Deep Learning-based Obstacle Detection and Depth Estimation

Yi-Yu Hsieh\textsuperscript{1}, Wei-Yu Lin\textsuperscript{1}, Dong-Lin Li\textsuperscript{2}, and Jen-Hui Chuang\textsuperscript{3}

\textsuperscript{1}Institute of Computer Science and Engineering, \textsuperscript{2}Computer Vision Research Center, \textsuperscript{3}Department of Computer Science
National Chiao Tung University, Taiwan

Sept. 24, 2019
Outline

- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based On YOLO
- Experimental Results
- Conclusion
Outline

■ Introduction

■ YOLO – a CNN for Deep Learning

■ The Proposed Depth Prediction Based On YOLO

■ Experimental Results

■ Conclusion
Introduction – Motivation

- Obstacle detection is a crucial issue in robotics and autonomous driving systems
- Because of perspective projection, obstacle depth information is lost
Introduction – Related Works

- To achieve **obstacle avoidance**, we need
  - Object detection
  - Depth prediction

- **Object detection** methods
  - Traditional methods
    - HOG + SVM
    - DPM
  - Deep learning-based methods
    - Fast / Faster R-CNN
    - SSD / R-FCN / FPN / FRCN

- **Depth prediction** methods
  - Traditional methods
    - Stereo matching
  - Deep learning-based methods (monocular)
    - FCRN
    - Godard *et al.*, CVPR, 2017
    - Kuznietsov *et al.*, CVPR, 2017
  → Too slow (10fps↓ on TITAN)

- Self-designed **real-time** architecture using YOLOv2/3
# Introduction – Related Works

<table>
<thead>
<tr>
<th>Method/CNN model</th>
<th>COCO test mAP</th>
<th>Speed (fps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast R-CNN</td>
<td>39.3</td>
<td>0.5</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>41.5</td>
<td>7.0</td>
</tr>
<tr>
<td>R-FCN</td>
<td>51.9</td>
<td>12</td>
</tr>
<tr>
<td>RetinaNet</td>
<td>57.5</td>
<td>5.1</td>
</tr>
<tr>
<td>FPN FRCN</td>
<td>59.1</td>
<td>5.8</td>
</tr>
<tr>
<td>SSD 300x300</td>
<td>41.2</td>
<td>46</td>
</tr>
<tr>
<td>YOLOv2 416x416</td>
<td>44.0</td>
<td>67</td>
</tr>
<tr>
<td>SSD 500x500</td>
<td>46.5</td>
<td>19</td>
</tr>
<tr>
<td>YOLOv2 608x608</td>
<td>48.1</td>
<td>40</td>
</tr>
<tr>
<td>YOLOv3 416x416</td>
<td>55.3</td>
<td>35</td>
</tr>
<tr>
<td>YOLOv3 608x608</td>
<td>57.9</td>
<td>20</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based On YOLO
- Experimental Results
- Conclusion
YOLO – a CNN for Deep Learning
– YOLO: Main Concept

A. Splits input image into 13x13 cells
B. Predicts 5 bounding boxes for each cell
C. Final detections → thresholding and NMS (Non-maximum Suppression)
YOLO – a CNN for Deep Learning – YOLOv3

- **Architecture design**
  - **Darknet-53** → learns better, computes faster
    - 53 convolution layers and 5 stride-2 convolution layers
  - No max-pooling → use **stride-2 convolution**
    - Preserve more information, each pixel is responsible for layer output
  - Up-sample layer → multi-scale **prediction (3 scales)**
    - To find objects at different sizes

- Stride-2 convolution reduces the dimensionality of each feature map
  - Use convolutions to produce output features
    - Each feature has contribution to output features
YOLO – a CNN for Deep Learning
– YOLOv3: Up-sample Layer

- Up-sample increases the dimensionality of each feature map
  - Larger feature map → detection of smaller objects
  - Concatenation of object information → better detection result
YOLO – a CNN for Deep Learning
– YOLOv3: Output Layers

