# ORB-SLAM-CNN: Lessons in Adding Semantic Map Construction to Feature-Based SLAM

Andrew M. Webb[0000−0001−7834−5250], Gavin Brown, and Mikel Luján

School of Computer Science, University of Manchester, Manchester, UK
{andrew.webb,gavin.brown,mikel.lujan}@manchester.ac.uk

**Abstract.** Recent work has integrated semantics into the 3D scene models produced by visual SLAM systems. Though these systems operate close to real time, there is lacking a study of the ways to achieve real-time performance by trading off between semantic model accuracy and computational requirements. ORB-SLAM2 provides good scene accuracy and real-time processing while not requiring GPUs [1]. Following a 'single view' approach of overlaying a dense semantic map over the sparse SLAM scene model, we explore a method for automatically tuning the parameters of the system such that it operates in real time while maximizing prediction accuracy and map density.

**Keywords:** Online parameter tuning · SLAM · Semantic segmentation

## 1 Introduction

We describe a method for associating semantic predictions with the 3D model of a sparse, feature-based SLAM system. We perform visual SLAM on the ScanNet dataset [3] using ORB-SLAM2 [11], while passing each frame used by the SLAM system through a convolutional neural network for semantic segmentation, based on MobileNets [6]. The output of the neural network is at one eighth of the resolution of the input image in each dimension; we use the fully connected conditional random field (CRF) method of Krähenbühl and Koltun [7] to upsample and refine the segmentation images.

The predictions associated with each pixel in each inference frame are projected into the 3D scene; we do not fuse predictions from multiple views. We show that this method has the advantage that we can fit a simple model to predict the accuracy of the 3D map as a function of the system parameters. We use this model to automatically tune the parameters such that the system operates in real time on a given platform while maximizing performance metrics. The following sections describe the various components of the system. For evaluation, we have two metrics for accuracy. The overall accuracy measures the proportion of correctly labelled points. The per-class accuracy is the mean of the proportion of correctly labelled points for each class, and is sometimes more useful than the overall accuracy in an imbalanced dataset.

Finally, in the appendix we describe our experience combining predictions from multiple views in order to predict the labels of points in the 3D map. This

technique does not work well in the sparse, feature-based setting, because it restricts predictions to object boundaries, which are harder to classify.

## 2 Related Work

In this section we describe other approaches for incorporating semantic predictions into the 3D model of a SLAM system. SemanticFusion [10] combines predictions from a convolutional neural network from multiple views, and associates them with a dense surfel based reconstruction produced by a SLAM system. Theirs was the first work to demonstrate, in real time, the use of pixel correspondences between frames from a SLAM system to fuse per-frame segmentations in order to produce an accurate 3D semantic map. McCormac et al. show that combining predictions from multiple views results in greater 3D semantic map accuracy when used in conjunction with a dense SLAM system, while our initial experiments (see Appendix) do not yield similar results for the case of an unmodified, sparse, feature-based SLAM system.

They use as their SLAM system ElasticFusion [16], which constructs a dense surfel-based model. One consequence is that, because they associate predictions with every surfel in the dense model, their Bayesian update step, which combines predictions from individual frames, takes almost as long as the forward pass of the CNN. In our work, we avoid combining predictions.

In order to achieve the real-time construction of a 3D semantic map, McCormac et al. reduce the image size fed into the neural network, to $224 \times 224$ or $320 \times 240$ depending on the network used, and perform the inference on a high end GPU (an Nvidia TITAN Black). In our system, the use of the efficient MobileNets networks reduces the computational cost of inference. This, and our use of parameters other than image resolution to reduce computational cost at the expense of accuracy, means we are able to run the system at real time at full VGA resolution on more moderate hardware. In our experiments, inference is performed on an Nvidia GTX 1050 (Notebook), with a peak power consumption of around 50 W, one fifth of that of the Titan Black.

As in our work, McCormac et al. use the CRF scheme of Krähenbühl and Koltun [7] to refine their segmentation map. However, they apply the CRF to the 3D surfel map, whereas we apply it to the 2D segmentation images before predictions are associated with the 3D model. We have not performed experiments to compare these two approaches.

Also related is the work of Pillai and Leonard, in which correspondences between frames, according to a SLAM system, are used to achieve strong object recognition performance [12]. As in our own work, they use ORB-SLAM to produce a sparse 3D map of the scene. In contrast to our work, the point cloud is clustered after points in low density regions are removed, and the clusters are taken to represent objects. Bounding boxes for each object are computed by projecting back down onto the keyframes, and a set of features are then computed for each object for each frame, and a linear classifier uses these features to

predict the class of the object. As in SemanticFusion, a conditional independence assumption is used when combining the predictions from multiple frames.

