Autonomous driving challenge: To Infer the property of a dynamic object based on its motion pattern using recurrent neural network

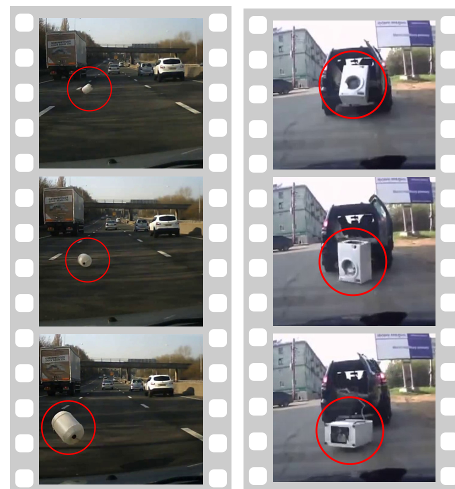
Mona Fathollahi, Rangachar Kasturi
Department of Computer Science and Engineering
University of South Florida, Tampa
(mona2,r1k)@mail.usf.edu

Abstract

In autonomous driving applications a critical challenge is to identify action to take to avoid an obstacle on collision course. For example, when a heavy object is suddenly encountered it is critical to stop the vehicle or change the lane even if it causes other traffic disruptions. However, there are situations when it is preferable to collide with the object rather than take an action that would result in a much more serious accident than collision with the object. For example, a heavy object which falls from a truck should be avoided whereas a bouncing ball or a soft target such as a foam box need not be.

We present a novel method to discriminate between the motion characteristics of these types of objects based on their physical properties such as bounciness, elasticity, etc.

In this preliminary work, we use recurrent neural network with LSTM cells to train a classifier to classify objects based on their motion trajectories. We test the algorithm on synthetic data, and, as a proof of concept, demonstrate its effectiveness on a limited set of real-world data.



(a) (b)

Figure 1: Selected frames of dynamic object in the road. (a): A plastic container which is safe to cross over. (b): a heavy object that should be avoided by the vehicle.

1 Introduction

Over the last few decades, the technology of self-driving cars has made dramatic progress. One of the critical challenges of this emerging technology is the safety of both occupants and other road users. The current prototype of autonomous cars are equipped by advanced sensors such as ultrasonic sensor, vision sensor, radar and LIDAR. These sensors along with sophisticated data fusion algorithms are able to detect and track obstacles in real-time with very good resolution.

When an obstacle is detected in the planned path, either its planned route should be modified or the vehicle should come to stop. Depending on the traffic situation and vehicle speed, this policy could cause collision to other vehicles. Therefore, obstacle avoidance may not always be the safest action. Similar challenge has been proposed in [6] for the first time.

The intuitive solution would be to recognize the object before taking an action. The IU should predict whether it is safe to pass over the object or it should inevitably follow avoiding policy.

A sample video for each scenario is downloaded from Youtube and a few frames are shown in the Figure 1. In the first video, an empty plastic container is bouncing in the road which is safe to pass. In the second video, a heavy object is falling out of the front car which should definitely be avoided.

The immediate solution that one might consider is to formulate the problem as a regular image classification task and collect a dataset of collision safe and unsafe object images. While there is an impressive progress in object detection/recognition [5], this approach has several challenges which makes it ineffective for this particular application.

First, collecting a dataset that contains different objects in different lighting conditions and viewpoints is a difficult

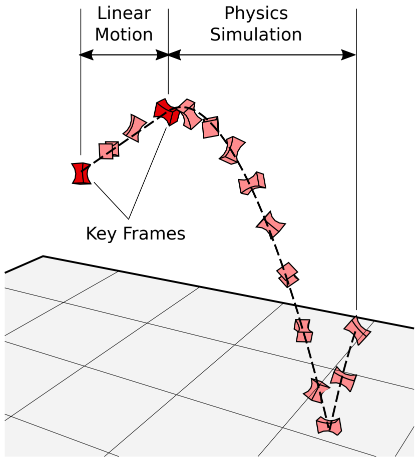


Figure 2: Object trajectory synthesis in Blender. The first few frames are to generate initial velocity (Linear motion). The second part, physics simulation, is recorded as trajectory of the object.

task by itself. Second, it is almost impossible to infer the weight of an object by its visual cue, for example two very similar boxes which one of them is full of metal pieces and the other one is empty have similar images.

