# Shape-From-Recognition: Recognition enables Meta-data Transfer

Alexander Thomas [a] Vittorio Ferrari [b] Bastian Leibe [c]
Tinne Tuytelaars [a] Luc Van Gool [b,a]

[a] *ESAT/PSI-VISICS/IBBT, Catholic University of Leuven, Belgium*
[b] *BIWI, ETH Zürich, Switzerland*
[c] *UMIC Research Centre, RWTH Aachen University, Germany*

**Abstract**

Low-level cues in an image not only allow to infer higher-level information like the presence of an object, but the inverse is also true. Category-level object recognition has now reached a level of maturity and accuracy that allows to successfully feed back its output to other processes. This is what we refer to as *cognitive feedback*. In this paper, we study one particular form of cognitive feedback, where the ability to recognize objects of a given category is exploited to infer different kinds of meta-data annotations for images of previously unseen object instances, in particular information on 3D shape. Meta-data can be discrete, real- or vector-valued. Our approach builds on the Implicit Shape Model of Leibe and Schiele [1], and extends it to transfer annotations from training images to test images. We focus on the inference of approximative 3D shape information about objects in a single 2D image. In experiments, we illustrate how our method can infer depth maps, surface normals and part labels for previously unseen object instances.

*Key words:* Computer Vision, Object recognition, Shape-from-X

## 1 Introduction

When presented with a single image, a human observer can deduce a wealth of information, including the overall 3D scene layout, material types, or ongoing actions. This ability is only in part achieved by exploiting low-level cues such as colors, shading patterns, textures, or occlusions. At least equally important is

*Email address:* alexander.thomas@esat.kuleuven.be (Alexander Thomas).


Fig. 1. *Humans can infer depth in spite of failing low-level cues, thanks to cognitive-feedback in the brain. In the left photo, recognizing the buildings and the scene as a whole injects extra information about 3D structure (e.g. how street scenes are spatially organized, and that buildings are parallelepipeda). In turn this enables, e.g., to infer the vertical edges of buildings although they do not appear in the image, and the relative depths between the buildings. Similarly, recognizing the car and knowing car lacquer is highly reflective allows to correctly estimate the depth for the center part of the right photo, in spite of contradictive local cues.*

the inference coming from higher level interpretations, like object recognition. Even in the absence of low-level cues, one is still able to estimate depth, as illustrated by the example of Fig. 1. [1]

These observations are mirrored by neurophysiological findings, e.g. Rockland and Hoesen [2], as 'low-level' areas of the brain do not only feed into the 'high-level' ones, but invariably the latter channel their output into the former. The resulting feedback loops over the semantic level are key for successful scene understanding, see e.g. Mumford's Pattern Theory [3]. The brain seems keen to bring all levels into unison, from basic perception up to cognition.

In this work, local object characteristics and other meta-data are inferred from a single image, based on the knowledge of similar data for a set of training images of other instances of the same object class. This annotation is intensely linked to the process of object recognition and segmentation. The variations within the class are taken into account, and the observed object can be quite different from any individual training example. In our approach, pieces of annotation from different training images are combined into a novel annotation mask that matches the underlying image data. By using 3D shape information as meta-data, we are effectively able to infer approximative 3D information about recognized object instances, given just a single 2D image. As example application, take a car entering a car wash (see bottom of Fig. 14). Our technique allows to estimate the relative depth and surface orientations for each part of the car, as well as to identify the positions of the windshields, car body, wheels, license plate, headlights etc. This allows the parameters of the

[1] Melbourne skyline photo by Simon Ho

car wash line to better adapt to the specific car.

The paper is organized as follows. After discussion of related work, we recapitulate the Implicit Shape Model of Leibe et al. [1] for simultaneous object recognition and segmentation (section 3). Then follows the main contribution of this paper, as we explain how we transfer meta-data from training images to a previously unseen image (section 4), for both discrete and real-valued meta-data. We demonstrate the viability of our approach by transferring depth maps and surface orientations for cars, as well as object part labels for both cars and wheelchairs (section 5). Section 6 concludes the paper.

## 2 Related Work

Several previous examples of cognitive feedback in vision have already been implemented. Hoiem et al. [4] propose a general framework which embeds the separate mechanisms of object detection and scene geometry estimation into a cognitive loop. Objects can be more reliably detected and false-positive detections in improbable locations are filtered out based on the automatically estimated geometry of the scene (e.g. people on trees). In turn, object detections allow to improve scene geometry estimation. In [5], a similar idea is applied to images taken from a moving vehicle, using car and pedestrian detections to improve ground-plane and scene depth estimation in a city environment. However, these systems only couple recognition and crude 3D scene information (the position of the groundplane). Here we set out to demonstrate the wider applicability of cognitive feedback, by inferring 'meta-data' such as 3D object shape, the location and extent of object parts, or material characteristics, based on object class recognition. Given a set of annotated training images of a particular object class, we transfer these annotations to new images containing previously unseen object instances of the same class.

The inference of 3D information from single 2D images has been an ongoing research topic for decades. Inspired by Biederman's component theory [6], the goal initially was to infer hierarchical 3D structure for objects in a 2D image. Many of the first systems used line drawings (e.g. [7]), implicitly assuming that the problem of obtaining an accurate line drawing from arbitrary 2D images would be solved in the future. Recently, there has been a trend towards inferring qualitative, rather than detailed 3D shape from single real-world photos. Hoiem et al. [8] estimate the coarse geometric properties of an entire scene by learning appearance-based models of surfaces at various orientations. The method focuses purely on geometry estimation, without incorporating an object recognition process. This means that in a complex scene, it is impossible to infer separate object identities from the inferred scene composition. Their system relies solely on the statistics of small image patches, and is opti-

mized for a very coarse set of surface orientations and a classification between ground, vertical and sky for the entire scene. In [9], Sudderth et al. combine recognition with coarse 3D reconstruction in a single image, by learning depth distributions for a specific type of scene from a set of stereo training images. The reconstructions are limited to sparse point-cloud based models of large-scale scenes (e.g. offices), not detailed models of individual objects which are the focus of our work. In the same vein, Saxena et al. [10] are able to reconstruct coarse depth maps from a single image of an entire scene by means of a Markov Random Field. As in [8], the method relies solely on statistics of image patches, and their spatial configuration inside a typical scene. Therefore it cannot exploit knowledge about specific object types in the scene, and conversely, the presence of objects cannot be inferred from the system's output. Han and Zhu [11] obtain quite detailed 3D models from a single image. Their method uses graph representations for both the geometry of the objects and their relations to the scene. To extract the graph representation from the image and estimate the geometry, a sketch representation of the objects is generated. This limits the method to objects that can be represented by a set of lines or have prominent edges, like trees or polyhedra. Hassner and Basri [12] infer 3D shape of an object in a single image from known 3D shapes of other members of the object's class. Their method is specific to 3D meta-data though, and the object is assumed to be recognized and segmented beforehand. Their analysis is not integrated with the detection and recognition of the objects, as is ours.

The above-mentioned works all focus on the estimation of depth cues from a single image. A more general framework is the work on image analogies, where a mapping between two given images $A$ and $A'$ is transferred to an image $B$ to get an 'analogous' image $B'$. As shown in work by Hertzmann et al. [13] and Cheng et al. [14], mappings can include texture synthesis, superresolution and image transformations like blurring and artistic filters. Most closely related to our work is the mapping that is called 'texture-by-numbers', where $A$ is a parts annotation of a textured image $A'$. This allows to generate a plausible textured image from a new annotation $B$. Even though no example is shown in the cited works, it should be possible to do the inverse mapping, i.e. annotate an unseen image. However, the image analogies framework is also limited to local image statistics, and does not involve a deeper understanding of the structure of the image.

Other methods focus on segmentation only, which can be considered a specific type of meta-data. Kumar et al. [15] combine Layered Pictorial Structures with a Markov Random Field to segment object class instances. Because the LPS correspond to object parts, a rough decomposition of the object into parts can also be inferred. Unsupervised learning of segmentations for an object class has been demonstrated by Winn and Jojic [16], and Arora et al. [17]. However, it is unclear whether these methods could be extended to arbitrary meta-data.

Although our method is able to infer 3D cues for a previously unseen recognized object instance, it is still limited to the pose in which it was trained. In [18], we extended the ISM system to the multi-view case, and we are investigating the integration of that approach with the meta-data annotation presented in this paper. A number of other multi-view approaches have emerged since then. For instance, Hoiem et al. [19] have augmented their LayoutCRF with a 3D model, and demonstrate the recognition of cars from multiple viewpoints. In principle, the inferred model pose might allow to infer 3D shape information for recognized objects, but this is not explored in their paper [19]. Other methods, such as Kushal et al. [20] and Savarese and Fei-Fei [21] propose a more qualitative approach towards multi-view object class recognition, by modeling objects in different poses using loosely connected parts. This makes it more difficult to extend those systems to produce a dense annotation of the recognized object.

