Unsupervised Generation of a Viewpoint Annotated Car Dataset from Videos

Nima Sedaghat, Thomas Brox
Computer Vision Group
University of Freiburg, Germany
{nima,brox}@cs.uni-freiburg.de

Abstract

Object recognition approaches have recently been extended to yield, aside of the object class output, also viewpoint or pose. Training such approaches typically requires additional viewpoint or keypoint annotation in the training data or, alternatively, synthetic CAD models. In this paper, we present an approach that creates a dataset of images annotated with bounding boxes and viewpoint labels in a fully automated manner from videos. We assume that the scene is static in order to reconstruct 3D surfaces via structure from motion. We automatically detect when the reconstruction fails and normalize for the viewpoint of the 3D models by aligning the reconstructed point clouds. Exemplarily for cars we show that we can expand a large dataset of annotated single images and obtain improved performance when training a viewpoint regressor on this joined dataset.

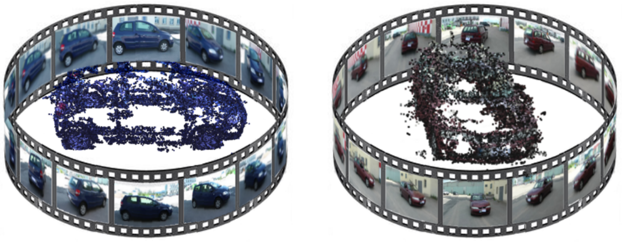


Figure 1: Two videos of different class instances are sought to be registered temporally such that frames showing the same viewpoint are in correspondence. We solve this problem via an explicit 3D reconstruction to automatically generate viewpoint and bounding box annotation for a video dataset.

1. Introduction

The viewpoint of an object is an important attribute in 2D recognition approaches. This is because the appearance of the 2D projection varies a lot with the changing viewpoint and because the viewpoint has an important semantic meaning. In the past, many viewpoint specific detectors have been proposed [3, 11, 4, 5, 23, 7] which aim on better detection performance. Another direction of research is on building explicit viewpoint classifiers that can infer the viewpoint for a given detection [21]. Recent approaches regard detector training and accurate viewpoint classification as a joint problem [19, 26].

Training viewpoint specific detectors or viewpoint classifiers usually requires images with viewpoint annotation to work well. As an alternative to explicit viewpoint annotation, keypoint annotation was proposed to infer the viewpoint and pose [3]. Automated viewpoint clustering approaches [6, 11, 5] provide only approximate viewpoint labels and can introduce many erroneous labels to the training.

As an alternative to a collection of single images, videos are a good data source to infer viewpoints. Walking with a video camera around an object, images from many viewpoints can be collected very efficiently. Even with low frame rates, the sampling density in a video is very high compared to image collections.

The challenging part is again the viewpoint annotation in these videos. Assume we get several videos showing many different object instances from the same class. The camera could have been moved along very different paths. In order to train viewpoint classifiers in the end, we are interested in the corresponding frames in the videos, *i.e.*, those frames that show the same (or very close) viewpoints. We can also regard this as a clustering problem. Given the collection of all the frames in all the videos, we are searching for viewpoint specific clusters exploiting the sequential nature of the recorded videos.

Retrieving such a clustering was first approached by Mei *et al.* [16]. It is still very challenging for multiple reasons: (1) In each of the videos we see a different object instance. The appearance of instances can vary a lot. (2) If the dataset does not provide a segmentation or bounding box (in videos such annotation is rare), there is much clutter in the background that may confuse clustering. (3) Not all videos will show the same viewpoints.

In case of a static scene, videos offer the possibility to reconstruct the 3D structure. With the 3D structure and the camera positions available, matching the viewpoints is much easier. Moreover, it is quite straightforward to infer an object segmentation from a 3D point cloud. A nice first approach using structure from motion to infer viewpoints was described by Glasner *et al.* [10]. They compute 3D point clouds from a set of photos of the same object and then build a voting based detector. Their approach still includes several manual steps. In contrast, we provide a fully automated approach to generate a training dataset with viewpoint labels from a set of videos.

A successful approach to integrate 3D viewpoint information into recognition methods is by using synthetic 3D surfaces as provided by CAD models [15, 18]. In this case the problem becomes much easier due to flawless 3D surfaces. On the other hand, reconstruction from videos yields a dataset that consists of natural images rather than synthetic ones.

Technically, robust alignment of the point clouds is the most important challenge in this setting. In [22], various 3D shape matching and alignment methods have been reviewed. Most of the existing methods are highly sensitive to noise and require flawless 3D reconstructions to work well. In practice, the point clouds comprise many errors. Moreover, we are interested in aligning different object instances. The work of [14], although it is robust to noise, is suitable only for matching models of the same shape. In our work, we approach the problem of matching different instances of a class by a global search on a hierarchical feature representation of the point cloud to determine pairwise alignments. Moreover, we optimize on all instances to detect bad alignments of some pairs and to retrieve a globally consistent alignment of the whole dataset. [1]

We show exemplarily for cars that the dataset created automatically by the proposed approach from videos allows to train a viewpoint regressor with good performance. We show that we can also join our data with the manually labeled ImageNet training set to obtain an overall improvement in performance.

