Vehicle Trajectory Extraction using Deep Learning

Xinhe Ren
UC Berkeley
jim.x.ren@berkeley.edu

Yue Zhang
UC Berkeley
y.zhang@berkeley.edu

Abstract

To collect extensive data on realistic driving behavior for use in simulation, we propose a framework that uses online public traffic cam video streams to extract data of driving behavior. To tackle challenges like frame-skip, perspective, and low resolution, we leverage recent advances in deep learning for object detection and tracking to extract trajectories from the video stream to corresponding locations in a bird's eye view traffic simulator. In this study, we benchmarked several pre-trained deep neural networks for object detection and constructed a video processing and vehicle trajectory extraction pipeline that is robust to vehicle occlusion.

1. Introduction

Autonomous driving simulators offer the potential to rapidly prototype behavioral algorithms. However, a significant concern with developing a simulation is how accurately it reflects the physical world. Currently, autonomous driving companies collect real-world traffic data through countless road trials and expensive sensor and telemetry equipment, a luxury possessed only by a few. In recent years, live streaming has become a popular internet trend. The availability of cheap, live traffic camera streams provides us an economical alternative method of studying traffic interactions.

We examine how to leverage online video streams (e.g., a YouTube stream of a four-way traffic intersection at 7 Ave and Main St in Canmore, Alberta[1]). Using online video streams for data collection has the potential to capture a significant amount of driver demonstration data, however, we must address the challenges of perspective, skipping, and low resolution.

We propose a video processing pipeline which applies recent deep learning advances in object detection [18] to detect vehicles in the video stream. We then explore deep learning based tracking [6]. The output of the system is a set of trajectories of vehicles.

[1] https://canmorealberta.com/webcams/main-street

2. Related Works

2.1. Automated Extraction of Traffic Data

Traffic data historically has been sourced from both traffic cameras and sensors as well as onboard sensor arrays. Research on traffic detection using traffic cameras historically used classical machine vision and digital signal processing techniques to produce real-time traffic scene analysis [12]. More recent work has investigated improving static intersection vehicle detection methods with existing video image vehicle detection systems [37] through improved sensor fusion with camera systems.

Current sources of traffic and vehicle detection data for autonomous vehicle research mostly focus on using vehicle sensors, such as onboard cameras or LIDAR, to detect and identify nearby objects [15]. Another approach is to equip vehicles with GPS sensors and then analyze the decision-making data afterward to examine traffic congestion in urban settings [2]. With the rising popularity of deep convolutional neural networks, it is interesting to explore collecting traffic behavior through fixed traffic cameras without the need for sophisticated vehicle telemetry.

2.2. Object Detection

Region-based CNNs (RCNNs) perform object detection by first proposing a set of features in the images, followed by classification [5]. Multiple improvements and optimizations have been made to RCNN, including Fast RCNN [4] which flipped the order of feature proposal and classification and performs shared convolution operations, and Faster RCNN [24] which has a built in region proposal network to further speed up the model runtime. Other approaches to this problem including training a predictive model using simulation data [19], and performing object detection with depth images [28, 26].

In contrast to RCNN, recent works such as YOLO [23] and SSD [18] use single neural networks that eliminate proposal generation and are able to perform multi-object detection in real time.

2.3. Trajectory Extraction

Object tracking and trajectory extraction are widely studied in the computer vision domain. Hu et al. [10] conducted a survey in 2004 on video surveillance of object motion and behaviors. Specifically, the survey examined region, active contour, feature, and model-based object tracking. One of the earlier studies in 1994 by Koller et al. [12] examines traffic flow on a highway by computing affine transformations of vehicles between sequential frames of the surveillance video. However, the affine transformation assumption breaks down when vehicles execute angle shifting motions such as turning. Vidal et al. [35] applied a Kalman filter to a feature based object tracking algorithm, greatly improving the performance. More recently, Li et al. [16] developed a model based pedestrian tracking system with human segmentation. The system applies a human model prior as a seed for segmentation. In our experiment, we first apply a simple likelihood-based method to classify time agnostic bounding boxes into different candidate trajectories.

