Learning to Walk: A Tale of Jackrabbots Path-finding Adventures
CS231N Winter 2015 Project Milestone

John Doherty
Stanford University
Stanford, California 94305
doherty1@stanford.edu

Kathy Sun
Stanford University
Stanford, California 94305
kathysun@stanford.edu

Abstract

The Jackrabbot is an autonomous delivery robot designed to share pedestrian walkways. In contrast to autonomous vehicles, this proposes the challenges of interacting safely with humans and bikers in a socially acceptable fashion. The path of the Jackrabbot then becomes nontrivial, as it cannot block pedestrian traffic or scare fellow travelers. The goal of our project is to develop a path-planning algorithm for the Jackrabbot that learns from observing the path-finding behaviors of the humans it will share walkways with, using both computer vision techniques and convolutional neural networks. This project will combine aspects of CS231A and CS231N with the Jackrabbot Research Project from Silvio Savarese's lab. We plan to use this for both class projects.

1. Introduction

Path-finding and obstacle avoidance has been an active area in research since the development of autonomous vehicles. The Jackrabbot, an autonomous delivery robot, faces the additional challenges of interacting safely with humans and bikers on pedestrian walkways in a socially acceptable fashion. We wish to allow the Jackrabbot to learn path-finding behaviors from being driven by humans, given what it is observing in its camera feed. We frame the problem as a classification problem in which the classes represent discretized orientation vectors the robot should take at each timestep based on the current image, or past images it has seen.

ALVINN, an autonomous land vehicle in a neural network, developed at Carnegie Mellon attempted to solve similar challenges but in a driving environment [1]. We wish to apply similar techniques of machine learning with respect to computer vision on a robot in a pedestrian environment and characterize the effectiveness of different algorithms. Not only will we compare several different convolutional neural network structures, but we will also compare our algorithm to supervised learning on the same dataset but using other computer vision techniques such as object detection and tracking as features. We will reference material from CS231A [3, 4, 5, 6] and CS231N [2].

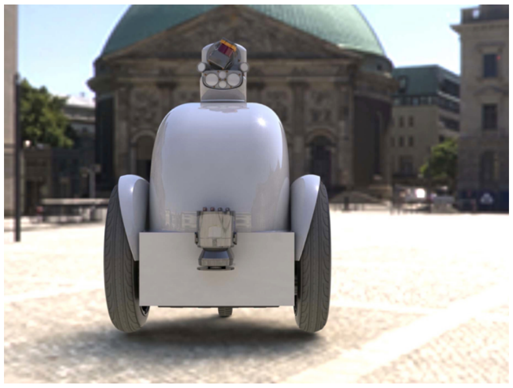
Figure 1. The Jackrabbot

2. Problem Statement

In order to train the Jackrabbot to navigate like a human, we will manually drive the robot through crowded environments. While driving the robot, we will record the images being captured by all of the cameras $(c_1 ... c_n)$ on the device as well as the manual control input to the robot. The manual control input at time t, θ_t will be recorded as the difference between the orientation at time t, ϕ_t and at time t-1, ϕ_{t-1}. To make this a classification problem, θ_t will represent a bin of angles. For example, if ϕ_t and ϕ_{t-1} are measured in degrees in [-180...180], we compute θ as:

$$\theta = \lfloor ((\phi_t - \phi_{t-1})/binsize) \rfloor$$

After driving the robot for some time, T, we will have recorded $c_1, t ... c_n, t, \theta_t$ for every t in 1..T. After recording sufficient data, we will build up a dataset of inputs and outputs that the robot can use to learn to replicate and generalize the navigation patterns recorded from the human driver. Each input will be a time sequenced set of images from all of the cameras $(c_{1,t}...c_{n,t}, c_{1,t-1}...c_{n,t-1}...c_{1,t-k}...c_{n,t-k})$, and the output will be θ_t.

Using this dataset, we will develop, train, and test a convolutional neural network to predict the desired orientation given a sequence of images from multiple cameras. As will be discussed in the next section, we will be using a slightly modified architecture to take these multiple images as input. We will try a different architectures and operate on different subsets of this data. For example, we can compare the effectiveness of using a single camera vs. multiple cameras or using images from varying sequence lengths (i.e. varying the value of k). How we will train these different models will be discussed more in the next section.