This object recognition system is unable to perform in real-time, taking 1.6 s per keyframe—with approximately 1 s between keyframes with default ORB-SLAM2 settings on most datasets—to perform the feature extraction and encoding steps. However, unlike this work and SemanticFusion, the SLAM supported object recognition work is done without the use of a GPU. Another difference with our work is that we construct a 3D semantic map, with a different prediction associated with every point in the map but no notion of 'objects', whereas Pillai and Leonard perform object recognition.

Sünderhauf et al. combine predictions from multiple views by using a Bayesian filter method [14], assuming first order Markov properties. Vineet et al. use a voxel-based map representation, with voxel predictions being updated based on sequential single-view predictions [15]. Their update rules are designed to quickly correct map corruptions caused by moving objects within the scene.

Kundu et al. [8] and Li and Belaroussi [9] construct a 3D semantic map from monocular input, rather than the RGB-D input we use here. Kundu et al. use semantic cues to constrain the 3D structure of the scene in a voxel-based model. Li and Belaroussi use a scale-drift aware method that allows them to transition between indoor and outdoor scenes. Häne et al. [4] treat image segmentation and dense 3D (voxel-based) reconstruction as a joint inference problem, allowing each task to inform the other.

## 3 Configuring the Convolutional Neural Network

### 3.1 The ScanNet Dataset

The ScanNet dataset [3] consists of 1513 video sequences with 2.5 million frames. For each frame there is an RGB image, a depth image, and per-pixel labels with 1163 distinct object classes, which we can use to evaluate semantic segmentation accuracy. The dataset also contains the groundtruth for the camera trajectory for each scene, which can be used to evaluate the SLAM system.

We use the same train/validation/test split of the dataset as used in the classification tasks in the original ScanNet paper; we use 1045 of the scenes to train the neural network, 156 to tune the network hyperparameters and CRF parameters, and 312 to evaluate our system.

### 3.2 MobileNet Semantic Segmentation Network

We use a modified version of MobileNets [6] for semantic segmentation. We train the network on the twenty most common classes in the dataset, and merge some similar classes, such as 'chair' and 'office chair'., We reserve class zero for unlabelled pixels. We modified the network to take four input channels, so that the network takes the RGB-D images as input. The RGB and depth inputs are normalized such that values lie between $-1$ and 1.

We removed the average pooling layer and reduced the stride from 2 to 1 in two convolutional layers, so that the output is a low resolution semantic segmentation of the input image. We also modified some layers after the reduced stride layers to be atrous convolutions with a dilation of 2 to compensate for the reduced receptive field, as in the DeepLab system [2].

The MobileNets networks have two parameters to trade off accuracy and computation. In the MobileNets paper, lowering the input image resolution reduces the accuracy of the prediction. In our work, lowering the resolution lowers the output segmentation resolution. We control the input resolution with the *rescale factor* parameter $r$, which scales both the height and width. The MobileNets networks also have a *width multiplier* $w$, which controls the number of feature channels throughout the network. In this work we train two versions of the MobileNets, with width multiplier values of $w = 0.5$ and $w = 1.0$.

### 3.3 Conditional Random Field Post-Processing

As in the work of Krähenbühl and Koltun, and in the DeepLab system [2], we apply bilinear upsampling and conditional random field (CRF) post-processing to the output of the network to obtain a segmentation the same size as the input image. The upsampled output is coarse, with smoothed edges and lacking fine detail. The CRF is used to refine the segmentation.

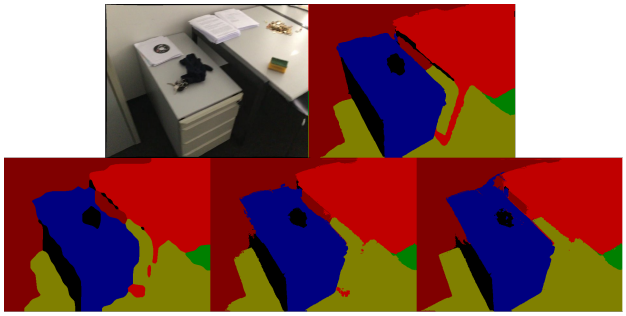

Fig. 1: Top left: a test image from ScanNet. Top right: ground truth labels. Bottom row: pre-CRF, then CRF-processed images with 3 and 5 iterations, in which the segmentation conforms more closely to the ground truth.

