J Sign Process Syst (2017) 88:205–217
DOI 10.1007/s11265-016-1209-3

Deep Learning for Automated Occlusion Edge Detection in RGB-D Frames

Soumik Sarkar[1,4] · Vivek Venugopalan[1] · Kishore Reddy[1] · Julian Ryde[2] · Navdeep Jaitly[3,5] · Michael Giering[1]

Received: 2 September 2015 / Revised: 3 November 2016 / Accepted: 20 November 2016 / Published online: 8 December 2016

Abstract Occlusion edges correspond to range discontinuity in a scene from the point of view of the observer. Detection of occlusion edges is an important prerequisite for many machine vision and mobile robotic tasks. Although they can be extracted from range data, extracting them from images and videos would be extremely beneficial. We trained a deep convolutional neural network (CNN) to identify occlusion edges in images and videos with just RGB, RGB-D and RGB-D-UV inputs, where D stands for depth and UV stands for horizontal and vertical components of the optical flow field respectively. The use of CNN avoids hand-crafting of features for automatically isolating occlusion edges and distinguishing them from appearance edges. Other than quantitative occlusion edge detection results, qualitative results are provided to evaluate input data requirements and to demonstrate the trade-off between high resolution analysis and frame-level computation time that is critical for real-time robotics applications.

Keywords Deep learning · Occlusion edge detection · Automatic feature extraction · Robotics applications

✉ Soumik Sarkar
soumiks@iastate.edu

Vivek Venugopalan
venugov@utrc.utc.com

Kishore Reddy
reddykk@utrc.utc.com

Julian Ryde
rydejc@utrc.utc.com

Navdeep Jaitly
ndjaitly@gmail.com

Michael Giering
GierinMJ@utrc.utc.com

[1] Decision Support & Machine Intelligence, United Technologies Research Center, East Hartford, CT 06108, USA

[2] Embedded Systems, United Technologies Research Center, Berkeley, CA, USA

[3] Department of Computer Science, University of Toronto, Ontario, Canada

[4] Present address: Iowa State University, Ames, IA, USA

[5] Present address: Google Inc., Mountain View, CA, USA

1 Introduction

Occlusion edge detection is a fundamental capability of computer vision systems as is evident from the number of applications and significant attention it has received [1–5]. Occlusion edges are useful for a wide array of tasks including object recognition, feature selection, grasping, obstacle avoidance, navigating, path-planning, localization, mapping and stereo-vision. In addition to numerous applications, the concept of occlusions edges is supported by the human visual perception research [6] where it is referred to as figure/ground determination. Once occlusion boundaries have been established, depth order of regions become possible [7, 8] which aids navigation, simultaneous localization and mapping (SLAM) and path planning. Specifically, there is utility of geometric edges for running SLAM algorithms for mobile robots. Many textureless environments are not suitable for feature based SLAM techniques despite being relatively common in indoor environments. However, maps based on occlusion and geometric edges will still allow

localisation even in these low texture regions (see example in Fig. 1). For indoor mapping, planes seem a natural consideration for landmarks and indeed numerous researchers have explored plane based mapping [9]. Although less common in robotic experiments, there are places not suitable for planar mapping including buildings with curved walls, natural outdoor environments and extremely cluttered scenes such as those found in search and rescue scenarios. Although planes can be a good way of compressing map information, an observed planar surface is not as constraining to robot pose as feature points and edges. Another strong motivation for using the geometric edges is that they allow localisation by both range and image sensors. For many indoor environments, considering only geometric edges, removes floors, walls and ceilings leaving the elements that lie along the intersection of planes or in cluttered regions, resulting in significant compression of the map data. While such compression may not be significantly effective in many unstructured outdoor environments, occlusion edge detection can still be useful in certain structured scenarios such as roads and pavements.

Occlusion edges also help image feature selection by rejecting features generated from regions that span an occlusion edge. As these are dependent on viewpoint position, removing these variant feature saves on further processing and increases recognition accuracy [10]. In many object recognition problems, the shape of the object is better for recognition rather than its appearance, which can be easily dramatically altered e.g., by painted objects, camouflage and people wearing different clothes. However, shape determination is not the approach for state-of-the-art SIFT based object recognition algorithms. Furthermore, knowledge of occlusion edges is a key component of virtual reality (VR), mixed reality (MR) [11, 12] and optic flow algorithms [7]. In robotics, geometric edges of objects demarcate their spatial extents helping with grasping, manipulation as well as maneuvering through the world without collision and therefore, knowledge of occlusion edges is essential.

In the context of Dynamic Data Driven Applications Systems (DDDAS), it is therefore essential to develop algorithms that enable faster and more accurate occlusion edge determination via intelligent processing of heterogeneous information sources such as RGB, depth information and motion related data. Such information can be dynamically accommodated in the map of the environment and system model to improve the accuracy of decision-making. More accurate and efficient decision-making strategies can in turn help the measurement system, in this case the RGB and Depth camera to improve scene understanding capabilities by performing necessary actions such as camera movement, pan, tilt and zoom. In this context, recent works [13, 14] show the effectiveness of the DDDAS framework for various vision and perception problems. Furthermore, to enhance automation and efficiency, meticulous hand-crafting of visual features should be avoided as much as possible. Finally, efficient frame-wide decision-making scheme is required along with other key information regarding the model space and control objectives for closing the control loop. In general, there are multiple specific DDDAS applications that this study will be relevant for such as target recognition, surveillance and tracking and video processing.

In this context, this paper evaluates the efficacy of Deep Learning tools [15] for the task of occlusion edge detection. Recently, this class of techniques have emerged as the top performing machine learning tool for various tasks such as object recognition [16], speech recognition [17], denoising [18], hashing [19] and data fusion [20]. While Deep Neural Networks (DNN) pre-trained using Deep Belief Networks (DBN) [21, 22] perform quite well in most data types, deep Convolutional Neural Networks [23, 24] have been shown to be most suited for images. The better performance is primarily attributed to the preservation of local structures (i.e., localized pixel dependencies) by CNN as opposed to DBN-DNN (where, typically layers are fully connected bipartite graphs). The occlusion edge detection task can

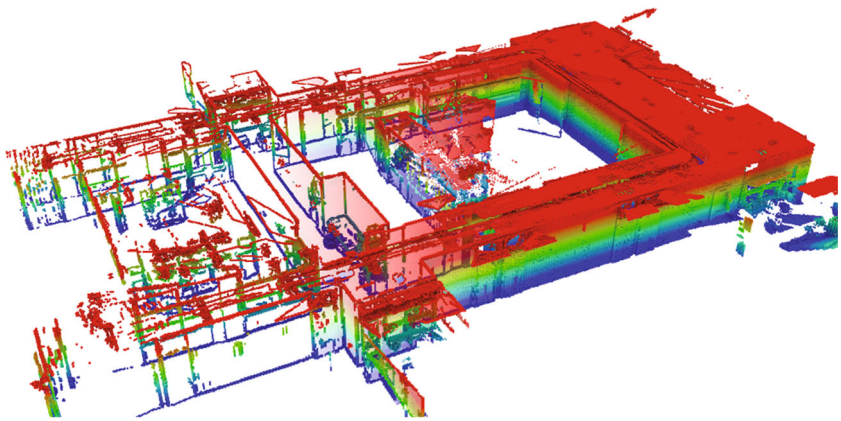

Figure 1 A voxel map and the corresponding geometric edges for the *mason hallway* dataset [25].

