Automatic Dataset Generation for Learning Physics from Occlusion Videos

Darpan Tushar Sanghavi Sri Sudhamsu Krishna Manne
University of Massachusetts Amherst
{dsanghavi,smanne}@cs.umass.edu

Abstract

An important aspect distinguishing machine vision from human vision is that we have pre-attentive vision. We know what to expect, and with practice, to the extent that we don't even need to think about what we're doing. This corresponds to developing intuitions of physics, which motivates us to mimic this learning using neural networks, and investigate the correlation of the hidden units with physical quantities such as depth, color, velocity, acceleration, etc. In the absence of any adequate dataset that meets our needs, we set out to make one on our own. We discuss our process to automate data generation (from raw videos) from which we expect to learn real-world physics, primarily depth cues due to motion and occlusions of objects.

1. Introduction and Motivation

One of the first milestones every one of us learns within the first few months of life is the ability to track objects. During the first 4-8 months [2], infants develops depth perception, color vision, and intuitions of physics critical to eye-body coordination. We take inspiration from an imagery of a baby moving its head around in a room full of objects, leading to several occlusions of objects by other objects. A baby's brain is able to infer depth from the scene, and even begin to expect which object would next occlude another object.

We model this scenario in our data by videographing stationary objects with a moving camera (like a moving head) and capturing occlusions between the objects. The larger aim is to enable training deep networks to understand physics rules. We model a surrogate learning task - predicting which object out of given two objects in a video is the occluding object and which gets occluded by the other - with the expectation that the hidden units in the network will learn physics rules. We expect the network to learn, primarily, depth cues like relative size, relative velocity, angle of motion, position relative to ground plane, and color gradients.

Due to the lack of suitable datasets for this task, we developed a fully-automatic and modular system to generate a dataset of mini-clips capturing occlusion events from raw videos. We also generate corresponding labels for the occlusion frame number and also the tracking boxes for the occluded and occluding objects. This system provides us a framework to generate virtually unlimited samples from web-mined videos. It is possible to extend it to specific domains by tuning the individual modules.

2. Related Work

One key task in our data creation process is detection of occlusions. While occlusion detection has been dealt with before several researchers, the methods have usually involved stereo images and depth perception as means to occlusion detection or prediction [9] [16] [10]. Our scenario is different, as we don't have stereo images to infer depth. Given a video from a single camera, and tracking information of an object or part of an object, we need to detect when the entity being tracked is occluded by another object.

A related area of research is Action Recognition from videos. We can take inspiration from Temporal Segment Networks [14] that employ temporal structure modeling, multi-resolution CNNs [6], and 2-stream spatio-temporal architectures [12]. Datasets used for those networks like UCF101 [13], HMDB [7] involve human motion videos only. The Sports-1M [6] dataset is too noisy for our purpose. Video Prediction networks like [4] would also inherently learn physics rules, but have been designed for "imaginative" vision in robotics - to predict the effect of a robotic arm push.

3. Dataset Requirements

There are two aspects to analyzing the dataset requirements - the final dataset and the initial raw videos from which the dataset will be generated. First we discuss the final dataset we want in order to get us a better idea of what kind of videos we need to collect.

Submitted as Final Report for CS696 Independent Study. University of Massachusetts Amherst. Spring 2017.