## 3 Object Class Detection with an Implicit Shape Model

In this section we briefly summarize the *Implicit Shape Model* (ISM) approach proposed by Leibe et al. [1], which we use as the object class detection technique at the basis of our approach (see also Fig. 2).

Given a training set containing images of several instances of a certain category (e.g. side views of cars) as well as their segmentations, the ISM approach builds a model that generalizes over intra-class variability and scale. The modeling stage constructs a codebook of local appearances, i.e. of local structures that occur repeatedly across the training images. Codebook entries are obtained by clustering image features sampled at interest point locations. Agglomerative clustering is used, and the number of codewords follows automatically by setting a threshold on the maximal distance between clusters [1]. Instead of searching for correspondences between a novel test image and model views, the ISM approach maps sampled image features onto this codebook representation. We refer to all features in every training image that are mapped to a single codebook entry as *occurrences* of that entry. The spatial intra-class variability is captured by modeling spatial occurrence distributions for each codebook entry. Those distributions are estimated by recording all locations of codebook entry occurrences, relative to the object centers (which are given as training annotation). Together with each occurrence, the approach stores a local segmentation mask, which is later used to infer top-down segmentations.

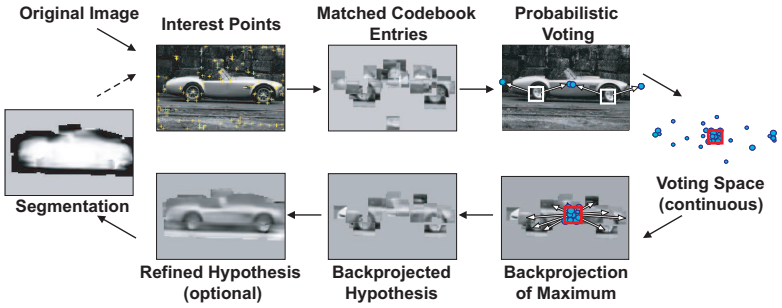


Fig. 2. *The recognition procedure of the ISM system.*

### *3.1 ISM Recognition.*

The ISM recognition procedure is formulated as a probabilistic extension of the Hough transform [1]. Let $e$ be an image patch observed at location $\boldsymbol{\ell}$. The probability that $e$ matches to codebook entry $c_i$ can be expressed as $p(c_i|e)$. Patches and codebook entries are represented by feature descriptors. In our implementation, two descriptors match if their distance or similarity (Euclidean or correlation, depending on the descriptor type), respectively, is below or exceeds a fixed threshold. Each matched codebook entry $c_i$ casts votes for instances of the object category $o_n$ at different locations and scales $\boldsymbol{\lambda} = (\lambda_x, \lambda_y, \lambda_s)$ according to its spatial occurrence distribution $P(o_n, \boldsymbol{\lambda}|c_i, \boldsymbol{\ell})$. The votes are weighted by $P(o_n, \boldsymbol{\lambda}|c_i, \boldsymbol{\ell})p(c_i|e)$, and the total contribution of a patch to an object hypothesis $(o_n, \boldsymbol{\lambda})$ is expressed by the following marginalization:

$$p(o_n, \boldsymbol{\lambda}|e, \boldsymbol{\ell}) = \sum_i P(o_n, \boldsymbol{\lambda}|c_i, \boldsymbol{\ell})p(c_i|e) \tag{1}$$

where the summation is over all entries $c_i$ in the codebook. The votes are collected in a continuous 3D voting space (translation and scale). Maxima are found using Mean Shift Mode Estimation with a kernel with scale-adaptive bandwidth and a uniform profile [22,1]. Each local maximum in this voting space yields a hypothesis that an object instance appears in the image at a certain location and scale.

### *3.2 Top-Down Segmentation.*

After the voting stage, the ISM approach computes a probabilistic top-down segmentation for each hypothesis, in order to determine its spatial support in the image. This is achieved by backprojecting to the image the votes contributing to the hypothesis (i.e. the votes that fall inside the mean-shift kernel at the hypothesized location and scale). The stored local segmentation masks are used to infer the probability that each pixel $\boldsymbol{p}$ is inside the *figure* or *ground*

area, given the hypothesis at location $\boldsymbol{\lambda}$ [1]. More precisely, the *figure* probability for $\boldsymbol{p}$ is only affected by codebook entries $c_i$ that match to a patch $e$ containing $\boldsymbol{p}$, and only by their occurrences that contribute to the hypothesis at location $\boldsymbol{\lambda}$. The probability is calculated as a weighted average over the corresponding pixels in these occurrences' segmentation masks. The weights correspond to the contribution of each occurrence to the hypothesis:

$$\begin{aligned} p\big(\boldsymbol{p} \in figure|o_n, \boldsymbol{\lambda}\big) &= \frac{1}{C_1} \sum_{e:\boldsymbol{p}\in e} \sum_i p\big(\boldsymbol{p} \in figure|e, c_i, o_n, \boldsymbol{\lambda}\big) p(e, c_i|o_n, \boldsymbol{\lambda}) \\ &= \frac{1}{C_1} \sum_{e:\boldsymbol{p}\in e} \sum_i p\big(\boldsymbol{p} \in figure|c_i, o_n, \boldsymbol{\lambda}\big) \frac{p(o_n, \boldsymbol{\lambda}|c_i) p(c_i|e) p(e)}{p(o_n, \boldsymbol{\lambda})} \end{aligned} \tag{2}$$

The priors $p(e)$ and $p(o_n, \boldsymbol{\lambda})$ are assumed to be uniformly distributed [1]. $C_1$ is a normalization term to make the equation express a true probability. The exact value of this term is unimportant because the outcome of eq. 2 is used in a likelihood ratio [1]. We underline here that a separate local segmentation mask is kept for every occurrence of each codebook entry. Different occurrences of the same codebook entry in a test image will thus contribute different local segmentations, based on their relative location with respect to the hypothesized object center.

In early versions of their work [23], Leibe and Schiele included an optional processing step, which refines the hypothesis by a guided search for additional matches (Fig. 2). This improves the quality of the segmentations, but at a high computational cost. Uniform sampling was used in [23], which became untractable once scale-invariance was later introduced into the system. Instead, in this paper we propose a more efficient refinement algorithm (section 4.3).

### *3.3 MDL Verification.*

In a last processing stage of the ISM system, the computed segmentations are exploited to refine the object detection scores, by taking only *figure* pixels into account. Moreover, this last stage also disambiguates overlapping hypotheses. This is done by a hypothesis verification stage based on Minimum Description Length (MDL), which searches for the combination of hypotheses that together best explain the image. This step prevents the same local image structure to be assigned to multiple detections (e.g. a wheel-like image patch cannot belong to multiple cars). For details, we again refer to [1].

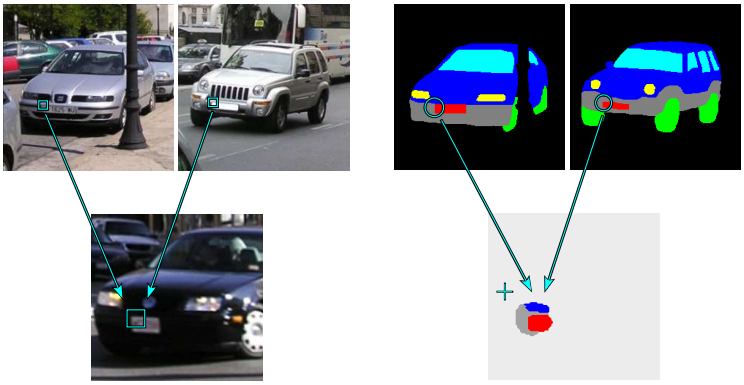

Fig. 3. *Transferring (discrete) meta-data. Left: two training images and a test image. Right: the annotations for the training images, and the partial output annotation. The corner of the license plate matches with a codebook entry which has occurrences on similar locations in the training images. The annotation patches for those locations are combined and instantiated in the output annotation.*

## 4 Transferring Meta-data

The power of the ISM approach lies in its ability to recognize novel object instances as approximate jigsaw puzzles built out of pieces from different training instances. In this paper, we follow the same spirit to achieve the new functionality of transferring meta-data to new test images.

Example meta-data is provided as annotations to the training images. Notice how segmentation masks can be considered as a special case of meta-data. Hence, we transfer meta-data with a mechanism inspired by that used above to segment objects in test images. The training meta-data annotations are attached to the occurrences of codebook entries, and are transferred to a test image along with each matched feature that contributed to a hypothesis (Fig. 3). This strategy allows us to generate novel annotations tailored to the new test image, while explicitly accommodating for the intra-class variability.

Unlike segmentations, which are always binary, meta-data annotations can be either binary (e.g. for delineating a particular object part or material type), discrete multi-valued (e.g. for identifying *all* object parts), real-valued (e.g. depth values), or even vector-valued (e.g. surface orientations). We first explain how to transfer discrete meta-data (Section 4.1), and then extend the method to the real- and vector-valued cases (Section 4.2).