Recently, we begin to see many deep learning based approaches to object tracking. In 2005, MD-Net [20] became the first deep learning method to win The Visual Object Tracking challenge (VOT) [13]. Leveraging large training datasets, many offline-trained neural networks can track objects with very high inference speed. Held et al. [8] proposed a convolutional neural network on tracking generic objects, which generalizes to novel objects and tracks at 100 fps. Bertinetto et al. [1] use a Siamese CNN network architecture to detect differences between video frames to track objects, also enabling the network to be trained offline. The Re3 (Real-Time Recurrent Regression Networks) by Gordon et al. [7], in addition to a CNN for vision processing, employs a recurrent neural network for temporal information handling. In our experiment, we test a pre-trained Re3 network as a potential deep learning based replacement for our likelihood estimator based tracking method.

2.4. Learning from Online Videos

There have been many instances of learning from online videos, which can be noisy and ill-constrained. Ulges et al. [34] utilized YouTube content for the autonomous training of a video tagger. Prest et al. [22] trained an object detector from weakly annotated videos on the internet using domain adaptation to improve the performance of the detector. In robotics, Yang et al. [38] explored learning robot manipulation tasks (grasping) by processing videos from the World Wide Web. The paper used a convolutional neural network for object recognition, and action grammar parse tree to interpret the videos' unconstrained semantic structures. Niebles et al. [21] developed a system to learn human motions from YouTube videos. Srivastava et al. [30] use LSTM neural networks to perform unsupervised learning on YouTube videos. Sorschag [29] conducted a survey of video annotation techniques by examining how to collect large amounts of video for machine learning algorithms. Another survey by Vishnu et al. [36] examined how the prevalence of web videos has enabled large-scale learning.

In this study, we use websites such as `earthcam.com` and YouTube, which feature more than hundreds of live traffic cam streams. We apply deep learning object detection networks to these videos to extract vehicle trajectories.

3. Vehicle Detection

Advances in deep learning over the last several years have significantly improved perception and object detection in images. Recently, deep neural networks such as Single Shot Detector (SSD) [18] and Faster-RCNN [25] have demonstrated surprising performance on many object detection datasets, including PASCAL VOC [3] and COCO [17], in which the goal is to classify objects and localize them using bounding boxes. These datasets include familiar objects, such as vehicles and pedestrians. After benchmarking several deep networks on a hand-labeled held-out dataset of collected images, we found that SSD has the highest performance. We utilize the pre-trained SSD network to label vehicles in the collected data with real-time performance.

3.1. Comparing Different Object Detection Networks

There are many deep neural networks architectures for object detection, often trading off between speed and accuracy. Also, sharing pre-trained network weights has become increasingly common. Many network models are available online for immediate, off-the-shelf inference. We are interested in building a fast pipeline for extracting trajectories of cars and people towards real-time data analysis and learning. Thus, we examine using $>$30 fps networks such as SSD. SSD has a simplified network architecture that combines a feature extractor with additional layers that perform object detection at various scales. We compare several feature extractors: VGGNet [27], InceptionNet [31], and MobileNet [9]. Google's open-source TensorFlow Object Detection API[2] [11] provides implementations of Faster-RCNN, SSD-InceptionNet, and SSD-Mobilenet trained on the COCO dataset. We also examine an implementation of SSD-VGGNet[3] trained on the PASCAL VOC dataset.

To benchmark these object detection networks, we hand-labeled held-out datasets of 100 daytime images collected. We used the BBox-Label-Tool[4] to draw bounding boxes

[2]https://github.com/tensorflow/models/tree/master/research/object$_detection$

[3]TensorFlow implementation of SSD-VGGNet: `https://github.com/balancap/SSD-Tensorflow`.

[4]Bounding box labeling tool: `https://github.com/puzzledqs/BBox-Label-Tool`.

