Hand-Object Interaction Detection with Fully Convolutional Networks

Matthias Schröder Helge Ritter
Neuroinformatics Group, Bielefeld University
{maschroe,helge}@techfak.uni-bielefeld.de

Abstract

Detecting hand-object interactions is a challenging problem with many applications in the human-computer interaction domain. We present a real-time method that automatically detects hand-object interactions in RGBD sensor data and tracks the object's rigid pose over time. The detection is performed using a fully convolutional neural network, which is purposefully trained to discern the relationship between hands and objects and which predicts pixel-wise class probabilities. This output is used in a probabilistic pixel labeling strategy that explicitly accounts for the uncertainty of the prediction. Based on the labeling of object pixels, the object is tracked over time using model-based registration. We evaluate the accuracy and generalizability of our approach and make our annotated RGBD dataset as well as our trained models publicly available.

1. Introduction

The visual detection and recognition of human actions by technical systems is a fundamental problem with many applications in domains such as virtual or augmented reality, tangible computing, robotic teleoperation, or human-computer interaction systems generally. Hand-object interactions are a particularly important type of such actions, since the way in which humans interact with the world is often based around the interplay between hands and objects. Visually detecting and processing hand-object interactions is a challenging problem due to the high complexity of the involved movements, and because the acquisition of such movements is made difficult by issues like occlusions or visual ambiguities, particularly in monocular vision setups.

Existing systems that process hand-object interactions typically use pre-specified visual markers or color information to discriminate between hands and objects in sensor data. We present a method that is capable of hand-object discrimination irrespective of object color or shape, and without the need for visual markers. To this end, we use a data-driven approach, in which a fully convolutional neural network is trained to discern the relationship between hands and objects, and then used to produce dense, pixel-wise predictions for the localization of hands and objects. By combining these pixel-wise predictions with the 3D information obtained from an RGBD sensor, we track the object's rigid pose in real-time as it is undergoing user manipulation. Figure 1 illustrates our method with an example.

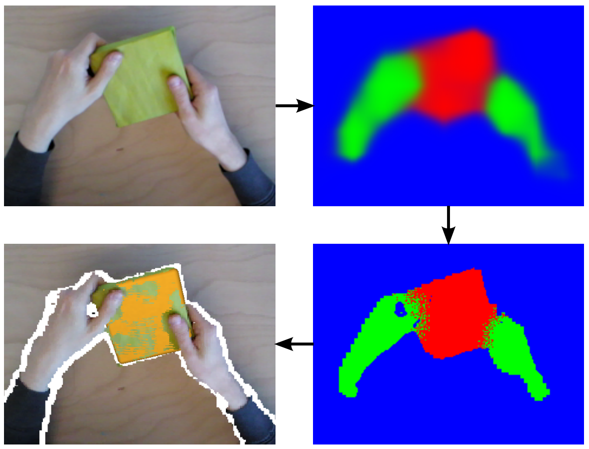
Figure 1: Our method localizes interacting hands and objects in an input image (top left) by computing pixel-wise probabilities. The resulting probability map (top right) discriminates between background (blue), hands (green) and objects (red). Based on this, a dense labeling of hand and object pixels is generated (bottom right). The classified pixels are then used to recover the 6D pose of the manipulated object (bottom left) in real-time.

Our data-driven approach is based on a purposefully designed dataset of approximately 50,000 RGBD frames containing a variety of hand-object interactions. We make this dataset as well as our trained convolutional neural network models publicly available.[1] In a thorough evaluation of our hand-object discrimination approach, we explore different variations of our system architecture and input data modalities. We show that, despite being trained with a low variation of objects, our method can generally detect previously unknown objects while they are manipulated by hands.

[1] https://ni.www.techfak.uni-bielefeld.de/node/3579

2. Related work

The outstanding performance of convolutional neural networks (CNNs) in image recognition tasks [12, 17, 19, 9] has been leveraged in a wide range of computer vision domains by employing transfer learning [27], where pre-trained CNNs are adapted and fine-tuned towards new tasks. For image segmentation tasks this typically involves using a pre-trained CNN as an encoder that generates feature maps, which are then processed by a decoder employing transposed convolutions to generate the segmentation map [13, 1, 14, 26]. Such end-to-end methods differ from approaches that use CNNs in addition to pre-processing or post-processing [7, 6, 3].