logically be conceived as a two step process: identifying edges in an image followed by distinguishing between occlusion and appearance edges. Therefore, deep neural networks are particularly interesting for this problems as they extract hierarchical features (features of features) from data and visualization of intermediate optimized filters [16] show that edge type features are very common. It also should be noted that such an approach eliminates the need for complicated hand-crafting of features that is commonly done in many current approaches. Due to availability of GPUs and recent advancements in the algorithmic/implementation side, model parameters of large CNNs can be effectively learnt using sufficient amount of data for complex problems [16] and overfitting problems can be avoided significantly. In fact, the CNN model size (depth and breadth) can be optimized iteratively for a certain problem. Often however, memory of the implementing GPU becomes the bottle-neck.

In this paper, the main contributions are: (i) formulation of an occlusion edge detection problem as a classification of center-pixels of an image patch with RGB, Depth (D) and optical flow field (UV) channels (ii) performance evaluation of CNN with various input information sources, namely RGB, RGB-D and RGB-D-UV for occlusion edge detection problem and (iii) fusion of patch predictions to generate frame-wide occlusion edges which can be used for robotic applications. Note, similar methods and studies exist in different contexts such as for tracking with occlusion detection [26], wearable multimodal sensor fusion [27], vehicle registration [28] and wide-area motion imagery [29]. This study uses a publicly available benchmark RGB-D (with multiple time frames) data set captured with moving camera in an indoor environment by the Computer Vision group at Technische Universität München (TUM) [30]. The optimized and hardware-accelerated CNN implementation has been done on NVIDIA K-40 GPU.

The paper is organized in seven sections including the introduction. The problem formulation along with the data set description is provided in Section 2. While Section 3 provides the details of architecture and training parameters for the CNN, testing and post-processing are discussed in Section 4. Various experiments with corresponding quantitative results are provided in Section 5 and qualitative observations are articulated in Section 6. Finally, the paper is summarized and concluded with future research directions in Section 8.

2 Problem Formulation and Generating the Training data

In general, it is difficult to define occlusion edge pixels rigorously. In an image, edges manifest along paths of high contrast and are due to four main reasons: (i) texture change, i.e., abrupt change in surface color, (ii) lighting change, i.e., sharp shadows, (iii) range discontinuity, i.e., abrupt change in distance from the observer and (iv) surface normal change, e.g., intersection of two planes. Throughout this work it is assumed that appearance edges are a necessary but not sufficient condition for occlusion edges. This assumption is rarely violated in real world environments but when it is then even the human visual system fails.

It is important to appreciate the distinction in the causes of image edges. Texture change and illumination edges are not observed by 3D sensors. Therefore, the remaining geometric edge types are range discontinuities and abrupt surface normal changes. Surface normal changes are pose invariant, however edges due to range discontinuities can vary with observer position. These surface normal and range discontinuities are illustrated in the last image of Fig. 2. The cylinder sides in Fig. 2 are examples of range discontinuities. The position of these edges varies in 3D space as the position of the observer shifts whereas the cylinder rim edge position is consistent regardless of observer position. For use in mapping, the following characteristics is desired from extracted edge voxels: they should be generally invariant to rotation and translation, and they should be helpful in terms of constraining pose. Therefore, in this study the focus is on identifying the third and fourth type of edges, i.e., edges due to range discontinuity and surface normal change.

Traditional approaches for detect geometric edges in 3D data include a keypoint detector based on a 3D extension of the Harris corner operator in the Point Cloud Library [31]. This detector operates on local normals of points. A related approach for selecting interest points on 3D meshes was introduced in [32]. In principle, this study is similar to a recent works on indoor scene segmentation [33] and depth map prediction [34]. However, this study focuses on if only occlusion edges can be isolated using CNNs and also if reasonable performance can be achieved without using the depth channel of the RGB-D data. As mentioned earlier, this paper uses a benchmark RGB-D data set the Computer Vision group at Technische Universität München (TUM). The data set contains RGB and depth images of a Microsoft Kinect sensor that was recorded at full frame rate (30 Hz) and sensor resolution 640 × 480 by moving camera in an indoor environment. The occlusion edge detection problem is formulated as a classification problem and the procedure of generating training data is provided in the following subsection.

2.1 Training Data

Although most of the occlusion edge detection exercises, training labels are generated manually, the occlusion edge information is largely present in a clean version of the depth

Figure 2 Image with associated edges due to appearance and due to geometry.

(D) channel. Therefore, occlusion edge label for a pixel, i.e., the ground truth can be automatically determined to some extent using the depth channel data. However, visually it can be observed that ground truth obtained in such a way may have a large percentage of missed detection. Still, we show that this automated label generation process enables training of a deep CNN. The label generation procedure is illustrated in Fig. 3. From left to right, the three plates in the figure shows an example RGB frame, the corresponding (clean) D channel data and classification frame generated using a simple thresholding only on the depth data. Other than gray (signifying no edge) and white (signifying occlusion edges) colors, the black color can be seen in the classification frame. This signifies bad depth measurements due to presence of absorbing surface or larger than maximum distance allowed between the sensor and the surface.

As shown in Fig. 4, the RGB-D data set was collected using a camera motion along a certain trajectory in an indoor environment. The trajectory is divided into disjoint training and testing sections so that the trained model can be tested using previously unseen data. The frames in the RGB-D data set are 480 × 640 in size. In order to create training examples for the Convolutional Neural Network (CNN), 32 × 32 patches are extracted from the large frames in the training section. The training label for each patch is determined by the pixels located at the center [35, 36]. As illustrated in Fig. 5, if majority of the pixels (2 × 2 in this case) at the center of a 32 × 32 patch contains occlusion edges, the patch is labeled as an **Occlusion** patch. On the other hand, if center pixels contain appearance edges or no edge, corresponding patch is labeled as a **No Occlusion** patch. Patches with considerable bad or unlabeled pixels are pre-filtered and not used for training. Furthermore, a class balancing is also performed between occlusion and no-occlusion examples within the training data set. As expected, we observe that balancing provides significant performance improvement as originally number of occlusion patches are significantly lower compared to no occlusion patches.