Network	Precision	Recall	Average IoU
SSD-VGGNet	**0.907**	0.354	**0.787**
SSD-InceptionNet	0.427	0.273	0.706
SSD-MobileNet	0.431	0.258	0.702
Faster-RCNN	0.570	**0.553**	0.702

Table 1: Results of evaluating four object detection networks on the hand-labeled daytime dataset. We used the default parameters of these deep neural networks in this experiment. We report the precision, accuracy, and average bounding box quality (IoU) of true positive detections for car class labels. The dataset contains a total of 718 cars.

around objects. To evaluate the networks, we use the following procedure:

- For a particular image, obtain the detection predictions from the network, which include a class label, confidence score, and a bounding box.
- Discard any predictions with a confidence score lower than 0.5, the default threshold defined in the balancap/SSD-Tensorflow repository.
- Compute the Intersection over Union (IoU) between each detected bounding box and each of the ground truth bounding boxes in that same image. If the best IoU match is greater than 0.5 for detection, it is considered a true positive, and we record the bounding box quality. If there are no matches of sufficiently high quality, the detection is considered a false positive.
- After processing all detections, any ground truth detection that was not matched by a predicted detection is a false negative.

We complete this procedure for each of the four networks on both hand-labeled datasets. Table 1 lists the resulting precision, accuracy, and average bounding box quality for the two object classes. We also show examples of the detections outputted by the four networks on an example image in Figure 1. We note that the recall values are quite low due to the difficulty of the hand-labeled test dataset. The images are of a busy intersection with many cars that are close together, often partially occluding each other. Also, there may be cars at a farther distance that are difficult for any network to detect.

Although we observe higher recall values with Faster-RCNN, we use SSD-VGGNet because of its higher precision and better bounding box quality on our dataset.

4. Deep Object Tracking

We explore end-to-end deep learning methods for object tracking, in particular, Re3 [6]. Re3 is a deep neural network that combines the image processing advantages of a

Figure 1: Example detections from each of the four networks on a sample daytime image: SSD-VGGNet (top left), SSD-InceptionNet (top right), SSD-MobileNet (bottom left), and Faster-RCNN (bottom right). Detections of cars are shown with blue bounding boxes, and detections of people are shown with green bounding boxes.

CNN with the temporal data processing power of an RNN. In this study, we use a pre-trained version of Re3[5] released by its paper's authors.

Re3 can be trained offline. This means that the network's tracking ability can generalize to arbitrary objects, and does not need to be trained on samples of specific target objects. The inputs to the Re3 network during tracking are the sequence of images and an initial bounding box of the object of interest. The network also supports the tracking of multiple objects in the same frame, which helps us track multiple vehicles in the same video clip.

We use SSD to generate and supply potential initial bounding boxes and Re3 to accomplish the tracking. Algorithm 1 illustrates how we combine these two networks. We define T to be a list of trajectories being tracked by Re3, f to be a frame of a video as an image, B_{Re3} to be a list of tracked bounding boxes from querying Re3, and B_{SSD} to be a list of bounding boxes from object detector SSD. When discussing if two bounding boxes "match", we consider the metric: intersection over union (IoU). If two bounding boxes have an IoU greater than 0.7, an empirically chosen threshold, then we consider the two bounding boxes "matching".

Figure 2 illustrates the comparison between SSD and Re3 outputs. Note that SSD is time and trajectory agnostic, and does not assign bounding boxes to trajectories. Contrarily, Re3 assigns each detected object to a trajectory. For example, "7_1" is the first vehicle trajectory in the video, and "15_2" is the second pedestrian trajectory in the video. Note that SSD failed to detect the car in trajectory "7_2" nor the person in trajectory "15_3" in this video frame. Without

[5]Pre-trained Re3 in TensorFlow: `https://gitlab.cs.washington.edu/xkcd/re3-tensorflow`

$T \leftarrow []$
while *has next frame,* f **do**
  $B_{Re3} \leftarrow$ Re3.getBBox(f,T)
  $B_{SSD} \leftarrow$ SSD.getBBox(f)
  for b *in* B_{Re3} **do**
    if b *does not match any in* B_{SSD} **then**
      T.add(Re3.track(f, b))
    end
    if b *outside intersection* **then**
      T.remove(Re3.track(f, b))
    end
  end
end