### *4.1 Transferring Discrete Meta-data*

In case of discrete meta-data, the goal is to assign to each pixel $\boldsymbol{p}$ of the detected object a label $a \in \{a_j\}_{j=1:N}$. We first compute the probability $p(label(\boldsymbol{p}) =$

$a_j$) for each label $a_j$ separately. This is achieved by extending eq. (2) for $p(figure(\boldsymbol{p}))$ to the more general case of discrete meta-data:

$$p\Big(label(\boldsymbol{p}) = a_j | o_n, \boldsymbol{\lambda}\Big) = \\ \frac{1}{C_2} \sum_{\boldsymbol{p} \in N(e)} \sum_i p\Big(label(\boldsymbol{p}) = a_j | c_i, o_n, \boldsymbol{\lambda}\Big) p\Big(\hat{a}(\boldsymbol{p}) = a_e(\boldsymbol{p}) | e\Big) p(e, c_i | o_n, \boldsymbol{\lambda}) \quad (3)$$

The components of this equation will be explained in detail next. $C_2$ is again a normalization term. The first and last factors inside the summation are generalizations of their counterparts in eq. (2). They represent the annotations stored in the codebook and the voting procedure, respectively. One extension consists in transferring annotations also from image patches *near* the pixel $\boldsymbol{p}$, and not only from those *containing* it. With the original version, it is often difficult to obtain full coverage of the object, especially when the number of training images is limited. By extending the neighborhood of the patches, this problem is reduced. This is an important feature, because producing the training annotations can be labor-intensive (e.g. for the depth estimates of the cars in Section 5.1). Our notion of proximity is defined relative to the size of the image patch, and parameterized by a scale-factor $s_N$, which is 3 in all our experiments. More precisely, let an image patch $e$ be defined by its location $\boldsymbol{\ell} = (\ell_x, \ell_y, \ell_s)$ obtained from the interest point detector (with $\ell_s$ the scale). The neighborhood $N(e)$ of $e$ is defined as:

$$N(e) = \{\boldsymbol{p} | \boldsymbol{p} \in (\ell_x, \ell_y, s_N \cdot \ell_s)\} \quad (4)$$

A potential disadvantage of the above procedure is that for $\boldsymbol{p} = (p_x, p_y)$ outside the actual image patch, the transferred annotation is less reliable. Indeed, the pixel may lie on an occluded image area, or small misalignment errors may get magnified. Moreover, some differences between the object instances shown in the training and test images that were not noticeable at the local scale can now affect the results. To compensate for this, we include the second factor in eq. (3), which indicates how probable it is that the transferred annotation $a_e(\boldsymbol{p})$ still corresponds to the 'true' annotation $\hat{a}(\boldsymbol{p})$. This probability is modeled by a Gaussian, decaying smoothly with the distance from the center of the patch $e$, and with variance related to the scale of $e$ and the scale $\lambda_s$ of the hypothesis by a factor $s_G$ (1.40 in our experiments):

$$p\Big(\hat{a}(\boldsymbol{p}) = a_e(\boldsymbol{p}) \,|\, e\Big) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-({d_x}^2 + {d_y}^2)/(2\sigma^2)\right) \\ \text{with} \qquad \sigma = s_G \cdot \ell_s \cdot \lambda_s \\ (d_x, d_y) = (p_x - \ell_x, p_y - \ell_y) \quad (5)$$

Once we have computed the probabilities $p(label(\boldsymbol{p}) = a_j)$ for all possible labels $\{a_j\}_{j=1:N}$, we come to the actual assignment: we select the most likely label for each pixel. Note how for some applications, it might be better to keep the whole probability distribution $\{p(label(\boldsymbol{p}) = a_j)\}_{j=1:N}$ rather than a hard assignment, e.g. when feeding back the information as prior probabilities to low-level image processing.

An interesting possible extension is to enforce spatial continuity between labels of neighboring pixels, e.g. by relaxation or by representing the image pixels as a Markov Random Field. In our experiments (Section 5), we achieved good results already without enforcing spatial continuity.

The practical implementation of this algorithm requires rescaling the annotation patches. In the original ISM system, bilinear interpolation is used for rescaling operations, which is justified because segmentation data can be treated as probability values. However, interpolating over discrete labels such as 'windshield' or 'bumper' does not make sense. Therefore, rescaling must be carried out without interpolation.

### *4.2 Transferring Real- or Vector-valued Meta-data*

In many cases, the meta-data is not discrete, but real-valued (e.g. 3D depth) or vector-valued (e.g. surface orientation). We will first explain how we obtain a real-valued annotation from quantized training data, and then how fully continuous meta-data is processed.

#### *4.2.1 Quantized Meta-data*

If the available meta-data is quantized, we can use the discrete system as in the previous section, but still obtain a continuous estimate for the output by means of interpolation. The quantized values are first treated as a fixed set of 'value labels' (e.g. 'depth 1', 'depth 2', etc.). Then we proceed in a way analogous to eq. (3) to infer for each pixel a probability for each discrete value. In the second step, we select for each pixel the discrete value label with the highest probability, as before. Next, we refine the estimated value by fitting a parabola (a $(D + 1)$-dimensional paraboloid in the case of vector-valued meta-data) to the probability scores for the maximum value label and the two immediate neighboring value labels. We then select the value corresponding to the maximum of the parabola. This is a similar method as used in interest point detectors (e.g. [24,25]) to determine continuous scale coordinates and orientations from discrete values. Thanks to this interpolation procedure, we obtain real-valued output even though the input meta-data is quantized. The advantage of only considering the strongest peak and its immediate neighbors

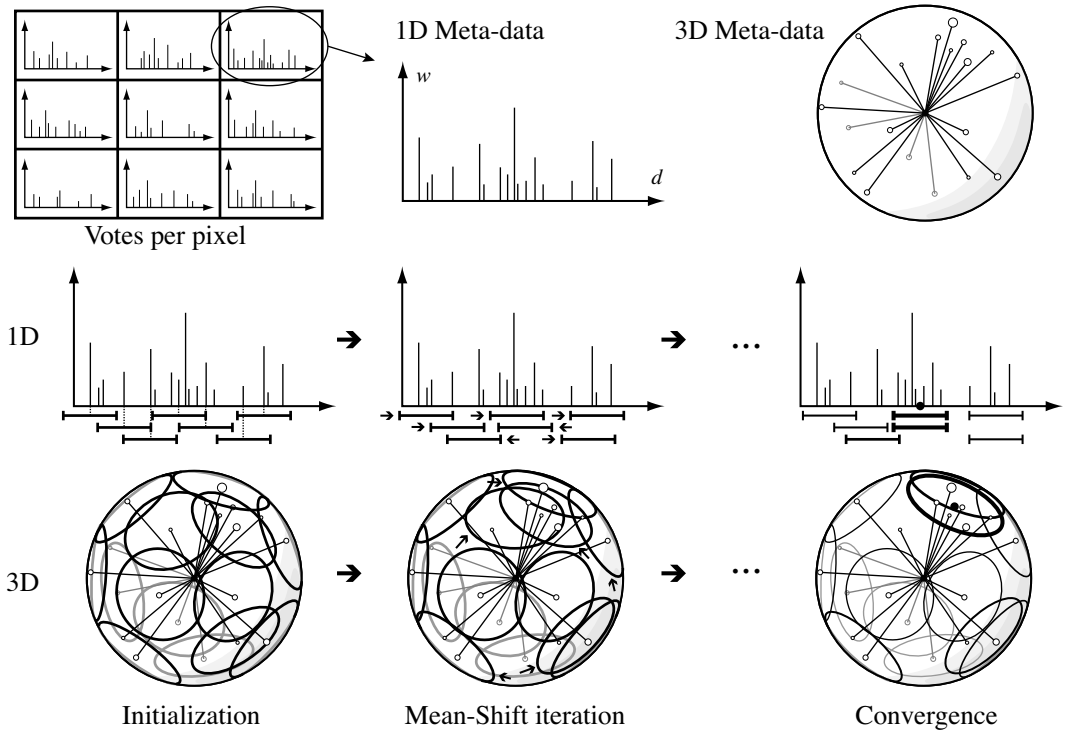


Fig. 4. *Mean-Shift mode estimation for continuous and vector-valued meta-data. The top left shows a* $3 \times 3$ *pixel fragment from an image, with 1D vote distributions for each pixel. The top right shows another possible distribution where each vote is a 3D normal vector (the size of the circles indicates the vote weights). The middle and bottom row show the Mean-Shift mode estimation procedure for both types of data. In the rightmost figures, the line width of the windows corresponds to their scores and the black dot is the final value.*

is that the influence of outlier votes is reduced (e.g. votes for discrete values far from the peak have no impact).