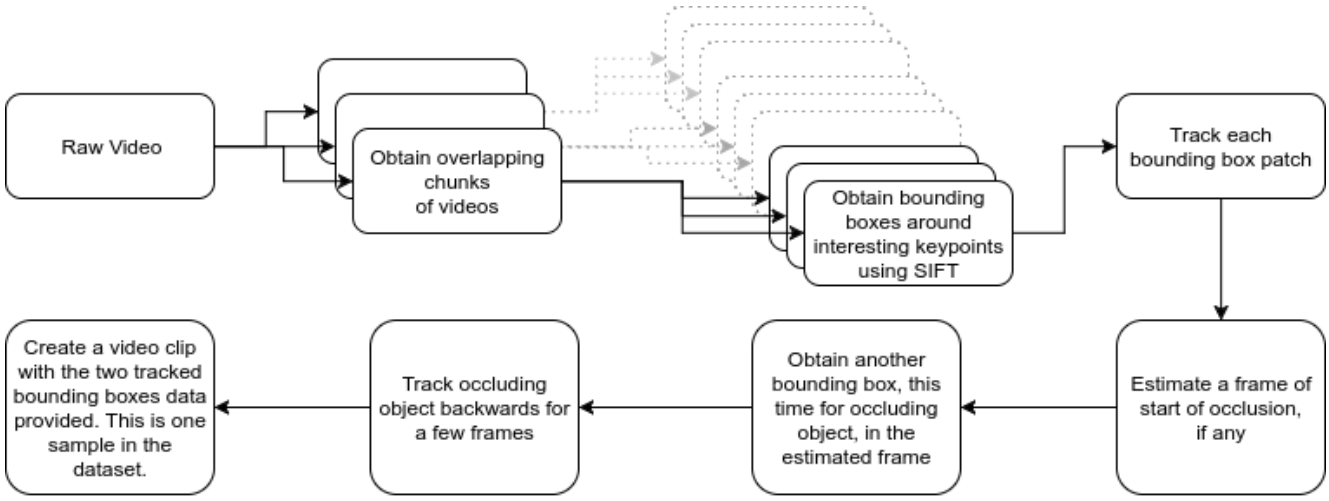


Figure 1: Dataset Generation Pipeline

3.1. Final Dataset

Each training example in the dataset is a mini-clip of two objects moving toward each other. We want our neural net to learn physics in order to predict which of the two objects is going to occlude the other. We want each mini-clip to be approximately 100 frames or more (about 3-4 seconds) capturing the movements just before occlusions. Corresponding to each training example, we also want a label indicating the frame number in which the occlusion occurs. Additionally, if we choose to have a more complex dataset (where an occlusion might happen or not), we also want a label indicating whether an occlusion happens or not.

To the best of our knowledge, this is the first work of its kind and so we want to first develop a dataset in a very restricted domain to keep the learning task easy (like what MNIST is to ImageNet). The restrictions imposed (as opposed to a dataset generated from web-mined videos) for the simplest dataset creation are:

- Guaranteed occlusion in each training example.
- No moving objects, resulting in occlusions caused only by camera motion.
- A limited number of objects in the raw videos.
- Plain backgrounds to minimize noise. However, it is important to have variety in backgrounds to avoid overfitting.
- Good resolution and smooth camera movements to reduce blur.
- High depth variance in order to capture the effect of depth on perceived velocity. The objects should not be far (low variance), but at an approximate distance of within 1-2 meters from the camera.

We could have grayscale videos, but chose to include color information to enable learning color gradients for perceiving depth.

3.2. Raw Video Data

In order to generate the mini-clips described above, we need clean raw videos of occlusions among objects. We aim to generate candidate bounding boxes, track them and predict if it gets occluded, and at which frame it starts to get occluded (f_{occ}), or if the tracker failed. Choosing the candidates resulting in occlusion, we generate a second bounding box on the occluding object, and back-track it to generate the mini-clip.

To be able to generate clean mini-clips, our videos need to have the following characteristics.

1. Unique objects that can be tracked easily. The generated patch for tracking need not cover to the whole object, but should be a distinctly trackable part on the object edge.
2. Occlusions can be due to 2 reasons: (a) Motion of an object in the video and/or (b) Motion of the camera. All videos can be classified into only object motion (a), only camera motion (b) or both (a+b). For example, the baby moving its head to see occluding objects is of type camera movement only (b). For the simple dataset, we choose the stationary object moving camera model (b only).
3. Duration - We want our mini-clips to be around 1 second, so we need videos lasting only a few seconds, maybe upto a minute if there are continuous occlusions. In the case of long videos, we divide the file into multiple overlapping chunks.
4. Smooth and stable movement - We should prefer slower moving camera/objects at a higher frame rate

Figure 2: Example of a tracked bounding box content showing occlusion. Displaying every 10th frame.

to ensure good quality of data. Rapid movements will cause blurring, resulting in noise and information loss.

We were unable to find any existing dataset that met all of our needs. We realized the need of a novel dataset, and explain the implementation details in the next section.

4. Methodology

To build a clean dataset fit for the problem, we collect 46 manually shot videos with occlusions. We design a system to extract mini-clips fully automatically from raw videos. We keep our architecture modular so that each block can be improved independently. We also keep in mind that the pipeline should be easily extendable/generalizable for open domains. Figure 1 shows our pipeline, consisting of pre-processing, generating candidate bounding boxes, tracking, detecting occlusion frame, and identifying the occluding object. With our ultimate aim being to generate mini-clips from any new raw video, this architecture allows us to keep improving the quality of our generated bounding boxes, occlusion predictions, and estimation of the occluding object independently.

To briefly explain our workflow, we generate candidates and tag them (manually) if they have an occlusion and at which frame it starts to happen. We learn from this data and try to auto-tag new unseen videos scraped from the internet or more manually shot videos to create a large data set. We then estimate a bounding box on the occluding object using the occlusion frame (because we expect the objects to be adjacent to each other at the start of occlusion) and track the video in reverse till δf previous frames. The following sections delve into the details of the pipeline.