2. 3D object surfaces derived from video

2.1. Dataset

We have collected videos of 52 different cars with a usual camcorder. Every full-view video consists of roughly 1500 to 3500 frames, which is uniformly downsampled to a number close to 120 to speed up 3D reconstruction.

The first 12 videos of the collected dataset where used as a validation set to develop our methodology and to optimize the (hyper-)parameters of the method's components, such as 3D reconstruction and alignment. With the method fixed, we ran it on the remaining videos to generate the final annotated dataset.

2.2. 3D reconstruction

We use the off-the-shelf structure from motion (SFM) package called VSFM [24, 25], which runs bundle adjustment [1] and PMVS/CMVS [9] to reconstruct the scenes shown in each video. The software gives as output a point cloud representing the scene, the normal vectors assigned to each point, as well as the camera locations and extrinsic parameters, all of which are used in this work.

We regard the SFM package as a black box. However, we need to utilize it in a smart way to benefit from the temporal ordering of video frames, as the SFM packages usually expect randomly taken images of the scene. Thus, instead of relying on the software to guess image pairs, we provide each frame paired with its 3 following frames. This way we can run the program and get a basic reconstruction of the scene. Then we use this basic model and obtain camera location estimates to restart the reconstruction, pairing each frame with its 5 neighbors, in addition to adding pairs representing a potential loop closure. This, compared to the simple automatic run of the SFM software, results in more robust and accurate reconstructions.

We finally apply a dense surface reconstruction method, in particular *Smooth Signed Distance Surface Reconstruction* (SSD) [13], to the point cloud for the further processing not being affected by the point density.

2.3. Ground plane estimation and object segmentation

Given the point cloud of the whole scene, we are only interested in those points that belong to the object of interest. To this end, we first detect the ground plane using RANSAC [8]. The assumption that the object of interest must be placed on a more or less planar surface is quite reasonable and a very weak restriction in practice. Then we remove the points belonging to the ground plane. This leaves the object of interest separated from other structures in the background, i.e., points not belonging to the object can be removed automatically via connected components. We assume that the object is dominant in the video (in the center of attention), which is naturally the case as we walk with the camera around the object.

The ground plane serves as xy plane for all objects in the dataset. Thanks to the external camera parameters estimated in structure from motion, this already provides the normalized *elevation* angle for the object in each individual frame. Moreover, projecting back the segmented object according to the camera parameters, makes it possible to obtain an approximate segmentation of the object in each

[1] The dataset and the code can be found at `http://lmb.informatik.uni-freiburg.de/Publications/2015/SB15`

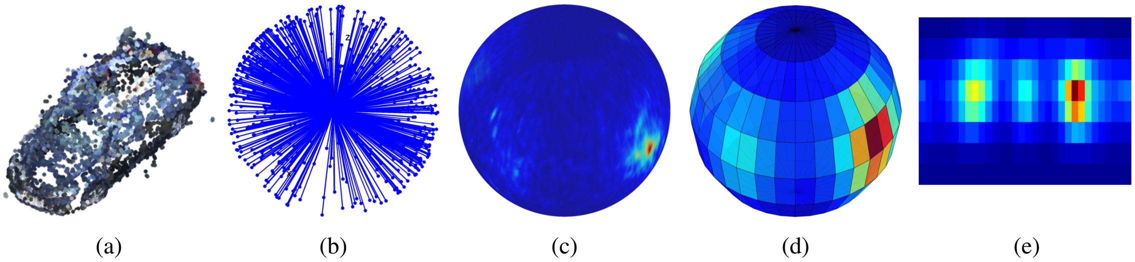


Figure 2: The orientation histogram. (a) shows a sample car. (b) & (c) depict its normal vectors and their projections on the surrounding sphere. (d) & (e) illustrate the binning in spherical and flat representations respectively.

image, which also directly yields classical bounding boxes in the images; see Fig. 6.

This also allows for normalization of the scale of the individual objects. But this is not necessary in the current work, since the introduced description of the objects is invariant to the scale and translations, as will be seen shortly.

3. Pairwise alignment of 3D models

Given two point clouds, M_1 and M_2 of two objects, we want to infer a rigid body transformation and scaling to best align them. The alignment problem can be formulated as a minimization problem on the following cost function:

$$J(\beta) = d(f(M_1), f(M_2(\beta))) \tag{1}$$

where β contains the transformation parameters. $M_2(\beta)$ represents the transformed version of M_2. $f(.)$ denotes some description of the models, and d is an appropriate distance metric. A descriptor for each point cloud that is invariant to some deformations is important because we align different object instances, i.e., a rigid body transformation cannot perfectly align the raw point clouds.

