Object Depth Estimation Using Stereo Vision and Monocular Ranging

Rohan Sharma
Electrical and Computer Engineering
M. Eng '11
rs423@cornell.edu

David Skiff
Computer Science
B.S. '12
des259@cornell.edu

Abstract

Object Depth Estimation (ODE) is a common problem faced in Computer Vision, and it's often solved using either Monocular or Stereo ODE algorithms. Since both use different tactics and have estimation problems at different distance ranges, it may be beneficial to use them in tandem and use Machine Learning techniques to combine the results in order to decrease errors.

1. Introduction

1.1. Motivation

The need to identify object locations relative to a fixed point, based on single or multiple 2-dimensional images is a common problem in Computer Vision, and comes up often in Robotics. This has led to an entire area of study, referred to as Object Depth Estimation (ODE), to form directly around solving this issue. Thus, there has been a considerable amount of research into both single (monocular) and multiple (stereo) image ODE algorithms.

Monocular algorithms typically parse an image into objects and determine depth based on the location of the object in the image. A big issue to this approach is the susceptibility to scaling errors. These can be corrected for in some cases, based on various features of the object, but it remains as a significant source of error. Stereo algorithms tend to focus on the differences between the views of an object based on angle. However, this tends to give inaccurate results on objects at a distance. Since both algorithms attempt to solve the same problem using different methods, but have individual drawbacks, it seems intuitive that a combination has the possibility of decreasing errors.

1.2. Approach Introduction

The new solution suggested by this paper is to use supervised machine learning techniques, such as Linear Regression, on the results of using both algorithms independently. For the purposes of training and evaluation, a large data set of images from two separate angles paired with a reading from a Hokuyo Laser Range Finder is used. Multiple Machine Learning techniques are compared on the monocular and stereo ODE algorithm outputs, but this paper focuses on a Markov Random Field model.

2. Approach

There are several important components to this project that must be incrementally completed to ensure successful implementation of object depth estimation. These sections are enumerated as follows:

2.1. Data set collection and Hardware setup

Before implementing any algorithms, we had to successfully complete taking a large set of data so that we have a database of stereo images and ground truth from laser range data. However, taking an accurate set of stereo images proved more of a challenge than anticipated. We had to get 2 identical cameras with similar intrinsic properties (such as focal length, skew, lens curvature etc.). These had to be mounted together with a predefined physical *vertical* separation along with the Hokuyo Laser Range finder and had to be axis aligned to minimize skew between the cameras and to align image pixels with laser range (this constitutes as manual alignment). The resulting data set contains over a hundred images of various rectangular and circular objects in different orientations, and locations in the field of view. Most of the data was collected in one session to reduce the amount of error introduced due to removal and repositioning of the camera between sessions.

2.2. Image Rectification, Laser Alignment and Edge Extraction

After taking the data sets, the 2 stereo images contained in each data set had to be rectified such that there is 1-1 correspondence between the columns. Since the cameras were mounted with a vertical separation, each column of the images was aligned with each other such that an accurate disparity calculation could be performed based on the base edges of objects observed in each image. Therefore, the extreme left and right ends of the 2 images were modified such that each vertical column in the 2 images matches each other. Since there was only a horizontal variation in the image (extra columns at the end), the end columns of the 2 images had to be compared against one another to find a close match and any additional columns were removed from the image. This rectification was constant throughout the entire image database (that was taken in one session).

As a second step, the laser range data had to be formatted and aligned with the rectified stereo images to get accurate ground truth information about each column of the images. This was done by manually aligning different parts of the laser range (starting from the center) to see if the laser data matched with the obstacles viewed (was performed once). Once the laser angles were aligned with the image edges, *linear interpolation* had to be performed to get range estimations for each column. S

Every time we take data sets with a new vertical separation, we would have to change the image alignment code as we would never be able to get the same angle orientation of the 2 cameras and laser. The results section contains 2 images that represent 1 set of images aligned with the laser data.