A CRF is a graphical model for expressing the posterior distribution over a collection of variables via the relative compatibilities of assignments to subsets of those variables. Here it is used to express a preference that pixels close to each other in the image and in colour-space should be labelled similarly. The effect is to assign more probability to segmentations that are smooth, without pixels that are assigned a different label to all of their neighbours, and to refine the edges in the segmentation to conform to sharp colour changes. An example of applying a dense CRF in this way is shown in Figure 1.

The posterior of the CRF is approximated iteratively. We can vary the number of *CRF iterations* $I$ in order to trade off accuracy against computational cost. A CRF can be implemented as a layer in a deep neural network [17], which would enable end-to-end training and would reduce CRF inference time. However, in this work we have used an available C++ implementation [7][1]. We first train the network and then separately fit the parameters of the CRF.

### 3.4 Training and Parameter Fitting

We use a cross entropy loss function with L2 regularization, and use batch normalization throughout the network. No loss is assigned for unlabelled pixels. We correct for class imbalance by making the per-class loss inversely proportional to the frequency of the class. Two modified MobileNets networks, with width multiplier parameters of 0.5 and 1.0, were trained for 85,000 and 165,000 iterations respectively using a batch size of 32 on a TITAN X (Pascal). Each batch was rescaled randomly between 0.25 and 1.0 times the original image size (in terms of area), and then cropped to a random $224 \times 224$ region.

We tune the hyperparameters—the learning rate and the L2 regularization parameter—by training until convergence ten times with randomly sampled hyperparameter settings, and choosing the settings that maximize the mean per-class accuracy on the validation set. To tune the CRF parameters, we randomly sample values 50 times, each time evaluating the post-CRF per-class accuracy on 312 images from the validation set—two images per scene.

Table 1: Overall / per-class accuracy (%) (with CRF iterations $I = 5$) on 312 images from validation and test sets.

(a) Width multiplier $w = 0.5$.

| | Validation | Test |
|---|---|---|
| Pre-CRF | 57.3 / 57.4 | 55.5 / 55.9 |
| Post-CRF | 58.2 / 59.0 | 56.6 / 56.3 |

(b) Width multiplier $w = 1.0$.

| | Validation | Test |
|---|---|---|
| Pre-CRF | 66.2 / 60.0 | 64.9 / 57.7 |
| Post-CRF | 67.4 / 60.7 | 65.9 / 58.4 |

Table 1 gives the accuracy of the two versions of the network on the frames of the ScanNet dataset, with width multipliers $w = 0.5$ and $w = 1.0$, before and after applying the CRF step (with iterations $I = 5$), on 312 images each of the validation and test sets. Figure 2 shows some example predictions made by the network with width multiplier $w = 1.0$ after applying the CRF.

## 4 Feature-Based SLAM: ORB-SLAM2

We use ORB-SLAM2 [11], a feature-based SLAM system that picks out a relatively small number of 'ORB features' in each frame. These features, also called

[1] We use `pydensecrf`, available at github.com/lucasb-eyer/pydensecrf.

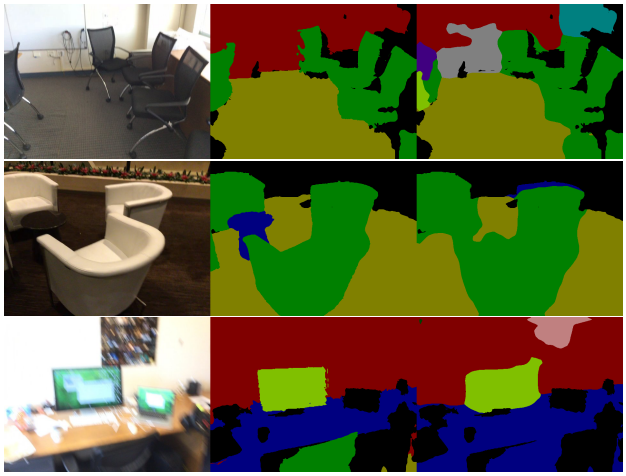

Fig. 2: Example images and predictions using width multiplier $w = 1.0$. Left: input images. Middle: ground truth labels. Right: predicted labels.

keypoints, are invariant to small changes in scale and rotation [13], and so corresponding features can be found between different frames. These correspondences are then used to jointly infer the trajectory of the camera and the positions of these points in 3D space. We will call the 3D points, which make up the model of the scene as constructed by ORB-SLAM2, MapPoints, in accordance with the terminology used in the ORB-SLAM2 source code.