Other than 3-channels input with RGB and 4-channels input with RGB-D, we also explore 'structure from motion' using optical flow information. As we are interested in depth discontinuity, change in video frames due to motion can be quite useful. Specifically, we use a 2-frame estimation of horizontal (U) and vertical (V) components of the optical flow field with an iterative reweighted least square (IRLS) formulation. Figure 6 shows two consecutive frames and the output of the off-the-shelf optical flow algorithm [37].

Remark II.1 In an absolute sense, occlusion edges depend on the gradient of the depth image which is very sensitive to noise in the depth map and the depth map derived from a single image is very noisy and has large errors. In our work, we are estimating the occlusion edges directly rather than estimating depth first and then calculating occlusion edges. Secondly there are additional cues (RGB, UV) other than depth which contribute to establishing occlusion edges that our technique is taking advantage of.

Figure 3 Example RGB, depth and classification frames from the training data generation procedure. In the classification frame *gray* signifies no edge, occlusion edges are white and black is for no or unreliable data.

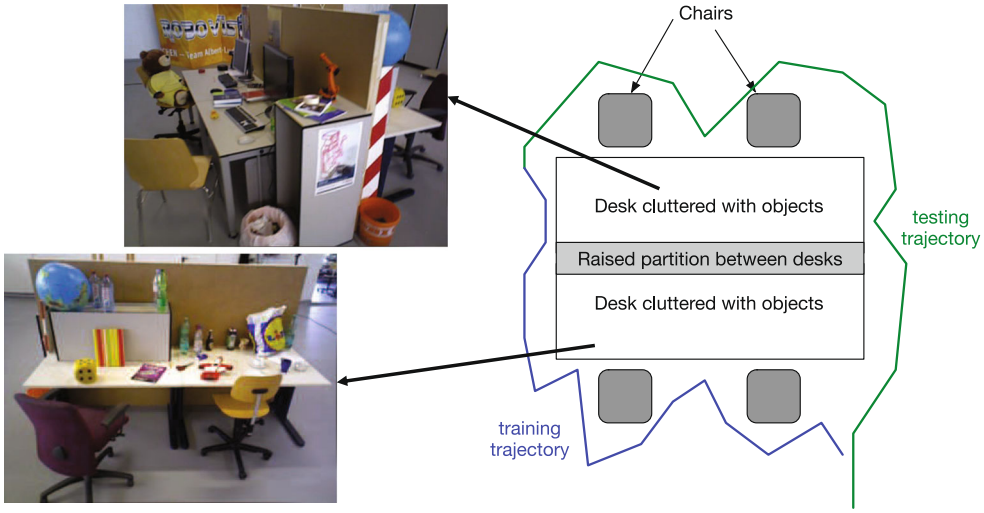

Figure 4 Partitioning of camera trajectory for collecting RGB-D data set into training and testing sections.

3 CNN Architecture and Model Learning

The architecture of the Convolutional Neural Network (CNN) used in this paper is illustrated in Fig. 7. The CNN has three pairs of convolution-pooling layers followed by softmax output layer [16]. This section articulates details of those layers as well as various hyper-parameters used for model learning.

Description of layers As described in Section 2.1, 32 × 32 patches were used as data for the CNN in this study. Depending on the experiment, different number channels are used for the input data. For example, while 4 channels were used for single (time) frame RGB-D data (as shown in Fig. 7), 6 channels were used for an RGB-D-UV sequence. Note, all these channels are passed independently through the convolution and max-pooling processes in parallel before combining them for the output layer. So, the convolution and max-pooling processes shown in Fig. 7 applies separately on patches with respect to every channel. More detailed description of various experiments will be provided in Section 5. The layer size parameters here correspond to the RGB-D experiment with 4 channels. The first convolutional layer uses 32 filters (or kernels) of size $5 \times 5 \times 4$ with a stride of 1 pixel and padding of 2 pixels on the edges. A two-fold sub-sampling or pooling layer follows the convolutional layer that generates the input data (of size $16 \times 16 \times 32$) for the second convolutional layer. This layer uses 32 filters of size $5 \times 5 \times 32$ with a stride of 1 pixel and padding of 2 pixels on the edges. A second pooling layer with the same specification as the first one is used after that to generate input with size $8 \times 8 \times 32$ for the third convolutional layer that uses 64 filters of size $5 \times 5 \times 32$ with same stride and padding strategies as before. The third pooling layer also has the same configuration as the two before it and leads to a softmax output layer with two labels corresponding to **No Occlusion** and **Occlusion** classes.

Hyper-parameters The CNN described above was trained using stochastic gradient descent with a mini-batch size of 100 examples. Although biases of convolutional layer neurons were initialized with constant values zero, weights of the neurons were initialized with zero-mean Gaussian distributions with standard deviations as: 0.0001 for first, 0.01 for second and 0.01 for third convolutional layer. Interestingly, the network performed better with a comparatively

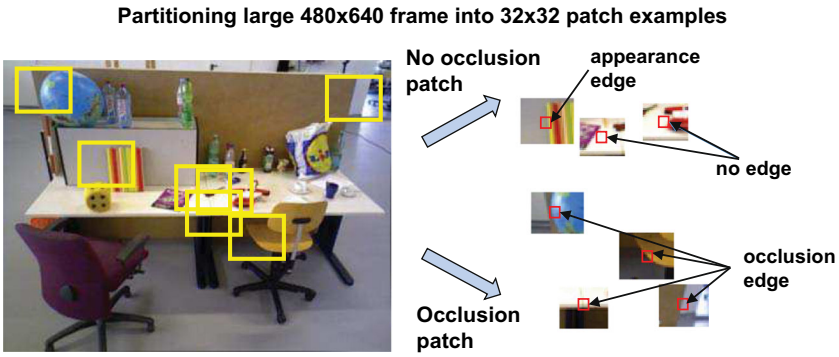

Figure 5 Generation of training data 32 × 32 patches from original 480 × 640 frames and labeling based on center-pixels.

Figure 6 Two consecutive RGB frames and the output of the optical flow algorithm.

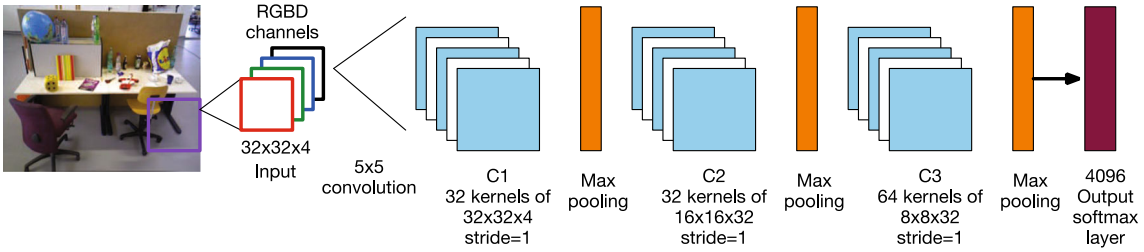

Figure 7 Illustration of Convolutional Neural Network (CNN) architecture used for Occlusion Edge classification.