2.3. Stereo Object Depth Estimation Algorithm

Stereo vision is a technique for inferring depth of an object using a combination of 2 or more cameras that are a certain distance apart from each other. For our implementation, we have a combination of 2 cameras that are a certain distance ***d*** apart from each other (physical separation). Thus, if we assume that a certain section of an image has a disparity ***x*** between the 2 images, the depth of the object can be calculated as follows:

$$Z = \frac{d.f}{Xb - Xt} = \frac{d.f}{x}$$

Where
d = Physical separation between the camera
f = Focal length of the camera
x = Disparity of the image patch in the stereo images (ground edges in our case)
Z = Actual depth estimation

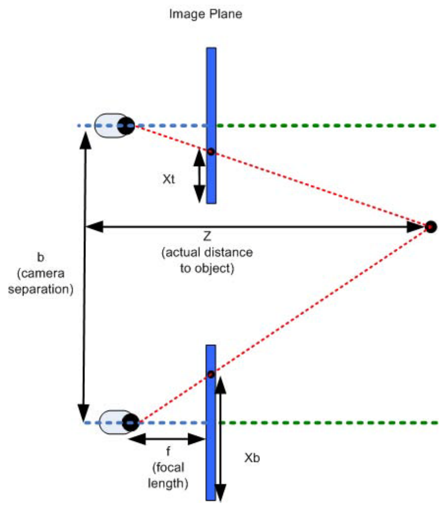


Figure1: Stereo vision setup illustration

The focal length of the camera was converted to pixel representation using the following formula:

$$Focal\ Length_{(pixels)} = Image_{width(pixels)} * \left(\frac{focal\ length_{(mm)}}{lens_{width(mm)}}\right)$$

To run this algorithm on the images, several pre-processing image steps had to be completed. We used a combination of filters to extract the edges from the picture. First, we converted the image into a grayscale image as color information is not required for this algorithm. We then ran the grayscale images through a 7x7 Gaussian blur filter to smooth out any edges created due to the carpet texture. This was implemented so that the first edge that we observe from the bottom of the images is the ground edge of the object and not random edges due to the texture of the carpet. This image was then run through a Canny edge detector with a threshold value of [low high] = [0.05 0.25] to give fine edges for the bottom for the object.

These images (rectified and edge extracted) are then compared for the bottom most edge in each image column/segment and the disparity in the pixel positions in between the 2 images is found. These values are then run through the formula to get an estimate of the depth of the object's edge. Comparison of this range data with the laser range information is provided in the results section.

2.4. Monocular Ground Ranging Object Depth Estimation Algorithm

The monocular ground ranging algorithm tries to find the depth using just the pixel information of the object edge. This algorithm is based on the theory that objects that are far away in the image are present at a higher pixel level (from the bottom) in the image and appear closer to the horizon. Thus, this algorithm tries to estimate the distance to an object based on the number of pixels from the bottom of the image to the object edge.

Using a similar concept from the previous algorithm, the colored image is converted into a grayscale image and a Gaussian filter will be convolved with it to blur out the edges caused by any carpet texturing. Following this, a canny edge detector is run to find the bottom most edge of the object and the pixel distance to the edge is stored. Using the ground truth information from the laser range data, a linear regression algorithm (using the least mean squares algorithm) is run on the top and bottom image sets to find the weight parameters for the 2 sets. These learned parameters are used along with the pixel information to estimate depth of the object's edge. The linear regression algorithm and feature set used is as follows:

$$\theta = (A'A)^{-1} * (A'y)$$

Where

Θ = Weights for linear regression (of size 3)
A = feature matrix
y = laser data vector

The feature vector is created using the 3 features as given below:

$$A = [x \;\; \log(x) \;\; x^2]$$

Here, each x is simply a very large column vector that consists of the pixel position for each column of the image (in the training set). This vector consists of all the columns of all images (#sets * 626, where 626 is number of columns after rectification).