Finally, there is a high possibility of recognition failure because the image resolution is usually poor for far objects. Also, the classifier should decide in short period of time, where motion blur might make the problem even more challenging. For example, the white plastic container in the first column of Figure 1 could be classified as a gas capsule.

The last two challenges could be easily resolved by a human observer by looking at the trajectory of empty box versus heavy box (e.g. plastic container versus a gas capsule). Therefore, assuming that the real-time trajectory of the dynamic object is provided [2], we claim that motion pattern provides strong cue to infer the object dynamics properly and to classify it as a ”safe to pass over” or ”must avoid” object.

2 Method

In this section, our goal is to design a classifier to infer object's bounciness characteristic based on its trajectory when it hits the ground. Our approach is based on the observation that the bouncing pattern of the objects is directly affected by their mass.

2.1 Data

To collect data, we should throw different objects with different masses and shapes and record their trajectories. On the other hand, since the bounciness of the object is also related to initial velocity, we should collect a lot of data to be able to learn the effect of mass on the trajectory. Therefore, dataset collection in this case is cumbersome and expensive because hitting the object with high velocity would damage it.

Therefore, we generate synthetic video with a binary label denoting heavy or light object trajectory. We utilized open source 3D creation suite, Blender, to generate motion data of bouncing objects. Blender uses Bullet Physics Library for collision detection, rigid body dynamic simulation and other Physics simulations tasks.

Each trajectory starts from a random coordinates and Euler angles and the object has random initial linear and angular velocities. To generate random initial velocities, two key-frames are inserted at first and seventh frames. Also, the height of object at first frame and both linear and angular positions at seventh frame are randomized. (Figure 2). The physics engine takes over the object animation after seventh frame. The world coordinates of the object after the seventh frame are recorded as object trajectory time series.

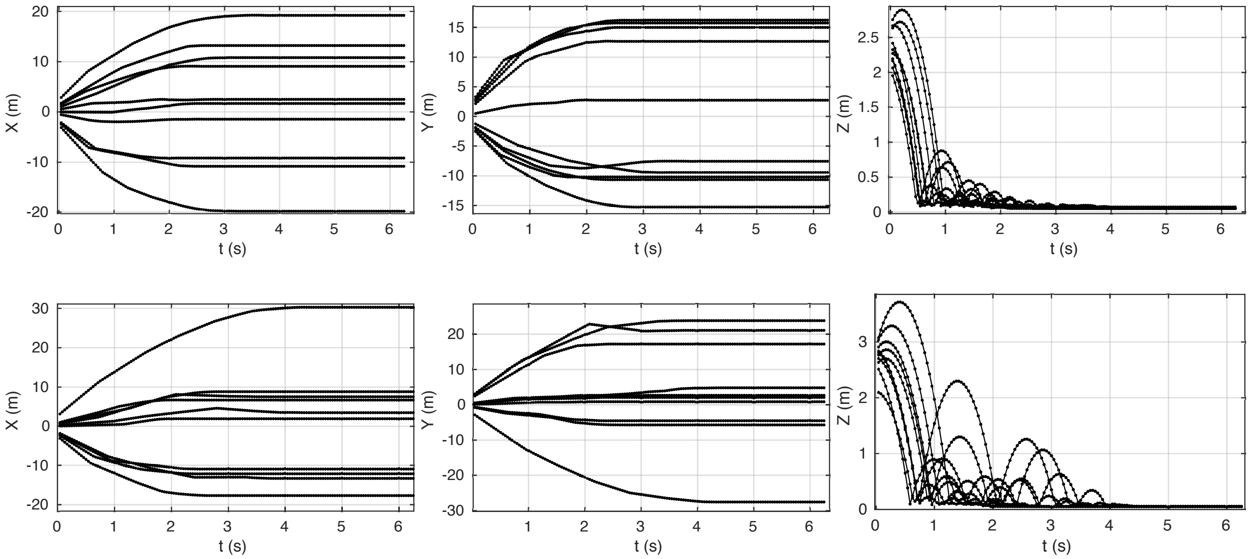
In the initial phase of our project, we only consider two object categories; the first class is the trajectory of light objects that have a high tendency to bounce when they hit to the ground, and the second class are the objects that are heavy and have more tendency to slide than bounce. We generated 1000 training videos, and 1000 test videos for both categories.