Since we aim to minimize computational overhead in our real-time application, we adopt the end-to-end fully convolutional network (FCN) approach of [13] to generate dense predictions for hand and object pixels. Our approach differs from theirs in the way the class labels are generated: instead of simply maximizing the FCN's output probabilities, we randomly sample from the pixel-wise probabilities and take depth information into account, which improves accuracy. Depth information has been previously utilized for segmentation with CNNs [4, 8]. In particular, [8] used the depth map to compute a 3-channel geocentric embedding that encoded information w.r.t. an estimated gravity direction. In contrast, we explore gravity-independent depth-based input modalities in addition to color, including normal maps and depth-based foreground masking.

A CNN approach for localization and recognition of hand-object interactions in color images has been proposed in [16]. Similar to our method, this approach uses a FCN to generate dense probability maps to predict the location of body parts and objects. These are then used in a complex multi-stage classification scheme for offline action recognition, while our approach uses the FCN outputs to discriminate the hand from the object and to generate pixel labels for real-time tracking. Similar to our approach, [28] used FCNs to discriminate between objects for 6D pose estimation. While this approach produces good results for object localization in cluttered environments, it requires a multi-camera setup and does not achieve real-time performance, which is crucial for dynamic hand-object interactions.

Real-time free-hand tracking using CNNs has been accomplished in previous works [22, 15]. Unlike our approach, these methods estimate joint positions of a freely moving hand after first detecting the hand's bounding box in a separate pre-processing step. In [25] the user's hands are segmented in input images with a two-part CNN architecture. Rather than transposed convolutional layers, this approach uses fully connected layers to generate the hand segmentation. Our method uses FCNs to generate hand-object probability maps, from which a dense pixel labeling is created. The resulting labels are used together with the input point cloud to track the object in real-time.

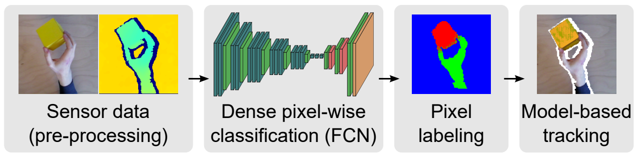

Figure 2: Overview of our method. Hand and object are discriminated in the input sensor data using a FCN and a probabilistic labeling strategy. The resulting labeled pixels are used as geometric correspondences for object tracking.

We adapt the model-based hand tracking method of [20, 21] towards rigid object tracking. This method is highly efficient, as well as robust against occlusions and noisy RGBD input data. Beyond free-hand tracking, the state of the art in simultaneous hand-object tracking has advanced significantly [24, 23, 18], but while these methods achieve highly accurate pose estimations, they typically still rely on simple color-based segmentation for distinguishing between hand and object, whereas our work represents a step towards fully automatic hand-object discrimination.

3. Method overview

Our method detects interacting hands and objects in sensor data and tracks the 6D pose of the manipulated object. Figure 2 shows an overview of the steps involved in this process. The input data is obtained using an RGBD sensor, which is mounted above the tabletop interaction area and captures the user's hands and the objects. Each captured frame is processed to conform with the input modalities required by the FCN (e.g. resizing to square dimensions). Using this input frame, the FCN produces a 3-channel probability map with probability values for each class (background, hand, object) at every pixel location. Based on these pixel-wise probabilities, and combined with the depth information of the input frame, a dense labeling of hand and object pixels is generated. These labeled pixels are then used to define data correspondences for model-based tracking, which estimates the object's 6D pose over time by fitting a model of the object to the labeled data. In the following sections we describe the details of each step.

4. Dense pixel classification

We follow the FCN architecture of [13] and fine-tune it towards the hand-object interaction domain. Generally, this architecture consists of an encoder part, which uses convolutions and pooling to compute feature maps with decreasing spatial resolution and increasing depth, and a decoder part, which uses transposed convolutions and element-wise fusion to generate class score maps with the spatial dimen-