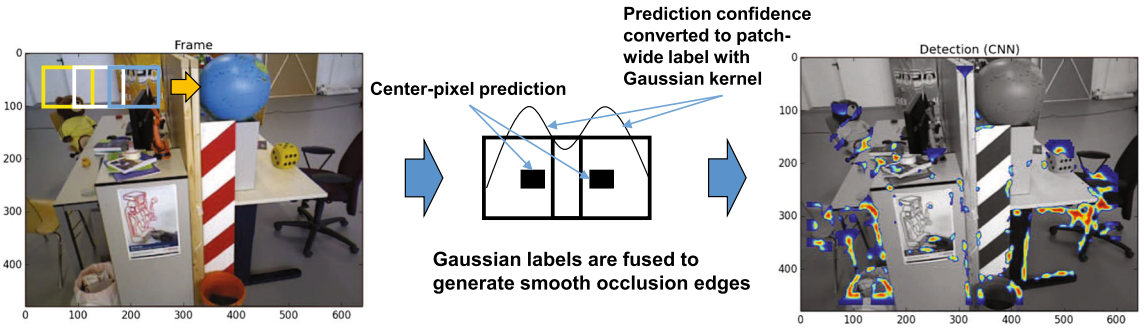

Figure 8 Post-processing at the testing phase involves collecting 32×32 overlapping patches with a constant stride from large frames; prediction confidence of a patch center pixel label is converted into a Gaussian kernel with Full Width at Half Maximum (FWHM); Gaussian labels are fused in a mixture model to generate smooth occlusion edges.

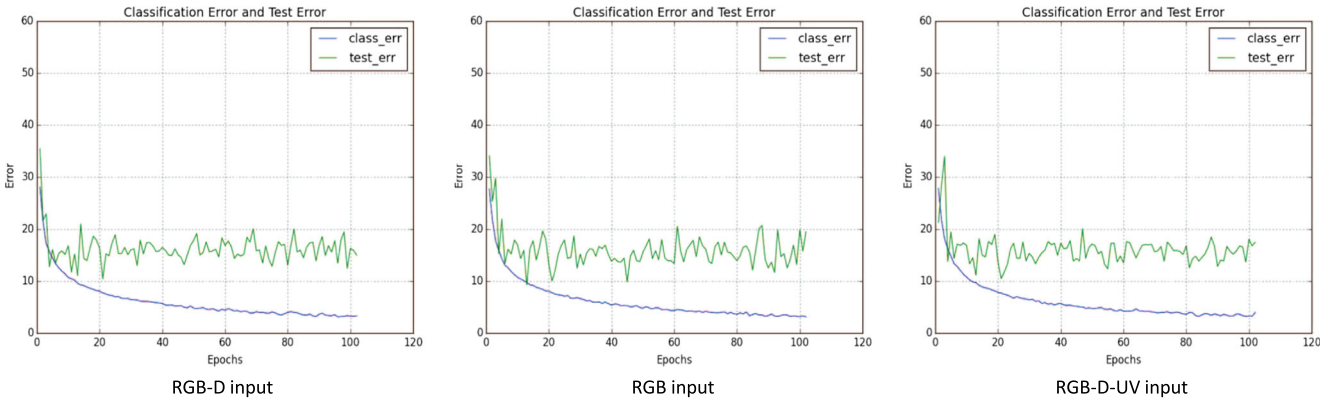

Figure 9 Training and testing error plots (for RGB-D, RGB and RGB-D-UV inputs) over various training epochs.

larger initialization of the weight standard deviation (0.3) for the output layer. The learning rate and momentum used for all the convolutional layers and for all training epochs were 0.001 and 0.9 respectively. Finally, $L2$-regularizers were used for all convolutional layers as well with weight 0.001. No dropout was used for model training in this study.

Training with GPU: The NVIDIA Kepler series K40 GPUs are FLOPS/Watt efficient and are being used to drive real-time image processing capabilities. The Kepler series GPU consists of a maximum of 15 Streaming Execution (SMX) units and up to six 64-bit memory controllers. Each SMX unit has 192 single-precision CUDA cores and each core comprises of fully pipelined floating-point and integer arithmetic logic units. The K40 GPUs consist of 2880 cores with 12 GB of on-board device memory (RAM). Deep Learning applications have been targeted on GPUs previously in [16] and these implementations are both compute and memory bound. Stacking of the channels for the RGB-timeseries and the RGB delta experimnets result in a vector of $32 \times 32 \times 12$, which is suitable for the Single Instruction Multiple Datapath (SIMD) architecture of the GPUs. At the same time, the training batch size caches in the GPU memory, so the utilization of the K40 GPU's memory is very high. This also results in our experiments to run succesfully on a single GPU instead of partitioning the different layers over multiple GPUs.

4 Testing and Post-processing

Performance testing of CNN is done in both quantitative and qualitative manner with various input information as will be explained in Section 5. For quantitative results, classification errors are computed based on the model's ability to predict label of the center pixels of a test patch collected from a frame captured in testing section of camera motion. The qualitative observations and visualization are made using a post-processing scheme as illustrated in Fig. 8. In this scheme, classification confidence for a patch center pixels is collected from the softmax posterior distribution and it is extrapolated across the patch using a Gaussian distribution with Full Width at Half Maximum (FWHM). Such Gaussian kernels from overlapping patches are fused in a mixture model to generate smooth occlusion edges in the testing frame.

5 Experiments and Quantitative Results

Different experiments are performed with different sets of input data for comparative evaluation. They are described below along with corresponding quantitative performance of the CNN model:

RGB-D frame The first set of experiments used single temporal frames of RGB-D data (i.e., 4 channels). This task may seem rather straight forward as the (noisy) depth information is directly available as one of the channels in the input data. However, majority of edges in the current frames are appearance edges and RGB channels clearly provide that information. Therefore, the task for the CNN model is to detect edges via automatic feature extraction and distinguishing occlusion edges from appearance edges.

RGB frame The second set of experiments used single temporal frames of RGB data (i.e., 3 channels). The goal here was to investigate if discriminative features exist and can be extracted by CNN from just RGB channels in order to classify patches into **Occlusion** and **No occlusion** edges. Ideally, without temporal information RGB channels may not carry a lot of occlusion information. However, occlusion information may remain in certain features such as shadows. Therefore, the goal here is to investigate if such features can be recognized by a CNN to detect occlusion edges.

RGB-D-UV frame The third set of experiments used UV channels (horizontal and vertical components of the optical flow field respectively) in addition to RGB-D channels (i.e., 6 channels). These additional channels provide the critical temporal information for occlusion edge detection.

