Obtaining 3D Information from 2D Images Efficiently using Feature Extraction and Laser Dot Projection

Pratham Gandhi, Samuel Schuur

Bronx, New York

Abstract

A system to extract 3D information about a space was created that could be trained simply in a matter of minutes. The system, nicknamed "Constellation", was designed to be implemented in situations requiring rapidly updating computations and environmental awareness, to run on relatively inexpensive hardware, and to perform its tasks with limited human intervention. To achieve this, Constellation uses a combination of hardware and software components, focused around a set of artificial neural networks to extract features and a physical grid of points refracted from a single laser. A Python 3.6 implementation of the system was tested on two machines in a variety of environments, and produced an average of x times faster environment generation as other comparable algorithms or approaches to similar problems.

NYCSEF 2018

1. Introduction

The current most popular approach to creating a three dimensional informational estimate from a two dimensional image is triangulation, or binocular, stereo vision approach. Such approaches take several images of the same scene from different locations and angles in order to generate a spatial awareness. Such systems, however, are limited in practical applications in our increasingly evolving digital world, due to the limiting factor that they require multiple images, the acquisition of which is sometimes impractical. Additionally, they can sometimes be quite slow, in that they need to first identify common features in the different photos, find out the positional relation of those features, and then only begin to start seeing the greater picture of the full space. The purpose of Constellation was to solve both of these problems with a generally applicable approach which can be implemented in a variety of different use-cases.

We chose to approach this problem first through the somewhat traditional technique of using artificial neural networks, mathematical functions modeled after nature which specialize in taking in large amounts of input to produce outputs, to identify features in a two dimensional image. Neural networks are commonly used in situations which are challenging to navigate via traditional "if-else" programming, which make them perfect for the fuzzy problems of feature extraction from images. Where our system differs is in its use of a grid of refracted laser points into the scene to judge the distances and orientations of various objects in relation to the camera. This allows superior environment generation in a shorter amount of time, due to the omission of the multiple-image stereo aspect. This approach and its implemented system was named Constellation for the star-like appearance of the laser dot grid it relies on.

2. Background

2.1. Neural Networks

A human brain is composed of billions of neurons, connected and excitable cells which allow humans to be intelligent. Each neuron's dendrites receive signals from other neurons. If the these electrical signals exceed a threshold, the neuron fires, thereby propagating a voltage out of its synapses and on to other neurons. Together, these neurons form an extensive network, where an immense set of sensory input i processed alongside expected outputs. The purpose of artificial neural networks is to simulate this same network of

interconnected neurons using mathematical properties to emulate electrical signals and approximate complex functions

2.1.1. Artificial Neurons

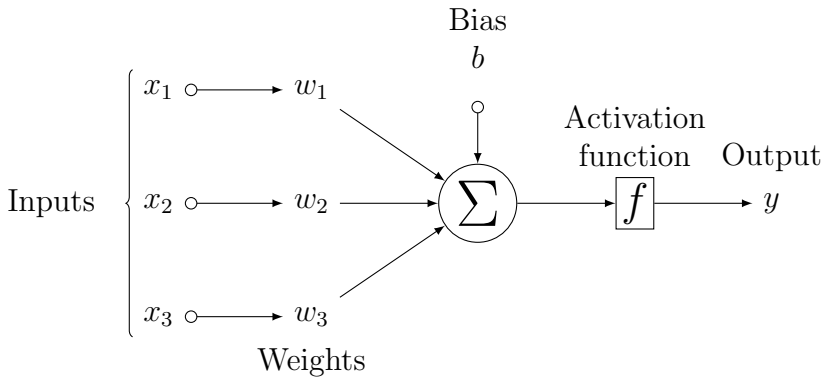

Figure 1: Artificial Neuron Diagram

An artificial neuron is an approximate mathematical model of a biological neuron, and is intended to fulfill similar purposes. Each artificial neuron has multiple inputs from other neurons. Along with the inputs, each neuron has a weight for each input, a bias term, and activation function, and only one output. The output of one neuron travels to other neurons and becomes of their inputs. The output of a neuron is expressed mathematically for n inputs x and weights w, a bias term b, and activation function $f(x)$ as follows:

$$activation = \sum_{i=0}^{n} x_i w_i + b$$

$$output = f(activation)$$

Mentioned above, weights are positive or negative numbers which scale the effect or relevance of their accompanying inputs on a neuron's single output. Besides the weights, another important component of a neuron is the activation function, whose primary is to transform the output of a neuron. Common activation functions include a sigmoid curve, a hyperbolic tangent, a binary step function, and a rectifier function.The sigmoid activation and hyperbolic tangent activation functions are often used for general purpose neural networks due to their gradient nature, fixed range, and the fact that

they are easily differentiated. Finally, the bias term is a number added to the summation of weighted inputs, which allows us to make transformations on the domain of the activation function.

