ShadowCam: Real-Time Detection Of Moving Obstacles Behind A Corner For Autonomous Vehicles

Felix Naser[1], Igor Gilitschenski[1], Guy Rosman[2], Alexander Amini[1], Fredo Durand[1], Antonio Torralba[1], Gregory W. Wornell[1], William T. Freeman[1], Sertac Karaman[3], and Daniela Rus[1]

***Abstract*— Moving obstacles occluded by corners are a potential source for collisions in mobile robotics applications such as autonomous vehicles. In this paper, we address the problem of anticipating such collisions by proposing a vision-based detection algorithm for obstacles which are outside of a vehicle's direct line of sight. Our method detects shadows of obstacles hidden around corners and automatically classifies these unseen obstacles as "dynamic" or "static". We evaluate our proposed detection algorithm on real-world corners and a large variety of simulated environments to assess generalizability in different challenging surface and lighting conditions. The mean classification accuracy on simulated data is around 80% and on real-world corners approximately 70%. Additionally, we integrate our detection system on a full-scale autonomous wheelchair and demonstrate its feasibility as an additional safety mechanism through real-world experiments. We release our real-time-capable implementation of the proposed ShadowCam algorithm and the dataset containing simulated and real-world data under an open-source license.**

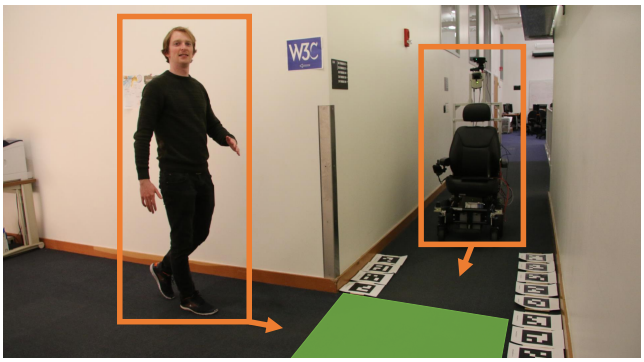


Fig. 1: The proposed ShadowCam algorithm detects moving obstacles behind a corner which cast a shadow in the marked green area. This makes driving with the autonomous wheelchair safer by avoiding dangerous potential collision situations.

I. Introduction

Safety is a key challenge and promise of future mobility solutions, specifically of autonomous cars. Advanced driver assistance systems and autonomous driving research have come a long way to make driving safer. We believe in addition to improvements to existing methods both on the hardware and the algorithmic side, we need to explore new ways of how perception, planning, and control can contribute to a safer driving future. This paper proposes a novel method for using shadows as features to avoid collisions with unseen moving obstacles.

In certain situations, human drivers and operators can perceive obstacles even when they are occluded. This, in turn, allows operators to anticipate collisions. One technique used by humans for detecting hidden dynamic obstacles is observing changes in illuminance which provides the ability to infer the approaching of a person, a vehicle around a corner, or when a car is backing out of a driveway.

A pronounced and easily observable cue is the night-time illumination change generated by car-lights, which is used by humans to anticipate an approaching vehicle. Inferring motion around corners becomes considerably more challenging during daytime or while operating a mobile robotic platform in a well-lit indoor environment. In these scenarios, shadows can be used as a cue, providing information on whether a potential obstacle behind a corner is in motion (Fig. 1). However, use of shadows in obstacle detection systems is particularly challenging as it requires motion detection given a poor signal-to-noise ratio (SNR) due to a sometimes barely visible shadow.

In this paper, we consider the problem of anticipating daytime collisions with moving obstacles that are occluded behind a corner. Our method performs a dynamic threshold on color-amplified images allowing the detection of even weakly visible shadows. It classifies an image sequence as either "dynamic" or "static", enabling the autonomous vehicle to react and avoid a potential collision by slowing down or stopping. An indoor dataset was created for the evaluation of the proposed method using different corner configurations during different times of day and motion regimes. Furthermore, a synthetically-generated dataset was created, which allows us to evaluate our algorithm under a broader variety of corners and lighting conditions, including difficult edge cases. Finally, the method was deployed on an autonomous wheelchair to validate its capabilities in challenging real-world experiments.

[1]Felix Naser, Igor Gilitschenski, Alexander Amini, Fredo Durand, Antonio Torralba, Gregory W. Wornell, William T. Freeman and Daniela Rus are with the Massachusetts Institute for Technology, Computer Science and Artificial Intelligence Lab (CSAIL), Cambridge, MA, USA. {fnaser, igilitschenski, amini, fredo, torralba, gww, billf}@mit.edu and rus@csail.mit.edu

[2]Guy Rosman is with the Toyota Research Institute, Cambridge, MA, USA. rosman@csail.mit.edu

[3]Sertac Karaman is with the Massachusetts Institute for Technology, Laboratory for Information and Decision Systems (LIDS), Cambridge, MA, USA. sertac@mit.edu