Numerical results are provided below for all of these cases. For training the CNN, 57,518 training patches extracted from large image frames (collected in training section of the camera trajectory) are used. During testing, 1,271,002 patches (collected in testing section of the camera trajectory) are used to provide quantitative performance data. Figure 9 shows training and testing error plots over various epochs and specifically the training error graph clearly demonstrates that the training process does not saturate. Table 1 provides percentages of overall error, false alarm and missed detection averaged over epochs 80 through 100 as well as the value of the Pratt Metric averaged over the same epochs. Note, while the overall error, false alarm and missed detection percentages measure pixel-wise accuracy, the Pratt Metric evaluates the similarity between

Table 1 Occlusion detection performance of CNN with RGB-D, RGB and RGB-D-UV inputs.

Channels	Overall error	False alarm	Missed detection	Pratt Metric
RGB-D	16.43 %	16.10 %	43.04 %	0.287
RGB	15.36 %	15.20 %	46.70 %	0.292
RGB-D-UV	15.18 %	14.95 %	46.31 %	0.363

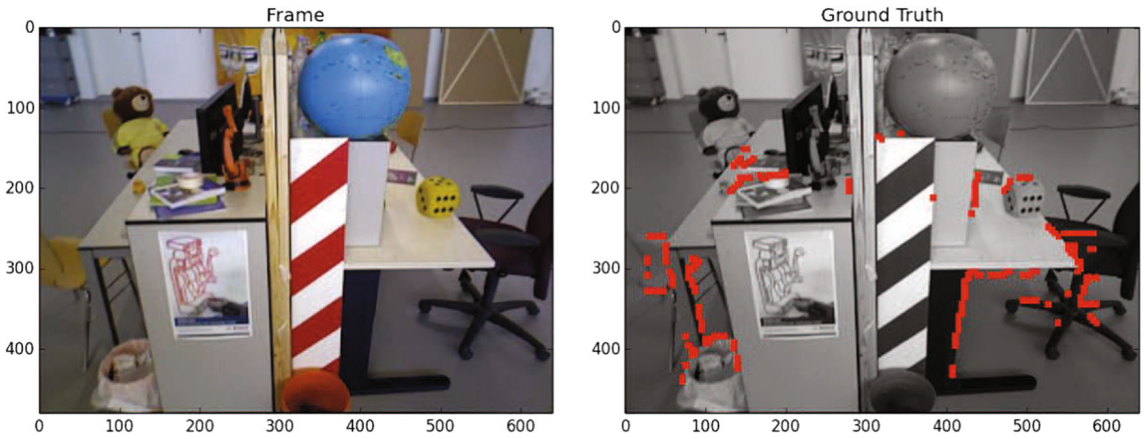

Figure 10 Example RGB frame and corresponding occlusion edge ground truth.

detected occlusion edges and the corresponding ground truth occlusion edges as described in [38].

As provided in Table 1, for both cases, false alarm performance is significantly better compared to missed detection performance. The primary reason for this is that the training labels are generated using an automated procedure using the imperfect depth channel and that can cause many missed detection training examples. For example, qualitative results in the next section show that the CNN based model captures certain occlusion edges which were not part of the ground truth. Numerically, overall error percentage is very close to false alarm rate as majority of the test example patches do not contain occlusion edges. The occlusion edge detection tool is more sensitive with RGB-D input compared to the RGB input. Therefore, missed detection percentage with RGB-D input is 3.65 % less compared to that with RGB input. However, that also causes false alarm rate to rise for RGB-D input. As majority of test example patches do not contain occlusion edges, overall error for RGB input is slightly lower than that of RGB-D input. Adding two more channels based on optical flow, reduces the false alarm rate without increasing the missed detection significantly. The overall performance is also found to be the best (both pixel-wise and in terms of the Pratt Metric) in the case of RGB-D-UV input. Overall, it is interesting to observe that performance is quite comparable even just with RGB channels compared to using all 6 channels. This is significant as it suggests that a vision system with deep learning algorithms can potentially recognize occlusion edges without any depth sensor.

6 Qualitative Observations

This section presents qualitative results in order to understand the efficacy of the deep learning tools for occlusion

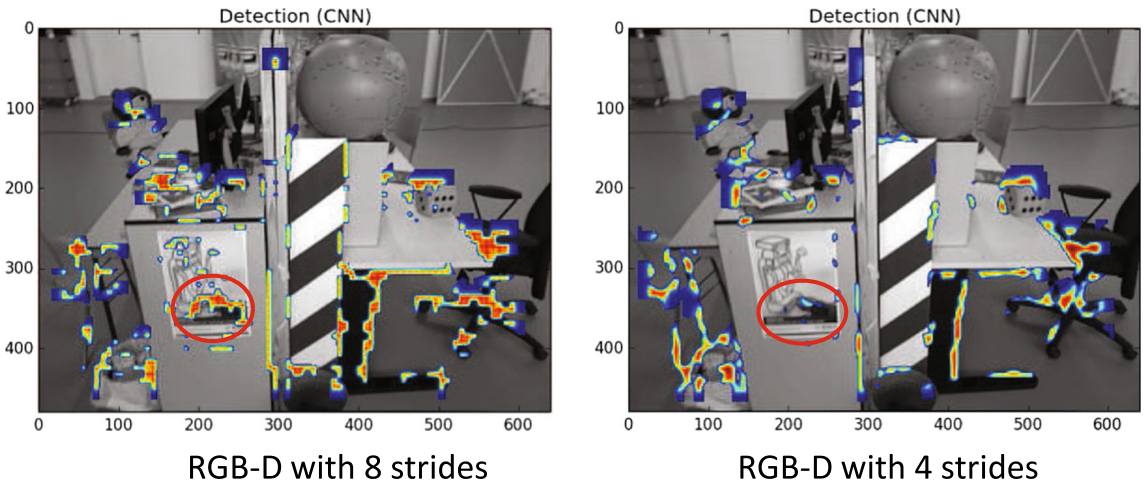

Figure 11 Occlusion edge detection performance on a test frame for RGB-D input with stride 8 and 4; heat map shows the fused detection confidence (red-yellow-blue signifies high-medium-low); red circled region shows example of confusing appearance edges as occlusion edges; performance improves while computational time increases with decrease in strides.