2.5. Machine Learning Algorithm

Finally, using the stereo vision and monocular ranging algorithms on the 2 images, we have 3 range values that estimate the depth of the object and this makes up our feature set for learning. The following figure illustrates our model and represents different modules used to get single range estimation by combining the algorithms explained above,

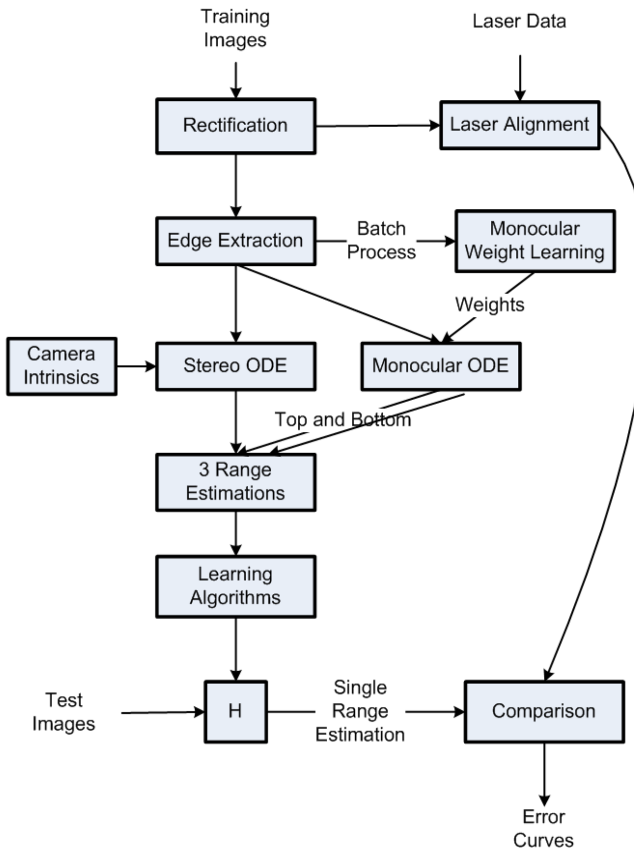


Figure2: Learning and Error analysis Model

The range data will be split into different segments based on the distance they are trying to estimate (that is 1 group for closer ranges - less than 1 meter, one for ranges between 1-3 meters and one group for ranges above 3 meters). We also have laser range data that represents the ground truth distance for the objects. Thus, the feature set and laser range data together represent the training set and test based on how we split up the data for testing.

Following this partition of data, a number of different Machine Learning algorithms were implemented in Matlab [3] in order to ascertain an optimal method. The following were tried:

1. Linear Regression – This is basic linear regression where each of the 3 range data are used as input features to the regression model and they are compared against laser data using least squares method to find out the required weights. These weights are then used against test images to calculate the range.
2. Locally Weighted Linear Regression – This is similar to Linear Regression. It merely allows neighboring points to have a greater effect on the algorithm than other points. The linear regression algorithm is similar to the one described in the Monocular ranging section. The input vector consists of the 3 ranges instead of features for the same pixel position. Tricubic weighting with an r-value of 50 was used [4].
3. KNN search – Use K nearest neighbors to find the range estimation from previously stored data. This might not be the optimal algorithm to run as we would need to have a very large number of data sets so that we cover the entire distance that we want to measure.
4. Markov Random Field model – A Markov Random Field (MRF) model that conditions each point by its neighbors as well as the results of the monocular and stereo ODE algorithms. (See 2.6)

To determine the accuracy of the algorithms, K-fold cross validation (10-fold) was used to determine performance by testing them against the generated training set. Thus, based on the performance of these learning algorithms, it is possible that different partitions of the range use different models to estimate the range of the object.

2.6 Markov Random Field model

One of the Machine Learning techniques revolves around a Markov Random Field (MRF). There was some indecision over the distribution to use, but a jointly Gaussian model was opted for. Similar to [Saxena et al, 2005], the outputs are conditioned on their neighbors as well as the input.

