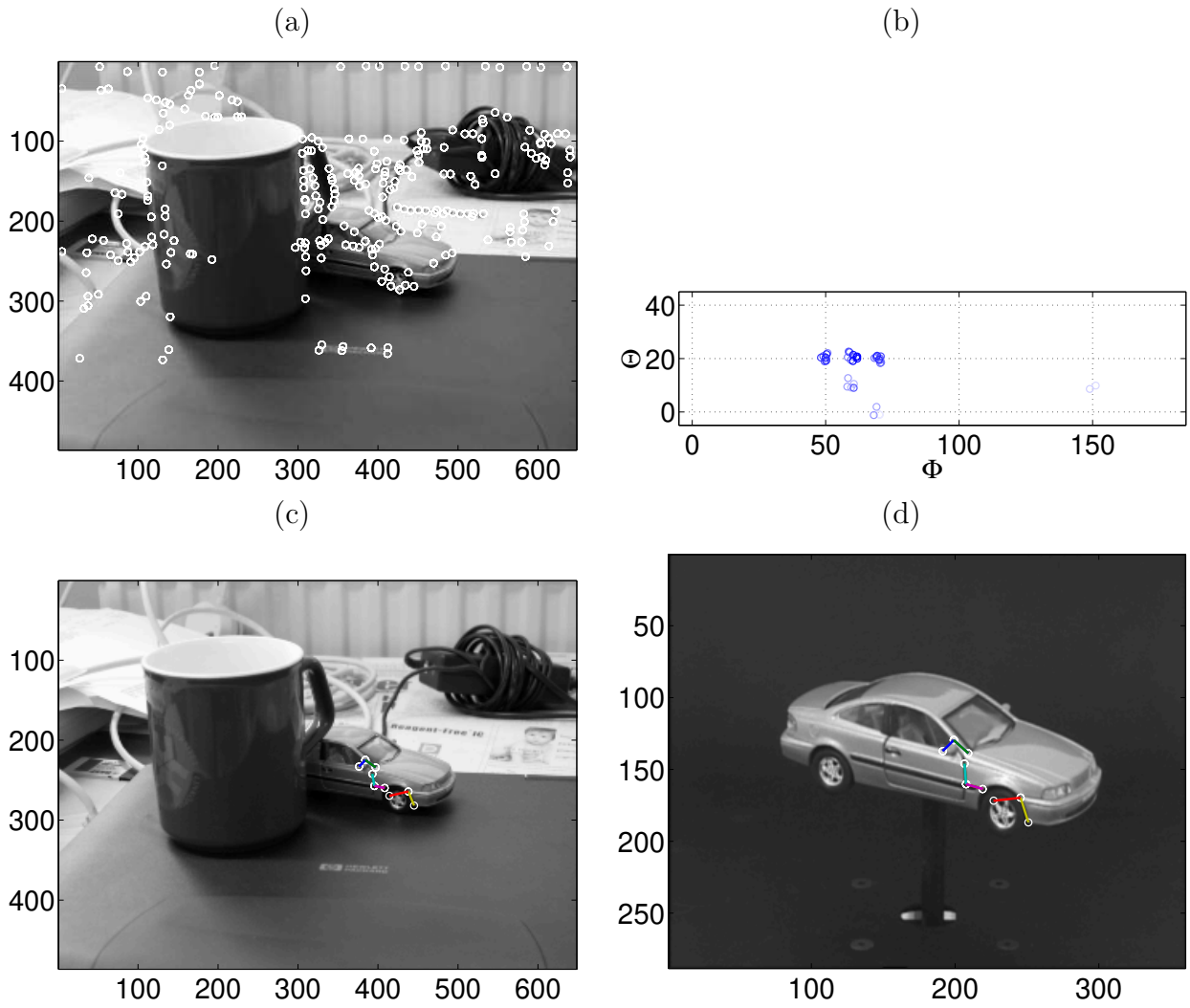


Figure 17: (a) Input image with feature points. (b) Estimates from the duplets. (c) Input image with the duplets corresponding to the votes in the largest cluster. (d) Image corresponding to the largest cluster and the duplets corresponding to the votes in that cluster.