4.1. Data Collection and Pre-processing

We shot 46 videos by hand, involving only camera movement and all stationary objects. Note that while collecting the videos we were only focusing on presence of occlusions, smoothness and speed of camera movement. Another focus was to collect a variety of videos in order to be able to develop algorithms extendable to open-domain videos.

We realized that in order to be able to scale our data generation pipeline to generate a million examples, we need to be able to work on videos mined from YouTube. Thus we also collect 23 videos mined from YouTube, consisting of short 5-10 second chunks from compilation videos. Because the videos are so small, this eliminates the chances of a video being shot from multiple cameras and the view switching in between (causing tracker failures, etc).

For every video, we create "chunks" which are just cut-down versions of the videos of length of up to 500 frames, with adjacent chunks having an overlapping region of 250 frames. Every chunk is then treated as a sample candidate for the final dataset.

4.2. Bounding Box Generation

For each chunk, we generate up to 10 candidate bounding boxes for an object (or part of an object) of interest to be tracked. To obtain these candidate boxes in each chunk, we use the following process:

1. Obtain SIFT [8] features/keypoints on first n frames in the chunk, separated by d frames. We used $n = 10$ and $d = 5$.
2. Match keypoints from the first frame with keypoints in each of the $n - 1$ other frames, and discard keypoints that couldn't be matched.
3. Discard keypoints that do not exist in all of the n frames.
4. Discard keypoints in border regions of the first frame. Border region is defined by a fraction f of the frame dimension. We use $f = 0.1$, i.e. keypoints in the first and last 10% of width and height of the frame are discarded.
5. If features overlap, then discard keypoints for the smaller features among the overlapping features.
6. The remaining keypoints are our final bounding box candidates. Constrain their size by imposing min-max limits, i.e. features smaller than min pixels in size will have a box of min pixels in size, and features larger than max pixels in size will have a box of max pixels in size. We use $min = 50$ and $max = 200$.

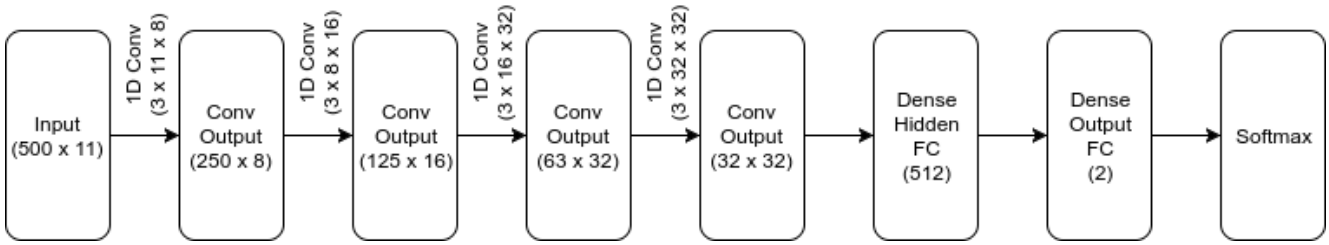


Figure 3: CNN for classifying mini-clips with occlusion/no-occlusion

4.3. Tracking and Occlusion Frame Detection

For each bounding box, we track it in the chunk using the Distribution Field tracker [11]. We tried several different trackers like CCOT [3], Staple [1], KCF [5], and Diagnose [15]. Only DF tracker and CCOT gave results stable enough for our work. CCOT was better than DF tracker in terms of absolute tracking performance, but it resulted in varying bounding box size as the tracked object moved and scaled. So, we chose DF tracker. An additional benefit with DF tracker is that we can access the DF model used for tracking and use it generate an L2-distance metric which we will use later.

Next, our task is to predict:

1. whether an occlusion occurs, or if it is a case of tracker failure, if necessary; and
2. if occlusion occurs, on which frame does it start?

For this purpose, after tracking we also calculate several image-space distances (averaged L2 distances between the image inside each tracked bounding box with the initial image inside the bounding box in the first frame). We end up with the following 9 plots for each candidate in the tracked mini-clips.

- L2 difference between the tracked Distribution Field model at each frame with Distribution Field model at first frame
- After splitting the bounding box into 4 halves - left and right halves, upper and lower halves, L2 difference of the left half of image in bounding box at each frame with the left half of image in bounding box at the first frame. Similarly, 3 more L2 differences for right, upper, and lower halves.
- 4 more differences exactly as in the step above, but using bounding boxes of double the size but centered at the same point. This doubling of the bounding box provides us more context around object patch being tracked.