### *4.2.2 Continuous and Vector-valued Meta-data*

Processing fully real- or vector-valued meta-data requires a different approach. Instead of building probability maps for discrete labels, we store for each pixel all values that have been voted for, together with their vote weights. We again use Eq. 5 to decrease the influence of votes with increasing distance from their patch location. By storing all votes for each pixel we obtain a sampling of the probability distribution over meta-data values. There are several ways to derive a single estimate from this distribution. In a similar vein as in the discrete system, we could take the value with the highest weight (argmax), but this has proven in experiments to give unreliable results, because it is very sensitive to outlier votes. A better method is to take the average, but this can still be offset by outliers. A third and more robust method is to estimate the mode of the sampled distribution.

We use a Mean Shift procedure [22] with a fixed window radius to estimate the mode for each pixel. This method works for 1-dimensional as well as vector-valued data. The mode estimation procedure uses a set of candidate windows, which are iteratively shifted towards regions of higher density until convergence occurs. Because the number of votes covering each pixel is in the order of one hundred, there is no need to initialize the windows through random sampling. Instead, we cover the entire distribution with candidate windows by considering the location of each vote as a candidate window, and removing all overlapping windows. Two windows overlap if their distance is less than the window radius. Depending on the type of data, distance can be defined as Euclidean distance, or as the angle between vectors. Next, we iterate over all windows by replacing each window's position by the weighted mean of all votes within its radius, until convergence occurs. The score of a window is the sum of the weights of all its votes. The coordinates of the window with the highest score yield the position $\hat{\boldsymbol{a}}$ of the mode. The estimate for the final value for $\boldsymbol{p}$ can be formulated as:

$$\hat{\boldsymbol{a}}(\boldsymbol{p}) = \underset{\boldsymbol{a}}{\operatorname{argmax}} \sum_{\mathbf{a}_i | d(\boldsymbol{a}, \boldsymbol{a}_i(\boldsymbol{p})) < \theta} w\Big(\boldsymbol{a}_i(\boldsymbol{p})\Big) \tag{6}$$

The scalar or vector value $\boldsymbol{a}_i(\boldsymbol{p})$ expresses the $i$-th vote for the value of pixel $\boldsymbol{p}$. There are as many votes as there are patches in the image that contribute to the pixel $\boldsymbol{p}$. Their weights $w\Big(\boldsymbol{a}_i(\boldsymbol{p})\Big)$ correspond to the weights of the object center votes in the Hough space cast by those patches, scaled by Eq. 5. The function $d(\boldsymbol{x}, \boldsymbol{y})$ is a distance measure between meta-data values (e.g. Euclidean distance or angle) and $\theta$ is the mean-shift window radius.

In case there are multiple modes with the same score, we take the average position (this occurs rarely in our experiments). The label 'background' is assigned if the score of the window around $\hat{\boldsymbol{a}}$ is smaller than the sum of the weights of background votes.

Figure 4 illustrates the mode estimation procedure for both 1-dimensional meta-data (e.g. depth values) and 3-dimensional normal vectors. In the latter case, the windows are circles on a unit sphere, and the distance measure between the votes and windows is the angle between their vectors. When updating the window positions, care must be taken to keep the resulting vectors normalized. When the meta-data consists of vectors that need to be compared using Euclidean distance (e.g. 3D points), the windows are (hyper)spheres of the same dimensionality as the vectors.

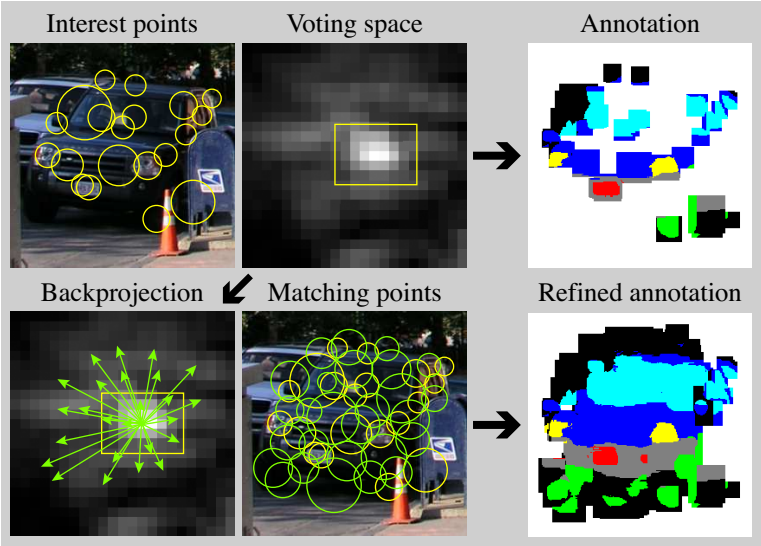


Fig. 5. *Refining a hypothesis. An image with poor contrast (top left) produces insufficient interest points to cover the whole object (top right). By backprojecting the occurrence locations from the detected peak in the Hough space (bottom left), additional points can be found (bottom center), and a more complete annotation can be constructed (bottom right).*

### *4.3 Refining Hypotheses*

When large areas of the object are insufficiently covered by interest points, no meta-data can be assigned to them. Using a large value for $s_N$ will only partially solve this problem, because there is a limit as to how far information from neighboring points can be reliably extrapolated. A better solution is to actively search for additional codebook matches in these areas. The refinement procedure in early, fixed-scale versions of the ISM system [23] achieved this by means of uniform sampling. A dense 2D grid of candidate points was generated around the hypothesis, which is intractable in the scale-invariant (3D) case. Therefore, we have developed a more efficient refinement algorithm which only searches for matches in promising locations.

For each hypothesis, new candidate points are generated by backprojecting all occurrences in the codebook, excluding points nearby existing interest points. We define two interest points to be nearby, if there is more than 85% mutual overlap between the neighborhoods over which their feature descriptors are computed. When the feature descriptor for a new point matches with the codebook cluster(s) that backprojected it, an additional hypothesis vote is cast. The confidence for this new vote is reduced by a penalty factor to reflect the fact that it was not generated by an actual interest point. In all our experiments, we use a penalty factor of 0.5. The additional votes enable the meta-data transfer to cover those areas that were initially missed by the interest point detector. This procedure is illustrated in Fig. 5.

This refinement step can either be performed on the final hypotheses that result from the MDL verification, or on all hypotheses that result from the initial voting. In the latter case, it will improve MDL verification by enabling it to obtain better figure area estimates of each hypothesis [1]. Therefore, we perform refinement on the initial hypotheses in all our experiments.

## 5 Experimental evaluation

We evaluate our approach on two object classes: cars and wheelchairs. For cars, we recover three types of annotations. The first is a 3D depth map, indicating for each pixel the distance from the camera (a real-valued labeling problem). The second is an orientation map, representing the surface normal for each pixel. This is a vector-valued labeling problem. We stress that both these results are achieved from a single image of a previously unseen car. In the third experiment, we aim at decomposing the car in its most important parts (wheels, windshield, etc.), which is a discrete labeling problem. We perform a similar part decomposition experiment on the wheelchairs. We first perform a series of experiments on controlled images to assess the annotation quality only. Then we show results on challenging images which demonstrate the recognition ability of our system.

### *5.1 Inferring 3D Shape*

In our first experiment, we infer 3D information, consisting of a depth map and surface orientations, as meta-data for the object class 'car'. A possible application is an automated car wash. Even though such systems mostly have sensors to measure distances to the car, they are only used locally while the machine is already running. It could be useful to optimize the washing process beforehand, based on the car's global shape (both depth and orientations) inferred by our system.

Our dataset is a subset of that used in [5]. It was obtained from the LabelMe website [26], by extracting images labeled as 'car' and sorting them according to their pose. For our experiments, we only use the 'az300deg' pose, which is a semi-profile view. In this pose parts from both the front (windscreen, headlights, license plate) and side (wheels, windows) are visible. Moreover, this is the least planar view, resulting in more interesting depth/orientation maps compared to purely frontal or side views. Note that while each ISM detector is only trained for a single viewpoint, it can be extended to handle multiple viewpoints [18]. The dataset contains a total of 139 images. We randomly picked 79 for training, and 60 for testing. We train an ISM system using

Fig. 6. *Obtaining depth and orientation maps for the car training images. Left shows the original image, middle the image with the best matching 3D model superimposed. At the right, the extracted depth map is shown for the top image, and the orientation map for the bottom image.*

the Hessian-Laplace interest point detector [27] and Shape Context descriptors [28], because this combination has been shown to perform best in [29]. The resulting codebook has 1576 entries, with a total of 84148 occurrences.