Algorithm 1: Tracking with SSD and Re3

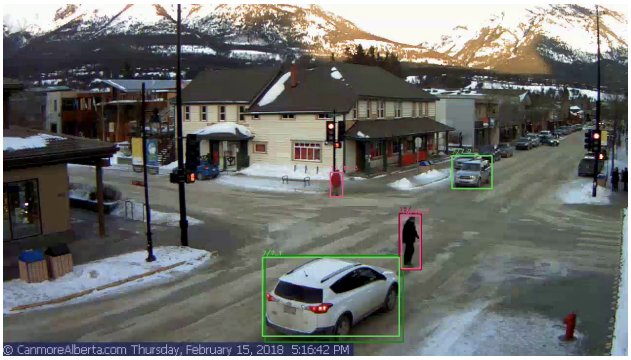

Figure 2: SSD and Re3 output bounding boxes with class and trajectory labels. Green represents cars (class 7) and pink represents humans (class 15). Dark green and pink rectangles show bounding boxes from SSD; light green and pink rectangles show bounding boxes from Re3. Both pre-trained neural networks produce consistent and robust bounding boxes for vehicles and pedestrians. In this image, we also show a failure mode of SSD, where the further objects are not successfully detected.

useing Re3, this failure causes frame skip. Re3, however, is guaranteed to output a bounding box for each trajectory in each frame.

5. Homography: From Camera to Bird's Eye Perspective

The bounding boxes generated by SSD gives the location of vehicles within the RGB image taken from the traffic camera perspective. To obtain the locations of the agents in the simulator, we utilize homography [32] to project the camera view to a bird's eye viewpoint.

Homography works by estimating a projective transformation matrix that morphs pixel locations from a source domain into a target domain. In this study, we are transforming pixel positions in the camera view domain to locations in the bird's eye view of the intersection. We estimated the

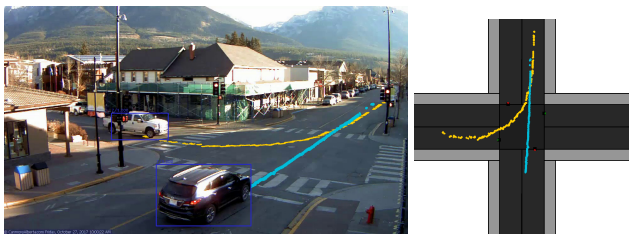

Figure 3: Trajectories of two vehicles extracted from camera view. Homography is then applied to each point in the trajectories to transform it to the bird's eye view.

matrix on four pairs of corresponding points between the camera and the top-down views. The points were selected such that the corners of the intersection in the traffic camera matched the corners of a square centered in the top-down view.

After homography, we still need to specify a point from the bounding box that corresponds to where the vehicle is centered on the road. To determine a point on the road, we use the midpoint of the bottom edge of the bounding box. However, this selected point may not correspond the vehicle's true point on the road, so we learn a linear mapping from the selected point to a corrected point to adjust for the inaccuracy using a hand-labeled dataset of corrections to apply. We propose some potential improvements in section 6

6. Conclusion and Future Work

By making use of recent advances in deep learning object detection and tracking, we create a framework that extracts driving behavior data from online video streams. We benchmarked four object detection networks and concluded that SSD with VGGNet is most suitable, due to its high precision and IoU quality of bounding boxes. By comparing the bounding boxes produced by SSD with those produced by the Re3 object tracking network, vehicle paths can be tracked with surprising accuracy with no human in the loop for initial frame labels. However, here we note a few areas for improvement.

6.1. 3D Reconstruction of Vehicle Orientations

Currently, we use the midpoint of the bottom edge of bounding boxes as an approximation of the of the vehicle's position, which is inaccurate if the vehicle is not exactly sideways towards the camera. To compensate for the inaccuracy in this heuristic, we trained a linear offset based on the vehicle's position in the scene. A more robust way to represent a vehicle as a point is to reconstruct the 3D model orientation of the car, and then project the centroid of that model onto the camera image plane.

This is a single-view 3D reconstruction task. Tulsiani et al. [33] improved existing techniques by enforcing consis-

tency between 3D model and 2D observation with the introduction of a differentiable ray consistency (DRC) term. Although, in this study, we only wish to identify the orientation and approximate centroid of a vehicle. We do not need our 3D models to accurately match those of cars in the scene. Leotta et al. [14] tracked vehicles by fitting a deformable, generic model of a vehicle to segments of a vehicle in a scene. Improving the projection and homography step can yield bird's eye view trajectories that better match the ground truth.