We can simplify the above problem in our case by reducing the parameter space significantly. As mentioned above, we already have a very good estimate of the elevation angle of the camera, due to the ground plane estimate. Moreover, the descriptor, $f(.)$, that we introduce in the next subsection, eliminates the need for scale and translation normalizations. Therefore, the only major parameter that must still be determined is the relative *azimuth* angle, ϕ, between M_1 and M_2.

3.1. Hierarchical Orientation Histogram (HOH)

A 3D orientation histogram, sometimes also termed *Extended Gaussian Image* [12], provides a global description of the point cloud which is independent of the location of the points in the point-cloud, and only depends on the surface normals. This gives higher robustness to intra-class variations, in addition to scale and translation invariance. As depicted in Figure 2, the normal vectors of all the points are drawn originating from a single point and are projected on the sphere surrounding it. The resulting points on the sphere, after binning form an orientation histogram, which gives a close-to-unique representation of convex objects [12]. To make it a practical feature descriptor, we normalize the bin values by their sum, and multiply by a weight roughly proportional to the inverse of their covering area on the sphere:

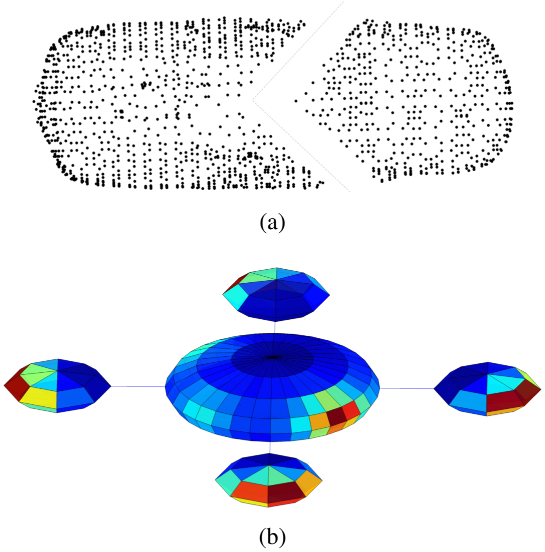


Figure 3: Hierarchical Orientation Histogram. (a): An example of slicings of an object, used for generation of the child histograms. (b): Sample root and child histograms.

$$w_{ij} = \frac{1}{|\sin \theta^*_{ij}|} \tag{2}$$

where θ^* is the value of θ in middle of each cell.

However, mere use of such a single histogram, causes too much loss of details for the sake of robustness to intra-class variations. Consequently, ambiguities occur and alignment will fail, as can be seen also in Table 1.

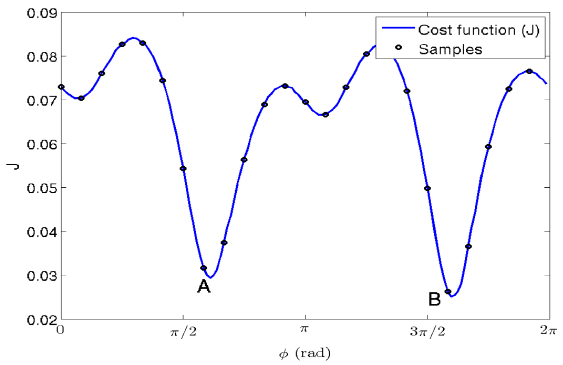


Figure 4: The cost function, J, of equation 1 for a pair of cars, in the whole search space. Points 'A' and 'B' are the two dominant candidates.

To overcome this problem, we introduce Hierarchical Orientation Histograms (HOH), which capture more spatial details of the shape of the objects, while still being robust enough to minor changes. Figure 3 illustrates an example of a HOH obtained from a sample 3D model. As depicted in this figure, in addition to the 'root' histogram, we use 'child' histograms computed from slices of the point cloud, obtained using radial cut planes with different ϕ values, originating from the centroid – Fig. 3b displays this from top view. These 'child' histograms use a smaller number of bins to remain robust to minor changes. The extra information is encoded in their order.

In this work, we have set the number of bins of the root histogram, b^{ϕ} and b^{θ}, to 32 and 8 respectively. We have 8 child histograms (N_c), and for each child we have 16 and 4 bins in the ϕ and θ directions (b_c^{ϕ} and b_c^{θ} respectively). Some of these values are changed in Figure 3b for illustration purposes.