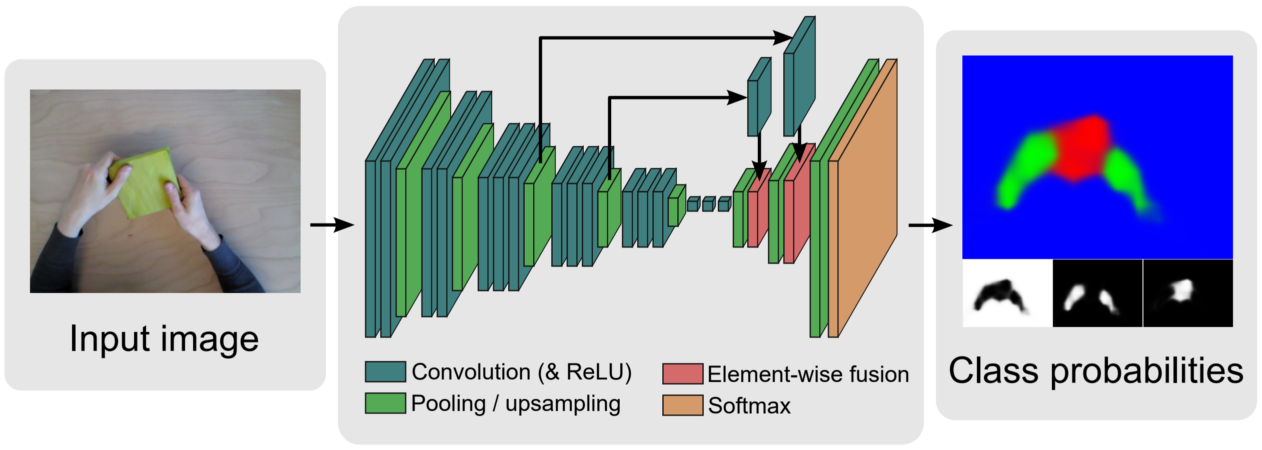

Figure 3: Dense pixel classification using a fully convolutional network with skip connections. The FCN computes a series of feature maps in the encoder part of the network (downsampling), which are then used to generate the class probability map in the decoder (upsampling). By using skip connections, the network successively fuses feature maps from different layers to obtain a high-resolution result. The classes between which our FCN distinguishes are background, hands, and objects.

sions of the input image. In the following, we describe the FCN architecture and training in more detail.

4.1. FCN architecture

Figure 3 illustrates the FCN architecture and the probability maps generated from an input image. The encoder part of the shown network is a fully convolutionalized version of the VGG-16 architecture [17, 13], which stacks a series of convolutional blocks, each consisting of several convolution-nonlinearity layers followed by a pooling layer. Overall, the encoder reduces spatial dimensions through max-pooling five times, each time downsampling the input by 50% (*pool1–pool5* layers).

The decoder upsamples score maps computed from the encoder's feature maps to the source image resolution using transposed convolutions. Score maps are produced by additional 1×1 convolution layers (without nonlinearity) generating class scores from a given feature map. The simplest decoder is one that directly upscales the score map computed from the last encoder layer to the input image dimension. However, after five pooling operations the encoder's receptive field is too large to capture finer silhouette details. To resolve this, [13] proposed using *skip connections*, which fuse score maps of higher-resolution layers by element-wise addition. This concept is illustrated in Figure 3 for two skip connections, after the *pool3* and *pool4* layers. Adding more than two skip connections reaches diminishing returns [13], and we found that using skip connections from layers that are not sufficiently deep yields unsatisfactory results (see Section 6). We refer to FCN architectures without skip connections as *single stream nets* and ones with skip connections as *skip nets*.

4.2. FCN training

We trained the FCN using a dataset of approximately 50,000 images containing hand-object interactions as well as free-hand movements. Since the purpose of our FCN is primarily to discern the *relationship* between hand and object, rather than only the *appearance* of specific objects, our dataset was designed with a high variation of grasps and manipulations and a low variation of objects. Our results show that even with low object variation in the training data, unknown objects can still be detected (see Section 6).

We recorded several interactions with two cuboid objects, one equilateral (see Figure 2) and one flat (see Figure 1), which are two shapes that afford a high variety of grasp and manipulation types. We followed the grasp taxonomy of [5] to capture as many different, distinct hand-object manipulation types as possible. The ground truth labels for hand, object, and background pixels were created with simple color classification, depth-based foreground masking, and manual annotation. An example of an image with its ground truth is shown in Figure 4a and Figure 4b.

The FCN training is based on transfer learning from a pre-trained model [13], and we follow their recommendations for the training parameters. We performed training for 20 epochs, using batch size 16, base learning rate 1×10^{-10}, momentum 0.99, and weight decay 5×10^{-4}. The dataset was randomly shuffled and 90% of the frames were used as training data, 10% as test data. To generate our particular class score maps, we added a 1×1 convolution before the softmax layer. The 320×240 sensor images are rescaled to be square FCN inputs. We experimented with 200×200 and 100×100 images, and found that smaller images improved runtime performance while not notably impacting accuracy.