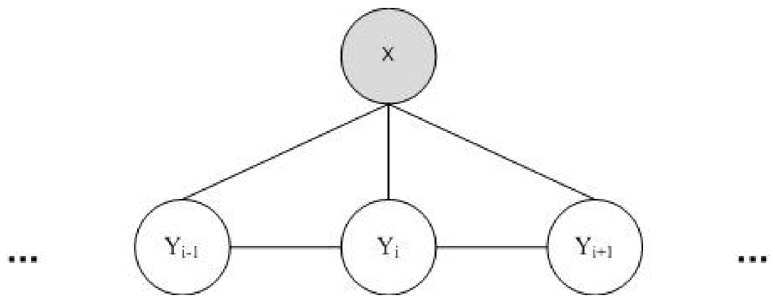


Figure3: Markov Random Field Model

We get the following probability function:

$$p(y|x;\ \theta) = \prod_{i=1}^{n} \exp\left(-\frac{(y_i - \theta^T \quad _i)^2}{2\sigma_1^2}\right) \prod_{j \in N(i)} \frac{(y_i - y_j)^2}{2\sigma_2^2}$$

Where N(i) is the set of the neighbors of i. θ can be found merely by using linear least squares, and we can solve for a closed form solution that maximizes the likelihood for y, which gives us:

$$y = argmin_y \sum_{i=1}^{n} \frac{(y_i - \theta^T X_i)^2}{2\sigma_1^2} + \sum_{j \in N(i)} \frac{(y_i - y_j)^2}{2\sigma_2^2}$$

This is a simple quadratic program, and we solve using Matlab's quadratic program solver [3].

3. Experiments and Results

3.1. Overview

The suggested ODE algorithm relies on Supervised Learning techniques, so a significantly large training data set is required, where each "set" contains two images taken from separate angles, and the readings from a Hokuyo Laser Range Finder, which is used to provide the ground truth. In the interest of getting decent results, the cameras have to have a decent resolution, roughly of the same focal length, and their positioning relative to each other must remain static. The cameras are mounted one atop the other with the range finder in between, such that all of them are aligned vertically. Finding cameras with a similar focal length, that didn't wobble, and gave decent images, was an issue at first, but two Axis M1011 cameras were used for our first data set acquisition, and they gave excellent results. We don't have permanent access to the mounting apparatus we've been using for the training sets, so we've been forced to reposition the cameras every time. Any new data sets will have the cameras located differently and thus will need to be trained separately. However, the various training sets still allow us to test various Machine Learning techniques on the problem, and evaluate their performance. The images include various rectangular and cylindrical objects placed randomly around a room. The object sizes and locations are restricted by the Range Finder. Thus, all objects were relatively large (greater than 6 inches wide and 3 feet tall), and were no more than 13 feet away. T he following are some physical characteristics for the setup:

Camera Used	– Axis M1011 Ethernet Camera
Laser Used	– Hokuyo URG-04LX Laser
Data Sets Taken	– 50 sets

Feature	Value
Axis M1011 Camera Focal Length	4.2 mm
Camera Lens Width	6.35 mm
Camera Aperture	2.0
Camera Angle of View	46.8035 degrees
Camera Vertical Separation	20.5 cm
Image Resolution	640x480
Max test object placement	4 m or ~13 feet
Laser Angular Resolution	0.36 degrees
Laser Scan Angle	240 degrees
Laser Detection Range	0.02 to 4m

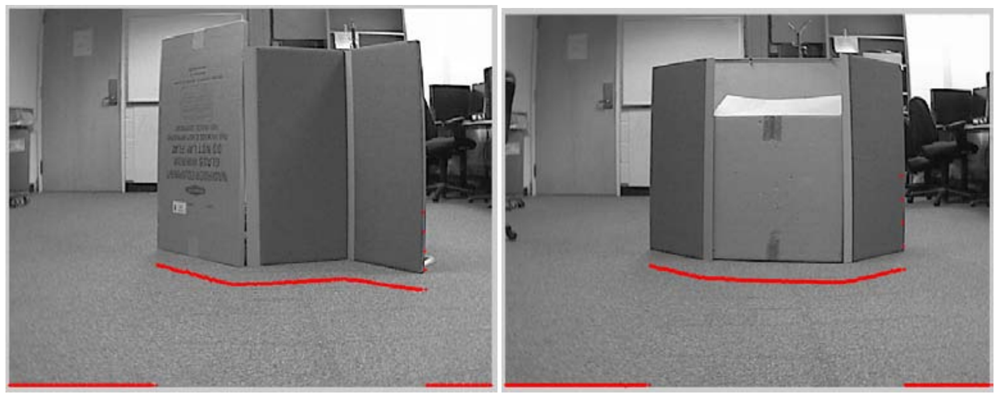
These statistics were used to place the test object and position the camera and laser units for taking data sets. The camera intrinsic mentioned in the table above were also used in range detection as well as laser alignment with the image view.