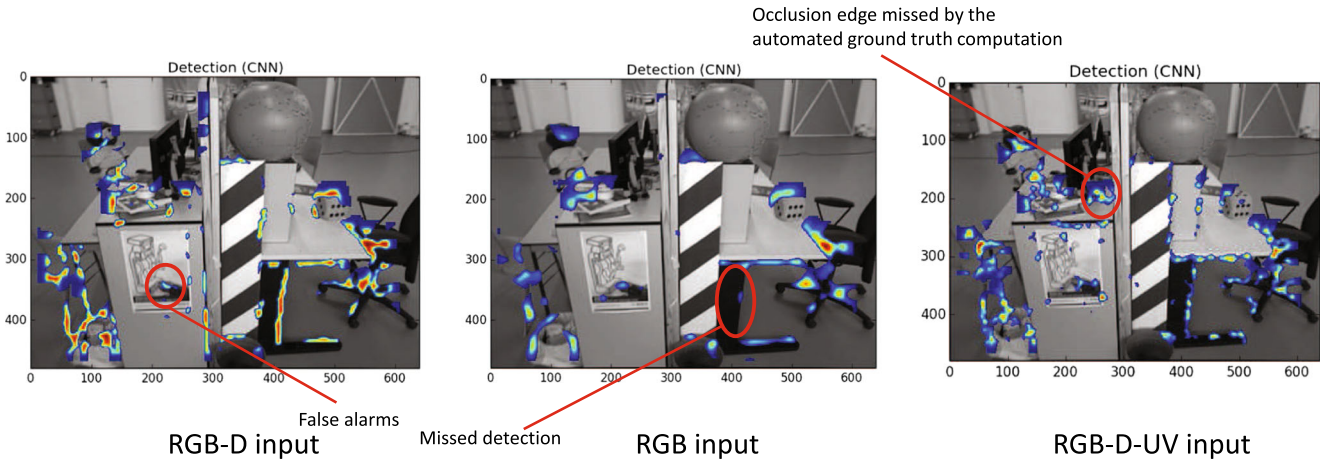

Figure 12 Occlusion edge detection performance on a test frame for all three input types with stride 4; heat map shows the fused detection confidence (red-yellow-blue signifies high-medium-low); RGB-D most sensitive, RGB the least, RGB-D-UV has the best overall performance; examples of missed detection, false alarm and true detection which was not labeled by automated ground truth determination process shown in red circled regions.

edge detection and for robotics applications as a whole. An example frame and corresponding ground truth obtained using the automated labeling process are shown in Fig. 10.

Figure 11 shows performances with RGB-D input with stride 4 and 8 (see Section 4 for details on strides) on the example testing frame. As expected, occlusion edge detection is better with a lower value of stride as more information is available per pixel in this case. It can be noted in the marked regions (circled in red) in the figures that false detection of occlusion edges reduces with a lower value of stride. The trade-off lies in computational speed. With a lower value of stride, the frame processing time increases linearly with increase in number of test patches. Therefore, this trade-off has to be chosen properly for real-time robotics applications.

Finally, Fig. 12 shows detection performance for all three input types with stride 4 and examples of missed detection, false alarm and true detection which was not labeled

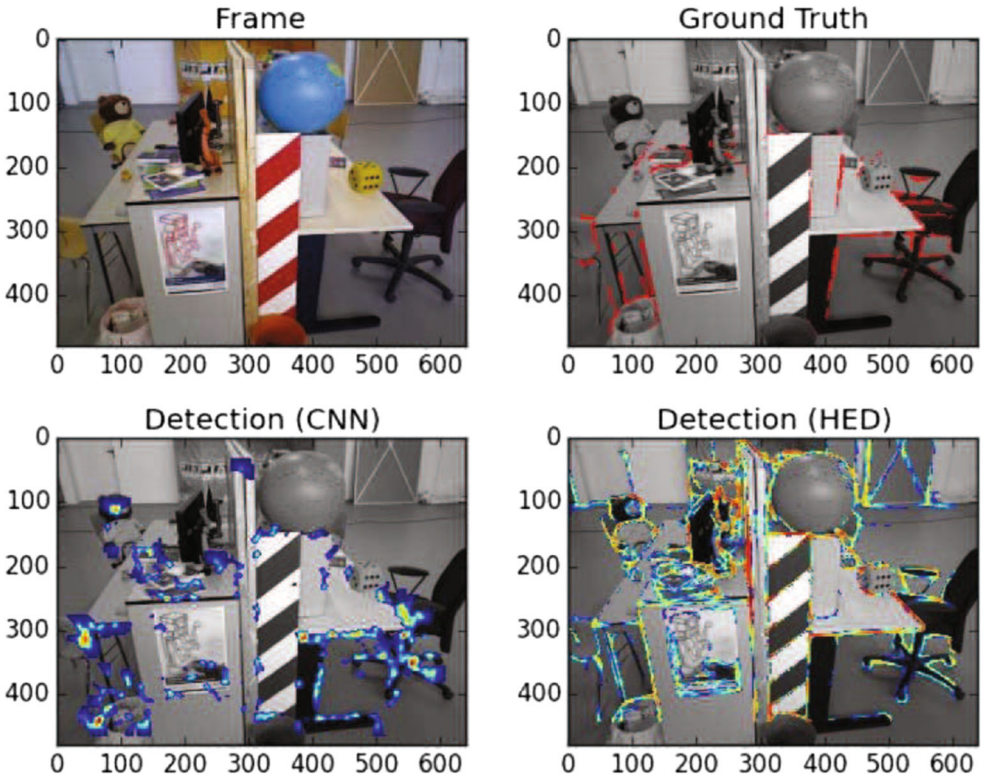

Figure 13 Occlusion edge detection performance comparison with Holistically-Nested Edge Detection (HED) technique on RGB frame.

by automated ground truth determination process are highlighted. From visual inspection, it is clear that detection confidence (shown by the heat map, red-yellow-blue signifies high-medium-low) is the highest for RGB-D input type which means the CNN model becomes most sensitive with that type of input. This corresponds to the quantitative results that shows RGB-D input type has the lowest missed detection rate but highest false alarm rate. On the other hand, RGB has the least detection confidence and RGB-D-UV has the best overall performance. The example of true detection labeled as false alarm because the ground truth was not labeled properly is significant. This shows that the CNN model is able to learn the occlusion features properly (not just memorize) and hence is able to detect certain occlusion edges that were missed by the automated ground truth determination process. This also demonstrates the need of manual labeling of occlusion edges to obtain a more accurate quantitative performance metrics. However, the large number of training and test examples becomes the barrier.

7 Comparative Evaluation

For performance comparison, we ran recently developed deep learning based Holistically-Nested Edge Detection [39] algorithm on test frames from the same TUM data set. We used the latest version of Caffe [40] and the pre-trained network from HED website. In general, the HED detector is a contour detector which prefers appearance edges that are likely to be range discontinuities and displays impressive performance on single images alone with RGB channels. Qualitative and quantitative results on one of the test frames are provided in Fig. 13 and Table 2 respectively. With visual inspection, it is evident that although the general edge detection performance is quite good for the HED method, it suffers from false alarms, i.e., identifying appearance edges as object contours due to range discontinuities. Numerical results also suggest the same as the false alarm rate of HED is significantly higher compared to our technique while missed detection performance is slightly better for HED. Overall error rate is lower for our method. Note, these performance metrics are only based on patches ($\approx$ 5000 occlusion patches and $\approx$ 120, 000 non-occlusion patches in the ground truth) obtained from the single frame presented in Fig. 13. Therefore, they should not be compared with the overall results in Table 1 that considers all patches from all 109 test frames.