The final descriptor f is then constructed by simple concatenation of the histograms, each reshaped to a 1-D vector. To compare the two descriptors, we use the χ^2 distance:

$$d(f_1, f_2) = \chi^2(f_1, f_2) = \sum_b \frac{(f_1(b) - f_2(b))^2}{(f_1(b) + f_2(b) + \epsilon)} \quad (3)$$

where ϵ is a small number (e.g. set to 10^{-20}) to prevent division by zero in case of empty bins.

3.2. Minimization of the alignment cost function

Figure 4 shows the plot of an example of the cost function, J, of equation 1 for a pair of cars, in the whole search space. Obviously it is not convex. Thus to find the global minimum, we sample the whole search space, pick the 2 best minima, initialize gradient descent with these candidates and pick the one that results in the lowest final cost.

3.3. Experiments on pairwise alignment

Table 1 compares the performance of our pairwise alignment method, with several baseline methods. To perform the comparisons we have used a fixed subset of 12 of our 3D models, from our 'test' set. To generate the ground truth, we manually aligned these 12 models to each other, and then the pairwise matching was tested on every possible pair in this set. Failure rate was considered as the percentage of the results which were more than 5.625°deviated from the ground truth (zero). This number corresponds to half of the azimuth covered in each cell of a histogram of length 36 in the ϕ direction.

ICP [2] is possibly the most used alignment method for point clouds and can be thought of as the main baseline. It heavily depends on a close-to-exact initialization, rendering it useless for the application at hand. Even though we ran the point-to-plane version of [20], with 10 uniformly sampled initializations, it had a high failure rate while being also quite slow. We helped it by pre-scaling the objects, using an estimate of their biggest dimension obtained from PCA – something that is not necessary in our method. The results improved a bit, but still are far from being applicable.

We also used PCA itself to align the objects based on their principal axis. However, as seen in the table, it had a failure rate of 17%, although cars are objects with a very well oriented 3D shape. For chairs it failed in 90% of all cases.

"Single-scale histogram" shows the results of alignment using the single basic orientation histogram introduced before, and the last row contains the results of our method.

4. Final consensus

Pairwise matching of every single model with all the others gives us the following matrix:

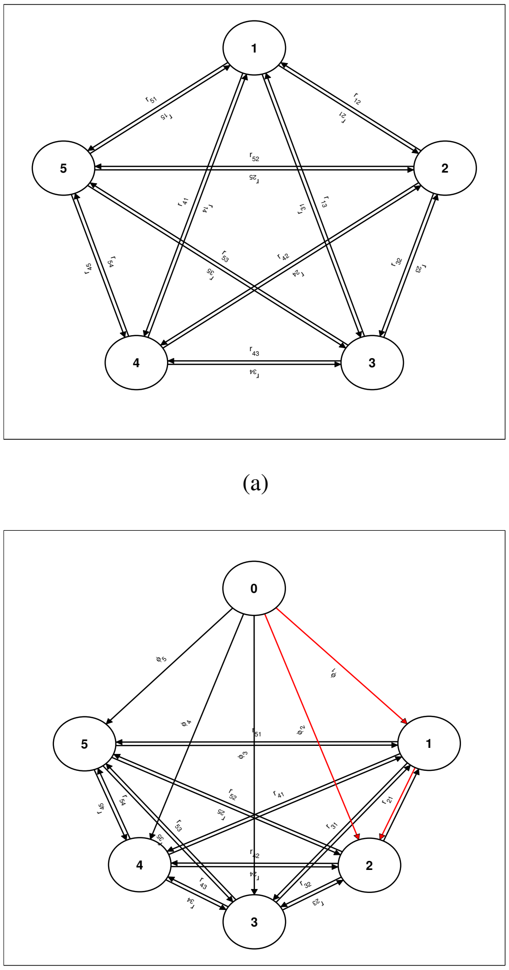
$$R = \begin{bmatrix} 0 & r_{12} & r_{13} & \dots & r_{1N} \\ r_{21} & 0 & r_{23} & \dots & r_{2N} \\ r_{31} & r_{32} & 0 & \dots & r_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N1} & r_{N2} & r_{N2} & \dots & r_{NN}, \end{bmatrix} \quad (4)$$

which corresponds to the graph displayed in Figure 5a.

If the pairwise alignments were ideally accurate, one row of this matrix would be enough to obtain the absolute pose for all the objects. That is not the case, however, as there is some uncertainty included in each element of this matrix. One source of this uncertainty is due to pairwise matching failures, such as the well-known front-back ambiguity which still exists in rare cases, no matter how good the pairwise matching performs. Another reason are erroneous reconstructions.

Method	Alignment failure rate (err>5.625°)	Average run time for single pair alignment
Point2Plane ICP with 10 initializations	42%	227s
Point2Plane ICP with 10 initializations + PCA pre-scaling	27%	99s
PCA	17%	0s
Single-scale histogram	44%	3s
Our method	**3%**	3s

Table 1: 3D Alignment methods comparison



(a)

(b)

Figure 5: (a) Graph R, corresponding to the Matrix R,representing pairwise alignments between all the models. (b) Graph R^*, based on which we optimize for the absolute pose values. A sample closed walk of length 3, is shown here in red.