The data set had the following properties:

Feature	Value
Test Data Mean	1.979417e+000
Test Data Variance	2.696225e+000
Lower Mono Data Mean	2.737936e+000
Lower Mono Data Variance	2.402528e-001
Upper Mono Data Mean	2.327267e+000
Upper Mono Data Variance	3.656286e-001
Stereo Data Mean	2.804435e+000
Stereo Data Variance	3.160992e+000

3.2. Image alignment results

The following images were created after aligning the images with the laser range data. As can be seen, they align perfectly with the obstacles and thus, this combination is used in algorithms to determine ground truth data for a particular pixel in the image (of course, some interpolation between laser data is done to get range data for each pixel).



3.3 Test set poisoning

The initial comparison method was to run the various algorithms using k-fold cross-validation on the columns of the images. However, the columns are far from independent, and thus this can cause poisoning of the test set during cross-validation. This is easily fixed by performing folds over the separate images, rather than the individual columns.

3.4 Results

The weights for the monocular ranging module were learnt and used to produce 3 ranges (along with stereo). These 3 ranges were compared with the laser data in the comparison module to produce the following mean square chart:

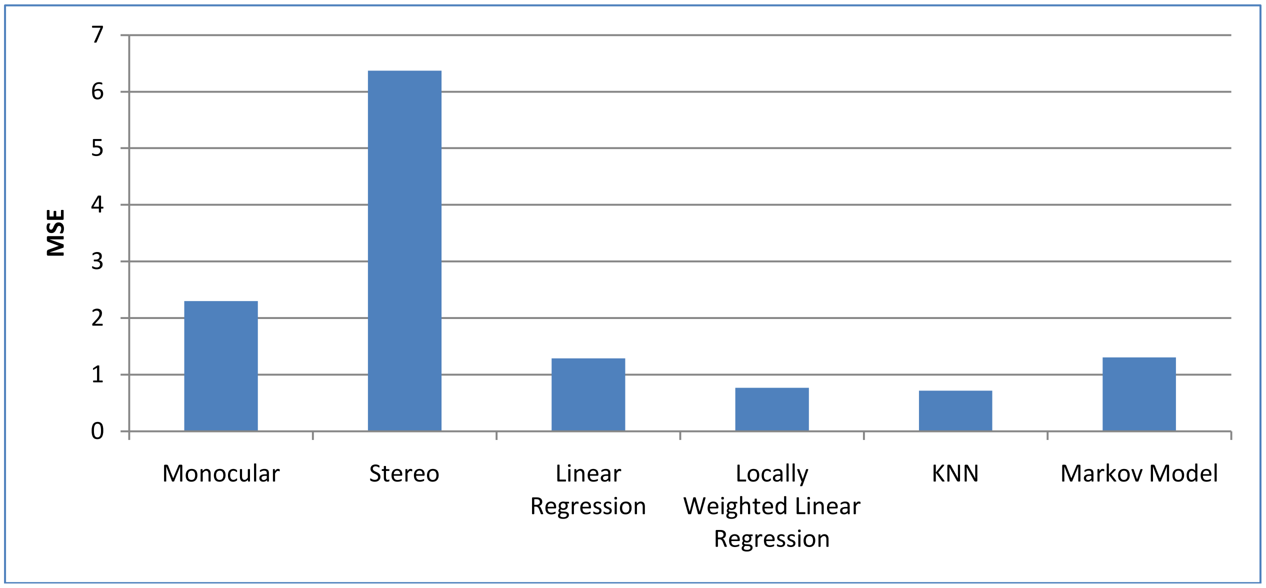


Chart: Mean Square Error for different models

As can be seen, neither of the individual ranges were able to correctly predict the value of laser range and thus gave large mean square errors. When these ranges were combined in a single learning model, we achieved much better results in comparison. As can be seen, the locally weighted linear regression model as well as KNN provided significant decreases in errors in comparison with Linear Regression or the Markov Model. Notably, all of the combination Machine Learning techniques performed better than merely the stereo or monocular algorithms on their own.