Table 2 Occlusion detection performance comparison between our method and HED technique with RGB input.

Method	Overall error	False alarm	Missed detection
OccCNN (ours)	25.46 %	2.76 %	48.15 %
HED	30.78 %	16.65 %	44.90 %

8 Conclusions and Future Works

In this study, we trained deep convolutional neural networks in a supervised manner in order to detect occlusion edges in RGB-D frames. The problem is formulated as a center-pixel classification problem for an image patch extracted from a larger frame. Apart from RGB-D inputs, experiments were performed to investigate the performance associated with dropping the depth (D) channel and adding motion related information. It is noted that although the missed detection rate increases without depth data, the false alarm performance actually becomes better. Overall performance is the best with use of all six channels, RGB-D-UV. A testing and post-processing scheme is developed to visualize the testing performance. The trade-off between high resolution patch analysis and frame-level computation time is discussed which is critical for real-time robotics applications. Future research directions primarily involve adding 'closing the control loop' capabilities to this deep learning based automated feature extraction and classification tool in order to realize an efficient DDDAS. Specific tasks are: (i) investigation of robustness to change in lighting conditions, textures and domain, (ii) design of motion planning using decisions from CNNs and (ii) analysis of computation speed vs accuracy trade-off for real-time operation.

References

1. Jacobson, N., Freund, Y., & Nguyen, T.Q. (2012). An online learning approach to occlusion boundary detection. *IEEE Transactions on Image Processing*, *21*(1), 252–261.
2. Ayvaci, A., & Soatto, S. (2011). Detachable object detection with efficient model selection. In *Energy Minimization Methods in Computer Vision and Pattern Recognition* (pp. 191–204): Springer.
3. Sargin, M.E., Bertelli, L., Manjunath, B.S., & Rose, K. (2009). Probabilistic occlusion boundary detection on spatio-temporal lattices. In *2009 IEEE 12th International Conference on Computer Vision,* (pp. 560–567).
4. Marshall, J.A., Burbeck, C.A., Ariely, D., Rolland, J.P., & Martin, K.E. (1996). Occlusion edge blur: a cue to relative visual depth. *JOSA A*, *13*(4), 681–688.
5. Stein, A.N., & Hebert, M. (2009). Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *International journal of computer vision*, *82*(3), 325–357.
6. Wagemans, J., Elder, J.H., Kubovy, M., Palmer, S.E., Peterson, M.A., Singh, M., & von der Heydt, R. (2012). A century of gestalt psychology in visual perception: i. perceptual grouping and figure–ground organization. *Psychological Bulletin*, *138*(6), 1172.
7. Sundberg, P., Brox, T., Maire, M., Arbeláez, P., & Malik, J. (2011). Occlusion boundary detection and figure/ground assignment from

optical flow. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* (pp. 2233–2240).
8. Smith, P., Drummond, T., & Cipolla, R. (2004). Layered motion segmentation and depth ordering by tracking edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(4), 479–494.
9. Pathak, K., Birk, A., Vaskevicius, N., Pfingsthorn, M., Schwertfeger, S., & Poppinga, J. (2010). Online 3D SLAM by registration of large planar surface segments and closed form pose-graph relaxation. *Journal of Field Robotics: Special Issue on 3D, Mapping*, *27*(1), 52–84.
10. Gil, A., Mozos, O.M., Ballesta, M., & Reinoso, O. (2010). A comparative evaluation of interest point detectors and local descriptors for visual slam. *Machine Vision and Applications*, *21*(6), 905–920.
11. Tian, Y., Guan, T., & Wang, C. (2010). Real-time occlusion handling in augmented reality based on an object tracking approach. *Sensors*, *10*(4), 2885.
12. Fukiage, T., Oishi, T., & Ikeuchi, K. (2012). Reduction of contradictory partial occlusion in mixed reality by using characteristics of transparency perception. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality* (ISMAR), (pp.129–139). Washington, DC, USA: IEEE Computer Society.
13. Blasch, E., & Aved, A. (2015). Dynamic data-driven application system (DDDAS) for video surveillance user support. *Procedia Computer Science*, *51*, 2503–2517.
14. Uzkent, B., Hoffman, M.J., Vodacek, A., & Kerekes, J.P. (2013). Feature matching and adaptive prediction models in an object tracking DDDAS. *Procedia Computer Science*, *18*, 1939–1948.
15. Bengio, Y., & Olivier, D. (2011). On the expressive power of deep architectures, *Algorithmic Learning Theory*, Springer, Berlin/Heidelberg.
16. Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks, in *NIPS*.
17. Hinton, G.E., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, *29*(6).
18. Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.A. (2008). Extracting and composing robust features with denoising autoencoders, in *ICML*.
19. Salakhutdinov, R., & Hinton, G.E. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, *50*, 969–978.
20. Srivastava, N., & Salakhutdinov, R. (2014). Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, *15*, 2949–2980.
21. Roux, N.L., & Bengio, Y. (2008). Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, *6*, 1631–1649.
22. Hinton, G., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313.5786*, 504–507.
23. Kavukcuoglu, K., Sermanet, Y.L., Boureau, P., Gregor, K., Mathieu, M., & LeCun, Y. (2010). Learning convolutional feature hierachies for visual recognition, in *NIPS*.
24. Lore, K.G., Akintayo, A., & Sarkar, S. (2017). Llnet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition*, *61*, 650–662.
25. Mason, J., Ricco, S., & Parr, R. (2011). Textured occupancy grids for monocular localization without features. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9-13.
26. Mei, X., Ling, H., Wu, Y., & Blasch, E.P. (2013). Efficient minimum error bounded particle resampling l1 tracker with occlusion detection. *IEEE Transactions on Image Processing*, *22*, 2661–2675.
27. Ordez, F.J., & Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition, *Ed. Yun Liu et al. Sensors (Basel, Switzerland)*.
28. Giering, M., Venugopalan, V., & Reddy, K. (2015). Multi-modal sensor registration for vehicle perception via deep neural networks. In *High Performance Extreme Computing Conference (HPEC), 2015* (pp. 1-6): IEEE.
29. Chen, X., Xiang, S., Liu, C.-L., & Pan, C.-H. (2013). Vehicle detection in satellite images by parallel deep convolutional neural networks. In *Proceedings of the 2013 2nd IAPR Asian Conference on Pattern Recognition, ACPR 13* (pp. 181-185). Washington, DC, USA: IEEE Computer Society.
30. Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*.
31. Rusu, R.B., & Cousins, S. (2011). 3D is here: Point cloud library (pcl). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), (Shanghai, China)*.
32. Sipiran, I., & Bustos, B. (2011). Harris 3D: a robust extension of the harris operator for interest point detection on 3D meshes. *The Visual Computer*, *27*(11), 963–976.
33. Couprie, C., Farabet, C., Najman, L., & LeCun, Y. (2013). Indoor semantic segmentation using depth**[[33]]** information. In *ICLR*.
34. Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network, *NIPS*.
35. Yu, S.X., Gross, R., & Shi, J. (2002). Concurrent object recognition and segmentation by graph partitioning. In *NIPS*.
36. Kontschieder, P., Bulo, S.R., Criminisi, A., Kohli, P., Pelillo, M., & Bischof, H. (2012). Context-sensitive decision forests for object detection. In *NIPS*.
37. Liu, C. (2009). Beyond pixels: Exploring new representations and applications for motion analysis, Doctoral Thesis. Massachusetts Institute of Technology.
38. Boaventura, G., & Gonzaga, A. (2007). Method to evaluate the performance of edge detector. *International Conference on Intelligent Systems Design and Applications,* pp. 341–346.
39. Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. *Proceedings of the IEEE International Conference on Computer Vision*, 1395–1403.
40. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding, arXiv preprint arXiv:1408.5093.