Finally, once we have trained and tested these models we need a good way to compare their performance. One measure will simply be the classification accuracy (the percent of the test samples for which we correctly predict θ). The problem with this metric is that it does not account for the correlation between the output classes. For example while two orientations may fall in different bins, they may actually be rather close to each other. We compute an evaluation metric on the average distance in bins between the correct θ and predicted θ. The final numeric metric that we will evaluate is the runtime of the network. For robotic applications, it is critical for the navigation system to work in real time.

While numeric metrics are important, it is equally critical to develop a set of qualitative metrics to debug mistakes being made by the classifier. To visualize these errors, we will create maps that compare the ground truth trajectory to the predicted trajectory. We will also visualize the weights for the first convolutional layer of the network. We can decompose the weights for our multi-image input to visualize filters that act over x, y, and t.

To evaluate the performance between different methods, we would also like to take benchmarks of the total training and test time of each method and plot them against the test accuracy.

3. Technical Approach

To solve this problem, we propose two main methods of visual processing as inputs into a linear classifier, SVM and Softmax, that will determine the direction the Jackraboot should turn at any given moment. We will also train a linear classifier, SVM and Softmax, on features obtained from computer vision techniques such as depth calculation, object detection, and motion tracking, as a baseline for the convolutional neural network based learning. After characterizing the performance of these two main methods, shallow vs. deep learning, variants of each method based on modifying the inputs and features will be explored to further optimize accuracy. Preprocessing will include zero-mean and normalization to [-1, 1]. The dataset will be randomly split into training, validation, and test sets into 70%, 20%, and 10% of the collected dataset to tune the hyperparameters and test the final performance.

3.1. Simple Computer Vision

The stereo camera will need to be calibrated to find the fundamental matrix. Once it is found, the following features will be extracted from the raw input image and put through an SVM and Softmax linear classifier.

- HOG features
- Depth map calculated from the stereo camera
- Motion map with tracked motion vectors in place of each pixel within the bounding box of detected objects

3.2. Convolutional Neural Network

We would like to implement and characterize all the following ideas, but in the interest of time and feasibility, we will start with the first two and pick which variations of the input to implement next based on the results we find.

- Transfer learning with single image inputs: Take pre-trained ConvNet on ImageNet with last fully-connected layer removed and retrain with collected Jackrabbot dataset.
- Custom architecture trained solely on the Jackrabbot dataset: Conv-Relu-Pool-Conv-Relu-Pool-FC with frames of 3 or more images as additional depth of the inputs to add the dimension of time/motion
- Concatenate images of extraneous cameras oriented in different directions like a panorama as inputs.
- Stack frames of stereo camera to account for physical depth in the inputs.
- Use the depth map and motion map features obtained above as inputs.

3.3. Intermediate/Preliminary Results

Since the Jackrabbot will not arrive until Tuesday, Feb 17, we have not been able to collect our dataset yet. Instead, to prepare for its arrival, we have set up OpenCV and Caffe and tested the library. We have also started implementing the data visualizations as well as converting the images into the proper format to stack multiple frames. Since we plan to pre-train our network on ImageNet, we have tested a couple networks and chosen GoogleNet to use for transfer learning.

References

[1] Pomerleau, Dean A., ”ALVINN, an autonomous land vehicle in a neural network”, Carnegie Mellon University, 1989. http://repository.cmu.edu/cgi/viewcontent.cgi?article=2874&context=compsci.

[2] Li, Fei-Fei and Karpathy, Andrej. ”CS231n: Convolutional Neural Networks for Visual Recognition”, 2015. http://cs231n.github.io.

[3] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach (2nd Edition)*. Prentice Hall, 2011.

[4] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. http://searchworks.stanford.edu/view/5628700.

[5] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011. http://searchworks.stanford.edu/view/9115177.

[6] D. Hoiem and S. Savarese. ”Representations and Techniques for 3D Object Recognition and Scene Interpretation”, *Synthesis lecture on Artificial Intelligence and Machine Learning*. Morgan Claypool Publishers, 2011. http://searchworks.stanford.edu/view/9379642.

[7] Gary Bradski, Adrian Kaehler. *Learning OpenCV*, O'Reilly Media, 2008. http://searchworks.stanford.edu/view/7734261.