In the following we take an optimization-based approach to deal with both sources of this problem at the same time: obtaining the most accurate pose estimate based on all the alignments together, and automatic removal of bad reconstructions. To this end, we add a reference node to the graph with index 0, creating the new graph, R^* – Figure 5b. This represents an imaginary model, based on which all the absolute poses are to be estimated. As seen in the figure, the edges connecting this node with the 'real nodes' are named with only a single index: ϕ_k.

The key idea here is that the edge values on each closed walk of length 3, on R^*, that starts from node 0 (and ends on it) should ideally sum up to multiples of 2π. This is true remembering that the directed edge values are in fact the mutual rotations required to align real/imaginary 3D objects. The reason we do not consider cycles of higher length on the graph is that triplets cover all the paths in the graph, such that any longer cycle can be expressed as a sum of a number of triplets. Based on this, we keep the currently estimated pairwise pose values (ϕ_{kl}) intact, and seek to find the vector $\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$ that minimizes the cost function

$$C = \sum_{k=1}^{n} \sum_{l=1}^{n} |e^{i\phi_k} e^{-i\phi_l} e^{ir_{kl}} - 1|^2, \tag{5}$$

where n is the number of nodes in the graph. This cost function is not convex. For minimization we use gradient descent, initialized with one row of the matrix to speed up the convergence, and increase the chance of finding the global minimum. On the other hand, since a constant shift in all the pose values (ϕ_k) does not affect the value of C, the gradient descent method could continue for ever. Hence, we put the variance of the components of $\partial C/\partial \phi_k$ as the stopping condition.

As stated before, we also need to detect and remove the 'bad' reconstructions from the dataset. We do this in the heart of the optimization phase, by calculating the following value, while computing the cost at each step:

$$S_k = \frac{1}{n} \sum_{l=1}^{n} \left(|e^{i\phi_k} e^{-i\phi_l} e^{ir_{kl}} - 1|^2 + |e^{i\phi_l} e^{-i\phi_k} e^{ir_{lk}} - 1|^2 \right) \tag{6}$$

These values are an estimate of the mean of errors over all the closed walks of length 3, passing through node k.