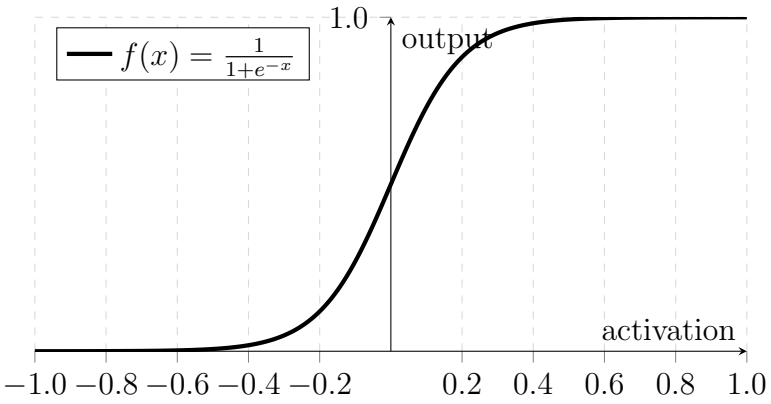

Figure 2: Sigmoid Function Graph

2.1.2. Feed-forward Neural Network

Constellation utilizes a feed-forward neural network, a collection of neurons organized into layers, as classifying mechanism for detection various features in a image. In a feed-forward neural network, the output of each neuron is the input of each neuron in the next layer. In this manner, the original inputs of the network are "fed forward" to new neurons in new layers, passing through a number of hidden layers in the middle, and finishing at the output layer.The outputs of the neurons in the final layer (output layer) are the outputs of the whole network and are used to judge the conclusion the network made.

2.1.3. Backpropagation

Supervised learning is the process of adjusting the weights of each neuron in a neural network in an effort to reach target outputs from a specific set of inputs, effectively "fine tuning" the neural network to make it as correct as possible. One of the most common methods of adjusting a neural network's weights is called backpropagation, first developed in Werbos in 1974. In order to train neural networks through backpropagation (BP), we use examples. Each set of example inputs, known as training sets, are connected to a target output, which the system is aware of, and together the

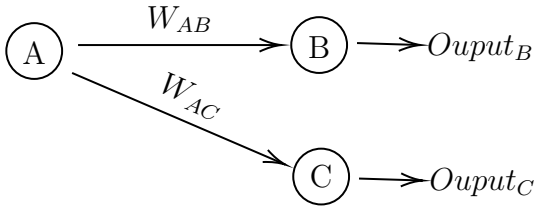

Figure 3: Single Branch of a Backpropagation Neural Network

two parts form a training pair. The first step of training a BP network is to pass the training set through the neural network with randomly generated weights. The error, represented as δ, for an output neuron is the difference between the actual output and the target output for that training set. Consider the abbreviated neural network illustrated in Figure 4. Assume this network has only an input layer consisting of an unspecified number of input neurons with unspecified outputs, a hidden layer containing only Neuron A, and an output layer consisting of Neurons B and C. Obviously, W_{AB} and W_{AC} are the weights of the edges connecting the neurons. We complete the backpropagation action by modifying weight W_{AB}, replacing it with W_{AB}^{+}, as follows:

$$\delta_B = Target_B - Output_B$$

$$W_{AB}^{+} = W_{AB} + (\delta_B \times Output_A)$$

This approach is what is also used to adjust the weight of Neuron C from the example. This approach, however, cannot calculate more appropriate weights for neurons whose outputs feed into the inputs for another neuron, referred to as hidden layer neurons, such as neuron A, because there is no predefined and confirmed target value. Therefore, we must calculate the error of the output of A indirectly by backpropagating from the output layer neurons, whose errors have already been calculated. For A, this could be shown as:

$$\delta_A = (W_{AB} \times \delta_B) + (W_{AC} \times \delta_C)$$

Once we have established the error for Neuron A, its updated weights can be calculated with the same approach used for B and C. This pattern is used throughout the full neural network to calculate the adjusted weights for hidden layers in order to converge upon the network's target output.

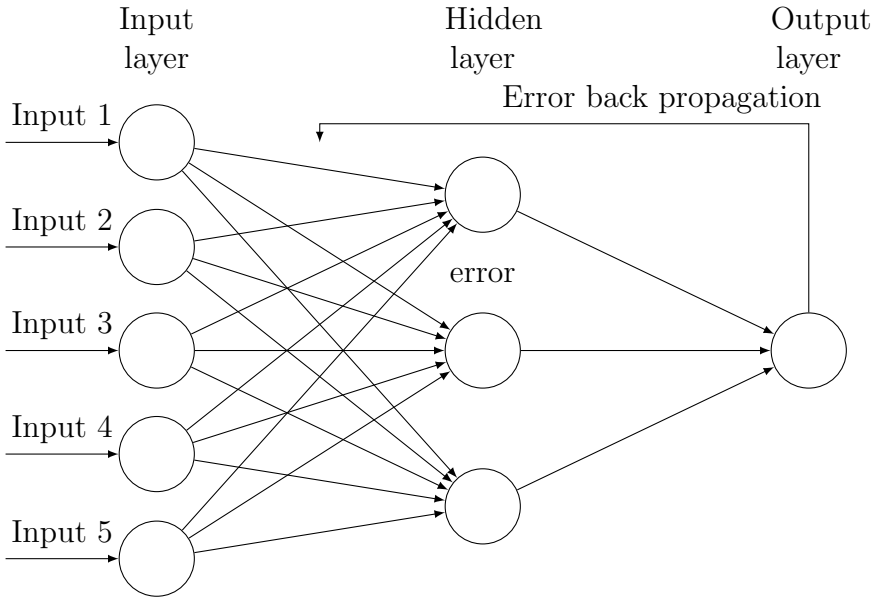

Figure 4: A Feed-Forward Neural Network