To obtain training and ground-truth data for both the depth and orientation maps, we manually align a 3D model on top of each training image. The most suitable 3D model for each image is selected from a freely available collection [2] (Figure 6). Depth is extracted from the OpenGL Z-buffer. In general, any 3D scanner or active lighting setup could be used to automatically obtain 3D shape annotations during training. We normalize the depths based on the dimensions of the 3D models by assuming that the width of a car is approximately constant. Orientations are encoded by mapping each surface normal vector $\boldsymbol{n} = (x, y, z)$ to a 24 bit color $\boldsymbol{c} = (r, g, b)$ (e.g. with a fragment shader):

$$\boldsymbol{c} = 255 \cdot \Big(\boldsymbol{n} + (0.5, 0.5, 0.5)\Big) \tag{7}$$

We test the system on the 60 test images, using the real-valued method from Section 4.2.2. For the Mean Shift mode estimation, we use a window radius $\theta$ of 24% of the total depth range, and 60 degrees for the orientations. The goal of this first experiment is to assess the quality of the annotations only, not the recognition performance, which will be demonstrated in Section 5.2. Because each image only contains one object, we therefore select the detection with highest score for meta-data transfer. Some of the resulting annotations can be seen in the third and fifth columns of figure 7.

[2] http://dmi.chez-alice.fr/models1.html

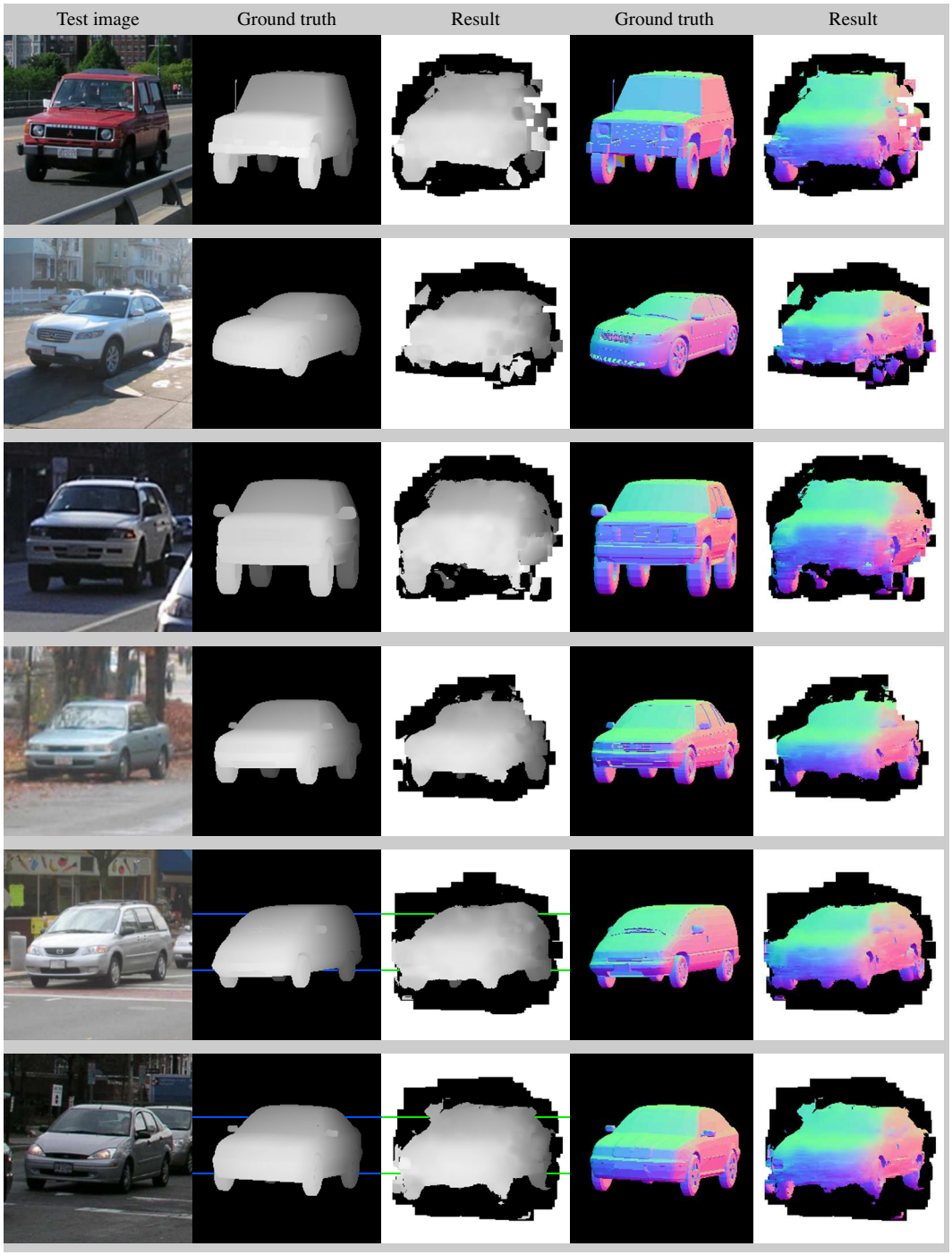


Fig. 7. *Results for the car depth map and surface orientation experiments. From left to right: test image, ground-truth and output of our system for the depth map experiment, and ground-truth and output for the surface orientation experiment. The R,G,B colors represent the components of the surface normal according to Eq. 7. White areas are unlabeled and can be considered background.*

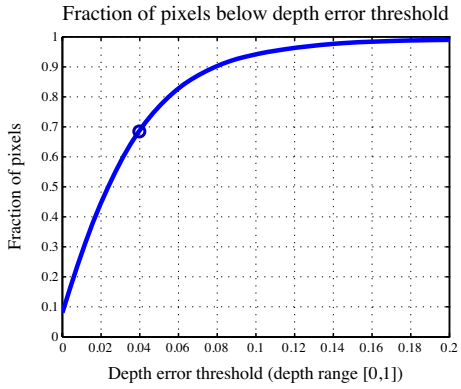


Fig. 8. *The fraction of pixels that are both labeled as 'figure' in ground-truth and output, that are below the absolute depth error threshold on the horizontal axis. The circle indicates the average depth error.*

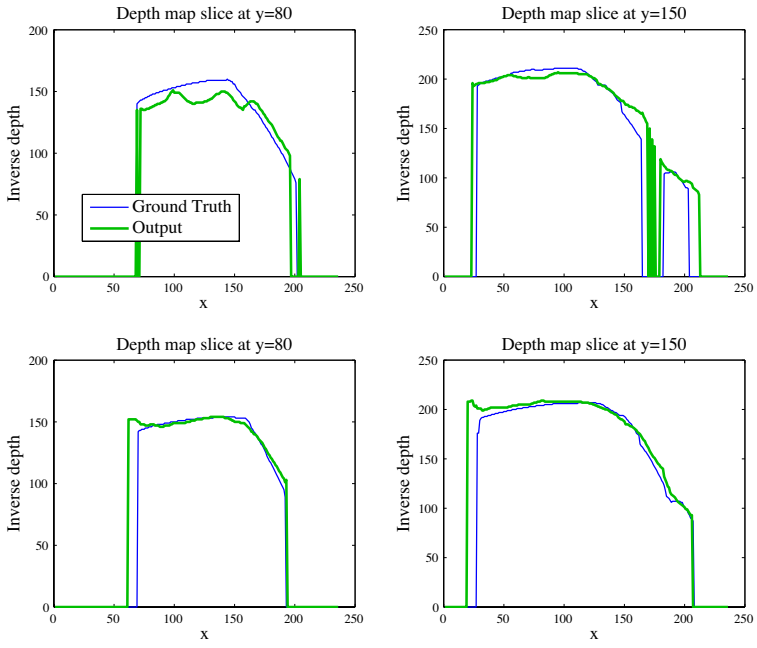


Fig. 9. *Horizontal slices through the ground-truth and output depth maps of the fifth car (top) and sixth car (bottom) in Fig. 7.*

To evaluate this experiment quantitatively, we use the ground-truth annotations to calculate the following error measures. We define *leakage* as the percentage of background pixels in the ground-truth annotation that were labeled as non-background by the system. The leakage for both the depth map and orientation experiments, averaged over all test images, is 5.7%. We also define a *coverage* measure, as the percentage of non-background pixels in the ground-truth images labeled non-background by the system. The coverage obtained by our algorithm is 94.6%. This means our method is able to reliably segment the car from the background.

All training and ground-truth depth maps are scaled to the interval $[0, 1]$ such

Fig. 10. *Some views of a texture mapped 3D model, generated from the depth map of the recognized car in the top left corner.*

that their depth range is 3.5 times the width of the car. The average absolute value of the depth error is 3.94% of this total range. This is only measured inside areas which are labeled non-background in both the ground-truth and result images, because the depth is undefined for the background. Fig. 8 shows how the fraction of this area varies in function of increasing absolute error threshold. It is possible to estimate the depth error in real-world units, by scaling the normalized depth maps by a factor based on the average width of a real car, which we found to be approximately 1.80 m. A plausible real-world depth error can be calculated by multiplying the relative error measure by $3.5 \cdot 1.80$m, which yields 24.8 cm for the mean absolute error. To better visualize how the output compares to the ground-truth, Fig. 9 shows a few horizontal slices through two depth maps of Fig. 7. Fig. 10 shows some views of a 3D model created by mapping the image of the recognized car onto the depth map produced by our system.