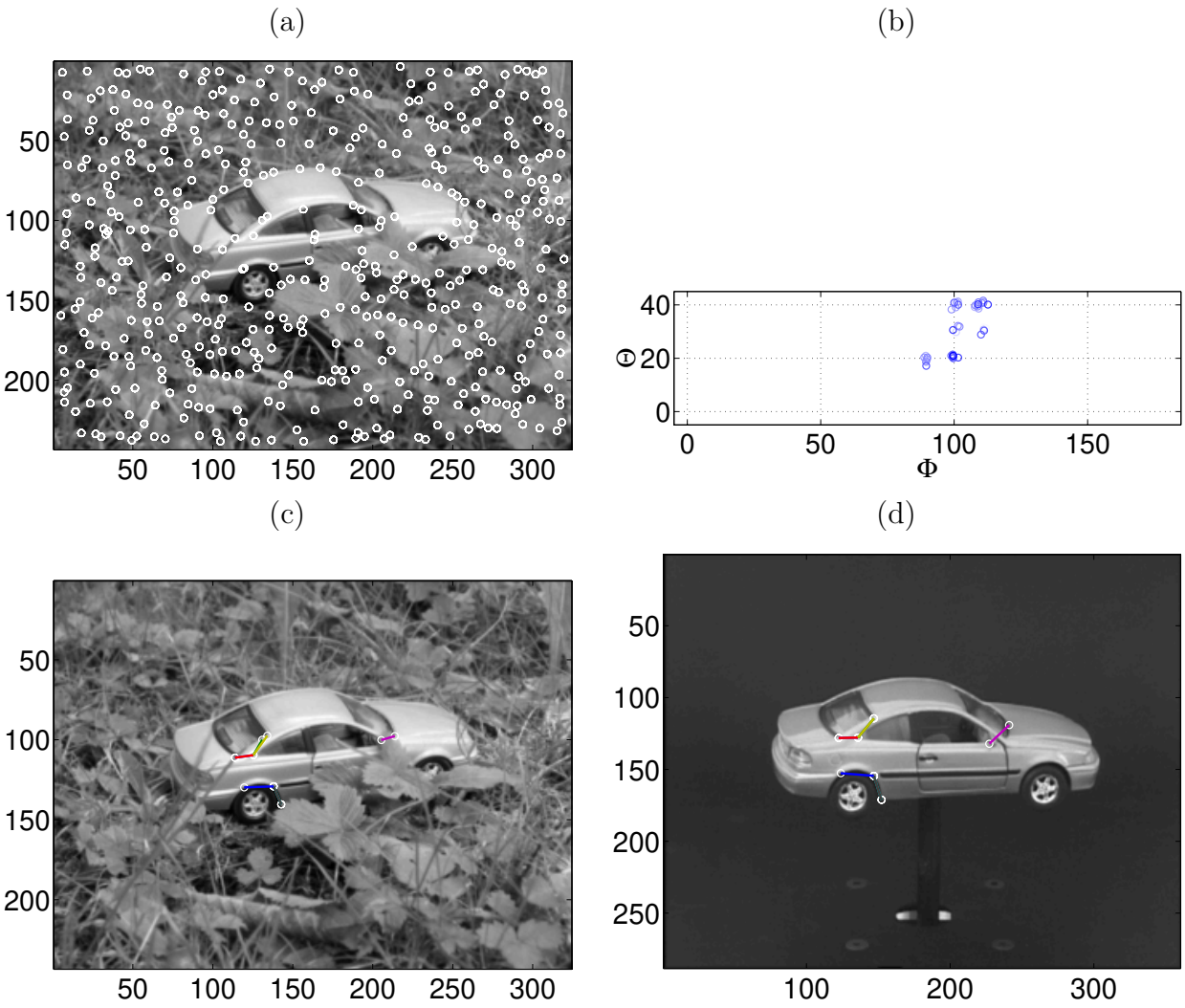


Figure 18: (a) Input image with feature points. (b) Estimates from the duplets. (c) Input image with the duplets corresponding to the votes in the largest cluster. (d) Image corresponding to the largest cluster and the duplets corresponding to the votes in that cluster.