- Output layer feature map size: $13 \times 13$, $26 \times 26$, $52 \times 52$
- For each scale
  - Each cell predicts 3 bounding boxes
  - Each bounding box needs 85 parameters
    - $x, y, w, h, confidence$
    - $class_1, class_2, \ldots, class_{80}$ (COCO has 80 classes)
  - The depth of output layer is $3 \times 85 = 255$
YOLO – a CNN for Deep Learning
– YOLOv3: Output Layers

- Output layer feature map size: $13 \times 13$, $26 \times 26$, $52 \times 52$
- For each scale
  - Each cell predicts 3 bounding boxes
  - Each bounding box needs 85 parameters
    - $x, y, w, h, \text{confidence}$
    - class1, class2, ..., class80 (COCO has 80 classes)
  - The depth of output layer is $3 \times 85 = 255$
<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Feature Map Number</th>
<th>Filter Size</th>
<th>Filter Stride</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional</td>
<td>512</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$13 \times 13$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>1024</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$13 \times 13$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>512</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$13 \times 13$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>1024</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$13 \times 13$</td>
</tr>
<tr>
<td>Convolutional (A1)</td>
<td>512</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$13 \times 13$</td>
</tr>
<tr>
<td>convolutional</td>
<td>1024</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$13 \times 13$</td>
</tr>
<tr>
<td><strong>Prediction 1 (scale 1)</strong></td>
<td><strong>255</strong></td>
<td>$1 \times 1$</td>
<td>1</td>
<td><strong>13 \times 13</strong></td>
</tr>
<tr>
<td>A1</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$13 \times 13$</td>
</tr>
<tr>
<td><strong>Up-sample</strong></td>
<td>256</td>
<td>$2 \times$</td>
<td>2</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td>Convolutional+A2</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>512</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>512</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td>Convolutional (B1)</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>512</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td><strong>Prediction 2 (scale 2)</strong></td>
<td><strong>255</strong></td>
<td>$1 \times 1$</td>
<td>1</td>
<td><strong>26 \times 26</strong></td>
</tr>
<tr>
<td>B1</td>
<td>128</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$26 \times 26$</td>
</tr>
<tr>
<td><strong>Up-sample</strong></td>
<td>128</td>
<td>$2 \times$</td>
<td>2</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td>Convolutional+B2</td>
<td>128</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>128</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>128</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>256</td>
<td>$3 \times 3$</td>
<td>1</td>
<td>$52 \times 52$</td>
</tr>
<tr>
<td><strong>Prediction 3 (scale 3)</strong></td>
<td><strong>255</strong></td>
<td>$1 \times 1$</td>
<td>1</td>
<td><strong>52 \times 52</strong></td>
</tr>
</tbody>
</table>
Outline

- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based On YOLO
- Experimental Results
- Conclusion
The Proposed Depth Prediction – YOLOv3-based Architecture

- Two modifications
  1. Multiple depth prediction branches
  2. Modify the output layer

![Diagram showing the architecture of the proposed depth prediction model based on YOLOv3. The diagram includes multiple depth prediction branches and modified output layers.](image-url)
The Proposed Depth Prediction – YOLOv3-based Architecture: Multiple depth branches

- 3 prediction layers in original YOLOv3
  → 3 depth prediction branches
    - Output layer feature map sizes: $13 \times 13$, $26 \times 26$, $52 \times 52$
    - 3 boxes per cell (for each scale)
    - One depth prediction for each box

➢ The sizes of output layer: $13 \times 13 \times 3$, $26 \times 26 \times 3$, $52 \times 52 \times 3$
The Proposed Depth Prediction – YOLOv3-based Architecture: Multiple depth branches
The Proposed Depth Prediction – YOLOv3-based Architecture: Output Layer

- Each bounding box now needs $85+1$ parameters
  - $x, y, w, h, \text{confidence}, \text{depth}$
  - $\text{class}_1, \text{class}_2, \cdots, \text{class}_{80}$
- Each cell predicts 3 bounding boxes
- The depth of output layer is $3 \times (85+1) = 255+3$
The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