To compare results between fully continuous meta-data and using quantized meta-data, we repeated the depth map experiment with the depth maps quantized to 20 levels. The interpolating method from Section 4.2.1 was used to obtain a continuous result. For this experiment, the leakage is 4.8%, the coverage 94.6% and the depth error estimate 26.7 cm. This shows that although the discrete method performs already well, the continuous method does better, and should be used whenever possible.

For the surface orientation experiment, we can calculate the average angular error over the area that is labeled foreground in both the ground-truth and test image. The average error over all test images is 23.3 degrees.

We examined the influence of the number of training images on annotation performance, by repeating the depth map experiment with ISMs trained from fewer images. We sorted the images according to the numbers of occurrences they have in the 79-image ISM. Images with fewer occurrences are removed

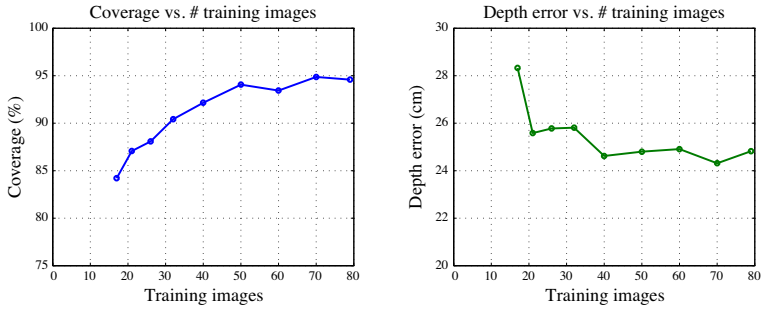


Fig. 11. *Evolution of coverage and depth error within non-background areas, in function of the number of training images for the car depth map experiment.*

first. We train ISMs in an identical way as described above, using less and less training images, from 70 down to 17. Figure 11 shows how the coverage and depth error within non-background areas evolve with varying numbers of training images. Performance remains comparable to the 79-image ISM, even when training from only 50 images. Below this point, both coverage and depth error start getting worse slowly. At 26 images, the position and scale of some detections starts deviating, and at 17 images, some cars cannot be recognized at all. This shows that the performance of the system degrades gracefully with the number of training images, and that it performs well even with considerably less than the original 79 images.

### *5.2 Object Decomposition*

In further experiments, the goal is to delineate certain areas of interest on the objects, which is a discrete annotation task. For our car wash scenario, a decomposition into parts would allow different washing methods to be applied to different car parts. For the class of wheelchairs, a possible application is a service robot. This robot's task could be to retrieve a wheelchair, for instance in a hospital or to help a disabled person at home. In order to retrieve the wheelchair, the robot must be able to both detect it and determine where to grab it. Our method will help the robot to get close to the grabbing position, after which a detailed analysis of scene geometry in a small region can establish the grasp [30].

We annotated our car dataset with ground-truth part segmentations for body, windshield/windows, wheels, bumper, lights and license plate. Aside from the different meta-data, the training phase is identical to the one in Section 5.1. The testing phase is performed with the method presented in Section 4.1. Results are shown in Fig. 12. The leakage for this experiment is 6.83% and coverage is 95.2%.

We report a quantitative evaluation for this experiment in the form of a con-

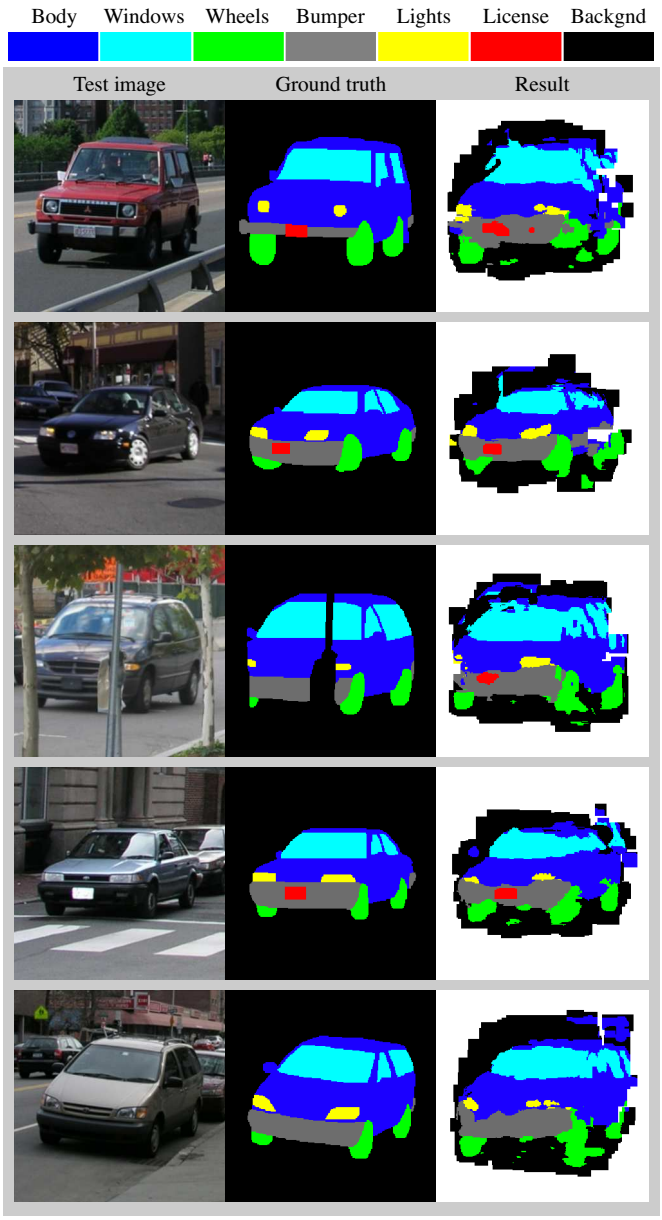


Fig. 12. *Results for the car parts annotation experiment. From left to right: test image, ground-truth, and output of our system. White areas are unlabeled and can be considered background.*

fusion matrix. For each test image, we count how many pixels of each part $a_j$ in the ground-truth image are labeled by our system as each of the possible parts (body, windows, etc.), or remain unlabeled (which can be considered background in most cases). This score is normalized by the total number of pixels of that label in the ground-truth $\hat{a}_j$. Table 1 shows the confusion table entries averaged over all test images. The diagonal elements show how well each part was recovered in the test images. Labeling performance is good, except for the headlights. This is due to the fact that they are the smallest parts in most of the images. Small parts have a higher risk of being confused

| | bkgnd | body | bumper | headlt | window | wheels | license | unlabeled |
|---|---|---|---|---|---|---|---|---|
| bkgnd | **23.56** | 2.49 | 1.03 | 0.14 | 1.25 | 1.88 | 0.04 | **69.61** |
| body | 4.47 | **72.15** | 4.64 | 1.81 | 8.78 | 1.86 | 0.24 | 6.05 |
| bumper | 7.20 | 4.54 | **73.76** | 1.57 | 0.00 | 7.85 | 2.43 | 2.64 |
| headlt | 1.51 | 36.90 | 23.54 | **34.75** | 0.01 | 0.65 | 0.23 | 2.41 |
| window | 3.15 | 13.55 | 0.00 | 0.00 | **80.47** | 0.00 | 0.00 | 2.82 |
| wheels | 11.38 | 6.85 | 8.51 | 0.00 | 0.00 | **63.59** | 0.01 | 9.65 |
| license | 2.57 | 1.07 | 39.07 | 0.00 | 0.00 | 1.04 | **56.25** | 0.00 |

Table 1
*Confusion matrix for the car parts annotation experiment. The rows represent the annotation parts in the ground-truth maps, the columns the output of our system. The last column shows how much of each class was left unlabeled. For most evaluations, those areas can be considered 'background'.*

with the larger parts (body, bumper) in their neighborhood.

For a second part decomposition experiment, we collected 141 images of wheelchairs from Google Image Search. We again chose semi-profile views, because they are the most complex and most widely available views. All images are annotated with ground-truth part segmentations for grab area, wheels, armrests, seat, and frame. In our assistive robot scenario, the grab area is the most important one. We included the rear right wheels in the 'frame' label, for two reasons. First, that wheel is often heavily or completely occluded by the frame itself. Second, this illustrates how our system can differentiate between similar-looking structures, based on their position on the object. A few representative images and their ground-truth annotations can be seen in the left and middle columns of Fig. 13.

The images are randomly split into a training and test set. We train an ISM system using 80 images in a similar way as for the car experiments. The codebook has 4289 entries, with a total of 133138 occurrences. Next, we test the system on the remaining 61 images, using the method from Section 4.1. Some of the resulting annotations can be seen in the third column of Fig. 13. The grab area is accurately localized.