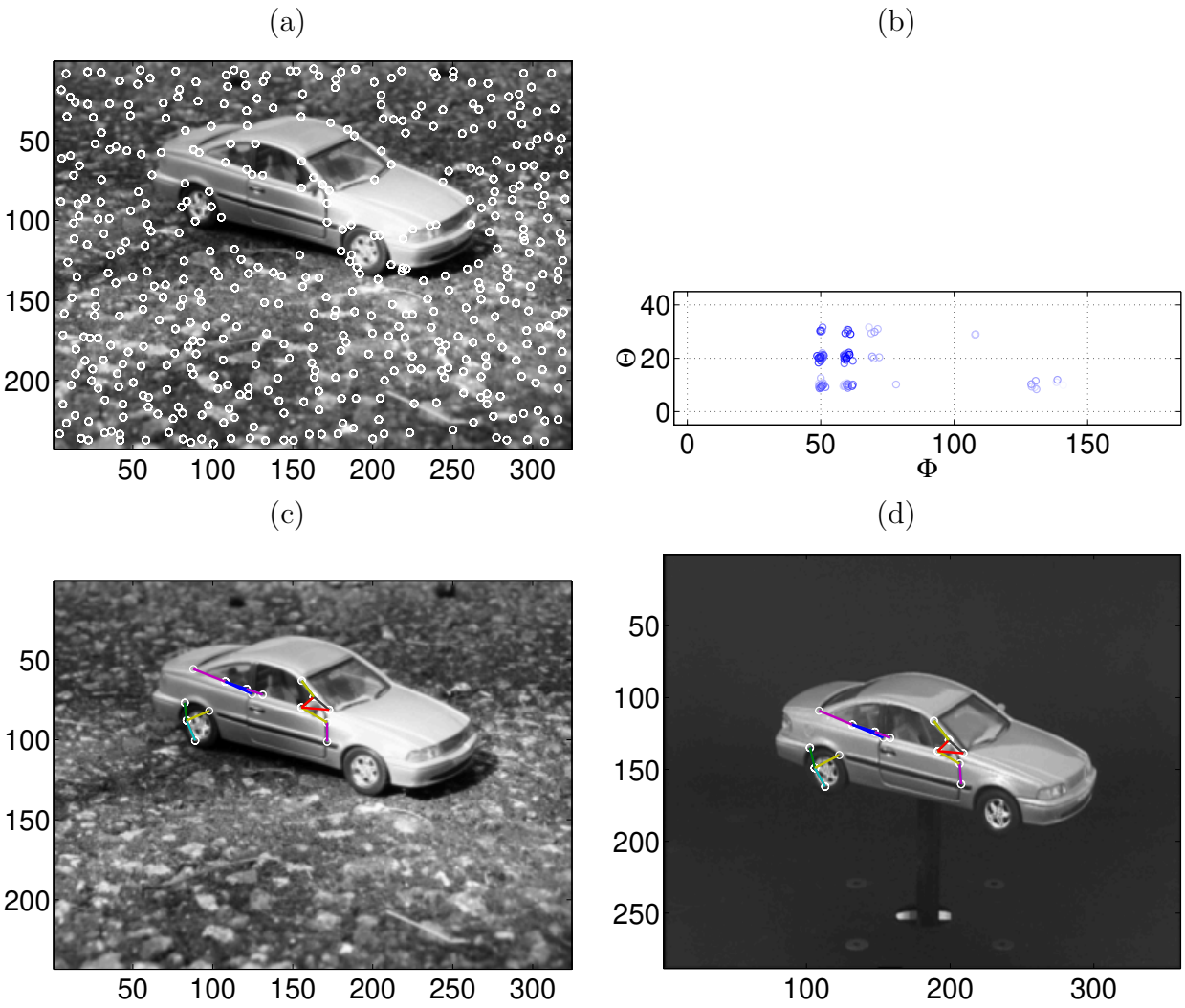


Figure 19: (a) Input image with feature points. (b) Estimates from the duplets. (c) Input image with the duplets corresponding to the votes in the largest cluster. (d) Image corresponding to the largest cluster and the duplets corresponding to the votes in that cluster.