- KITTI has RGB image and corresponding depth image
- To train our model: use ground truth of object depth
  - Use RGB images to locate objects
  - Use depth images to calculate ground truth of object depth
The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

1. Use original YOLOv3 to locate objects
   - The input of original YOLOv3 is square(1:1), and may cause object distortion and feature loss
The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

2. Split images to near square
   - Original: 1242×375 (3.3:1)
   - Split: 480×375 (1.2:1)
The Proposed Depth Prediction – Adapt KITTI Dataset as Our Experimental Data

3. Refine object location using Mask-RCNN
   - Object bounding box is not accurate enough
     - object depth may be erroneous
The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

3. Refine object location using Mask-RCNN
   - Object bounding box is not accurate enough
     ➢ object depth may be erroneous

object mask:
The Proposed Depth Prediction
– Adapt KITTI Dataset as Our Experimental Data

4. Define object depth (for obstacle detection)
   - Use average depth of the nearest 20% object points
   - KITTI dataset: 60K training images & 130K objects/depths
The Proposed Depth Prediction
– Build a Dataset Using AirSim

- AirSim – a program to generate training data
  - Load different scenes – different data domains
  - Generate different types of ground truth
    - RGB images / depth images / segmentation images
  - Use different vehicles
    - Car
    - Drone

RGB  Depth  Segmentation
The Proposed Depth Prediction
– Build a Dataset Using AirSim — Data Collection

- Camera position
  - Equally spaced samples along red lines: 1m spacing
  - Height: 1, 2 … 10m

- Camera direction
  - Random samples from normal distribution
  - Yaw: $\mu=0$, $\sigma=30$; Pitch: $\mu=0$, $\sigma=15$; Roll: $\mu=0$, $\sigma=15$
The Proposed Depth Prediction – Build a Dataset Using AirSim — Generate GT

- **Bounding box** ground truth

- **Object depth** ground truth
  - Nearest 20% depth average in the **mask**

- **Dataset detail**
  - Number of training images: 32,800
  - Number of objects: 60,000
The Proposed Depth Prediction
– Training Details

- Pre-trained COCO dataset
- Use data augmentation
  - Flip, rotate, random crop, adjust hue, saturation, exposure
- Add depth prediction loss ($L_1$ distance)
  \[ \sum_{i}^{N} |depth_i - depth_i^*| \]
- For KITTI-depth dataset
  - Detection result of original YOLOv3 is good
    - Train depth prediction branch only
- For AirSim Dataset
  - Detection result of original YOLOv3 is no good
    - Train full architecture
The Proposed Depth Prediction – Training Details

- Pre-trained on ImageNet
- Use data augmentation
  - Flip, rotate, random crop, adjust hue, saturation, exposure
- Add depth prediction loss
  \[
i_{\text{depth}}(i) \times i_{\text{depth}}(i) \]
- For KITTI-depth dataset
  - Detection result of original YOLOv3 is good
    - Train depth prediction branch only
- For AirSim Dataset
  - Detection result of original YOLOv3 is no good
    - Train full architecture
The Proposed Depth Prediction

– Evaluation Metrics

- For object detection

\[ \text{Precision} = \frac{\text{# of correct detections}}{\text{# of total detections}}, \quad \text{Recall} = \frac{\text{# of correct detections}}{\text{# of ground truths}} \]

- For depth prediction

  - Absolute relative difference (ARD):
    \[ \frac{1}{N} \sum_{i}^{N} \left| \frac{y_i - y_i^*}{y_i^*} \right| \]

  - Root mean square error (RMSE):
    \[ \sqrt{\frac{1}{N} \sum_{i}^{N} (y_i - y_i^*)^2} \]

  - Threshold: percentage of \( y_i \) such that
    \[ \delta = \max \left( \frac{y_i}{y_i^*}, \frac{y_i^*}{y_i} \right) < thr \]

\( y_i \): predicted depth
\( y_i^* \): ground truth depth
\( N \): total object number
Outline