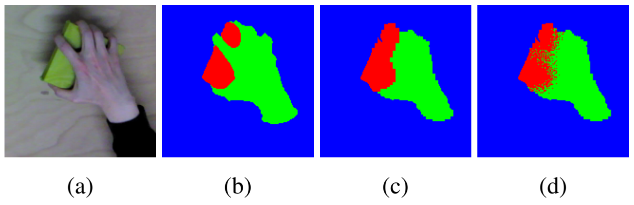
Figure 4: Computed pixel labeling compared to ground truth. (a) Input image. (b) Ground truth labeling. (c) Result of argmax labeling. (d) Result of fuzzy labeling. Unlike the argmax approach, our fuzzy labeling is able to coarsely capture the index finger that covers the object.

5. Pixel labeling and object tracking

Based on the FCN outputs, we generate a pixel-wise hand-object labeling and use the object labels to track the object's 6D pose. Pixel-wise classification is is typically done by selecting the class with the highest probability. However, since the FCN predictions are relatively coarse and inherently uncertain, the accuracy of the region outlines resulting from this approach is limited.

We address this by only labeling pixels whose depth values are within a given interaction volume. While this produces accurate outlines, resolving the ambiguity in areas where hand and object overlap is still challenging. In the following we describe a labeling strategy that directly incorporates the probabilistic nature of the FCN outputs, and we show that using this strategy improves object tracking accuracy.

5.1. Probabilistic labeling

We seek the class label l_i at pixel i given the N class probabilities $\{p_{c,i}\}_{c\in\{1,\ldots,N\}}$ at this pixel. The labeling resulting from maximizing the probabilities can be written as

$$\hat{l}_i = \underset{c\in\{1,\ldots,N\}}{\mathrm{argmax}} \{p_{c,i}\}. \tag{1}$$

We refer to this labeling strategy as *argmax labeling*. This labeling produces clear delineations between regions, but does not capture the uncertainty in the prediction. Instead, we determine the class label probabilistically by randomly sampling from the probability distribution at the current pixel, which can be written as

$$l_i = \underset{c\in\{1,\ldots,N\}}{\mathrm{sample}} \{p_{c,i}\}, \tag{2}$$

where sample is a function that returns a class index $c \in \{1,\ldots,N\}$ randomly selected according to the corresponding probabilities $\{p_{c,i}\}$. We refer to this labeling strategy as *fuzzy labeling*.

Figure 4 compares argmax labeling and fuzzy labeling with the ground truth of an example image. The argmax approach produces a clear delineation between hand and object, but does not capture the index finger pixels, whose probability values are below the object probability. Conversely, the fuzzy approach is able to recover more details of this ambiguous area.

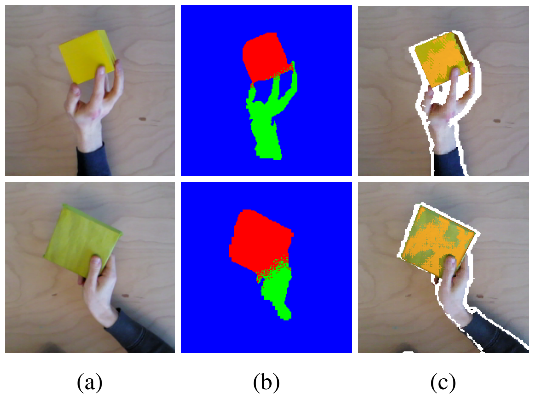
Figure 5: Different object tracking examples. (a) Input image. (b) Fuzzy labeling result. (c) Model fitted to point cloud.

5.2. Model-based tracking

Given the pixels labeled as the object class, we estimate the object's 6D pose by fitting a pre-specified geometric model of the object to the observed sensor point cloud data. Rather than using standard ICP [2], we adapt the robust registration method of [20, 21] from articulated tracking to rigid object tracking. This method is robust w.r.t. outliers and handles occlusions by accounting for visibility constraints during correspondence computation. In particular, correspondences between data and model are computed bidirectionally, which improves fitting accuracy.

Applied to our context, optimizing the rigid transformation $\boldsymbol{\theta}$ of the model $\mathcal{M}$ to match the object-labeled point cloud data $\mathcal{D}$ involves minimizing an objective function

$$\begin{aligned} E(\mathcal{D},\mathcal{M},\boldsymbol{\theta}) = \quad & \omega_1 \sum_{\mathbf{x}\in\mathcal{D}} \|\mathbf{x} - \Pi_{\mathcal{M}(\boldsymbol{\theta})}(\mathbf{x})\|_2^1 \\ & + \omega_2 \sum_{\mathbf{p}\in\mathcal{M}(\boldsymbol{\theta})} \|\mathbf{p} - \Pi_{\mathcal{D}}(\mathbf{p})\|_2^1, \end{aligned} \tag{3}$$

where $\Pi_{\mathcal{G}}(\mathbf{x})$ denotes the closest point projection of vector $\mathbf{x}$ onto geometry $\mathcal{G}$. The first term in (3) measures the fit of sensor data points $\mathbf{x} \in \mathcal{D}$ to the model, and the second term measures the fit of rendered model silhouette pixels $\mathbf{p} \in \mathcal{M}(\boldsymbol{\theta})$ to the sensor silhouette. For more details on the correspondence computation and the minimization of the objective function, we refer to [20, 21].