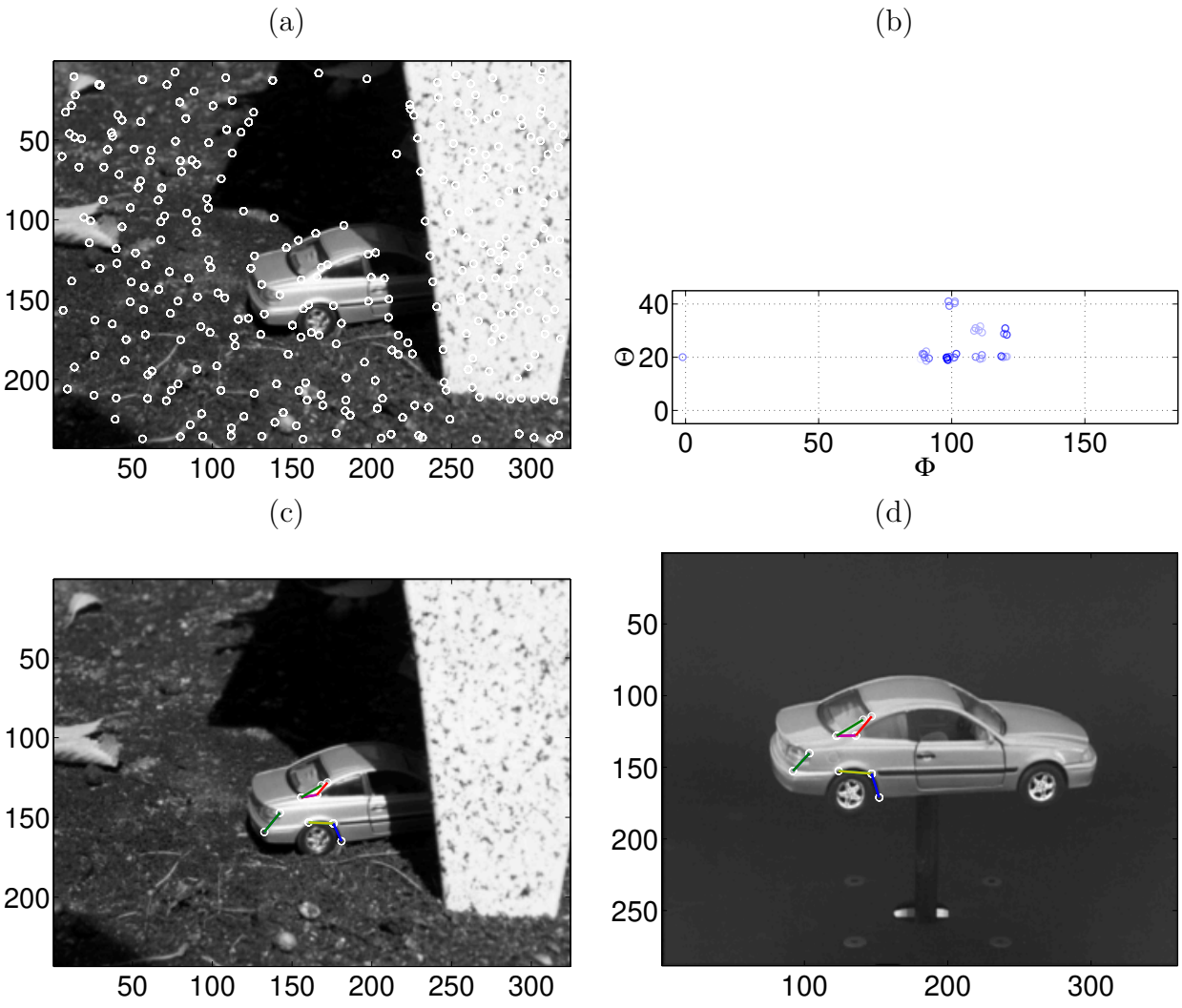


Figure 20: (a) Input image with feature points. (b) Estimates from the duplets. (c) Input image with the duplets corresponding to the votes in the largest cluster. (d) Image corresponding to the largest cluster and the duplets corresponding to the votes in that cluster.