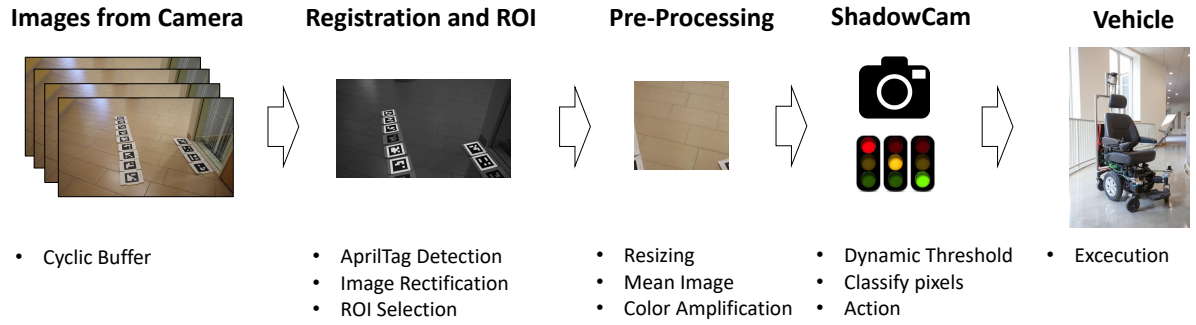


Fig. 2: Overview of the ShadowCam Algorithm: We run a cyclic buffer over the frames of the camera mounted on top of the autonomous vehicle. After the pre-processing steps we classify sequences and ensure that the vehicle avoids collisions with unseen obstacles.

Overall, the contributions of this work can be summarized as follows:

- A novel method for shadow-based motion detection of dynamic obstacles occluded behind corners.
- Extensive evaluations on synthetic data and recordings of real-world corners.
- Implementation of our algorithm as an open-source Robotic Operating System (ROS)[1] package[2], as well as publication of a comprehensive dataset of evaluation scenes[3].
- Integration of all code to a run in real-time on a full-scale autonomous wheelchair[4].

The remainder of the paper is structured as follows. We discuss the related work in Sec. II. Subsequently, we present and explain our detection algorithm in Sec. III. In Sec. IV, we go into more detail about our experimental setup and present the results of the experiments in Sec. V. The work is concluded in Sec. VI.

II. Related Work

There have been several works on perception for mobile robotics in non-line-of-sight (NLoS) scenarios. Most research in that context considers ultra-wideband (UWB) systems for localization, as e.g. in [21], and ranges up to using WiFi signals for NLoS perception [1]. A first approach for explicitly seeing around corners for mobile robotics was presented in [25] using a drone which can be launched from a car as an additional source of information. The present work uses a vision-based approach and does not require hardware infrastructure, assumptions about the occluding material, or deployment of drones.

Handling object occlusion: Consideration of occlusion for intelligent transportation systems mostly focused on improved tracking by improving detectors of other vehicles [16], [7] and pedestrians [18] while assuming partial visibility or a merely a temporary occlusion. In [5], explicit modelling of occlusions was also used for broader scene understanding. In contrast to these approaches, we do not assume even partial visibility of the potential obstacle but use, when available, its shadow instead.

Shadow Processing: Shadow processing typically focuses on its removal [11], [20], [6]. For mobile robotics, this is particularly relevant for improving visual localization, because it enables generating a more weather invariant representation of the environment [4], [14]. In contrast to these works, we explicitly use shadows as cues in our system. While in [13] shadows are also used in motion detection, that work assumes a different scenario involving a static camera and also considers visibility of the tracked object.

Hidden Scene Recovery: Computer vision approaches which infer about hidden scenery usually rely on Time-of-flight cameras [19], [22], [23], [12], [10], [8] which are prone to interference from other unpredictable lighting sources and therefore mostly rely on carefully controlled environments. It was recently shown that lighting from behind the corner and the created faint penumbra on the ground can be used for creation of a 1D video [2] from a static camera. Drawing inspiration from this work, our proposed approach considers shadows and uses them for motion detection from a moving platform. This is in contrast to most perception systems which explicitly consider shadows so far, since they mostly focus on its removal.

III. Approach

It is hard to come up with a general solution for the described NLoS problem. At some corners moving objects are physically not able to cast a shadow and when they do the shadow is mostly a low SNR signal that highly depends on various factors — these include size of the object, speed of the movement, lighting, reflection properties of the floor, color of the floor, ego motion and speed, among others. Despite these difficulties (as shown in [2]) it is possible to create a signal of a moving obstacle behind a corner from a static camera. But to actually make use of the signal in a practical way, e.g. as a safety feature for autonomous vehicles, it needs to work on a moving platform. This requirement adds even more noise to the system and makes it harder to achieve high detection accuracy.

[1] `http://www.ros.org/`

[2] Open-source ROS package ShadowCam before acceptance here `https://goo.gl/RpTT1m` upon acceptance here `https://github.mit.edu/fnaser/shadow_cam`.

[3] The dataset (both synthetic and real-world) is available here `https://goo.gl/k5ixRd`.