### 4.1 Semantic Map Construction

We construct the 3D semantic map by projecting all predictions of the segmentation network into the 3D scene maintained by ORB-SLAM2. We can do this by using the SLAM system's prediction for the camera pose and the depth values from the depth camera. The result is that we overlay a dense or semi-dense semantic map on top of the sparse 3D model produced by the SLAM system.

Since the SLAM system does not know how pixels other than keypoints/features correspond to each other in different frames, we cannot combine predictions from multiple views; the role of the SLAM system in constructing the semantic map is restricted to estimating the current camera pose. The structure of the system in this 'single view' setting is shown in Figure 3. For comparison, a feature-based 'multi-view' approach is described in the appendix.

## 5 Performance Models

In the single view setting, single-frame predictions are embedded directly into the 3D semantic map using the current estimate of the camera pose from the SLAM system and the depth channel of the images. If we assume that both are

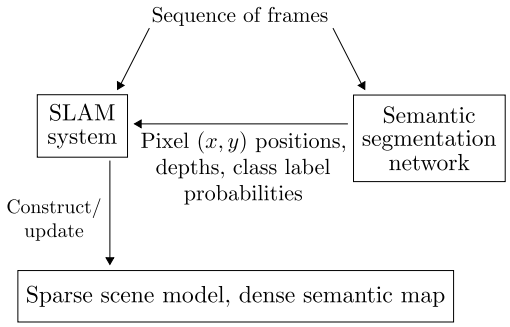


Fig. 3: Class label predictions from a deep network for semantic segmentation are passed to the SLAM system along with depth information. The SLAM system overlays this dense semantic map over its sparse 3D scene model.

accurate, then the accuracy of the single view 3D semantic map on a scene in the ScanNet dataset is the same as the accuracy of the semantic segmentation network averaged across the frames in that scene. This allows us to, unlike in the multiview setting described in the appendix, efficiently estimate the 3D semantic map accuracy by measuring the accuracy of the segmentation on a subset of frames.

The parameters listed in Section 5.1 affect the map accuracy, the map density, the inference time per frame, and the number of frames between inference frames. The key idea of this section is that we can fit an accuracy and a density model *offline* and then, on a given platform, or with a given new implementation of one of the components of the system, we can efficiently fit a timing model *online*. The three models—for the accuracy, density, and inference time—then allow us to choose parameter settings such that we can run the system in real time while maximizing the accuracy and density of the semantic map. The following sections describe the density, accuracy, and timing models used and the fitting process for the timing model.

### 5.1 Accuracy, Density, and Computation Trade Off Parameters

We have various means to reduce the amount of computation required at the cost of reducing the accuracy and density of the produced 3D semantic map. The MobileNets semantic segmentation network has a width multiplier $w$ and input resolution rescale factor $r$ parameters. As well as rescaling the input images, we can crop them using a *crop factor* $c$, feeding only a region of each frame to the segmentation network. Another possibility is to skip keyframes using a *skip factor* $s$, performing inference on one frame in every $s$.

We can also vary the number of *CRF iterations* $I$. The method used to compute the posterior of the CRF is iterative, with the approximation improving with the number of iterations. We vary the number of iterations between zero—leaving the network output unaltered— and five.

### 5.2 Density Model

The density of the semantic map, measured by the number of points in the map, is a simple function of the rescale and crop factors $r$ and $c$ and the skip parameter $s$; it is simply the number of individual pixel predictions that are projected into the map. The resolution of the semantic segmentation output scales with the square of each of the rescale and crop factors, whereas the number of inference frames with predictions being inserted into the map scales inversely with the skip parameter. The map density is therefore proportional to

$$f_{\text{density}}(r, c, s) = \frac{r^2 c^2}{s} \quad .$$

### 5.3 Accuracy Model

We evaluate the accuracy of the 3D semantic map by projecting the predictions onto individual frames and comparing to the groundtruth. Therefore, the accuracy of the 3D map on a scene is the same as the accuracy of the semantic segmentation network averaged over frames in that scene. The accuracies (overall and per-class) of the semantic segmentation network are functions of the width multiplier $w$, the number of CRF iterations $I$—we have already seen the effect of these parameters on the accuracy of the segmentation network in Table 1—and possibly of the rescale and crop factors $r$ and $c$. We express these as two functions $f_{\text{overall}}(w, I, r, c)$ and $f_{\text{per-class}}(w, I, r, c)$. We wish to fit offline two functions $\hat{f}_{\text{overall}}$ and $\hat{f}_{\text{per-class}}$ that will be used to estimate the accuracies for given parameter settings.