Figure 5 shows examples for the model-based object tracking. Figure 6 compares results for the model-based tracking when using the argmax labeling strategy and the

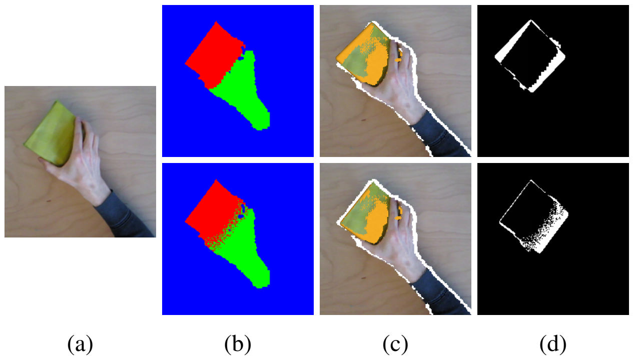

Figure 6: Tracking results with argmax labeling and fuzzy labeling. (a) Input image. (b) Label image. (c) Model fitted to point cloud. (d) Difference image between rendered model silhouette and object class labels (top: argmax, bottom: fuzzy).

fuzzy labeling strategy. When using argmax labeling, the fitting produces a bad overlap between the model and the detected object pixels, causing the model to shift from the true object position. With fuzzy labeling, there are enough object pixels to be used as fitting correspondences in areas where the hand covers the object, such that the fitting can align the model well with the object pixels.

6. Results

We evaluated our method on a validation dataset with approximately 10,000 frames of hand-object interactions that were not used during training, including both known and unknown movements and objects. To measure the accuracy of the pixel labeling w.r.t. ground truth, we use the mean pixel accuracy and mean region intersection over union (IoU) metrics as they are defined in [13, Section 5].

In the following, we evaluate the accuracy of our approach with respect to the basic CNN encoder architecture, the use of skip connections in the decoder, and variations of input data types. In addition to these quantitative results, we also provide some examples for the detection of unknown objects. All results were produced on a PC with an i7 CPU, 16 GB RAM, and a GTX 1070 GPU, using the Caffe framework [11].

6.1. FCN architecture variations

In addition to the VGG [17] architecture described in Section 4, we performed experiments with smaller CNN architectures for the encoder part of the FCN to reduce computational costs. In particular, we experimented with AlexNet [12], which is approximately 44% of VGG's size, and SqueezeNet [10], which is less than 10% of VGG's size. We refer to the different fully convolutionalized architectures as FCN-VGG, FCN-AlexNet, and FCN-SqueezeNet in the following. In the single stream versions of FCN-AlexNet and FCN-SqueezeNet the final layers generating class scores were substituted with upsampling layers to generate the class probability maps. For the skip net versions, we added skip connections after the *pool5* and *norm2* layers in FCN-AlexNet and before the *pool5* and *pool3* layers for FCN-SqueezeNet. We used transfer learning as described in Section 4 to train our FCNs, adding gradient accumulation and a higher learning rate for FCN-AlexNet and FCN-SqueezeNet.

Architecture (two skip conn.)	Model size	Inference time	Pixel acc.	Mean IoU
FCN-VGG	513 MB	32.8 ms	72.6	67.2
FCN-AlexNet	224 MB	8.6 ms	56.7	55.2
FCN-SqueezeNet	4.9 MB	4.6 ms	56.1	54.9

Table 1: Comparison of different FCN architectures. While using smaller models significantly reduces file size and inference time, the top accuracy drops notably.