These 9 distance measures are plotted in the bottom of Figure 5. The correspond to the difference of the currently being tracked box from its initial patch, and will be used as features in the classifier, along with the (x, y) location of the tracked bounding boxes at each frame, giving us a total of 11 features per frame.

We use a hybrid approach between machine learning and procedural algorithmic techniques. We use a 1D CNN classifier to detect whether the occlusion happens or not (including tracker failure) in each mini-clip. For the mini-clips where occlusion happens, we developed a filter and gradient based method to estimate the frame at which occlusion starts to occur.

4.3.1 Occlusion Classifier

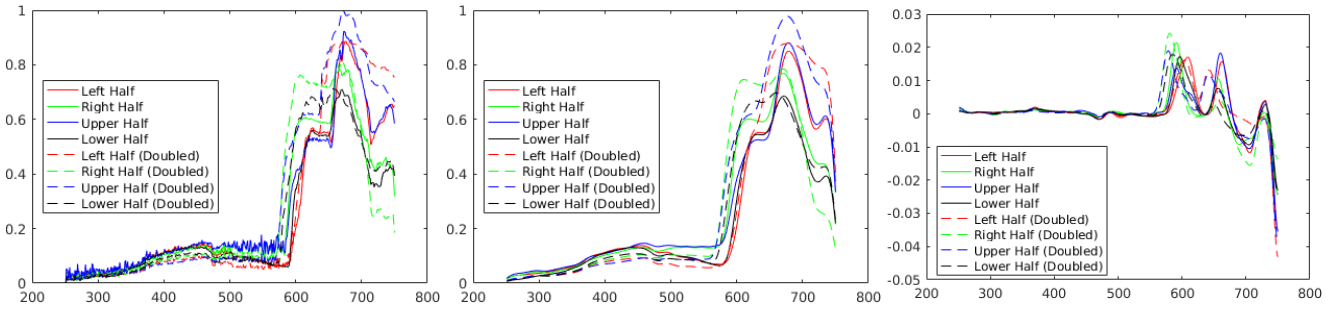
We used the 11 features per frame as described above to classify whether a candidate has valid occlusions or not. To get the ground truth labels of occlusion for each bounding box, we manually tagged 700 videos using a custom-made MATLAB app (discussed more in a later section) to easily view, annotate, and review the mini-clips. The labels are: -1 for tracker failure, -2 for bad bounding box initialization, 0 for no occlusion, and an integer greater than 0 indicating the frame number of the start of an occlusion. The moment/frame we choose as start of occlusion is the frame at which a human (the annotator) can confidently see that the object has begun to get occluded.

$$f_{occ} = \begin{cases} \text{n} & \text{if occlusion occurs at frame } n \\ 0 & \text{if no occlusion happens} \\ -1 & \text{if tracker failure} \\ -2 & \text{if bad bounding box} \end{cases}$$

We then convert this into a binary classification problem by merging classes -2, -1 and 0 into 0, and the remaining occlusion labels into 1. Our classification network is shown in figure 3. We get an accuracy of up to 80% in this classification task.

4.3.2 Estimating f_{occ}

For the mini-clips which got classified as having an occlusion in the previous sub-task, we further process the L2-difference metrics we generated above, to estimate a frame



(a) Normalized to [0,1] **(b)** Filtered (a) with a Gaussian **(c)** Gradients of (b)

Figure 4: Plots of various L2 distances (in image-space) we make use of to estimate f_{occ}. Frame number is on X-axes. Ground truth f_{occ} was 587. Prediction using out method was 579.

number to indicate the start of an occlusion. The process for this is as follows:

1. Apply a Gaussian filter on the 8 sectional L2-distances (obtained from the 4 halves each in the bounding boxes and the doubled bounding boxes). A sample output is in figure 4b.
2. Calculate gradients of the filtered L2-distances at each frame
3. Extract peaks (of prominence above a threshold) in the gradients curves. A sample output is in figure 4c.
4. We observe that for occlusion, the curves for doubled bounding boxes will see peaks a few frames before the the curves for the normal bounding boxes. Thus,
5. We discard the peaks in the curves for doubled bounding boxes if they are not followed by a peak in the curves of the normal bounding boxes within some n frames. We use $n = 30$.
6. Similarly, we discard the peaks in the curves for normal bounding boxes if they are not preceded by a peak in the curves of the doubled bounding boxes with same n frames.
7. By this point, we only have a few candidate peaks in each curve. Now we try to see which of the peaks appear in multiple curves. For an occlusion a peak would appear simultaneously in 2-3 out of the 4 curves of the doubled bounding boxes. And are followed by corresponding peaks in the curves of the normal bounding boxes within some frames. We discard the peaks which don't have this property, i.e. don't appear simultaneously across curves, as these are just peaks due to factors other than occlusions.
8. We pick the location of the first peak in the remaining candidate peaks in the curves of the doubled bounding boxes as our estimate of the beginning of the occlusion.