[4] Video of autonomous wheelchair with ShadowCam `https://youtu.be/rHqwBFEslu4`

We are providing a motion detection algorithm based on shadows as features from a moving vehicle. We look at corners where moving objects are physically able to cast a shadow. Since we want to focus our efforts on this new idea, we assume we can rely on an image registration method and know the Region of Interest (ROI) of each frame. The ROI could be determined using the map that the autonomous wheelchair uses to localize itself, other place recognition algorithms [24], or a deep-learning-based detector, but determining the ROI is not the focus of this paper. Instead we use AprilTags [17], [26] to provide extra features for sequence stabilization and for cropping the ROI where we expect to see a shadow. We take the furthest tag and crop a dynamic rectangle using the size of that tag, which results in a rectangle that is larger the closer we are to the corner.

Fig. 2 shows how we embedded the ShadowCam in the perception, planning and control cycle of the autonomous vehicle, which is in our case a wheelchair. First, we observe the corner and then we provide the frames as inputs for the ShadowCam, which outputs a decision on whether or not it is safe to continue along the path.

Algorithms 1 and 2 sketch out how we implemented the ShadowCam. For the actual C++ code we refer to the open-source ROS package. Algorithm 1 first gives an overview of the main loop. The classification procedure, Algorithm 2, shows how we determine "dynamic" or "static" which is our core algorithmic contribution.

Algorithm 1 ShadowCam Algorithm

```
list ← global var
while true do
    f ← getFrame()
    d ← getAprilTagDetections(f)
    if checkDetections(d) then
        s ← createSequence(f, list)
        c ← classifySequence(s)
        vehicleInterface(c)
```

Our system uses a cyclic frame buffer approach to achieve real-time performance. This means we can output a detection result whenever a new image fulfills the requirements to get appended to the sequence, e.g. we require a maximum number of detected AprilTags above a defined quality level.

Once a new image gets appended to the sequence we identify corresponding tags to compute a homography h for each frame to get projected to the viewpoint of the first frame in the sequence. In other words we apply image registration to all frames i in the current sequence j according to

$$f_{j,i}(x, y) = f'_{j,i}(h(x), h(y)) \ . \tag{1}$$

After the image registration step we crop according to the tags the ROI. To reduce noise, we down-sample each cropped and registered image using bilinear interpolation to a 100×100 patch,

$$f_{j,i} = \text{resize}(f_{j,i}, (w, h)) \ . \tag{2}$$

Algorithm 2 Classify Sequence

```
procedure CLASSIFYSEQUENCE(S)
    f̄ ← mean(S)
    c ← 0
    for all f ∈ S do
        f ← colorAmplification(f̄, f)
        f ← temporalFilter(f)
        f ← dynamicThreshold(f)
        f ← morphologicalFilter(f)
        sum ← sum + sumPixels(f)
    if sum >= camThreshold then:
        c ← 1
    return c.
```

Then we compute the mean image over all down-sampled images, i.e.,

$$\bar{f}_j = \frac{1}{n} \sum_{i=1}^{n} f_{j,i} \ . \tag{3}$$

We subtract the mean image from each frame in the current sequence and apply a Gaussian blur before we amplify the difference to the mean. That is, we compute

$$d_{j,i} = |G\big((f_{j,i} - \bar{f}_j), k, \sigma\big)| \cdot \alpha \tag{4}$$

where G is a linear blur filter of size k using isotropic Gaussian kernels with covariance matrix diag(σ^2, σ^2). We chose σ depending on k according to $\sigma = 0.3 \cdot ((k-1) \cdot 0.5 - 1) + 0.8$ as in [3]. We call α the amplification parameter since it amplifies the difference to the mean image. (Based on empirical observations, k has been set to 3 and α has been set to 5 for all experiments.) This process serves as color amplification and helps to improve the detectability of a shadow (sometimes even if the original signal is invisible to the human eye). In other words, this process increases the signal-to-noise ratio. Fig. 3 depicts an example of an image before and after the color-amplification process. After the frame is color amplified we run a temporal low-pass filter

$$t_{j,i} = d_{j,i} \cdot t + d_{j,i-1} \cdot (1 - t) \tag{5}$$

where $t_{j,i}$ is the filtered result of the difference images $d_{j,i}$. To become more robust against different corners and light conditions, we take inspiration from [9] and apply a "dynamic" threshold. We take the difference from the mean of each channel of the filtered frame as a criterion to determine motion with respect to the standard deviation of the filtered image,

$$c_{j,i} = \begin{cases} 0, & \forall |t_{j,i} - \bar{t}_{j,i}| \leq w \cdot \sigma(t_{j,i}) \\ 1, & \forall |t_{j,i} - \bar{t}_{j,i}| > w \cdot \sigma(t_{j,i}) \end{cases} \tag{6}$$

where w is a tune-able parameter that depends on the noise distribution. We set $w = 2$ for all our experiments. The underlying assumption here is that dynamic pixels are further away from the mean, since they change more drastically. A combination of dilation and erosion is used to first connect pixels which got classified as motion and then erosion is

used to reduce noise. We are applying morphological ellipse elements with two different kernel sizes [3], i.e.,

$$c_{j,i} = \text{dilate}(c_{j,i}, 1) \ , \qquad c_{j,i} = \text{erode}(c_{j,i}, 3) \ .$$

At the end, we sum up all pixels under the intuitive assumption that more movement in between frames will result in a higher sum

$$s_j = \sum_{i=1}^{n} c_{j,i}(x, y) \ . \tag{7}$$

To classify the whole sequence as either "dynamic" or "static" we then apply a camera-specific threshold. We show in Sec. V how the threshold can be determined. The data appears to prove what sounds intuitively correct: A less noisy image results in fewer mis-qualified pixels which results in a lower threshold. This implies that a better camera (frame rate and resolution) and a better image registration quality lead to a smaller threshold.