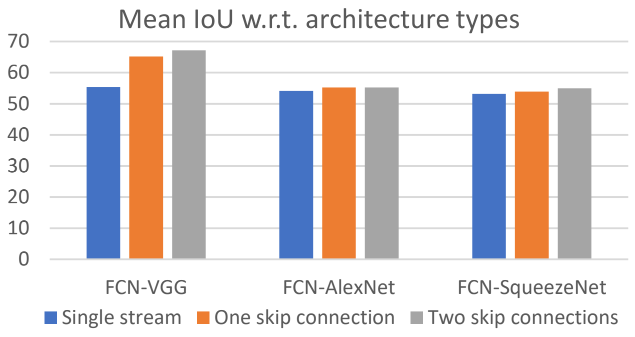

Figure 7: Mean IoU for all FCN variants. Adding skip connections improves the accuracy of FCN-VGG, but has no significant impact on FCN-AlexNet and FCN-SqueezeNet.

Table 1 shows the model parameter file size, the inference time for one image, the mean pixel accuracy, and the mean IoU for each FCN variant using two skip connections. Architectures with less parameters greatly reduce memory requirements and inference time, but this comes at the cost of accuracy. Figure 7 shows the mean IoU for all single stream and skip net variants. While the skip connections noticeably improve the performance of FCN-VGG, they have no significant impact on the other two smaller alternatives, which remain around the same accuracy level as the single stream net. In order to learn features to delineate regions in an image, the corresponding layers must be sufficiently deep, or have large convolution kernels. The skip connection layers in the smaller FCNs do not fulfill these criteria and therefore these architectures do not benefit from the skip net design.

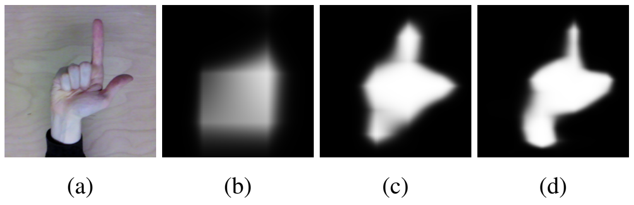

Figure 8: Comparison of probability map outputs for different FCN-VGG variants. (a) Input image. (b) Output for single stream net. (c) Output for net with one skip connection. (d) Output for net with two skip connections.

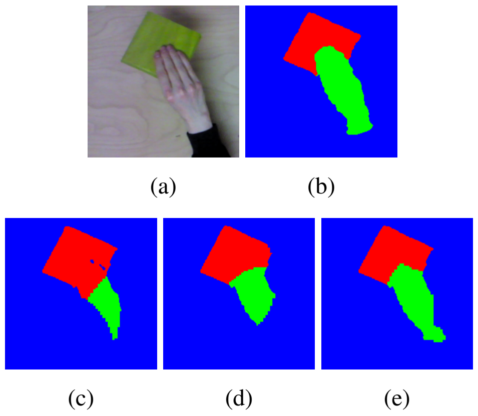

Figure 9: Comparison between ground truth and argmax labeling results for different FCN variants. (a) Input image. (b) Ground truth. (c) Single stream net result. (d) One-skip net result. (e) Two-skip net result.

Figure 8 compares the probability map outputs of the single stream, one-skip, and two-skip nets for the *hand* class using FCN-VGG. The resolution of the skip nets is higher and therefore better able to capture details in the outline of the hand. Figure 9 shows the segmentations resulting from argmax labeling for single stream and skip nets in comparison with the ground truth. The boundary between hand and object is better approximated by the skip nets due to their higher resolution. Overall, while using small single stream nets is highly efficient and can yield acceptable results, the improved detail and accuracy in the results of the skip nets outweigh the performance gain. As the inference time of FCN-VGG still allows for real-time operation (32.8 ms), we use this architecture in our system.

6.2. Input data modalities

We explored different possibilities of incorporating depth information in addition to color as the input to our system, in order to exploit the additional geometric information this provides. Since our method is based upon transfer learning from models pre-trained on color data, we considered ways to incorporate such geometric information with as little changes to the base architecture as possible. The depth-based scene descriptor of [8] is not directly applicable to our setup, as we do not estimate or presuppose a geocentric coordinate frame. Instead, we experimented with foreground masking using a depth threshold to eliminate any possible background clutter, and using normal maps as input instead of color. We found that using raw depth maps did not yield acceptable results.

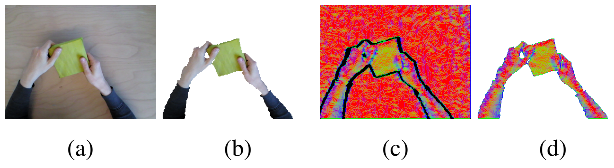

Figure 10: Evaluated input data modalities. (a) Full color image. (b) Color image with depth-based foreground masking. (c) Full normal map. (d) Normal map with depth-based foreground masking.