With a leakage of 3.75%, and a coverage 95.1%, the segmentation performance is again very good. The confusion table is shown in Table 2. Not considering the armrests, the system performs well as it labels correctly between 67% and 77% of the pixels, with the highest score being for the part we are the most interested in, i.e. the grab area. The lower performance for the armrests is again due to the fact that they are the smallest parts in most of the images.

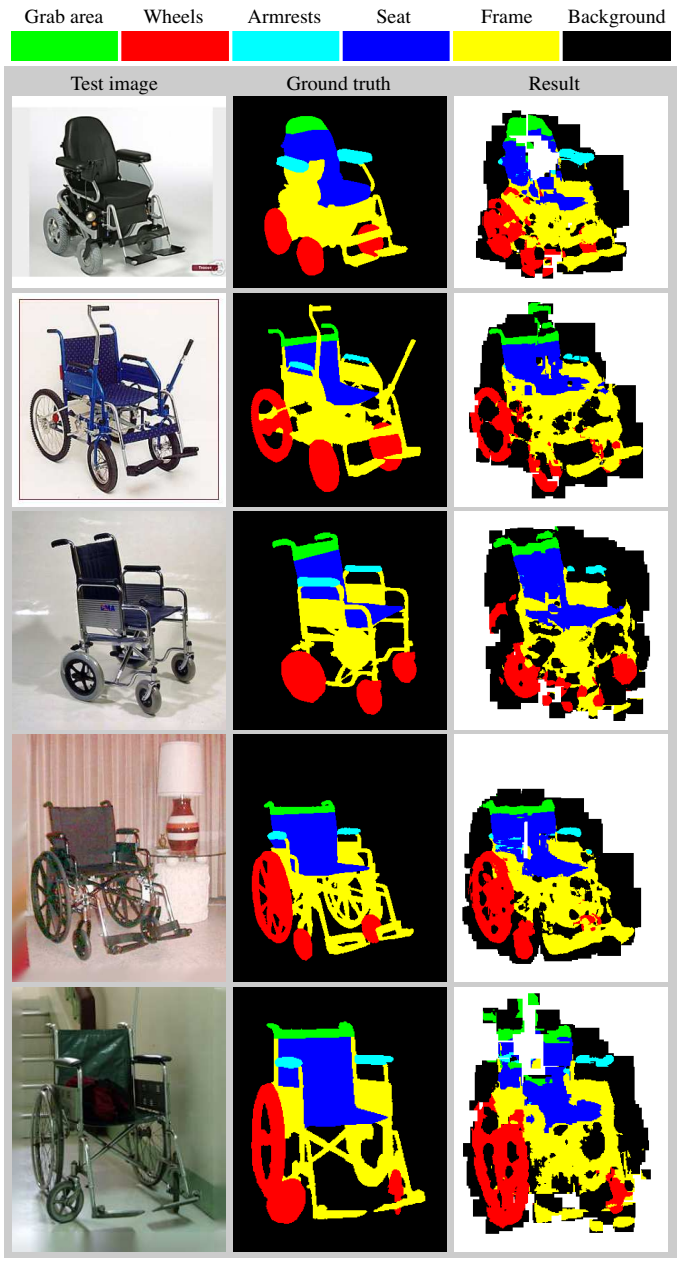


Fig. 13. *Results for the annotation experiment on wheelchair images. From left to right: test image, ground-truth, and output of our system. White areas are unlabeled and can be considered background.*

### *5.3 Combined recognition and annotation in cluttered images*

To illustrate the ability to simultaneously detect objects in cluttered scenes and infer meta-data annotation, we have performed the part decomposition experiment on challenging real-world images for both the car and wheelchair classes. Results for the cars are shown in Fig. 14.

| | bkgnd | frame | seat | armrest | wheels | **grab-area** | unlabeled |
|---|---|---|---|---|---|---|---|
| bkgnd | **32.58** | 1.90 | 0.24 | 0.14 | 1.10 | 0.37 | **63.67** |
| frame | 15.29 | **66.68** | 6.47 | 0.46 | 6.90 | 0.10 | 4.10 |
| seat | 2.17 | 15.95 | **74.28** | 0.97 | 0.33 | 1.55 | 4.75 |
| armrest | 11.22 | 5.62 | 29.64 | **49.32** | 1.25 | 0.63 | 2.32 |
| wheels | 13.06 | 9.45 | 0.36 | 0.07 | **71.39** | 0.00 | 5.67 |
| **grab-area** | 6.48 | 1.28 | 9.77 | 0.11 | 0.00 | **76.75** | 5.62 |

Table 2
*Confusion matrix for the wheelchair part annotation experiment (cfr. Table 2).*

For the wheelchairs, we collected 34 images with considerable clutter and/or occlusion. We used the same ISM system as in the annotation experiment, to detect and annotate the chairs in these images. Some results are shown in Fig. 15. We consider a detection to be correct when its bounding box has at least 50% overlap with the ground-truth bounding box. Out of the 39 wheelchairs present in the images, 30 were detected, and there were 7 false positives. This corresponds to a recall of 77% and a precision of 81%.

Processing time is in the order of 1 min for small images and 6 min for large cluttered images on a Core 2 Quad PC. Memory usage is between 200 MB for small images and 360 MB for large images. Little to no attempts at optimization have been done yet, so there is a lot of potential for making the software more efficient.

## 6 Conclusions

We have developed a method to transfer meta-data annotations from training images to test images containing previously unseen objects, based on object class recognition. Instead of using extra processing for the inference of meta-data, this inference is deeply intertwined with the actual recognition process. Low-level cues in an image can lead to the detection of an object, and the detection of the object itself causes a better understanding of related low-level cues, like depth, orientations or part labels. The resulting meta-data inferred from the recognition can be used as input for other systems, e.g. as a prior for a 3D reconstruction algorithm.

Future research includes closing the cognitive loop by using the output from our system as input for another system. For instance, inferred depths, orientations and/or part labels can be used to guide a robot's actions, possibly in combination with other systems. Another interesting extension would be a method to improve the quality of the annotations by means of relaxation or Markov Random Fields.

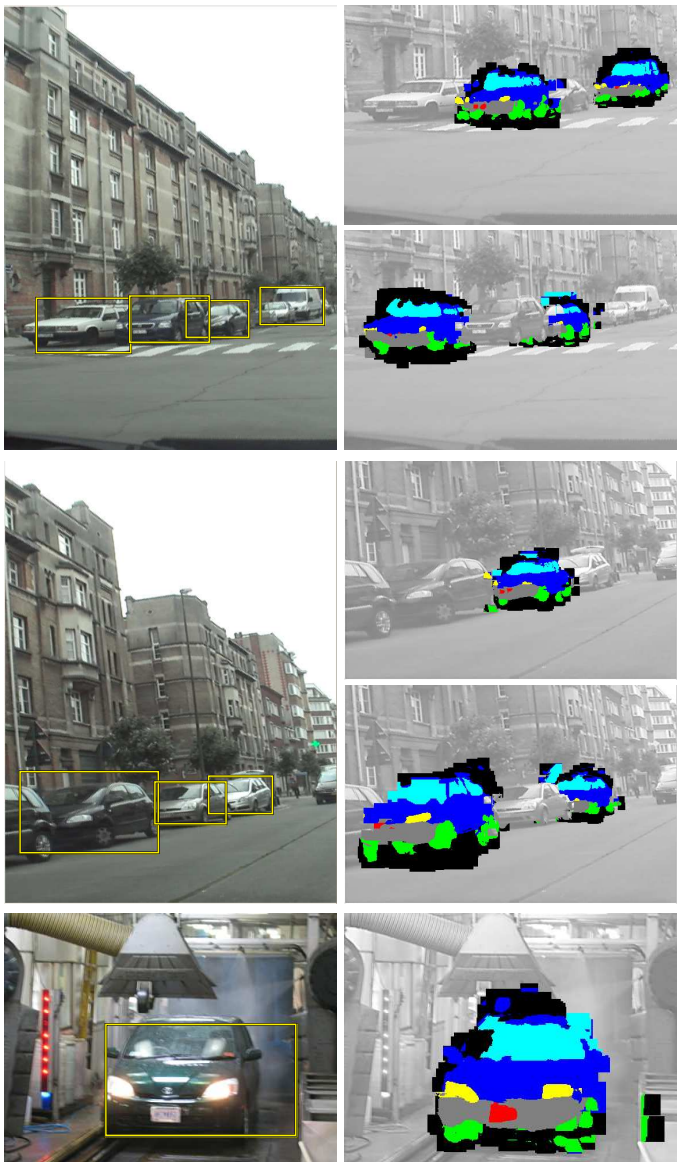

Fig. 14. *Car detection and annotation results on real-world test images. Even though the car in the car wash scene (bottom) is in a near-frontal pose, it was still correctly detected and annotated by the system trained on semi-profile views.*

## Acknowledgment

The authors gratefully acknowledge support by IWT-Flanders, Fund for Scientific Research Flanders and European Project CLASS (3E060206).