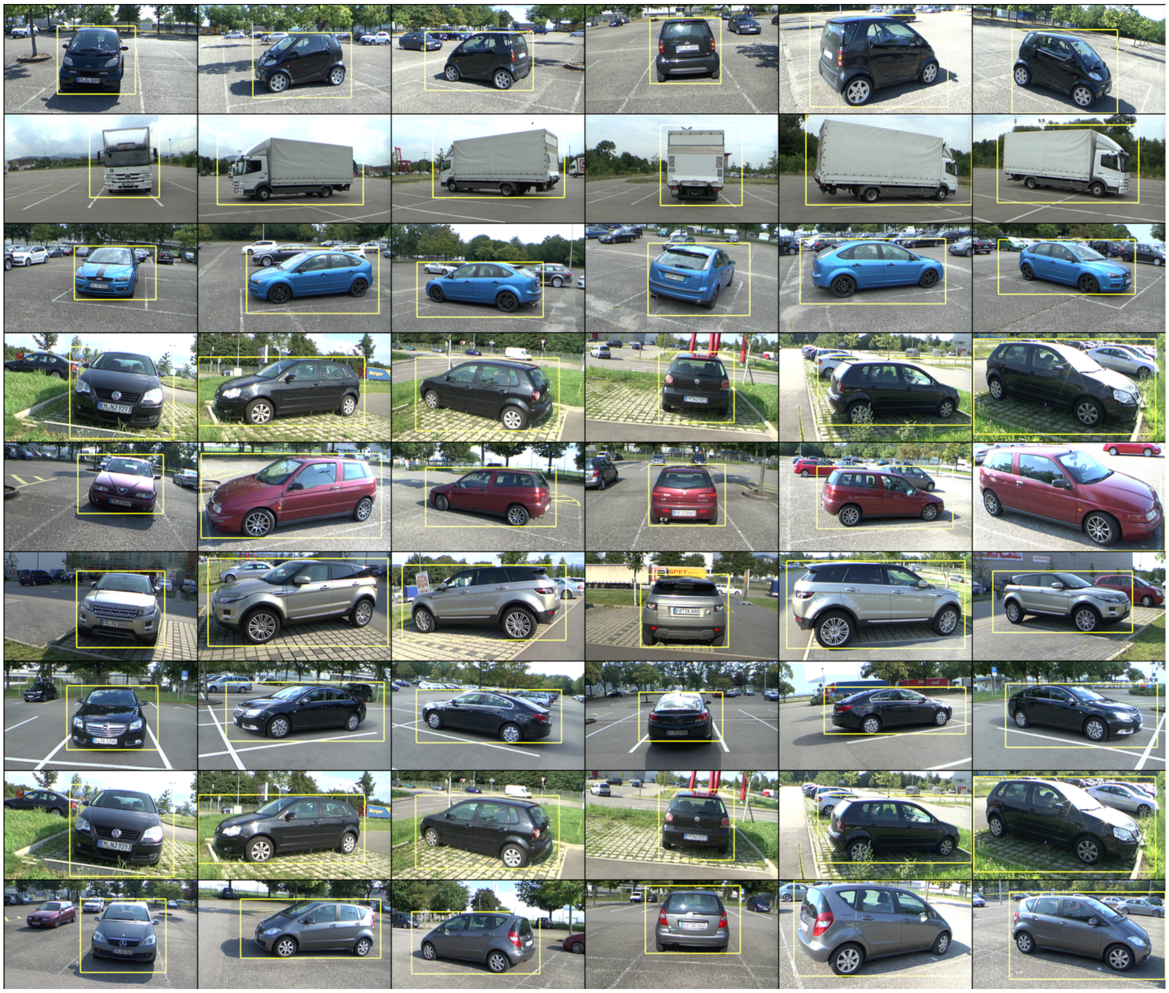


Figure 6: Samples from the automatically generated car dataset aligned by the azimuth angle. The elevation angle relative to the surface plane is available, too, due to the camera pose.

This is a good measure of unreliability for the node and is readily available after C converged. This way we detect nodes with too high S value and eliminate them automatically from the dataset. Based on our experiments on the validation set, a threshold of $5 \times \ 10^{-3}$ eliminates the bad objects while keeping the usable ones in the dataset. In the end, a total of 6 out of 52 cars were marked as bad models in our overall dataset.

5. Resulting datasets

Figure 6 illustrates some sample frames of the videos of cars automatically aligned using the introduced method. In total we recorded 52 car videos, of which we set 12 as the validation set and 40 as the test set. However, as stated before, 6 of the reconstructions failed: 1 from the validation set and 5 from the test set. Therefore, the effective size of the generated car dataset is 35, only considering the test set, and 46 in total.

On a machine with 16 CPU cores and a Geforce Titan, the whole process of converting the video frames to the 3D models and segmentation takes roughly 10 minutes per video. Matching of a single model to a reference set of size 10-30 takes only a few seconds. The whole car dataset was processed in less than 9 hours.

The framework is designed to allow for incremental improvement of datasets, such that the generated datasets, such as the one presented in this paper, are not limited in size and any number of videos from various sources can extend it in future.

While the method was developed for cars, it generalizes also to other static objects. This is demonstrated for chairs and mugs in Fig. 7. The parameters and methodology were kept exactly as for the car data set. From the 27 videos of