Input data	Mean pixel acc.	Mean IoU
Color, full	72.6	67.2
Color, masked	72.1	67.8
Normals, full	63.6	57.6
Normals, masked	67.2	61.0

Table 2: Mean pixel accuracy and mean IoU of FCN-VGG using different input data modalities.

Figure 10 shows an example for the different input data modalities. Normal maps trade discriminative features found in color images for additional geometric features and invariance to lighting, distance and scale. We generated training data for all four modalities from the same ground truth data and performed training as described in Section 4. In order to transfer weights learned on 8-bit, 3-channel color data to normal data, we transformed the normal map values from $[-1, 1]^3$ to $[0, 255]^3$. Transfer learning with untransformed normal data did not converge in our experiments.

Table 2 compares the accuracy of FCN-VGG using the different input data types. For color input, using the full images yields the best results and foreground masking does not significantly impact accuracy, which may be due to the fact that in our setup the background is fairly homogeneous and uncluttered, so the discrimination between foreground and background is not very problematic. Using normal maps as input yields lower overall accuracies due to the loss of discriminative color features, but when combined with foreground masking, the accuracy is improved. The accuracy of skip net architectures using normals still surpasses those of single stream nets using color, which suggests that normal maps may be a viable alternative when color data is unavailable, or when geometric information should be exploited in addition to color.

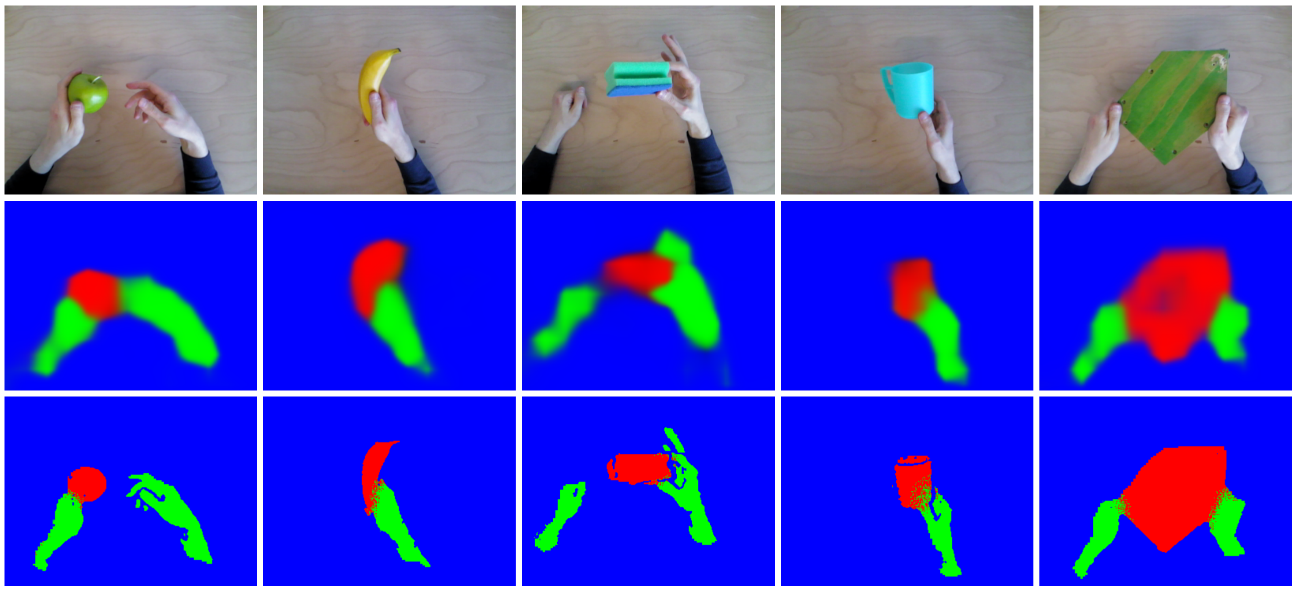

Figure 11: Examples of our hand-object discrimination with unknown objects. First row: input images. Second row: FCN class probability outputs. Third row: fuzzy labeling results. Our method is capable of detecting objects of different shapes, sizes, colors and textures, although these objects were not contained in the training database.