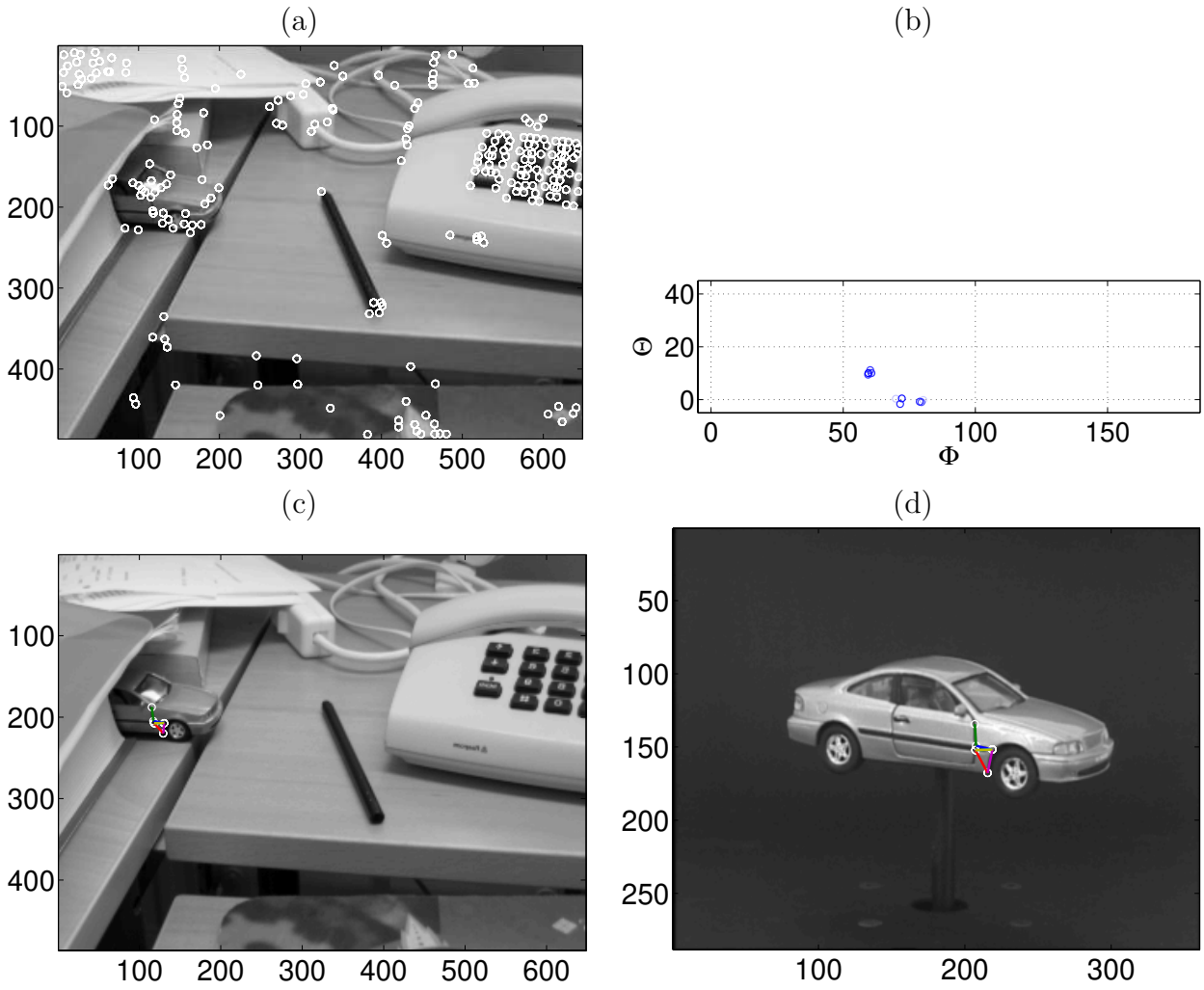


Figure 21: (a) Input image with feature points. (b) Estimates from the duplets. (c) Input image with the duplets corresponding to the votes in the largest cluster. (d) Image corresponding to the largest cluster and the duplets corresponding to the votes in that cluster.

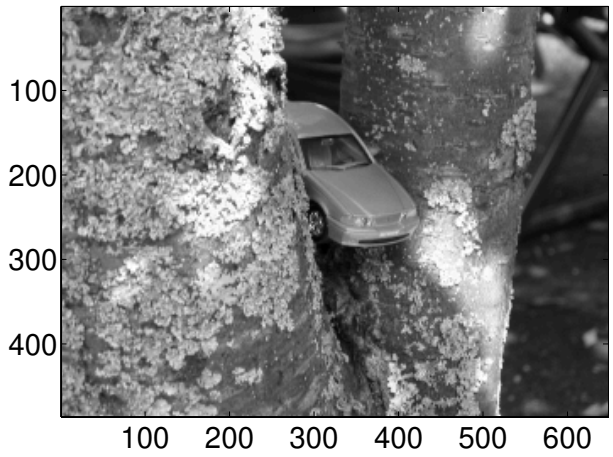


Figure 22: Input image for which the method fails.