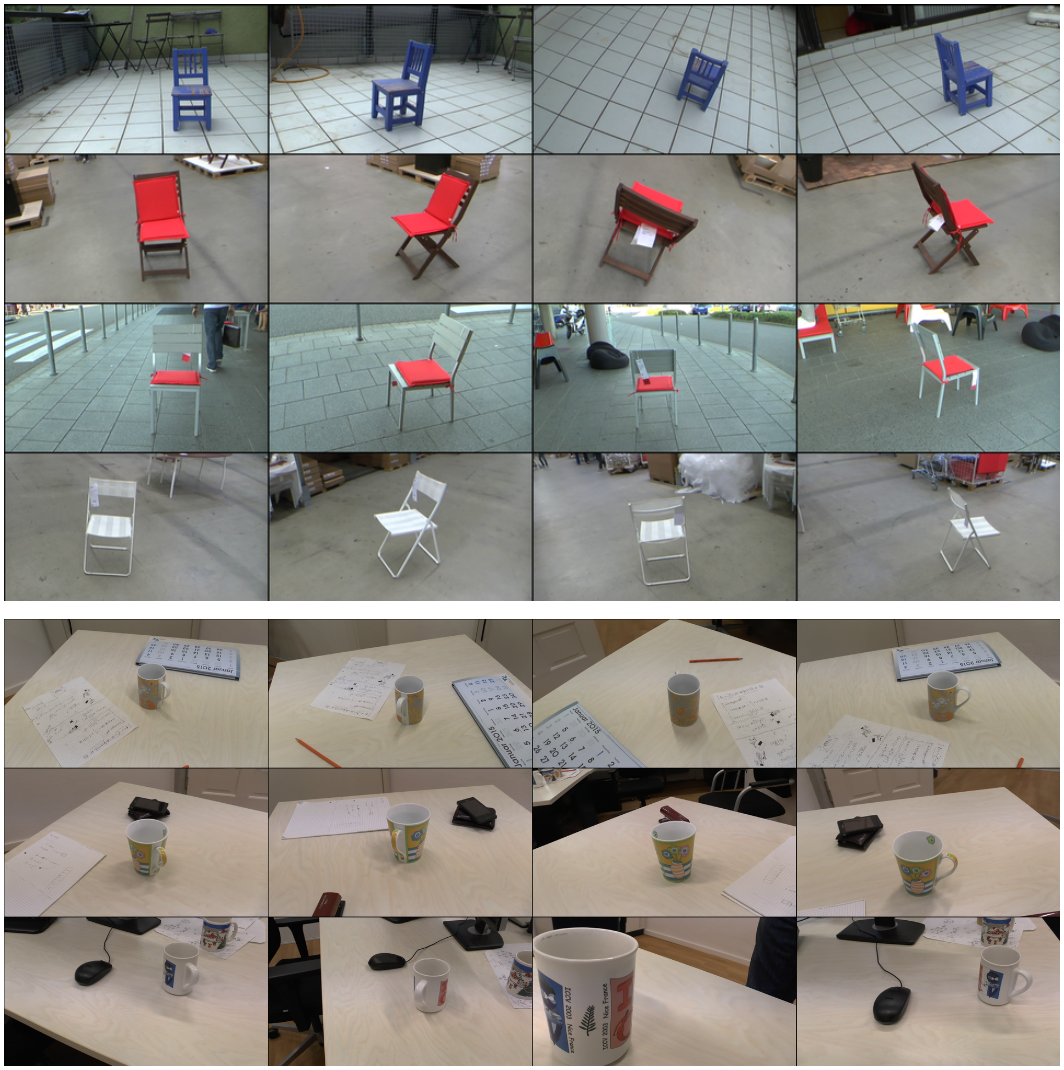


Figure 7: Samples from the automatically generated chair and mug datasets. The samples are ordered just based on the azimuth angle of their pose, disregarding the elevation angle relative to the surface plane, for illustration purposes.

mugs, the method kept and annotated 13 mugs. From the 50 chair videos, only 7 were kept and annotated. All annotations were correct with an azimuth error below 12 degrees. In practice one would like to increase the ratio of videos that pass the method's internal quality checks. The main reason for the many refused videos was problems in the 3D reconstruction, which was selected to be robust to the many reflections on cars. For the fine details and topography of chairs another reconstruction package or adaptation of parameters is likely to lead to larger annotation ratios.

6. Application example: training a convolutional network for viewpoint estimation

To provide a concrete example for possible use cases of such a viewpoint annotated dataset, in this section we implemented a convolutional network for viewpoint regression and trained and tested it with several combinations of datasets.

6.1. Network architecture

The convolutional network takes as input a 2D image and yields as output the continuous azimuth. It follows the standard architecture and consists of 3 convolutional layers with their respective max-pooling layers followed by 3 fully connected layers. Input images are assumed to contain a complete car inside them. The bounding box is extended to become a square around the car, and enlarged by 20% in both dimensions. The area inside the bounding box is cropped and resized to 150x150. The actual input size to the network is 128x128, which is randomly selected from this image in the training phase together with some other spatial

	Test set					
Training Set	Pascal’12	EPFL	Weizmann	Imagenet (val)	Imagenet (val) without trucks	Our Videos
Pascal’12	N/A	37.0°	25.6°	29.3°	28.7°	26.9°
EPFL	71.6°	N/A	34.8°	53.2°	46.3°	34.6°
Weizmann	65.2°	26.9°	N/A	49.5°	38.1°	11.7°
Imagenet (train)	47.7°	**17.4**°	13.6°	12.3°	11.7°	**10.6**°
Our Videos	61.5°	34.4°	12.2°	38.0°	27.6°	N/A
Imagenet+Our Videos	**44.8**°	19.9°	**10.7**°	**11.6**°	**10.9**°	N/A

Table 2: Mean absolute error of pose estimation using the convolutional network regressor trained and tested on various datasets.

and color-space augmentations. At test time, the image is cropped in the center, which gives the square containing the original car in the center. The output is a complex number representing $e^{i\phi}$, or equivalently two scalar outputs containing sin and cos of the azimuth, to deal with the problems caused by the periodic nature of the azimuth.

6.2. Experiments on various train/test sets

We selected the following most well-known viewpoint annotated datasets of cars, along with our generated dataset:

- The EPFL cars dataset [17] consists of videos with a static camera and cars rotating on a turntable. The timing of the rotating cars is given, based on which one can obtain an approximation of the pose of the car.
- The Weizmann dataset [10] consists of 22 cars, for each of which there are images from multiple viewpoints. Manual annotation of azimuth and elevation are provided.
- The PASCAL’12 dataset is the popular dataset from the PASCAL challenge with viewpoint annotation added Savarese *et al.* [26]. Since many examples are heavily truncated, it is challenging both for training and testing. We only use the non-truncated cars.
- Also for the cars in ImageNet, Savarese *et al.* [26] provided manual viewpoint annotation. This dataset contains only non-truncated cars. With 5000 images, the size and diversity of the dataset is most comfortable to train a strong viewpoint classifier.
- Our video dataset with automatically generated viewpoint annotation, which consists of 42 videos and a total of 5669 images. Even more images would be available due to the dense spatial sampling in the video.