6.2. Pedestrians

We hope to expand our pipeline experiment to include pedestrians. Since pedestrians are much smaller than vehicles, pedestrians are much harder for SSD to robustly detect and track. Identifying individual pedestrians in groups is also difficult. Table 2 shows some initial testing results on detecting humans with the default parameters. All the metric are lower compared to those in Table 1 for vehicles.

Network	Precision	Recall	Average IoU
SSD-VGGNet	**0.869**	0.201	**0.709**
SSD-InceptionNet	0.402	0.189	0.687
SSD-MobileNet	0.469	0.170	0.663
Faster-RCNN	0.532	**0.466**	0.668

Table 2: Results of evaluating the same four object detection networks on the hand-labeled daytime dataset of pedestrians. The dataset contains a total of 264 pedestrians.

References

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[2] R. Carli, M. Dotoli, N. Epicoco, B. Angelico, and A. Vinciullo. Automated evaluation of urban traffic congestion using bus as a probe. In *CASE*, pages 967–972. IEEE, 2015.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

[4] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016.

[6] D. Gordon, A. Farhadi, and D. Fox. Re3 : Real-time recurrent regression networks for object tracking. *CoRR*, abs/1705.06368, 2017.

[7] D. Gordon, A. Farhadi, and D. Fox. Re3 : Real-time recurrent regression networks for object tracking. *CoRR*, abs/1705.06368, 2017.

[8] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[10] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(3):334–352, 2004.

[11] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, 2017.

[12] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 126–131. IEEE, 1994.

[13] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, F. Porikli, L. Cehovin, G. Nebehay, G. Fernandez, T. Vojir, A. Gatt, et al. The visual object tracking vot2013 challenge results. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 98–111. IEEE, 2013.

[14] M. J. Leotta and J. L. Mundy. Vehicle surveillance with a generic, adaptive, 3d vehicle model. *IEEE transactions on pattern analysis and machine intelligence*, 33(7):1457–1469, 2011.

[15] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE, 2011.

[16] K. C. Li, H. C. Wang, and J. M. Chiu. Pedestrian tracking system by using human shape prior model. In *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1139–1143, Aug 2014.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[19] C. Mitash, K. E. Bekris, and A. Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. *arXiv preprint arXiv:1703.03347*, 2017.

[20] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 4293–4302. IEEE, 2016.

[21] J. C. Niebles, B. Han, A. Ferencz, and L. Fei-Fei. Extracting moving people from internet videos. In *European conference on computer vision*, pages 527–540. Springer, 2008.

[22] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3282–3289. IEEE, 2012.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2017.

[26] M. Schwarz, H. Schulz, and S. Behnke. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1329–1335. IEEE, 2015.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014.

[29] R. Sorschag. A high-level survey of video annotation and retrieval systems. *International Journal of Multimedia Technology*, 2(3):62–71, 2012.

[30] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[32] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[33] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, volume 1, page 3, 2017.

[34] A. Ulges, C. Schulze, M. Koch, and T. M. Breuel. Learning automatic concept detectors from online video. *Computer vision and Image understanding*, 114(4):429–438, 2010.

[35] F. B. Vidal and V. H. C. Alcalde. Window-matching techniques with kalman filtering for an improved object visual tracking. In *2007 IEEE International Conference on Automation Science and Engineering*, pages 829–834, Sept 2007.

[36] M. K. Vishnu and P. Scholar. A survey on the propagation of web videos. 2014.

[37] Y. Wang, Y. Zou, H. Shi, and H. Zhao. Video image vehicle detection system for signaled traffic intersection. In *Hybrid Intelligent Systems, 2009. HIS'09. Ninth International Conference on*, volume 1, pages 222–227. IEEE, 2009.

[38] Y. Yang, Y. Li, C. Fermüller, and Y. Aloimonos. Robot learning manipulation action plans by" watching" unconstrained videos from the world wide web. In *AAAI*, pages 3686–3693, 2015.