As a final experiment we also tested the method on an image with a real car of the same model, see Figure 23. The result seems promising.

Only the pose estimates of the object have been shown here. However, the position, scale and orientation of the object can easily be estimated from the found correspondences by computing how the duplets have been translated, rotated, and scaled between the found prototype image and the query image.

7 Conclusions and Discussion

We have presented an algorithm for object recognition and object pose estimation that has shown promising results on the test images in the experiments. But many details and steps in the algorithm can be replaced and improved, and some of the steps involved may not even be necessary in order to make the system work (e.g. the orientation inhibition). We may for example use other interest points, e.g. Harris points. But the number of points generated by the Harris detector is often rather high, which makes the algorithm more computationally complex. The star-patterns used here need to be further evaluated with respect to the alleged scale invariance. The feature matching step can be improved by using more intelligent matching algorithms, e.g. vector quantization. One topic for future research is also to use some type of learning structure, e.g. [5], which may aid in the matching, clustering, and interpolation procedures.

An idea to reduce the number of duplets may also be to use visual keys such as color, texture, depth, and motion.

More careful design of the training data may also improve the system. For example, as can be seen in Figure 13, the pole on which the car is mounted is visible, which will give rise to false features. Furthermore, the light source used for the training data should be more diffuse in order to avoid sharp reflexes in the car windows and chassis.

Nothing has been mentioned about computational complexity. The algorithm is implemented in Matlab and takes several minutes for a query image. However, it should be