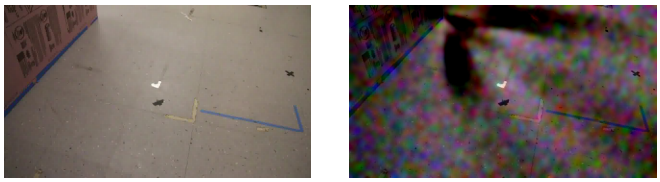


Fig. 3: Color Amplification: On the left side is an original Canon camera frame and on the right side the corresponding color-amplified frame.

Fig. 4: Ego-view: The autonomous wheelchair approaches a corner without direct sight of what is going on behind the corner.

Fig. 5: Left: The cropped, re-sized and registered image. Middle: Example of a frame which contains no moving obstacle behind the corner. Right: Example of a frame which contains movement. The white areas correspond to a shadow signal.

On the vehicle interface side, we run a temporal filter on the detection results to further smooth the signal. Once the ShadowCam detects an obstacle behind the corner the autonomous vehicle stops until it is safe to continue.

IV. Experimental Setup

To evaluate our algorithm in different scenarios and analyze its performance statistically, we collected data in a motion-capture room (referred to as the Holodeck), created synthetic corners with different properties in a Blender simulation, and collected real-world data using different cameras. We composed the entire dataset to cover a broad range of the mentioned nuisance factors, such as size of the object, speed of the movement, lighting, reflection properties of the floor, and color of the floor.

The experiments in the Holodeck and in simulation allow us to analyze the spatial performance of the algorithm, since we know the exact position of the moving obstacle behind the corner. Additionally, we can label each sequence based on the ground truth. In real-world experiments, we label whole videos as "dynamic" or "static" depending on whether or not a person was moving behind the corner. This leads to potentially mislabeled sequences in the videos, since the person behind a corner does not move at all times.

Holodeck: The controlled test setup in the Motion Capture environment enables labeling of each sequence automatically. There, we collected data with a stationary Canon EOS 70D camera using a EFS $17 - 58$ mm lens. The frame-rate was set to 30 fps, using codec H.264 at a resolution of 1920×1080. The motion capture system tracks the moving object behind the corner at around 100 Hz. We synchronize the video stream with the motion-capture data and label a sequence as "dynamic" when more than half of the frames contain a moving person behind the corner, otherwise it is labeled as "static".

In the Holodeck we were also able to vary a few scene parameters, such as the size of the person behind the corner, the light conditions produced by switching on and off different ceiling lights and the material of the ground on which the ShadowCam tries to detect shadows. Intuitively, we would expect a better signal when the moving object is closer to the corner, which is confirmed by our spatial analysis of the signal in the Holodeck test setup shown in Fig. 7.

Simulation in Blender: We created a synthetic dataset with Blender[5], to test the algorithm under a greater variety of lighting conditions, textures, person sizes, and material properties. With a Python script we access Blender's API to change the scene parameters dynamically and get ground-truth data for image registration and object motion. We use Blender's "Cycles Renderer" with only 10 samples to create one frame of size 960×540. This results in more noisy images making it harder for the algorithm to detect a shadow and thus providing more realistic data. The chosen area light casts shadows with soft edges based on ray tracing.

We change the textures of the scenes randomly. Floor and walls are changed independently which results in more combinations. We chose from 30 texture images (Fig. 8) to

[5]https://www.blender.org/

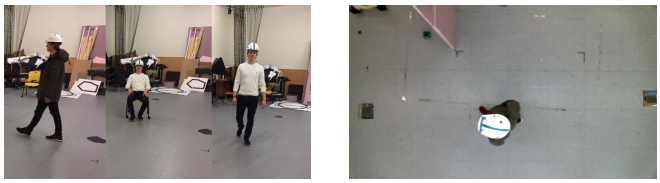


Fig. 6: The left image shows how we varied the size of the moving object behind the corner: Wearing a winter jacket, sitting on a chair with rollers and walking normally. The right image shows the corner wall and a person walking from the top view.

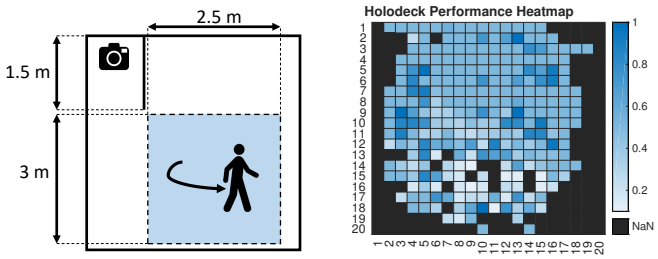


Fig. 7: The left plot shows the region in which the person behind the corner walked randomly from the top-view. The right plot shows the spatial performance of the algorithm (with regard to the mean position of the moving person in the blue area) to correctly classify a sequence as "dynamic" or "static". The darker the blue, the higher the classification accuracy for both classes.

create 30 different corners. We render for each corner $1,000$ frames for both classes ("dynamic" and "static") which sums up to $50,000$ synthetic images and around 6.2 GB.