Using this method we get an estimation accuracy of up to 75%. An estimate is taken to be correct if it falls within $[f_{occ} - 50, f_{occ} + 10]$, f_{occ} is the ground truth label for frame of start of occlusion.

4.4. Estimating Occluding Object

The occlusion frame predicted in the previous step is where an occluding object occludes our tracking box. We estimate another bounding box on the occluding object to get the final training examples. To obtain this new bounding box, we calculate sectional L2 distances in the images space (the 4 halves like in previous sub-tasks) between the doubled bounding boxes of the estimated occlusion frame and a frame n frames before it. We use $n = 20$. This halves showing greater difference gives us an indication of the direction from which the occluding object is entering our tracked bounding box. By calculating the ratio of the difference between the L2 distance in the left and right halves, and the difference between the L2 distance in the upper and lower halves, we obtain a unit vector for the direction from which the occluding object is coming. We generate a new bounding box of the same size as the tracked bounding box, but translated by an amount equal to the size of the bounding box in the direction of the unit vector obtained in previous step.

Once we have the bounding box on the occluding object in the estimated frame of occlusion, we backtrack this box for up to n frames before the estimated frame of occlusion. We'll mark this as a new start frame, and the end frame being the estimated frame of occlusion. Now we have two bounding boxes in this start frame, one on the object that will get occluded eventually, and one of the object that will occlude the other one.

These small clips, a few dozen frames with two tracked bounding boxes provided for two objects of interest, will be our training examples for further learning. The task will be to predict which tracked object will occlude the other. In order to do this prediction, we expect the network to learn concepts of depth and motion.