- Introduction
- YOLO – a CNN for Deep Learning
- The Proposed Depth Prediction Based on YOLO
- Experimental Results
- Conclusion
Experimental Results
– Testing Dataset Detail

- KITTI-depth
  - Number of testing images: 5,200
  - Number of objects: 14,200

- AirSim
  - Number of testing images: 3,400
  - Number of objects: 5,100
Experimental Results – From Depth Image to Object Depth

- Evaluation
  - Baselines
    - Depth per pixel
  - Our method
    - Depth per object

- To make fair comparison
  - Transform the result of baseline
Experimental Results
– Comparison with other methods

- Observation – KITTI-depth dataset
  - YOLOv3-based model compares favorably with other methods (and better than YOLOv2-based model)

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE (meter)</th>
<th>ARD</th>
<th>Threshold (No Cap)</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Cap</td>
<td>Cap 50</td>
<td>Cap 30</td>
<td></td>
</tr>
<tr>
<td>Godard et al. (CVPR, 2017)</td>
<td>6.011</td>
<td>4.939</td>
<td>2.853</td>
<td>0.207</td>
</tr>
<tr>
<td>Kuznietsov et al. (CVPR, 2017)</td>
<td>4.958</td>
<td>3.483</td>
<td>1.903</td>
<td>0.131</td>
</tr>
<tr>
<td>Ours (YOLOv2-based)</td>
<td><strong>4.373</strong></td>
<td>3.908</td>
<td>2.887</td>
<td>0.159</td>
</tr>
<tr>
<td>Ours (YOLOv3-based)</td>
<td><strong>2.927</strong></td>
<td>2.655</td>
<td><strong>1.899</strong></td>
<td><strong>0.086</strong></td>
</tr>
</tbody>
</table>

*Cap 50: only objects within 50m are calculated
*Tested on GTX1080

Lower is better
Higher is better
Experimental Results
– Comparisons between YOLOv2 and YOLOv3-based Model

- YOLOv2-based model testing result on KITTI-depth dataset
Experimental Results
– Comparisons between YOLOv2 and YOLOv3-based Model

- YOLOv3-based model testing result on KITTI-depth dataset
Experimental Results
– Comparisons between YOLOv2-based and YOLOv3-based Model

Observations
- YOLOv3-based model is better than YOLOv2-based model
- Fewer training data → larger relative error

Object depth distribution of training set

- **Object Count**
- **Percentage (%)**

Ground Truth Object Depth (m)

Object count

Cumulative percentage
Experimental Results
– Comparisons between different input sizes

- Observations – KITTI-depth dataset
  - Increasing input size decreases performance
  - Larger input size → lower FPS

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Size</th>
<th>RMSE (meter)</th>
<th>ARD</th>
<th>Threshold (No Cap)</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Cap</td>
<td>Cap 50</td>
<td>Cap 30</td>
<td>δ &lt; 1.25</td>
</tr>
<tr>
<td>Ours (YOLOv3)</td>
<td>416 x 416</td>
<td>2.927</td>
<td>2.655</td>
<td>1.899</td>
<td>0.086</td>
</tr>
<tr>
<td></td>
<td>480 x 480</td>
<td>2.981</td>
<td>2.671</td>
<td>1.871</td>
<td>0.092</td>
</tr>
<tr>
<td></td>
<td>544 x 544</td>
<td>2.983</td>
<td>2.695</td>
<td>1.909</td>
<td>0.093</td>
</tr>
</tbody>
</table>

Lower is better
Higher is better
### Table: Model Performance

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Size</th>
<th>RMSE (meter)</th>
<th>ARD</th>
<th>Threshold (No Cap)</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Cap</td>
<td>Cap 30</td>
<td>Cap 30</td>
<td>δ &lt; 1.25</td>
</tr>
<tr>
<td>Ours (YOLOv3)</td>
<td>416 × 416</td>
<td>2.927</td>
<td>2.655</td>
<td>1.899</td>
<td>0.086</td>
</tr>
<tr>
<td></td>
<td>480 × 480</td>
<td>2.981</td>
<td>2.671</td>
<td>1.871</td>
<td>0.092</td>
</tr>
<tr>
<td></td>
<td>544 × 544</td>
<td>2.983</td>
<td>2.695</td>
<td>1.909</td>
<td>0.093</td>
</tr>
</tbody>
</table>