Since the number of different car instances is not large enough in any of the datasets except ImageNet, we have disregarded the possible partitioning for train/validation/test sets and used the whole set either as a training or as a test set – except Imagenet, for which we consider train/validation sets separately.

Results of these experiments are shown in Table 2. Training the regressor on our automatically annotated dataset outperforms training on any other dataset except the very large ImageNet dataset. In comparison to ImageNet, our dataset lacks diversity. For instance, there is only one (special) truck in our dataset and many other more exotic car types are missing. After removing trucks from the ImageNet test set, regression performance with our dataset as training set increases significantly.

While our dataset lacks diversity with regard to different car types it contains many viewpoints of a single instance. Expanding the ImageNet training set with our dataset improves performance in almost all cases.

7. Conclusions

We have presented a fully automated approach to generate viewpoint and bounding box annotations for a set of videos of static scenes. We have demonstrated for the example of cars that such a dataset can be used to train a reasonably good viewpoint regressor. A practical limitation of the approach is that it requires collecting videos of static objects with many viewpoints of the object of interest being visible. So far, this kept us from applying the approach to many more object classes. Related is also the problem of diversity: while the approach yields a dataset that represents viewpoints in much detail, there are fewer object instances in the dataset than in usual training sets. However, our results on augmenting the ImageNet dataset with videos show that one can get the best of both by merging the diversity of single images with the richer viewpoint information in videos.

Acknowledgments

The project was funded by the ERC Starting Grant – VideoLearn. We also thank Lina Ronneberger and Elahe Eslami for helping us collect the videos.

References

[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *Computer Vision–ECCV 2010*, pages 29–42. Springer, 2010.

[2] P. Besl and N. Mckay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[3] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *Computer Vision–ECCV 2010*, pages 168–181. Springer, 2010.

[4] S. K. Divvala, A. A. Efros, and M. Hebert. How important are "deformable parts" in the deformable parts model? In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 31–40. Springer, 2012.

[5] B. Drayer and T. Brox. Training deformable object models for human detection based on alignment and clustering. In *Computer Vision–ECCV 2014*, pages 406–420. Springer, 2014.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, Sept. 2010.

[7] V. Ferrari, T. Tuytelaars, and L. Van Gool. Integrating multiple model views for object recognition. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–105. IEEE, 2004.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 11671.

[9] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, Aug. 2010.

[10] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich. Viewpoint-aware object detection and continuous pose estimation. *Image Vis. Comput.*, 30(12):923–933, Dec. 2012.

[11] C. Gu and X. Ren. Discriminative mixture-of-templates for viewpoint classification. In *Computer Vision–ECCV 2010*, pages 408–421. Springer, 2010.

[12] B. K. P. Horn. Extended gaussian images. *Proc. IEEE*, 72(12):1671–1686, 1984.

[13] J. Kautz, T.-y. Lee, M. C. Lin, F. Calakli, and G. Taubin. SSD: Smooth signed distance surface reconstruction. *Comput. Graph. Forum*, 30(7):1993–2002, 2011.

[14] H. Li, T. Shen, and X. Huang. Approximately global optimization for robust alignment of generalized shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(6):1116–1131, June 2011.

[15] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[16] L. Mei, M. Sun, K. M. Carter, A. O. Hero III, and S. Savarese. Unsupervised object pose classification from short video sequences. Technical report, DTIC Document, 2009.

[17] M. Ozuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 778–785. IEEE, 2009.

[18] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3d geometry to deformable part models. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3362–3369. IEEE, 2012.

[19] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Multi-view priors for learning detectors from sparse viewpoint data. *ArXiv Prepr. ArXiv13126095*, 2013.

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

[21] S. Savarese and F.-F. Li. 3D generic object categorization, localization and pose estimation. In *ICCV*, pages 1–8, 2007.

[22] J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3D shape retrieval methods. *Multimed. Tools Appl.*, 39(3):441–471, Sept. 2008.

[23] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards multi-view object class detection. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1589–1596. IEEE, 2006.

[24] C. Wu. VisualSFM: A visual structure from motion system, http://ccwu.me/vsfm/, 2011.

[25] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064. IEEE, 2011.

[26] Y. Xiang and R. Mottaghi. Beyond pascal: A benchmark for 3D object detection in the wild. In S. Savarese, editor, *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.