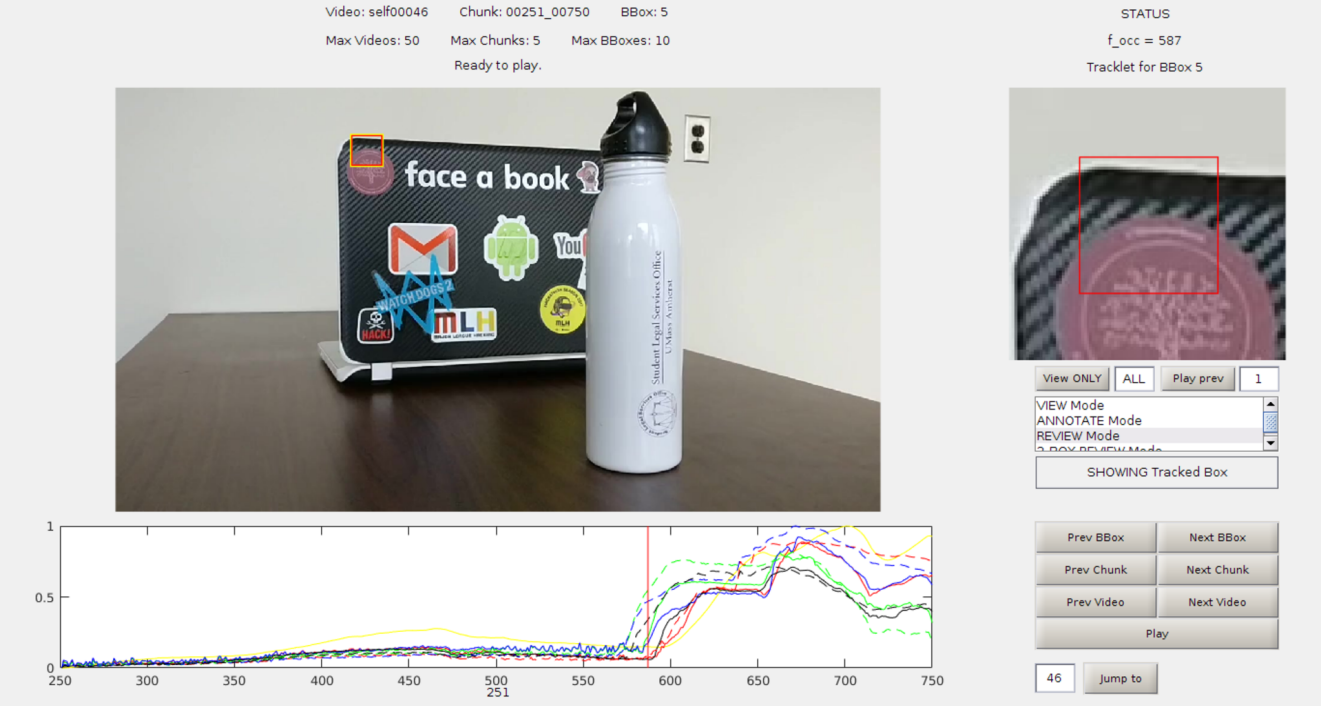


Figure 5: Visualization software screenshot in Review Mode. The biggest image is the full video frame. Small image on the right is the mini-image centered on the tracked bounding box. Plots below show the calculated L2-distance metrics. Vertical red line in the plot indicates frame of occlusion.

5. Visualization Software

We used MATLAB to create a GUI-based app to enable us to integrate visualization, browsing, annotation and analysis of mini-clips and feature plots. A screenshot is provided in figure 5. The software allows us to:

- Visualize the bounding boxes generated for each chunk
- Play each chunk, with features like forward/back and next/previous bounding box, chunk and video
- Annotate each bounding box with a $f_{occ} \in$ {-2,-1,0,x} where $x \in [f_{start}, f_{end}]$
- Review/rewrite the annotations
- Visualize the distance plots vs frame number with a seek bar for current play position and a f_{occ} indicator bar
- Visualize the final version of 2 tracking bounding boxes

The software integrates many keyboard shortcuts, operating modes and configuration options for start up.

6. Results and Conclusions

We are able achieve up to 80% accuracy on classification of a video as containing an occlusion or not, and up to 75% accuracy on estimating the frame of start of occlusion in videos where we have an occlusion. Combining these two, we would have up to 60% accuracy on correctly classifying and estimating the frame of start of occlusion.

Some common causes of failures we observed:

- substantial jerks
- speed of camera movement too high
- substantial rotation of object instead of simple translation
- similarly colored/shaded backgrounds and occluding objects confuses the DF tracker and also affects the distance metrics
- camera refocusing and blur
- unexplained tracker failures

A major pain-point in our data was that the bounding boxes didn't align with the object reasonably well. It would be ideal to have medium-sized object almost entirely inside a bounding box.

7. Future Work

The project has evolved as we have developed a better understanding of it. For example, we started off with an aim to generate mini-clips of mini-patches by centering the mini-clip to the tracked patch and include a buffer region around it for context. The initial idea was that the bounding box would occlude or get occluded by some other object. In this case, the training examples would be in the frame

of reference of the object being tracked. However, we soon realized that the crux of the problem is simply to perceive which object is closer. Using this insight, we modified our goal to generate mini-clips containing two bounding boxes - one each on the two objects of interest - instead of one.

Our current approach is a good starting point to further refine the dataset generation process. It should be possible to generate more features using the tracking information. And also use neural networks to estimate the frame of occlusion as well. We believe a final accuracy (currently classified and estimated frame of occlusion) of 80-90% or even higher is possible. This will lead to a very rich dataset for training a network.

Future work should proceed in two directions - (a) improving and extending the data generation framework to different domains, and (b) training deep networks using this data and investigating physics rules learnt by it. Because our purpose in data collection was to develop this system, we collected a variety of data, including videos of forests, hockey game, etc. As the immediate next step, we should collect new videos with strict specifications of objects, background and camera motion and apply this pipeline to generate clean data.

8. Acknowledgements

We would like to thank Prof. Erik Learned-Miller for guiding us through the many twists and turns this study took. His inputs and suggestions were key in developing this process the way it is. We also acknowledge Huaizu Jiang for sharing with us the codebase for the trackers.

References

[1] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401–1409, 2016.

[2] J. J. Campos, S. Hiatt, D. Ramsay, and C. Henderson. The emergence of fear on the visual cliff. *The development of affect*, 1:149–182.

[3] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg. Beyond correlation filters: Learning continuous convolution operators for visual tracking. In *European Conference on Computer Vision*, pages 472–488. Springer, 2016.

[4] C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016.

[5] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[6] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA, 2014. IEEE Computer Society.

[7] H. Kuhne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[9] Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta. Occlusion detectable stereo-occlusion patterns in camera matrix. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 371–378. IEEE, 1996.

[10] R. Raman, P. K. Sa, and B. Majhi. Occlusion prediction algorithms for multi-camera network. In *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, pages 1–6. IEEE, 2012.

[11] L. Sevilla-Lara and E. Learned-Miller. Distribution fields for tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1910–1917. IEEE, 2012.