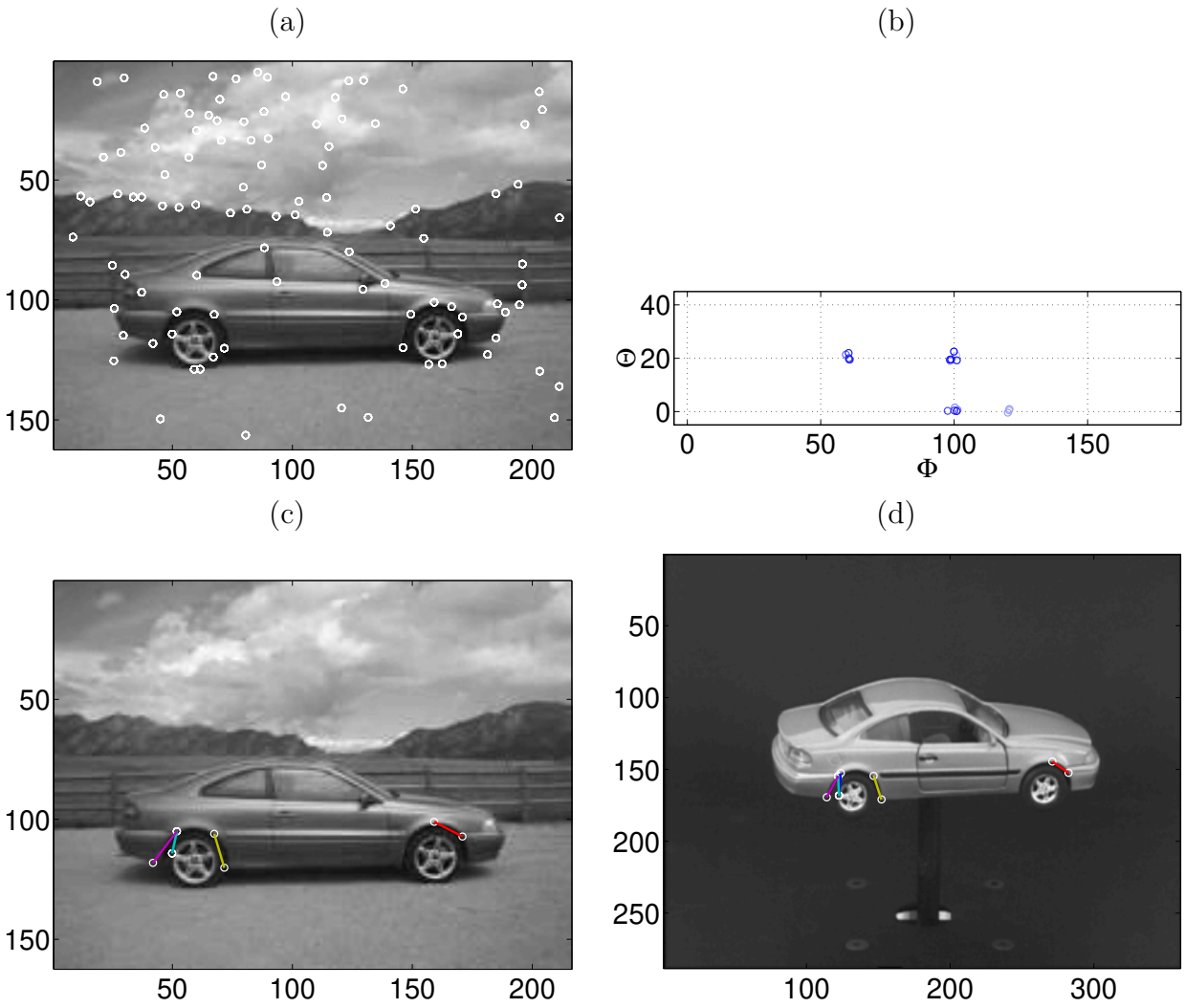


Figure 23: (a) Input image with feature points. (b) Estimates from the duplets. (c) Input image with the duplets corresponding to the votes in the largest cluster. (d) Image corresponding to the largest cluster and the duplets corresponding to the votes in that cluster.

possible to reduce this time a great deal. For example, Lowe [11, 12] reports a running time of about a second on a system with similar complexity.

Finally, preliminary experiments suggest that projection of the feature vector onto the largest PCA components can improve the performance in the case of sparsely sampled prototype views. The projected feature vectors are more slowly varying, which allows for a larger distance between the query image and the closest prototype view in the matching process. However, a slower variation also means that the projected feature vectors are less informative and less selective.

Acknowledgments

We gratefully acknowledge the support from the Swedish Research Council through a grant for the project *A New Structure for Signal Processing and Learning*, from SSF within the VISCOS project (VISion in COgnitive Systems), and from WITAS (Wallenberg laboratory on Information Technology and Autonomous Systems).

References

[1] J. Bigün. *Local Symmetry Features in Image Processing.* PhD thesis, Linköping University, Sweden, 1988. Dissertation No 179, ISBN 91-7870-334-4.

[2] J. Bigün. Pattern recognition in images by symmetries and coordinate transformations. *Computer Vision and Image Understanding*, 68(3):290–307, 1997.

[3] W. Förstner. A framework for low level feature extraction. In *Proceedings of the third European Conference on Computer Vision*, volume II, pages 383–394, Stockholm, Sweden, May 1994.

[4] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision.* Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.

[5] Gösta Granlund, Per-Erik Forssén, and Björn Johansson. HiperLearn: A high performance learning architecture. Technical Report LiTH-ISY-R-2409, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, January 2002.

[6] Gösta H. Granlund and Anders Moe. Unrestricted recognition of 3-D objects using multi-level triplet invariants. In *Proceedings of the Cognitive Vision Workshop*, Zürich, Switzerland, September 2002. URL: http://www.vision.ethz.ch/cogvis02/.

[7] Gösta H. Granlund and Anders Moe. Unrestricted Recognition of 3-D Objects for Robotics Using Multi-Level Triplet Invariants. *Artificial Intelligence Magazine*, 2003. To appear.

[8] C.G. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, September 1988.

[9] B. Johansson. Multiscale curvature detection in computer vision. Lic. Thesis LiU-Tek-Lic-2001:14, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, March 2001. Thesis No. 877, ISBN 91-7219-999-7.

[10] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, June 2003.

[11] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV'99*, 1999.

[12] David G. Lowe. Local feature view clustering for 3D object recognition. In *Proc. CVPR'01*, 2001.

[13] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[14] A. Selinger and R. C. Nelson. A perceptual grouping hierarchy for appearance-based 3D object recognition. *Computer Vision and Image Understanding*, 76:83–92, 1999.

[15] Stephen. M. Smith and J. Michael Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[16] Hagen Spies and Per-Erik Forssén. Two-dimensional channel representation for multiple velocities. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, LNCS 2749, pages 356–362, Gothenburg, Sweden, June-July 2003.