In addition to the change of the texture, we change the material properties such as surface quality (e.g. roughness) and reflection strength (e.g. mirror or carpet) randomly. The position and color of the light, the path of the moving platform with the camera, and the scale and texture of the person behind the corner are randomly modified within certain boundaries. E.g. the height of the occluded person ranges from 0.75m to 2.2m (see Fig. 9 and Fig. 10). For both camera and person the speed of motion changes independently for each corner randomly within the range $1-3$ m/s.

Fig. 8: Examples of the 30 textures we use to create different scenes with Blender. We chose various textures ranging from dark to bright colors.

Real-World Corners: Besides the controlled environments in the Holodeck and Blender we evaluated how the algorithm performs "in the wild". We created a real-world dataset with 3 different cameras and 15 corners. In total, we collected $85,000$ images of around 1 hour of data and 7.4 GB. To not only cover different types of corners, but also different image qualities, we chose the following three cameras:

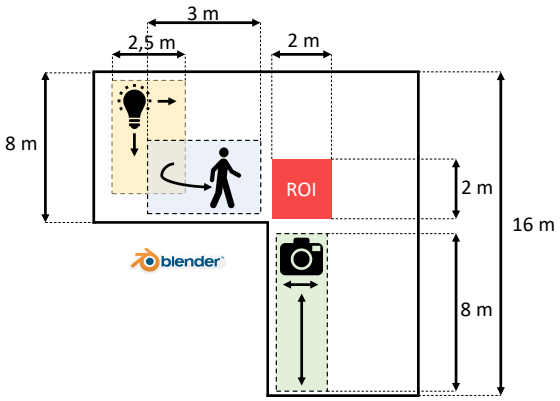


Fig. 9: Blender Scene: In addition to material changes (such as texture or reflection properties) we change the position of the light and the path of the camera. The person walks randomly at different speeds within the blue box.

- **Canon** EOS 70D and the EFS $17-58$ mm lens (single-lens reflect (SLR) camera): The frame-rate was set to 30 fps, with codec H.264 and image dimensions 1920×1080.
- **Webcam** Logitech HD Webcam C525 (low-end webcam): The frame rate was set to 24 fps, with codec VP8 and image dimensions 1280×720.
- **Webcam** Logitech HD Webcam C925-e (high-end webcam): The frame rate was set to 20 fps, with codec VP8 and image dimensions 1280×720.

AprilTags were used in order to focus on the detection problem. We placed 13 AprilTags on the same plane at which we expect to detect a shadow. In theory one tag would be sufficient, but most of the time only a subset of the AprilTags gets accurately detected and by adding more tags we increase numerical stability. AprilTag is a visual fiducial system. It's targets can be created from an ordinary printer, and the AprilTag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera. Real-time capable implementations are available[6].

We chose mainly corners where it is physically possible for a moving object behind a corner to cast a shadow. This implies for these corners that humans, if they pay close attention, might be able to see a shadow of an approaching person on the ground. We collected data ranging from high reflection floors and stone to dark carpet (see Fig. 11). In all real-world videos, we label each video as a whole as "dynamic" or "static" depending on whether or not a person was asked to walk behind the corners. The camera is moving in a range of 1 to 3 meters back and forth, whereas the person behind the corner moves randomly in a similar range.

Autonomous Vehicle: The algorithm was also tested on an autonomous wheelchair (see Fig. 12) which has been designed as an indoor counterpart to the autonomous car presented in [15]. Therefore, the wheelchair has a very similar sensor configuration and a software stack based on ROS. This enables us to run (besides vehicle specific

[6] https://april.eecs.umich.edu/software/apriltag/

Fig. 10: Simulation example corners.

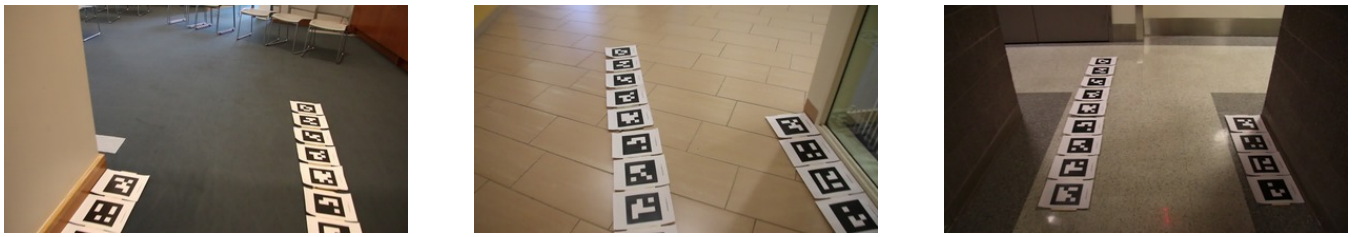


Fig. 11: Real-world example corners.

software parts, such as the low-level control) the same software packages on different vehicle types. This setup enables easily deploying a similar functionality to the real car. For the experiments, we added a Logitech HD Webcam C925-e on top of the Wheelchair's top laser scanner to increase the look-ahead distance and improve the angle at which the camera perceives the tags on the ground. Once the ShadowCam detects movement behind the corner we adjusted the control algorithm of the wheelchair so that it stops as long as movement is detected. As soon as the way is clear again the wheelchair resumes the forward motion.