[12] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[13] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[14] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool. Temporal segment networks: Towards good practices for deep action recognition. *CoRR*, abs/1608.00859, 2016.

[15] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia. Understanding and diagnosing visual tracking systems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3101–3109, 2015.

[16] C. L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on pattern analysis and machine intelligence*, 22(7):675–684, 2000.

Some sample sequences from the dataset generated from our videos

Object in green box will occlude the object in red box. But this distinction won't be provided to the network for learning; it will just see two boxes with no significance attached to them about their relative depth.

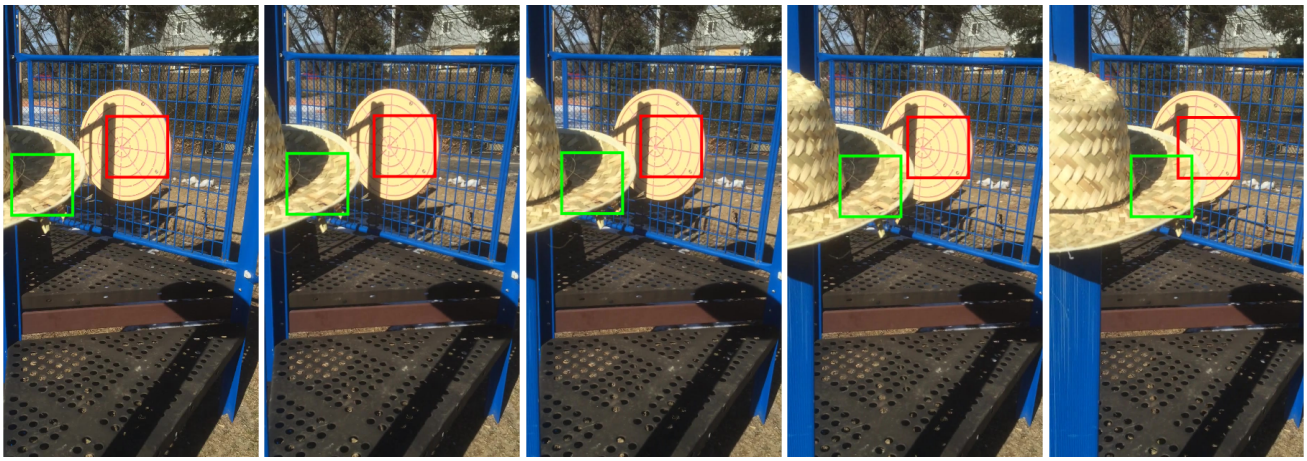


Figure 6: Data sample 1. Time increases from left to right.

Figure 7: Data sample 2. Time increases from left to right in first row, then left to right in second row.

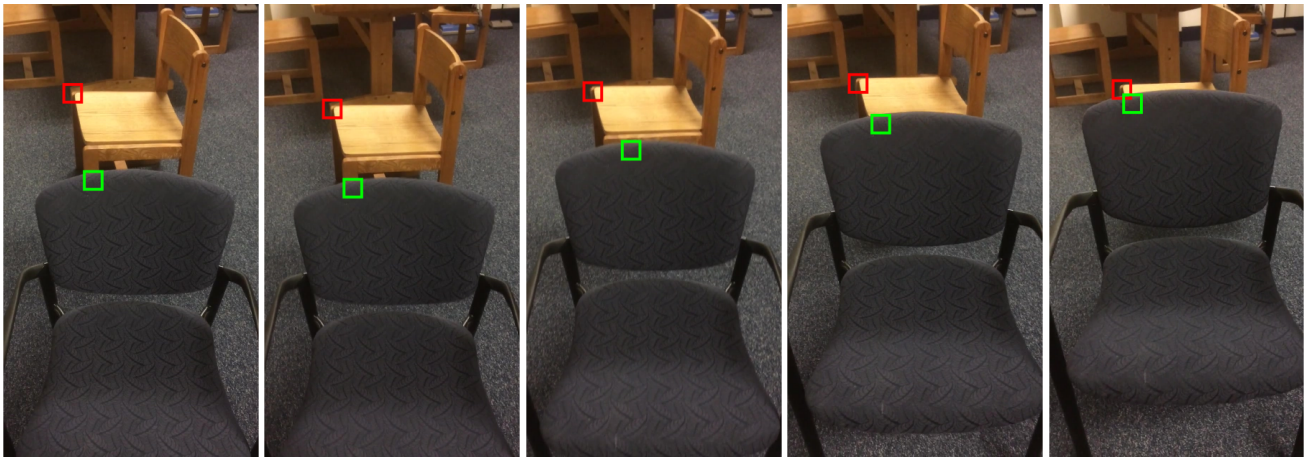


Figure 8: Data sample 3. Time increases from left to right.