We randomly sampled 3500 times from the joint parameter space, with $w \in \{0.5, 1.0\}$, $I \in \{0, 1, \ldots, 5\}$, and with $r$ and $c$ sampled uniformly from $[0.469, 1.0]$, rejecting the sample if $r \cdot c < 0.469$. This reflects the way that rescaling and cropping were performed during training, with the input image resolution always at least $224 \times 224$. For each setting of the parameters, we measure the overall and per-class accuracy on 32 random frames from the dataset.

We found by inspection and by conditional mutual information measures that, conditional on the width multiplier and number of CRF iterations, the rescale and crop factors are not informative for predicting accuracy values[2]. Since the width multiplier and number of CRF iterations are discrete parameters with a small number of values—there are twelve combinations in total—we use the empirical mean overall and per-class accuracies for each parameter setting as our estimate for $\hat{f}_{\text{overall}}(w, I)$ and $\hat{f}_{\text{per-class}}(w, I)$. These estimates are shown in Figure 4, along with their associated standard errors.

### 5.4 Timing Model

The inference time of the semantic segmentation network is a function of the width multiplier $w$ and the rescale and crop factors $r$ and $c$, with the rescale and

[2] To measure conditional mutual information, we used the *scikit-feature* feature selection library available at github.com/jundongl/scikit-feature.

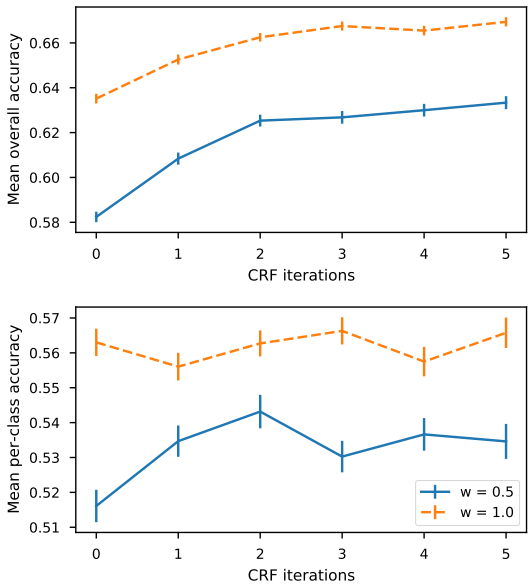


Fig. 4: Mean overall and per-class accuracies and standard errors. The mean values are used as a predictive model of the accuracy of the 3D semantic map as a function of the width multiplier and number of CRF iterations.

crop factors acting symmetrically to reduce the size of the input and subsequent layers in the network. We express the inference time as $f_{net}(w, rc)$. The inference time of the CRF step is a function of the number of CRF iterations $I$ and the rescale and crop factors, expressed as $f_{CRF}(I, rc)$. We wish to fit online two functions $\hat{f}_{net}$ and $\hat{f}_{CRF}$ in order to estimate inference time when running the system on a new platform, or when we have new implementations of the components of the system. For example, we may have switched from running the neural network on a GPU to a CPU, or switched the CRF inference to a neural implementation.

We make the assumption that each of these functions can be approximated by a low order polynomial, with terms up to the second power in each variable. I.e., we assume that $f(x, y) \approx \{a_{ij}x^i y^j | (i, j) \in \{0, 1, 2\}^2\}$. We also know that we can set the CRF part of the timing model to zero if the number of CRF iterations is zero. We fit the timing models by expanding the independent variables out to the appropriate polynomial basis, and then use linear regression with a mean squared error loss function to fit the parameters.