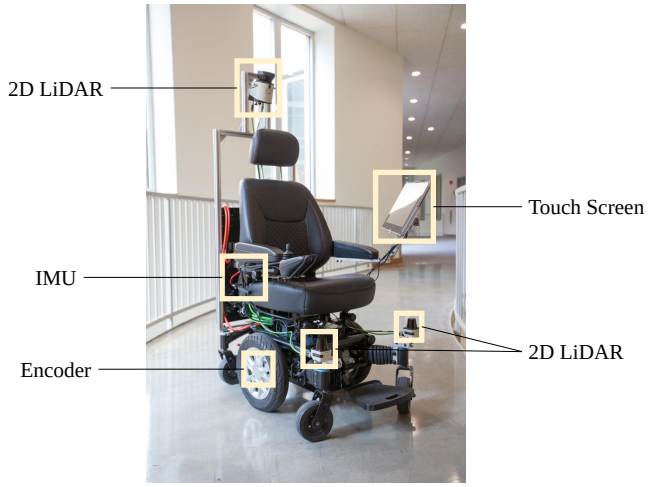


Fig. 12: Autonomous wheelchair and the main sensors. We mounted the webcam for the ShadowCam on top of the top LiDAR. Only the webcam is required for the ShadowCam algorithm.

V. Results

We quantitatively analyze the classification accuracy, real-time capability of the algorithm and demonstrate the ShadowCam integrated into an autonomous wheelchair. Our results show that it is possible to detect moving obstacles based on shadows out of the line of sight from a moving platform at indoor corners given that the object can physically cast a shadow, we have access to reliable image registration, and the region of interest is known a priori (e.g. through a specific detector or a map). Our algorithm is easy to deploy (it has only a few tune-able parameters) and generalizes to different corner settings. Importantly, we analyzed performance on different floors, light conditions and object sizes.

Histograms over sequences are used to visualize the performance of the algorithm on the respective datasets in Fig. 15. The ShadowCam algorithm computes one value for each sequence which represents the sum over all "dynamic"-classified pixels. The better the distributions of this value (for the cases with and without a moving obstacle) can be separated, the higher the classification accuracy can be when the threshold is set to the optimal point. For example in a static sequence with 1% as dynamic misclassified pixels, the method would yield a value of $255,000$ (100 px $\times$ 100 px $\times$ 10 images $\times$ 1% $\times$ 255 pixel value).

In order to determine the threshold value upon which we classify a sequence as "dynamic" or "static", we examine the histograms over all corners of a specific recording setup (e.g. simulation or real-world) and found the noise of the camera to be correlated with the choice of the threshold. This conforms to the intuition that higher noise levels lead to higher misclassification rates. Table I gives an overview of the final thresholds we chose to create the mean classification accuracy plots in Fig. 13 and Fig. 14.

For evaluation on the wheelchair, we implemented the algorithm in C++, using the AprilTag 2 detector and a cyclic frame buffer. This enabled us to compute a classification output at 30 Hz for a sequence of 10 frames. But since the camera we use on the wheelchair only runs at 20 Hz and only in around 1/3 of the images enough AprilTags get detected, the rate on the real system is around 7 Hz. The low rate of AprilTag detection is mostly due to motion blur. However, for the speed of the wheelchair 7 Hz is fast enough,

TABLE I: Dataset Overview

Camera Type	Threshold	Percent of Pixel	Number of Corners
Holodeck	200000	$\approx 1\%$	1
Blender	220000	$\approx 1\%$	30
Canon	500000	$\approx 2\%$	11
Webcam	650000	$\approx 2.5\%$	3

and the performance could be easily increased by switching to a better camera with a higher frame rate and/or image registration method. Our experiments with the autonomous wheelchair show that even with consumer grade cameras (such as the Logitech Webcam) the signal can be detected reliably.

Because our system is designed as an additional safety feature, we aim for a low rate of false "dynamic" classifications to provide a driving experience (Fig. 13) without unnecessary interruptions. That is, when the algorithm detects movement then it is very likely someone is actually moving behind the corner. Thus, the majority of sequences gets classified as "static" even if they sometimes contain a moving obstacle. It requires a strong movement behind the corner for the sequence to get classified as "dynamic". Currently the classification accuracy is based on videos which got labeled as a whole. We are expecting better results with the same algorithm if the labels were more accurate, since "dynamic" labeled videos contain also "static" sequences.