Some randomly chosen examples of the generated trajectory data are shown in Figure 3. Even though there is a clear distinction between Z dimension of the trajectories, we still see subtle difference in X and Y components. For example, for a light object (higher bounciness) it takes more time to come to full stop and this is reflected in X and Y coordinates and this justify the superior performance of classification when 3D data is used (Table 1). Finally, although some statistical differences are detectable between the two categories, the plots in this figure suggest that no simple rule can be proposed based on, for example, number of bounciness or time series duration and a more involved classification algorithm is required.

2.2 Classifier

In this section, we assume that the trajectory of the object is given using a tracking algorithm. Therefore the problem is reduced to the time series classification.

In this work, we adopted, Recurrent Neural Network models (RNN) for trajectory classification. RNN is a type of artificial neural network that is able to process data with arbitrary input sequence lengths. Their internal memory units and feedback loops have made them a very successful tool in sequential data prediction in various domains [4] [3].



(a) (b) (c)

Figure 3: Samples of synthesized trajectory data set. The figures shows time plots of (a)X, (b)Y, (c)Z coordinate of the (top)heavy, (bottom)light object.

Recently, they have been used in the context of time series classification [8]. In this work we also use RNN architecture with peePhole Long Short-Term Memory (LSTM) units [7] for motion trajectory classification.

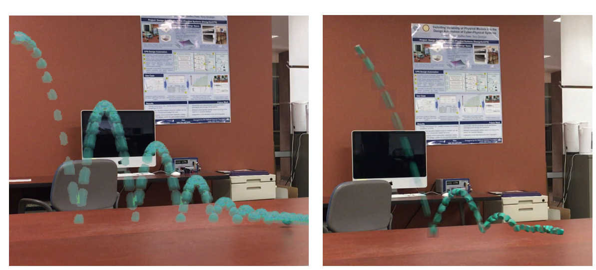
The input to the network is the first T seconds of objects' trajectory. Training and test time series are normalized by their standard deviations in each dimension. Our network architecture is a two-layered LSTM with 64 hidden units in each layer. The hidden state at the last time step of LSTM is fed into a softmax layer. The entire model is minimized by the cross-entropy error metric. We also add a dropout layer between second LSTM layer and softmax layer with rate 0.8. In our first experiment, we compare the influence of input dimension on the classification accuracy. We consistently get superior performance on 3D inputs. To demonstrate this observation, we grid search the parameters to get the maximum accuracy when only z-coordinate of the trajectories is used for training. The same parameters are used to train the network with 3D input and the results are compared in the Table 1.

To compute parameter gradients, the truncated back-propagation-through-time (BPTT) approach [9] is used to reduce the probability of vanishing gradient problem for long input sequences. The entire implementation is done using Tensorflow [1] package.

Table 1: Best accuracy for 3D and 1D input sequence.

Trajectory dimension(s)	Best Accuracy(%)
X, Y, Z	0.81
only Z	0.781

Input sequence length plays a critical role in the tracker speed and accuracy. If input length is too short, the classifier has limited data in the time window where the object is detected, and might not be able to learn distinguishable pattern. On the other hand, if the input sequence is too long, increasing the sequence length beyond some limit would cause the gradient of LSTM network to start vanishing or exploding, which consequently leads to accuracy drop. In the Figure 6, we have shown the classification accuracy for different lengths of input sequence.