Once the timing model has been fit, it is used to restrict the choice of parameters to those that will allow inference to be performed in real time. On the ScanNet dataset, ORB-SLAM2 selects a keyframe every 1040 ms on average. Since inference is performed on one in every $s$ keyframes, where $s$ is the skip

parameter, we restrict ourselves to parameter settings where

$$\frac{\hat{f}_{net}(w, rc) + \hat{f}_{CRF}(I, rc)}{s} \leq 1040\,\text{ms} \quad .$$

### 5.5 Example use of the Parameter Tuning Method

We collected timing information for 170 settings of the parameters with the semantic segmentation network running on an Nvidia GTX 1050 (Notebook) graphics card (GPU), and another 170 settings with the segmentation network running on an Intel i7-7700HQ (CPU). Table 2 shows the results of fitting the timing model to this data. The first column gives the range of values for inference time across the parameter settings sampled. The second column gives the root mean squared error (RMSE) for the predictive timing models.

Table 2: Range of times (ms) taken for neural network and CRF inference and RMSE of the corresponding predictive timing model.

(a) CPU

| | Range | RMSE |
|---|---|---|
| Network inference time | 33.3–82.2 | 4.0 |
| CRF inference time | 108–586 | 32 |

(b) GPU

| | Range | RMSE |
|---|---|---|
| Network inference time | 91.8–627 | 7.2 |
| CRF inference time | 124–1013 | 50 |

The final stage of the tuning tool is to take a large number of samples from the parameter space, and then exclude from consideration those that—according to the timing model—would result in slower than real time performance. We then select the Pareto front with respect to the semantic map accuracy and map density, according to the accuracy and density models. The resulting Pareto fronts in this case are shown in Figure 5.

## 6 Conclusions

ORB-SLAM2 provides good 3D model accuracy and real-time processing [1]. Following a 'single view' approach of overlaying a dense semantic map over the sparse SLAM scene model (ORB-SLAM), we explore a method for automatically tuning the parameters of the system such that it operates in real time while maximizing prediction accuracy and map density.

We can fit models of the semantic map accuracy and density offline, and efficiently fit an inference time model online on a given, new platform, or when we have new implementations of the system components. Given a platform on which ORB-SLAM2 runs in real time, this allows us to construct a 3D semantic map, also in real time, while maximizing map accuracy and density. There is room for the development of more sophisticated predictive accuracy and timing models.

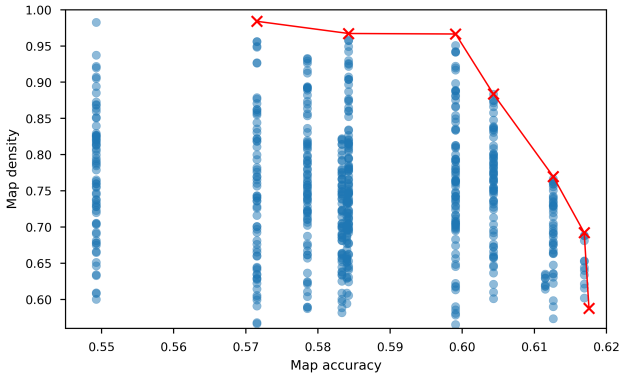


Fig. 5: Pareto front (red crosses) of accuracy and density. Each point (red or blue) is a setting of the parameters that allows the system to run in real time.

## References


1. Bodin, B., Wagstaff, H., Saeedi, S., Nardi, L., Vespa, E., Mawer, J., Nisbet, A., Luján, M., Furber, S.B., Davison, A.J., Kelly, P.H.J., O'Boyle, M.F.P.: Slambench2: Multi-objective head-to-head benchmarking for visual slam. 2018 IEEE International Conference on Robotics and Automation (ICRA) pp. 1–8 (2018)
2. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In: ICLR (2015), http://arxiv.org/abs/1412.7062
3. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2017)
4. Häne, C., Zach, C., Cohen, A., Angst, R., Pollefeys, M.: Joint 3D Scene Reconstruction and Class Segmentation. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 97–104 (June 2013). https://doi.org/10.1109/CVPR.2013.20
5. Hinton, G.E.: Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation **14**(8), 1771–1800 (2002). https://doi.org/10.1162/089976602760128018, https://doi.org/10.1162/089976602760128018
6. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR **abs/1704.04861** (2017), http://arxiv.org/abs/1704.04861
7. Krähenbühl, P., Koltun, V.: Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 24, pp. 109–117. Curran Associates, Inc. (2011), http://papers.nips.cc/paper/4296-efficient-inference-in-fully-connected-crfs-with-gaussian-edge-potentials.pdf
8. Kundu, A., Li, Y., Dellaert, F., Li, F., Rehg, J.M.: Joint Semantic Segmentation and 3D Reconstruction from Monocular Video. In: Fleet, D., Pajdla, T., Schiele,