6.3. Detecting unknown objects

To test the generalizability of our approach, we used it to detect hand-object interactions with objects that were not part of the FCN's training data. Figure 11 shows several such examples, along with the FCN class probability outputs and fuzzy labeling results. Despite the objects exhibiting variations in shape, size, color and texture, our method is capable of successfully detecting them and discriminating between hands and objects. This indicates that our trained model detects the *relationship* between hands and objects, rather than only their *appearance*. Notably, in the class probability map of the first example (Figure 11, leftmost column), the right hand is anticipated to be connected with the object despite being separated, which is resolved during the pixel labeling. Conversely, in the third example (Figure 11, third column), the left hand and object are correctly predicted to be separated. We also found that hands and objects can be detected in isolation, which reaffirms similar observations made in previous works [7, 28].

Our real-time system runs at 28 fps, with the majority of the computation time per frame being spent on FCN inference (32.8 ms). The labeling and model-based tracking do not impact runtime performance significantly (3 ms).

7. Discussion

We have presented a method for detecting hand-object interactions by densely classifying an input image obtained from an RGBD sensor into hand, object, and background categories and generating a labeling of hand and object pixels to facilitate real-time object tracking. The dense pixel classification is computed using a fully convolutional network, which extracts features from the input image and produces a class probability map of the same resolution as the input. Based on this output, a probabilistic pixel-wise labeling is performed, in which the uncertainties of the FCN classification are explicitly accounted for. Using the resulting object pixels, the 6D pose of the object is accurately tracked by continuously fitting a geometric model of the object to the sensor point cloud using robust registration.

In several experimental evaluations, we explored the effects of different variations of the FCN architecture as well as different input data modalities. There is a trade-off between the computational complexity of a FCN model and the accuracy of its results, and in order to benefit from skip connections added for higher resolution outputs the corresponding CNN layers must strike a balance between depth and receptive field size. CNN models pre-trained on color data can be fine-tuned to normal data, which trades discriminative color information for geometric information, at the expense of accuracy. We showed that our dense pixel classification and labeling method generalizes to interactions with objects that were not part of the training data.

The FCN employed for localization of hands and objects was fine-tuned towards hand-object discrimination by training on a new dataset of approximately 50,000 densely annotated RGBD frames containing various hand-object interactions. The dataset was designed in a principled manner in order to effectively cover all the different ways in which hands and objects can interact. Making our dataset and trained models publicly available opens up new possibilities to study and develop methods for detection, tracking, or recognition of hand-object interactions. Future avenues of research involve the development of new FCN architectures optimized for efficiency, simultaneous hand and multi-object tracking, and recognition of gestures and actions.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

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

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[4] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor semantic segmentation using depth information. *CoRR*, abs/1301.3572, 2013.

[5] M. R. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *Trans. Robotics and Automation*, 5(3):269–279, 1989.

[6] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *Trans. Pattern Anal. Mach. Intell.*, 35(8):1915–1929, 2013.

[7] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[8] S. Gupta, R. B. Girshick, P. A. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[10] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems*, pages 1106–1114, 2012.

[13] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[14] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision*, pages 1520–1528, 2015.

[15] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a feedback loop for hand pose estimation. In *International Conference on Computer Vision*, pages 3316–3324, 2015.

[16] A. Rosenfeld and S. Ullman. Hand-object interaction and precise localization in transitive action recognition. In *Conference on Computer and Robot Vision*, pages 148–155, 2016.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[18] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from RGB-D input. In *European Conference on Computer Vision*, pages 294–310, 2016.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[20] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly. Robust articulated-icp for real-time hand tracking. *Comput. Graph. Forum*, 34(5):101–114, 2015.

[21] A. Tkach, M. Pauly, and A. Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *Trans. Graph.*, 35(6):222:1–222:11, 2016.

[22] J. Tompson, M. Stein, Y. LeCun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *Trans. Graph.*, 33(5):169:1–169:10, 2014.

[23] D. Tzionas, L. Ballan, A. Srikantha, P. Aponte, M. Pollefeys, and J. Gall. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision*, 118(2):172–193, 2016.

[24] D. Tzionas and J. Gall. 3d object reconstruction from hand-object interactions. In *International Conference on Computer Vision*, pages 729–737, 2015.

[25] T. Vodopivec, V. Lepetit, and P. Peer. Fine hand segmentation using convolutional neural networks. *CoRR*, abs/1608.07454, 2016.

[26] J. Wang, Z. Wang, D. Tao, S. See, and G. Wang. Learning common and specific features for RGB-D semantic segmentation with deconvolutional networks. In *European Conference on Computer Vision*, pages 664–679, 2016.

[27] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Conference on Neural Information Processing Systems*, pages 3320–3328, 2014.

[28] A. Zeng, K. Yu, S. Song, D. Suo, E. W. Jr., A. Rodriguez, and J. Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. *CoRR*, abs/1609.09475, 2016.