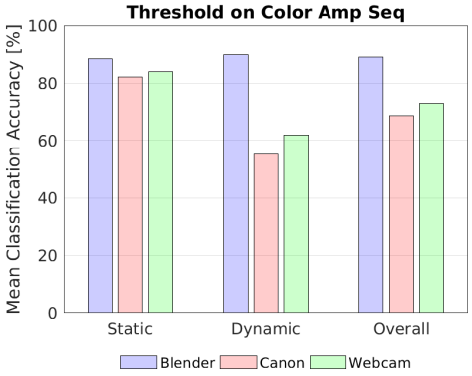


Fig. 13: As the distributions of the histogram suggest, we observe a low number of false positives. The classification accuracy for "static" sequences is high, whereas it is harder to detect movement based on shadows. Overall is the mean performance of the algorithm on both real-world data acquisition methods around 70%.

VI. Discussions And Conclusions

From a high-level point of view we look at a low SNR signal and reduced the SNR even further by adding camera movement. By making two assumptions (given image registration and ROI) we reduce our problem to the core research problem of motion detection based on shadows. We developed a real-time capable motion detection algorithm which is robust against noise, but still able to detect a low SNR signal. To the best of our knowledge we present the first autonomous wheelchair which is able to detect and react

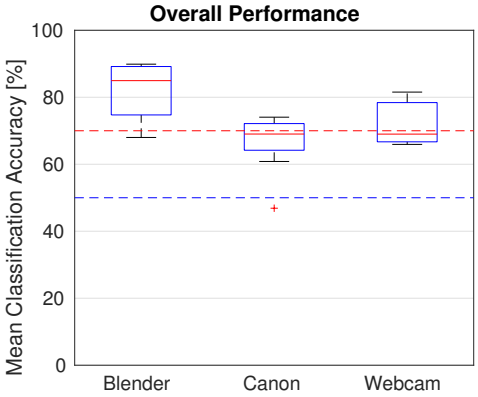


Fig. 14: Looking more closely into the overall performance for both classes "dynamic" and "static", we can observe our algorithm performs apart from one outlier corner with a very weak shadow signal ($(P(s)+P(d))/2 \approx (100\%+0\%)/2 = 50\%$) strictly better than random. The outlier is caused by a corner where the shadow isn't casted within the ROI.

to out of the line of sight moving obstacles based on their shadows.

We were able to show that a shadow cast by a moving obstacle out of the line of sight shouldn't only be treated as unwanted noise in an image but can actually provide safety relevant features. We hope this will inspire others to treat shadows more like an additional signal than as unwanted noise. The new dataset can be used for further research and in classes related to signal processing and computer vision.

We are excited to further explore new application areas for our motion detection algorithm and ways to improve the accuracy. We are also planning to collect more data and test the algorithm at higher speeds with a camera at higher frame rates. This will pave the way to bring the ShadowCam to autonomous cars.

Acknowledgments

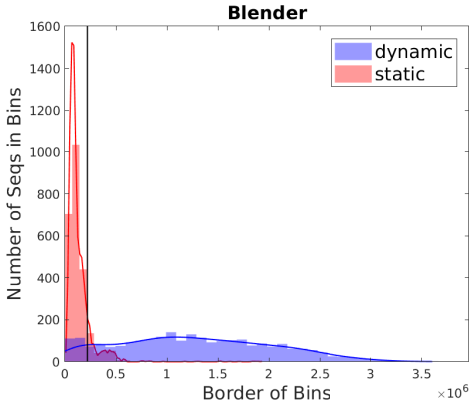
Toyota Research Institute (TRI) provided funds to assist the authors with their research, but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We acknowledge the generous research support from Amazon. We want to thank our colleagues: Christina Liao for her help with the dataset, plots and videos, Prafull Sharma for the help with simulation, Steve Proulx for the camera mount, Thomas Balch for the help with the wheelchair, Vickie Ye, Adam Yedidia, and Manel Baradad for discussions and feedback.

References

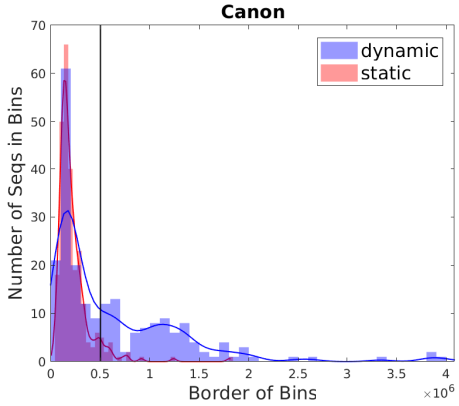
[1] F. Adib and D. Katabi. See Through Walls with WiFi! In *Proceedings of the ACM SIGCOMM Conference*, 2013.

[2] K. L. Bouman, V. Ye, A. B. Yedidia, F. Durand, G. W. Wornell, A. Torralba, and W. T. Freeman. Turning Corners Into Cameras: Principles and Methods. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

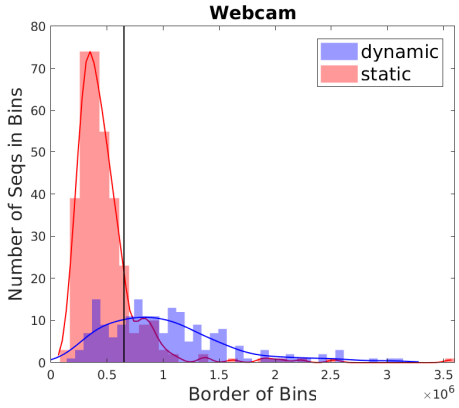
[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.