B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 703–718. Springer International Publishing, Cham (2014)
9. Li, X., Belaroussi, R.: Semi-Dense 3D Semantic Mapping from Monocular SLAM. CoRR **abs/1611.04144** (2016), http://arxiv.org/abs/1611.04144
10. McCormac, J., Handa, A., Davison, A.J., Leutenegger, S.: SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. 2017 IEEE International Conference on Robotics and Automation (ICRA) pp. 4628–4635 (2017)
11. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. IEEE Transactions on Robotics **33**(5), 1255–1262 (2017). https://doi.org/10.1109/TRO.2017.2705103
12. Pillai, S., Leonard, J.: Monocular SLAM Supported Object Recognition. In: Proceedings of Robotics: Science and Systems (RSS). Rome, Italy (July 2015)
13. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: Computer Vision (ICCV), 2011 IEEE international conference on. pp. 2564–2571. IEEE (2011)
14. Sünderhauf, N., Dayoub, F., McMahon, S., Talbot, B., Schulz, R., Corke, P., Wyeth, G., Upcroft, B., Milford, M.: Place categorization and semantic mapping on a mobile robot. In: IEEE International Conference on Robotics and Automation (ICRA 2016). IEEE, Stockholm, Sweden (May 2016)
15. Vineet, V., Miksik, O., Lidegaard, M., Niener, M., Golodetz, S., Prisacariu, V.A., Khler, O., Murray, D.W., Izadi, S., Prez, P., Torr, P.H.S.: Incremental Dense Semantic Stereo Fusion for Large-Scale Semantic Scene Reconstruction. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 75–82 (May 2015). https://doi.org/10.1109/ICRA.2015.7138983
16. Whelan, T., Leutenegger, S., Moreno, R.S., Glocker, B., Davison, A.: ElasticFusion: Dense SLAM Without A Pose Graph. In: Proceedings of Robotics: Science and Systems. Rome, Italy (July 2015). https://doi.org/10.15607/RSS.2015.XI.001
17. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: ICCV. pp. 1529–1537 (2015)

# A Appendix: Multi-View Semantic Map Construction with Feature-Based SLAM

In this section we describe a multi-view approach to associating semantic predictions with the 3D scene model of ORB-SLAM2, by using the correspondence between keypoints in different frames recorded by the SLAM system to combine predictions. We show that this approach—similar to SemanticFusion [10]—has drawbacks when used with sparse, feature-based SLAM systems.

We have modified ORB-SLAM2 to, for each keyframe, pass the $(x, y)$ positions of keypoints to the code implementing the segmentation network. The segmentation network performs inference on each keyframe, and passes the prediction probability vector for each keypoint back to ORB-SLAM2. ORB-SLAM2 then computes an aggregate prediction for each MapPoint by combining the predictions of the associated keypoints. This setup is illustrated in Figure 6. The aggregate MapPoint prediction probabilities were computed by taking the element-wise product of the keypoint prediction probabilities and then renormalizing. This is like a *product of experts* in ensemble machine learning methods [5]. Other aggregation methods were tried, such as taking the arithmetic mean or a maximum vote, with similar results.

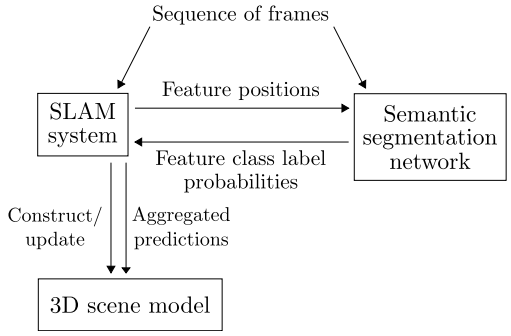


Fig. 6: Class label predictions from a deep network for semantic segmentation are made based on multiple views of the same objects and then associated with parts of the 3D model constructed by the SLAM system.

We report the accuracy of the segmentation network across all pixels and across all keypoints in the test set, and the accuracy of the 3D semantic map based on multi-view (aggregate) feature predictions. For each of these, we compute the 'overall accuracy', which is the total proportion of correctly classified pixels or MapPoints, and the 'per-class accuracy', which is the mean of the proportion of correctly classified pixels or MapPoints for each class.