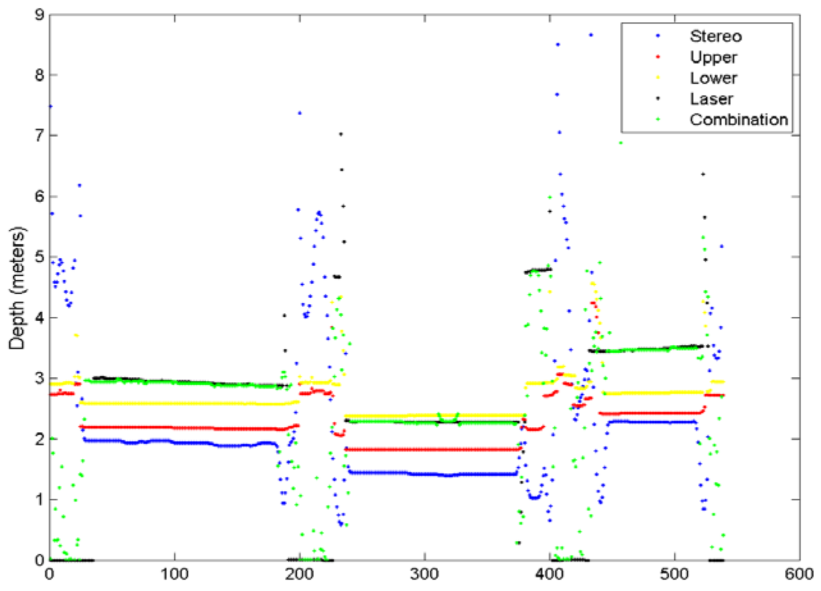


Figure: Final calculated range compared with other ranges for locally weighed linear regression. Calculated range overlaps laser range value.

The figure above is the result we get for one test image. As can be seen, the computed range estimate overlaps very well with the actual laser range data giving a very low MSE error. Consequently, the locally weighted regression model was the strongest for combining the 2 ranging techniques (Stereo vision and Monocular Ground Ranging) to give a single range estimate for the objects.

4.1 <u>Future Work</u>

4.1 Probabilistic Model

Based on the results of [Saxena et al, 2005], it seems that a Laplacian model for the neighbor constraints may provide better results, and is something we're considering implementing. Another idea is to add a latent variable S that indicates whether or not to disable either the stereo or monocular inputs (similar to [Li-Jia Li et al, 2009]), since both are invalid under certain circumstances (e.g. object is too far, too close, only visible in one camera, etc.) However, we would no longer to be able to find a closed form solution to maximize the likelihood of our model, and would have to change our learning methods entirely.

4.2. Beyond

There are number of interesting areas we can use this system on once it's completed. One interesting problem is that of object tracking. The idea is to track an object's movement throughout an image using depth information (by keeping a track of the environment and sensing the ranges that change over time). Another idea is to use it to learn a route based on the surrounding objects, or use it in conjunction with 3-D reconstruction algorithms to find a specific location where an image was taken.

References

[1] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. *Learning Depth from Single Monocular Image*. 2005

[2] Li-Jia Li , Richard Socher, and Li Fei-Fei. *Towards Total Scene Understanding: Classification, Annotation and Segmentation in an Automatic Framework*. 2009

[3] The Mathworks. *Matlab version 7.9*. 2009

[4] William Cleveland. *Robust Locally Weighted Regression and Smoothing Scatterplots*. 1979

[5] Christian Plagemann, Felix Endres, J¨urgen Hess, Cyrill Stachniss, Wolfram Burgard. *Monocular Range Sensing: A Non-Parametric Learning Approach.* 2008