(a) All 30 simulated Blender corners combined in one histogram.



(b) One example corner where we collected the data with the Canon.



(c) One example corner where we collected the data with a webcam.

Fig. 15: Histograms of detector values for different examples. (a) Since the distributions of "dynamic" and "static" sequences are quite distinct the mean classification accuracy of around 80% is expected when the threshold is set to 220.000 as the black vertical line indicates.

[4] P. Corke, R. Paul, W. Churchill, and P. Newman. Dealing with Shadows: Capturing Intrinsic Scene Appearance for Image-Based Outdoor Localisation. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[5] V. Dhiman, Q.-H. Tran, J. J. Corso, and M. Chandraker. A Continuous Occlusion Model for Road Scene Understanding. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] G. D. Finlayson, M. S. Drew, and C. Lu. Entropy Minimization for Shadow Removal. *International Journal of Computer Vision*, 85(1):35–57, 2009.

[7] T. Frank, M. Haag, H. Kollnig, and H.-H. Nagel. Tracking of Occluded Vehicles in Traffic Scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 1996.

[8] G. Gariepy, F. Tonolini, R. Henderson, J. Leach, and D. Faccio. Detection and Tracking of Moving Objects Hidden from View. *Nature Photonics*, 10(1):23–26, 2016.

[9] W. Jing, D. Xin, Z. Yun-fang, and G. Wei-kang. Adaptive Fuzzy Filter Algorithm for Real-Time Video Denoising. In *Proceedings of the International Conference on Signal Processing (ICSP)*, 2008.

[10] A. Kadambi, H. Zhao, B. Shi, and R. Raskar. Occluded Imaging with Time-of-Flight Sensors. *Transactions on Graphics (ToG)*, 35(2):15, 2016.

[11] S. H. Khan, M. Bennamoun, F. Sohel, and R. Togneri. Automatic Shadow Detection and Removal from a Single Image. *Transactions on Pattern Analysis and Machine Intelligence*, 38(3):431–446, 2016.

[12] M. Laurenzis, A. Velten, and J. Klein. Dual-mode Optical Sensing: Three-Dimensional Imaging and Seeing Around a Corner. *Optical Engineering*, 56(3), 2017.

[13] A. Leone and C. Distante. Shadow Detection for Moving Objects based on Texture Analysis. *Pattern Recognition*, 40(4):1222–1233, 2007.

[14] W. Maddern, A. D. Stewart, and P. Newman. LAPS-II: 6-DoF day and night visual localisation with prior 3D structure for autonomous road vehicles. In *Proceedings of the Intelligent Vehicles Symposium (IV)*, 2014.

[15] F. Naser, D. Dorhout, S. Proulx, S. D. Pendleton, H. Andersen, W. Schwarting, L. Paull, J. Alonso-Mora, M. H. Ang, S. Karaman, R. Tedrake, J. Leonard, and D. Rus. A Parallel Autonomy Research Platform. In *Proceedings of the Intelligent Vehicles Symposium (IV)*, 2017.

[16] E. Ohn-Bar and M. M. Trivedi. Learning to Detect Vehicles by Clustering Appearance Patterns. *Transactions on Intelligent Transportation Systems*, 16(5):2511–2521, 2015.

[17] E. Olson. AprilTag: A Robust and Flexible Visual Fiducial System. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2011.

[18] W. Ouyang and X. Wang. A Discriminative Deep Model for Pedestrian Detection with Occlusion Handling. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[19] R. Pandharkar, A. Velten, A. Bardagjy, E. Lawson, M. Bawendi, and R. Raskar. Estimating motion and size of moving non-line-of-sight objects in cluttered environments. In *CVPR*, pages 265–272. IEEE Computer Society, 2011.

[20] R. Ramakrishnan, J. Nieto, and S. Scheding. Shadow compensation for outdoor perception. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015.

[21] C. K. Seow and S. Y. Tan. Non-Line-of-Sight Localization in Multipath Environments. *Transactions on Mobile Computing*, 7(5):647–660, 2008.

[22] D. Shin, A. Kirmani, V. K. Goyal, and J. H. Shapiro. Photon-Efficient Computational 3-D and Reflectivity Imaging with Single-Photon Detectors. *Transactions on Computational Imaging*, 1(2):112–125, 2015.

[23] D. Shin, F. Xu, D. Venkatraman, R. Lussana, F. Villa, F. Zappa, V. K. Goyal, F. N. Wong, and J. H. Shapiro. Photon-efficient imaging with a single-photon camera. *Nature communications*, 7, 2016.

[24] K. Van De Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[25] A. Wallar, B. Araki, R. Chang, J. Alonso-Mora, and D. Rus. Foresight: Remote Sensing for Autonomous Vehicles Using a Small Unmanned Aerial Vehicle. In *Proceedings of the Conference on Field and Service Robotics (FSR)*, 2018.

[26] J. Wang and E. Olson. AprilTag 2: Efficient and Robust Fiducial Detection. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2016.