These results are shown in Table 3. The first three rows show the accuracies computed for various settings of the parameters described in Section 5.1. The first row gives the results for setting the parameters to maximize accuracy at the

Table 3: Overall / per-class accuracy (%) of the semantic segmentation network averaged across pixels and features, and the accuracies of the multiview-constructed 3D semantic map. The 'full network' has width parameter $w = 1.0$ with CRF iterations $I = 5$. The 'no CRF' network is the same but with CRF iterations $I = 0$. The third row is the same as the 'full network' but with a width multiplier of $w = 0.5$. The final row results are for a modified version of ORB-SLAM2 which uses more keyframes.

| | Accuracies (OP/PC) (%) | | |
|---|---|---|---|
| | Per pixel | Per feature | Multiview |
| Full network | 65.9 / 58.4 | 58.0 / 53.3 | 59.9 / 54.9 |
| No CRF | 64.9 / 57.7 | 53.4 / 51.1 | 55.9 / 53.1 |
| $w = 0.5$ | 56.6 / 56.3 | 49.9 / 53.7 | 51.9 / 54.9 |
| More frames | 64.9 / 57.7 | 52.5 / 51.0 | 55.7 / 53.7 |

cost of increased computation; the full network is used, with width multiplier $w = 1.0$, there is no cropping, rescaling, or frame skipping, and we apply $I = 5$ CRF iterations. The second row shows the results with the same parameters, but without any CRF post-processing. For the third row the parameters are the same as the first, but with the 'half width' network, with $w = 0.5$.

The multi-view per-feature predictions consistently give a two to three percentage point improvement in accuracy over the per-frame per-feature accuracy; combining predictions does result in increased accuracy. This low improvement—compared to the three to seven percentage point improvement seen from multi-view predictions in SemanticFusion—seems to be due to low diversity amongst predictions based on multiple views; in cases where the multiview predictions are wrong, the corresponding pairwise single view predictions (i.e., the predictions being combined) are the same in approximately 75% of cases, and the KL-divergence of the pairwise prediction probabilities are low. These diversity measures are shown in Table 4. This low diversity may in turn be due to ORB features being invariant under only small changes in orientation and scale, so that the multiple views that are combined are very similar.

Another feature of the results is that restricting predictions to only keypoints—as we are required to in order to take advantage of ORB-SLAM2 to combine predictions from multiple views—results in a reduction in accuracy by around 3-6 percentage points compared to the accuracy measured over all pixels; this drop in accuracy more than compensates for the increase in accuracy that comes from combining predictions from multiple views. This may be because ORB features are likely to be found on corners and edges, and so may be likely to be found on the boundary between objects. These points will be harder to classify, and a lower accuracy will result if the segmentation edges do not align well with object edges. Some evidence is lent to this interpretation by the fact that the drop in accuracy when restricting predictions to keypoints is higher when no CRF iterations are applied, as seen in Table 3, and that the use of the CRF

drastically reduces the KL-divergence between predictions associated with the same MapPoint, as shown in Table 4; the CRF, by aligning segmentation edges with object edges, has removed a major source of uncorrelated errors between predictions.

In the multiview setting, a surprisingly small number of observations/predictions are associated with each MapPoint; the mean number of observations per MapPoint is a little over four. It is possible to modify ORB-SLAM2 to create more keyframes per frame. The final row of Tables 3 and 4 give the results for a modified version of ORB-SLAM2, with a mean number of 7.0 observations per MapPoint. This modified version still shows only a small improvement in accuracy for multiview predictions over single view predictions.

Table 4: Mean number of observations/predictions per MapPoint. The second and third column show, in the case that the aggregate prediction for a MapPoint is wrong, the pairwise probability that two predictions agree and the pairwise KL-divergence between the prediction probabilities.

| | Obs. | Agreement | KL div. |
|---|---|---|---|
| With CRF | 4.35 | 0.778 | 0.0375 |
| Without CRF | 4.34 | 0.754 | 0.337 |
| More frames, no CRF | 7.0 | 0.752 | 0.292 |

In this section, we have shown that the popular method of combining predictions from multiple views in conjunction with a SLAM system to build a 3D semantic map is not suitable in the sparse, feature-based SLAM setting. Restricting predictions to ORB-SLAM2 keypoints, as is required for the multiview approach, reduces the semantic map accuracy by more than the increase in accuracy from combining predictions from multiple views can compensate for, suggesting that multiview semantic map construction using a sparse, feature-based SLAM system is not viable if the features are likely to appear on object boundaries, as will often be the case. It may be possible to do multiview prediction with a feature-based SLAM system by modifying the features such that they are more likely to appear in object interiors, but this is likely to affect SLAM tracking performance.