Soumik Sarkar is an Assistant Professor of Mechanical Engineering at the Iowa State University since Fall 2014. Previously, he was with the United Technologies Research Center for 3 years as a Senior Scientist. He received his Bachelors in Mechanical Engineering from Jadavpur University, Kolkata, India in 2006, Masters in Mathematics and Mechanical Engineering from the Pennsylvania State University in 2009 and PhD in Mechanical Engineering from the Pennsylvania State University in 2011. Dr. Sarkar's research interests include Statistical Signal Processing, Machine Learning, Sensor Fusion, Fault Diagnostics & Prognostics, Distributed Control and Complexity Analysis with applications to complex Cyber-Physical Systems such as aerospace, energy & smart building systems, transportation, manufacturing and agriculture systems. He co-authored 75 peer-reviewed publications including 25 journal papers, 3 book chapters and one magazine article. He has also served as a reviewer and session chair for several technical journals and conferences. Dr. Sarkar is currently serving as an Associate Editor of Frontiers in Robotics and AI: Sensor Fusion and Machine Perception journal. He is a recipient of the prestigious 2017 Young Investigator award from the US Air Force Office of Scientific Research (AFOSR).

Vivek Venugopalan is a Senior Research Scientist with UTRC and his research focuses on reconfigurable hardware acceleration to areas such as cybersecurity, image processing, high performance computing, and machine learning. He has more than 10 years of experience in the design and implementation of algorithms for embedded systems. His recent work on hardware security titled "Unraveling the security puzzle - a distributed framework to build trust in FPGAs" won the Best Paper Award at NSS 2015. He has authored over 20 peer-reviewed journal and conference papers, has 5 patents pending and is a recipient of an Outstanding Achievement Award from UTRC. Education: Pursuing Ph.D., Computer Engineering, Virginia Tech [ABD]; M.S. Electrical Engineering, Wright State University; B.E. Electronics Engineering, University of Pune, India.

Kishore K. Reddy is a Research Scientist at United Technologies Research Center (UTRC). He is currently leading the Data Analytics (Deep Learning) initiative at UTRC. His current research interests include Deep Learning for aerospace and building systems to perform outliers and anomalies detection, multi-modal sensor fusion and data compression. Prior to working at UTRC, he has successfully applied his skills in developing advanced video and image analysis algorithms, primarily segmentation and activity classification, for multiple contracts funded by DARPA, IARPA and NIH. Dr. Reddy during his Ph.D., has led teams from UCF-CRCV lab for VIRAT-DARPA, and NIH projects. Over the past 10+ years he has worked in the areas of controls, computer vision, big data analytics, Human Machine Interaction (HMI) and machine learning (deep learning). He has co-authored more than 20 research articles. Dr. Reddy has a Ph.D. in Electrical Engineering from the University of Central Florida, M.S. in Electronic Systems and Engineering Management from Fachhochschule Südwestfalen, Germany and a bachelor's degree in Electrical and Communications Engineering from JNTU, India. He interned at Kitware Inc., NY and Haute école valaisanne, Switzerland.

Julian Ryde received a BA in Natural Sciences (2000) and MSci in Physics (2001) from the University of Cambridge, UK. He received a PhD (2008) from the department of computer science and engineering at the University of Essex, UK. After his PhD he joined the robotics group at CSIRO's Autonomous Systems Laboratory in Brisbane Australia, as a postdoctoral research fellow until 2010. He was then a Research Assistant Professor in the Computer Science and Engineering Department of the State University of New York at Buffalo. Since 2013 Dr Ryde is a Senior Scientist at United Technologies Research Center (UTRC) based in Berkeley California. He has authored over 30 peer-reviewed journal/conference publications and book chapters. He was awarded best conference paper at the international conference on Robotics and Biomimetics in 2006 (IEEE-ROBIO). His research has been funded by ACARP, CSIRO, DARPA and FHWA and his research interests include multiple mobile robot mapping and autonomous system perception, predominantly with cameras and range sensors.

Navdeep Jaitly is a Research Scientist at Google. He received his PhD from the University of Toronto under the supervision of Geoffrey Hinton. His interests lie in Deep Learning, Speech Recognition and Computational Biology. At University of Toronto he worked on deep generative and discriminative models focusing mostly on speech data. During an internship at Google in 2011, he showed that Deep Neural Networks improved the accuracy of state of the art speech recognizers built on thousands of hours of data. He has since been working on end-to-end models for sequential data, applying these techniques to speech recognition for the purpose of developing purely neural models. Prior to his PhD he was a Senior Research Scientist at the Pacific National Laboratory and at Caprion Pharmaceuticals working on algorithms for analysis of Proteomics and Mass Spectrometry data.

Michael Giering, Ph.D., is Group Leader of the Decision Support & Machine Intelligence group at United Technologies Research Center (UTRC). In this role he's led the internally funded Service Technologies Initiative since 2013, where multi-modal deep-learning methods have been developed for PHM, navigation and real-time human in the loop systems. He also heads UTRC's machine learning, human-machine interaction (HMI) and cloud analytics portfolio. Giering has more than eight years of experience at UTRC developing analytics systems such as Bayes Net building diagnostics and fleet diagnostic methods for aerospace. Previously, he was a member of the central research organization at Mars Inc. for 10 years working in the areas of machine learning, large scale data mining, currency and commodity hedging, and long-term weather forecasting. He earned an interdisciplinary Ph.D. in physics & mathematics from SIU-C an M.S. in physics from Northeastern University and Bachelors degrees in mathematics and physics from Alfred University.