Figure 9: Data sample 4. Time increases from left to right in first row, then left to right in second row.

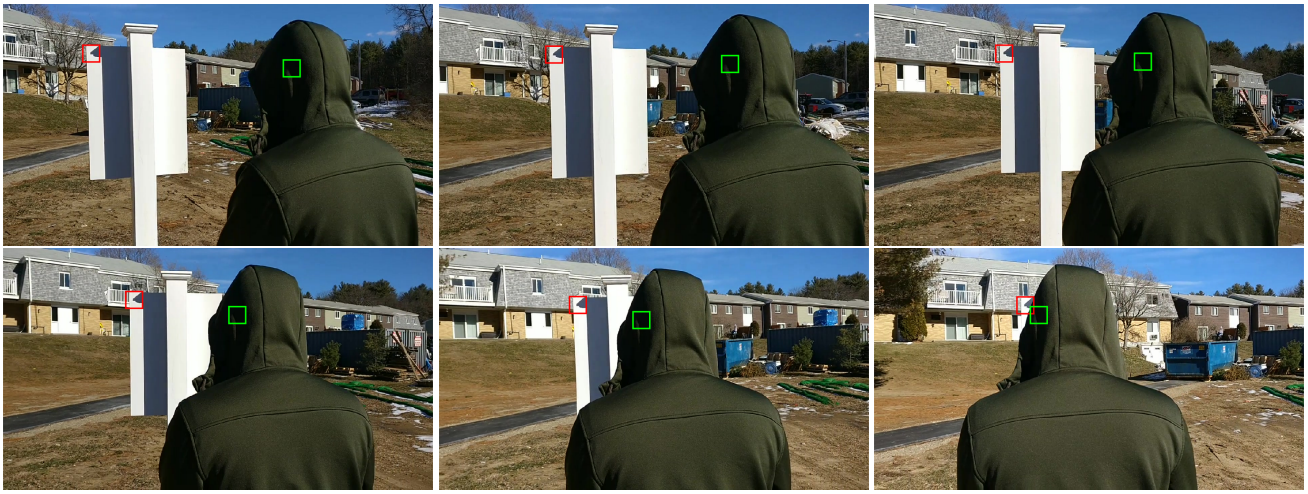


Figure 10: Data sample 5. Time increases from left to right in first row, then left to right in second row.

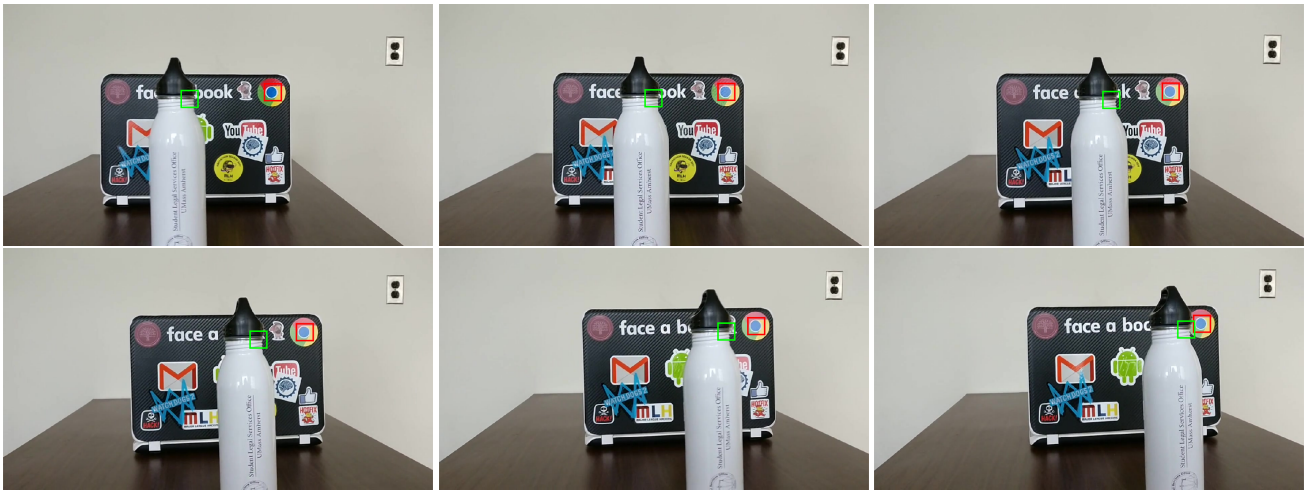


Figure 11: Data sample 6. Time increases from left to right in first row, then left to right in second row.