(a) (b)

Figure 4: Real-world experiment sample trajectories: (a) golf ball (b) wooden cube.

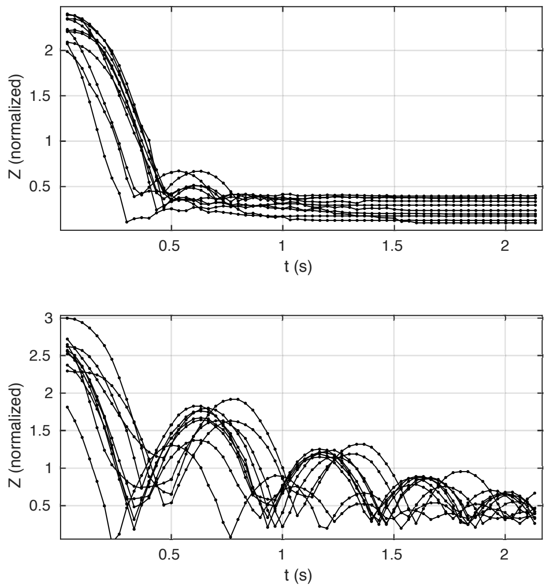


Figure 5: Samples of real-world trajectory data set for (top) wooden cube (bottom) golf ball

2.3 Experiment on real-world data

In this section, we leverage the trained network on synthetic trajectories to analyze the trajectory of real-world objects. One extreme example is chosen from each category: golf ball as an object with high bounciness, and wooden cube as an object with low bounciness. We throw them from different height with different initial velocities and record the video until they come to a full stop, Figure 4.

Additionally, the objects are marked with a distinct color with respect to the background to be able to use a simple color tracker since in this paper the focus is not on object tracking or detection. Lastly, when frames get blurred due to the fast motion of the object, missing part of the trajectory is reconstructed by a simple interpolation. The trajectories that are shorter than input sequence length get zero-padded. For each category, we collected 20 videos and trajectories are plotted in the Figure 5. We trained the RNN-LSTM network in the previous section except that it is trained on only z-channel of the trajectory samples. We got accuracy of 93% on the ball and 100% on the wood experiment.

3 Conclusion and Future work

In our preliminary experiments, we have found that motion pattern of an object provides strong cue on the object property. This has a potential application in autonomous driving technology which reduces the number of dangerous stops or maneuvers when on object suddenly appears in the front of the vehicle. Our preliminary experiments show promising results on synthetic and small set of real-world data. In the future, we are planning to collect more real-world.

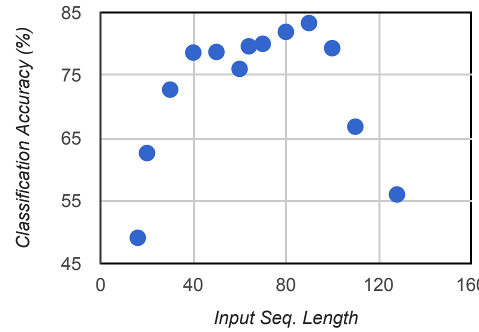


Figure 6: Classification accuracy versus trajectory length in LSTM-RNN

References

[1] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Alex Bewley, Vitor Guizilini, Fabio Ramos, and Ben Upcroft. Online self-supervised multi-instance segmentation of dynamic objects. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1296–1303. IEEE, 2014.

[3] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[4] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772, 2014.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[6] Sebastian Ramos. The dream of self-driving cars. *ICCV Tutorial on Computer Vision for Autonomous Driving*, 2015.

[7] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, pages 338–342, 2014.

[8] Rajiv Shah and Rob Romijnders. Applying deep learning to basketball trajectories. *arXiv preprint arXiv:1608.03793*, 2016.

[9] Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.