FindDepth

CS 293 Final Project Demo

Vipul Venkataraman - 110050084
C. Yeshwanth - 110050083
Outline <for a total of 30 mins>

- Aim of project (1 mins)
- Demo (5 mins)
- Teamwork Details (0.5 min)
- Design Details – Algorithm (5 mins)
- Design Details – Implementation (8 mins)
- Viva (9 mins)
- Transition time to next team (2 mins)
Aim of the project

- The aim of the project is to give an estimate of the depth-map for an image that it receives from the user.
- It produces this depth-map by extracting information about texture, color, intensity and the edges in the image.
- It then computes a summary feature vector of all of these features and fits a linear regression model to fit the data.
Aim of the project

- The training set was the same as the dataset used for the Make3D project.
- While the algorithm is a gross simplification of how depth estimation actually works in humans, we find that the model actually learns to effectively use texture and intensity cues to distinguish between the depths of various patches in the image.
Demo

- The program begins by reading the image and dividing the picture into patches.
- It then proceeds to compute the relevant feature vector for each patch that it has split the image into.
- The parameters obtained from the training are used to estimate the depth of each patch in the image.
- The output of the program is shown to the user in the form of a greyscale image with the darker colours representing patches closer to the camera and the lighter colours representing more distant patches.
Teamwork Details

- Vipul Venkataraman: Focused on researching and developing the algorithm for our project. Understood how filters work. Wrote the code for the implementation of our project in C++. Implemented Convolution. Wrote the Feature Extraction code in C++ that is used to obtain the final depth map using the parameters obtained from training.

- C.Yeshwanth: Focused on the training of the linear regression model for the system. Research and Implemented the training code in Octave to calculate the values of the parameters which define the model. Implemented the Feature Extraction code in Octave for the training. I also spent some time understanding the usages of filters.

<table>
<thead>
<tr>
<th>Overall Contribution by Team</th>
<th>Overall Contribution by Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member 1</td>
<td>Member 2</td>
</tr>
<tr>
<td>50 %</td>
<td>50 %</td>
</tr>
</tbody>
</table>
Algorithm Design

- The problem was to estimate the depth of various objects in an image which has various applications in robotics and augmented reality gaming.
- We solved the problem by extracting relevant features from the image for different patches and by fitting a linear model to it.
- The training of the model was done using a gradient descent computation.
Algorithm Design

- **Feature Extraction**
  - We begin by convolving the intensity image with the 9 Laws' masks which detect the texture properties of the image and the 6 Nevatia Babu Filters which are used for Edge Detection.
  - We then split the image into patches as the features of a single pixel cannot be used to predict its depth.
  - We then collect the features for a single patch by accumulating (Summing up over the absolute value of the output of the filters multiplied by their intensity).
Algorithm Design

- Features of single patch are alone not enough for estimating it's depth
- Therefore, we also add to the feature vector of a patch, the features of the patches surrounding it.
- We also take into account the more global features of an image by taking into account the summary of much larger patches (9 times and 27 times larger than the patch concerned) which will include the more global features of the patch and use this to estimate it's depth
Algorithm Design

- Training
  - For training, we fit a linear regression model that we fit using gradient descent, with mini-batch learning, feature scaling and adaptive learning rate
  - We used a squared error cost function for the learning
  - We used a different set of parameters for each row of the image
  - This had to be done because patches of the image at different heights behave differently and cannot be represented by a common set of parameters
Algorithm Design

- For example, a patch of blue at the top of the image can represent the sky which is very far away while a patch of blue at the bottom could represent a river or an ocean in close proximity to the person taking the photograph.
- Gradient descent is an optimization technique which is used for finding the optimum set of parameters for the task at hand.
- It does this by always moving the weights in the direction opposite to the direction that the gradient specifies.
- This has the nice property that all the weights are always changed in the direction that brings about the maximum reduction of the cost function.
Algorithm Design

- Since the learning rates play a big part in the behaviour of a gradient descent system, we used a clever trick which automatically reduces the learning rate when it finds that the gradient is beginning to oscillate around its optimum value.
- We increase the learning rate when the gradient for a particular weight has been consistently in one direction and reduce it when the gradient with respect to that weight oscillates.
- We run the algorithm iteratively until the weights converge to their optimum value.
Gradient Descent

This is the path followed by the optimizer to reach the global minimum \([0.22777 \quad -1.6257]\).
Gradient Descent
Algorithm Design

- Implementation
  - The program reads an image and computes the feature vector as described in the earlier part of the presentation.
  - It then proceeds to use the parameters obtained from the training to estimate the depth for every patch in the model.
  - The viewer is then shown the depthmap of his image which is shown in the form of a greyscale version of the original image where the lighter shades represent patches farther from the camera and the darker shades represent patches closer to the camera.
Sample Output
Sample Output
Brief Conclusion

- Our program has successfully learnt a set of parameters for predicting the depth-map of an image by extracting certain useful features from the image.
- Features extracted include texture cues, edge cues and other global and local cues about a certain patch.
Thank You – Questions?

Bring it on!
Back Up Slides
Back Up Slides
Back Up Slides
Back Up Slides