- **Lower is better**
- **Higher is better**
Experimental Results
– Comparisons between different input sizes

- Observation – AirSim dataset
  - Larger input size → higher recall rate
  - Larger input size → higher RMSE error

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Size</th>
<th>RMSE (meter)</th>
<th>ARD</th>
<th>Threshold (No Cap)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Cap</td>
<td>Cap 50</td>
<td>Cap 30</td>
<td>δ &lt; 1.25</td>
<td>δ &lt; 1.56</td>
<td>δ &lt; 1.95</td>
</tr>
<tr>
<td>Ours Fix (YOLOv3)</td>
<td>416 × 416</td>
<td>6.473</td>
<td>5.376</td>
<td>3.257</td>
<td>0.195</td>
<td>0.401</td>
<td>0.513</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>3.323</strong></td>
<td><strong>2.815</strong></td>
<td><strong>1.719</strong></td>
<td><strong>0.075</strong></td>
<td><strong>0.868</strong></td>
<td><strong>0.922</strong></td>
</tr>
<tr>
<td></td>
<td>480 × 480</td>
<td>3.773</td>
<td>2.910</td>
<td>1.342</td>
<td>0.085</td>
<td>0.929</td>
<td>0.961</td>
</tr>
<tr>
<td></td>
<td>544 × 544</td>
<td>3.935</td>
<td>2.963</td>
<td>1.431</td>
<td>0.092</td>
<td><strong>0.928</strong></td>
<td><strong>0.962</strong></td>
</tr>
</tbody>
</table>

Lower is better
Higher is better
**Experimental Results**  
– Comparisons between different input sizes

\[ \delta = \max \left( \frac{y_i}{y_i^*}, \frac{y_i^*}{y_i} \right) \]

<table>
<thead>
<tr>
<th>Predicted Depth</th>
<th>Ground Truth</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object 1</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td>Object 2</td>
<td>22</td>
<td>25</td>
</tr>
<tr>
<td>Object 3</td>
<td>26</td>
<td>30</td>
</tr>
</tbody>
</table>

RMSE: 3.36

<table>
<thead>
<tr>
<th>Predicted Depth</th>
<th>Ground Truth</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object 1</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td>Object 2</td>
<td>22</td>
<td>25</td>
</tr>
<tr>
<td>Object 3</td>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td><strong>Object 4</strong></td>
<td>55</td>
<td>65</td>
</tr>
</tbody>
</table>

RMSE: 5.78
Experimental Results
– Interactions between AirSim and YOLOv3-based Model

Observations
- Training can improve precision and recall rates
- Depth architecture helps detector learn better
- Detector learns better → depth predicts better

### Experimental Results

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE (meter)</th>
<th>Detection metric</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Cap</td>
<td>Cap 50</td>
</tr>
<tr>
<td>Original YOLOv3 (Not trained)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Original YOLOv3 (Trained)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours Fix</td>
<td>6.473</td>
<td>5.376</td>
</tr>
<tr>
<td>Ours Full</td>
<td>3.323</td>
<td>2.815</td>
</tr>
</tbody>
</table>

Detector fixed

Lower is better
Higher is better
Experimental Results
– Qualitative Results

Go to demo video
Outline

■ Introduction

■ YOLO – a CNN for Deep Learning

■ The Proposed Depth Prediction Using YOLO

■ Experimental Results

■ Conclusion
Conclusion

- KITTI dataset is adapted and have ground truth object depth
- The original YOLOv2 and YOLOv3 are modified to incorporate depth prediction
- The proposed architecture compares favorably on other depth prediction methods (KITTI)
- Extra depth prediction architecture can enhance the performance of object detection (AirSim)
Thank you