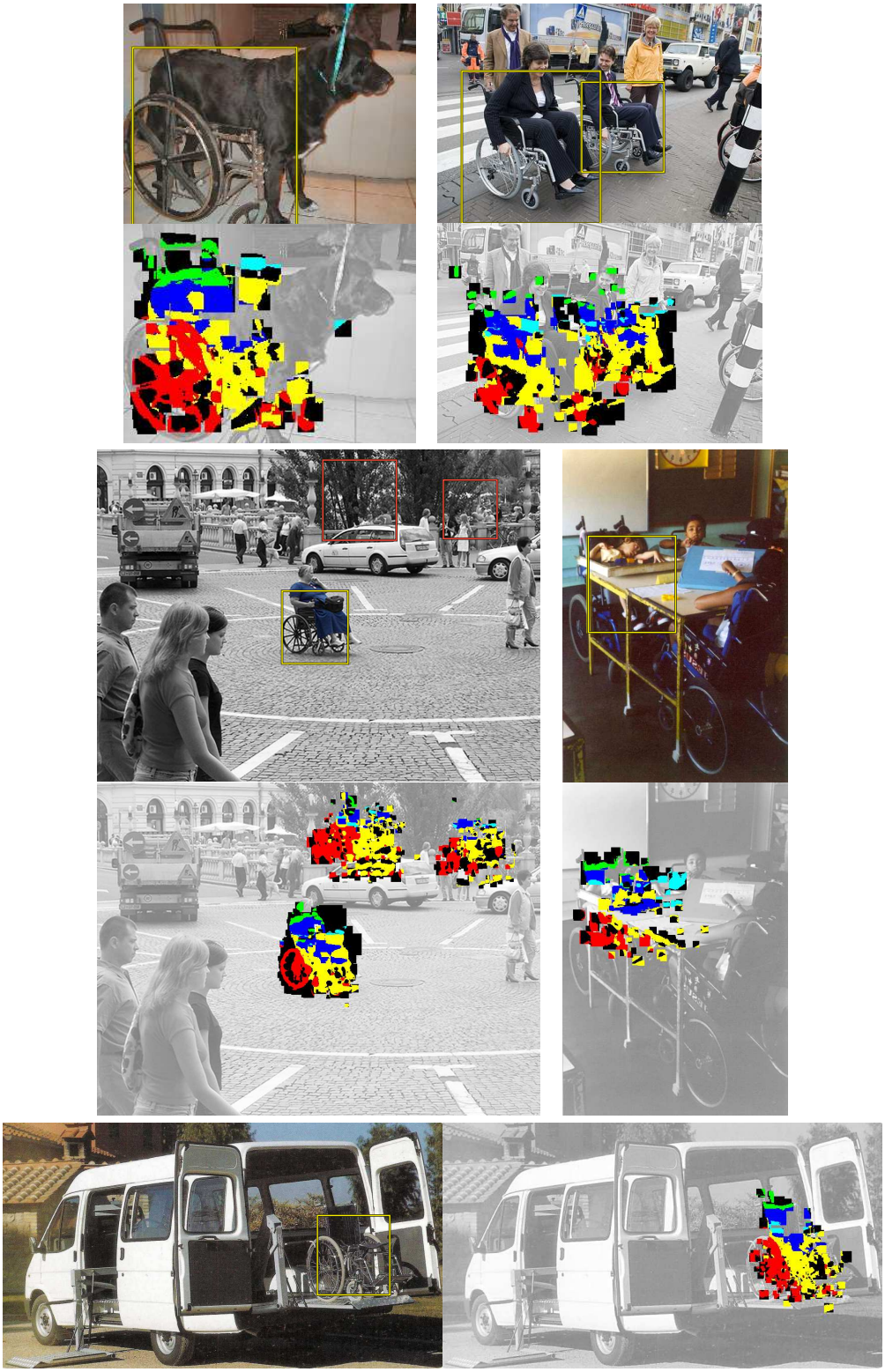

Fig. 15. *Wheelchair detection and annotation results on challenging real-world test images. All detections are correct except for the two topmost ones in the center left image. Note how one wheelchair in the middle right image was missed because it is not in the pose used for training.*

## References


[1] B. Leibe, A. Leonardis, B. Schiele, Robust object detection with interleaved categorization and segmentation, International Journal of Computer Vision 77 (1-3) (2008) 259–289.

[2] K. Rockland, G. V. Hoesen, Direct temporal-occipital feedback connections to striate cortex (V1) in the macaque monkey, in: Cereb. Cortex, Vol. 4, 1994, pp. 300–313.

[3] D. Mumford, Neuronal architectures for pattern-theoretic problems, Large-Scale Neuronal Theories of the Brain (1994) 125–152.

[4] D. Hoiem, A. Efros, M. Hebert, Putting objects in perspective, CVPR (2006) 2137–2144.

[5] B. Leibe, N. Cornelis, K. Cornelis, L. Van Gool, Dynamic 3D scene analysis from a moving vehicle, CVPR (2007) 1–8.

[6] I. Biederman, Recognition-by-components: A theory of human image understanding, Psychological Review 94 (2) (1987) 115–147.

[7] R. Bergevin, M. D. Levine, Generic object recognition: Building and matching coarse descriptions from line drawings, PAMI 15 (1) (1993) 19–36.

[8] D. Hoiem, A. A. Efros, M. Hebert, Geometric context from a single image, ICCV (2005) 654–661.

[9] E. Sudderth, A. Torralba, W. Freeman, A. Willsky, Depth from familiar objects: A hierarchical model for 3D scenes, CVPR 2 (2006) 2410–2417.

[10] A. Saxena, J. Schulte, A. Y. Ng, Learning depth from single monocular images, NIPS 18 (2005) 1–8.

[11] F. Han, S.-C. Zhu, Bayesian reconstruction of 3D shapes and scenes from a single image, Workshop Higher-Level Knowledge in 3D Modeling Motion Analysis (2003) 12–20.

[12] T. Hassner, R. Basri, Example based 3D reconstruction from single 2d images, in: Beyond Patches Workshop at IEEE CVPR06, 2006, p. 15.

[13] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, D. Salesin, Image analogies, SIGGRAPH Conference Proceedings (2001) 327–340.

[14] L. Cheng, S. V. N. Vishwanathan, X. Zhang, Consistent image analogies using semi-supervised learning, CVPR (2008) 1–8.

[15] M. Kumar, P. Torr, A. Zisserman, OBJ CUT, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[16] J. Winn, N. Jojic, LOCUS: Learning object classes with unsupervised segmentation, in: Proc. IEEE Intl. Conf. on Computer Vision (ICCV), Vol. 1, 2005, pp. 756–763.

[17] H. Arora, N. Loeff, D. A. Forsyth, Unsupervised segmentation of objects using efficient learning, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[18] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, L. Van Gool, Towards multi-view object class detection, CVPR 2 (2006) 1589–1596.

[19] D. Hoiem, C. Rother, J. Winn, 3D LayoutCRF for multi-view object class recognition and segmentation, CVPR (2007) 1–8.

[20] A. Kushal, C. Schmid, J. Ponce, Flexible object models for category-level 3d object recognition, CVPR (2007) 1–8.

[21] S. Savarese, L. Fei-Fei, 3d generic object categorization, localization and pose estimation, in: ICCV, 2007, pp. 1–8.

[22] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Anal. Mach. Intell. (1995) 790–799.

[23] B. Leibe, B. Schiele, Interleaved object categorization and segmentation, BMVC (2003) 759–768.

[24] D. G. Lowe, Distinctive image features from scale-invariant keypoints, IJCV 60 (2) (2004) 91–110.

[25] H. Bay, T. Tuytelaars, L. Van Gool, SURF: Speeded up robust features, Proceedings ECCV, Springer LNCS 3951 (1) (2006) 404–417.

[26] B. Russell, A. Torralba, K. Murphy, W. Freeman, LabelMe: a database and web-based tool for image annotation, MIT AI Lab Memo AIM-2005-025.

[27] K. Mikolajczyk, C. Schmid, Scale & affine invariant interest point detectors, IJCV 60 (1) (2004) 63–86.

[28] S. Belongie, J. Malik, J. Puzicha, Shape context: A new descriptor for shape matching and object recognition, in: NIPS, 2000, pp. 831–837.

[29] E. Seemann, B. Leibe, K. Mikolajczyk, B. Schiele, An evaluation of local shape-based features for pedestrian detection, in: Proceedings of the 16th British Machine Vision Conference, Oxford, UK, 2005, pp. 11–20.

[30] A. Saxena, J. Driemeyer, J. Kearns, A. Y. Ng, Robotic grasping of novel objects, NIPS 19 (2006) 1209–1216.