Shift R-CNN

DEEP MONOCULAR 3D OBJECT DETECTION WITH CLOSED-FORM GEOMETRIC CONSTRAINTS

Andretti Naiden, Vlad Paunescu*, Gyeongmo Kim, ByeongMoon Jeon, Marius Leordeanu*

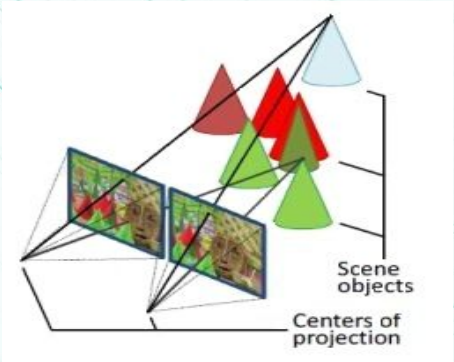
IEEE link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8803397

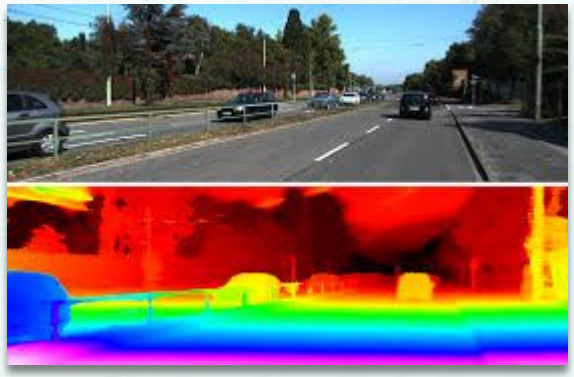
TOPICS

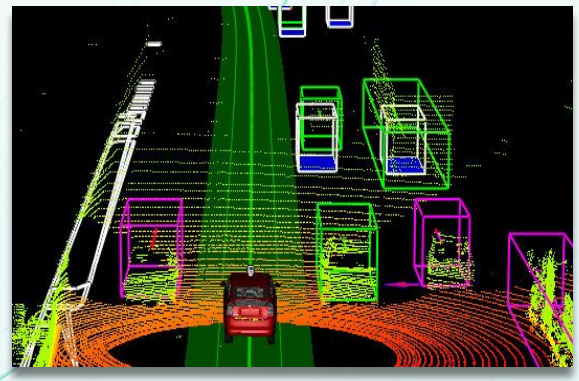
1. Motivation
2. Purpose
3. Contributions
4. Architecture
5. Results and Demo
6. Conclusion

Motivation

- **State-of-the-art self-driving cars** are based on Sensor Fusion methods
  - Stereo (multi RGB images), Depth sensors, LiDAR systems
  - Higher cost of production, more susceptible to hardware malfunctions, non-ergonomic



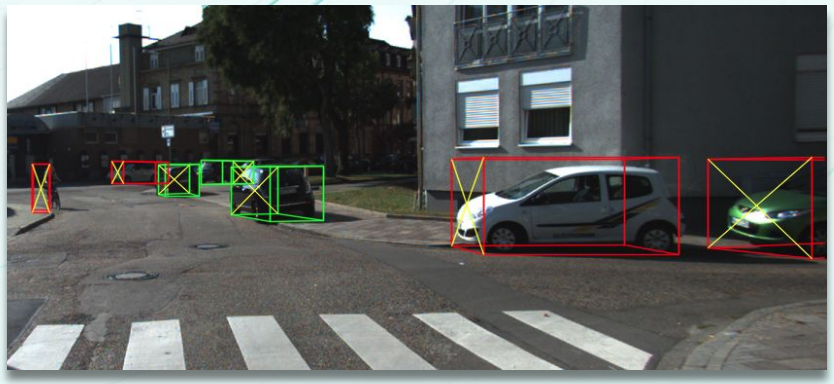




- **Humans drive using only visual cues** - autonomous driving using vision alone
  - RGB cameras are ubiquitous, less expensive, and have a significantly higher pixel resolution
- 2D object detection from a single RGB image (monocular) is already solved
- **Monocular 3D perception** remains a very difficult challenge

Purpose

- **Hybrid 3D Object Detection model** combining deep learning with geometry
- Monocular object detection - using only one RGB camera on the car
- No pretrained network for Depth Estimation
- **Detection** - 3D Bounding Box reconstructed from:
  a. *2D Bounding Box*
  b. *3D angle orientation* around the Oy axis
  c. *3D object dimensions* - height, width, length
  d. *3D object translation (3D center coordinates)* - (x, y ,z)



Contributions

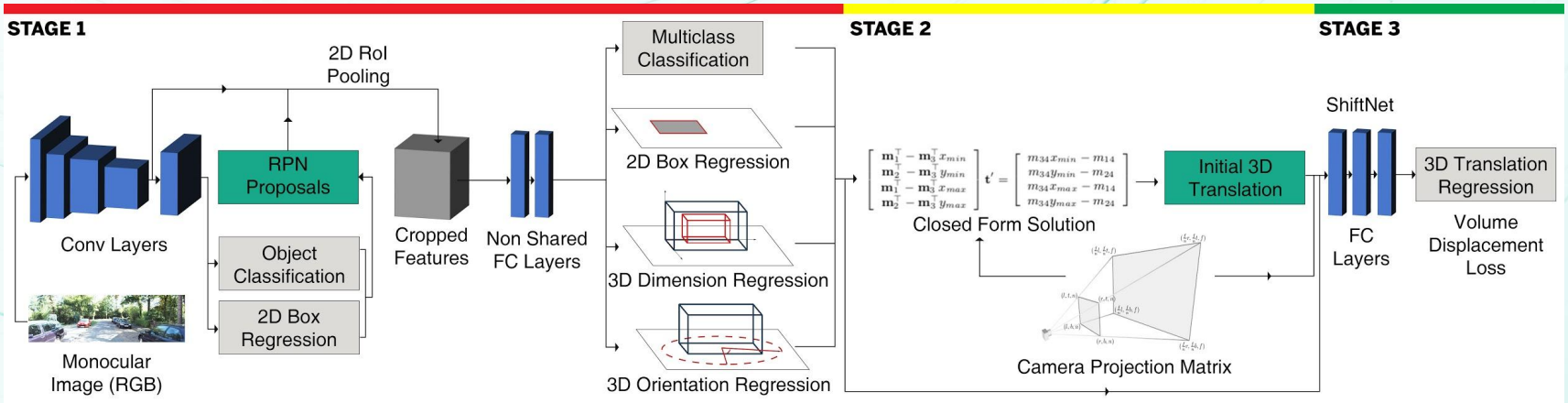
- **Modified Faster R-CNN**, 3D object detection architecture, estimating
  - 2D location, **orientation and 3D object properties (dimensions)**
- **Mathematical system of equations** solved using least squares, estimating
  - 3D object translation, based on previous values and camera projection matrix
  - Enforcement of the 3D to perfectly fit it's 2D projection

- Fully connected refinement network - **ShiftNet**
  - Learns the error dependency between the first 2 stages and corrects the final 3D translation
- A novel **Volume Displacement Loss (VDL)**
  - Differentiable alternative to 3D IoU metric
  - Used for training ShiftNet by approximating the maximization of 3D IoU

Architecture

Main Stages

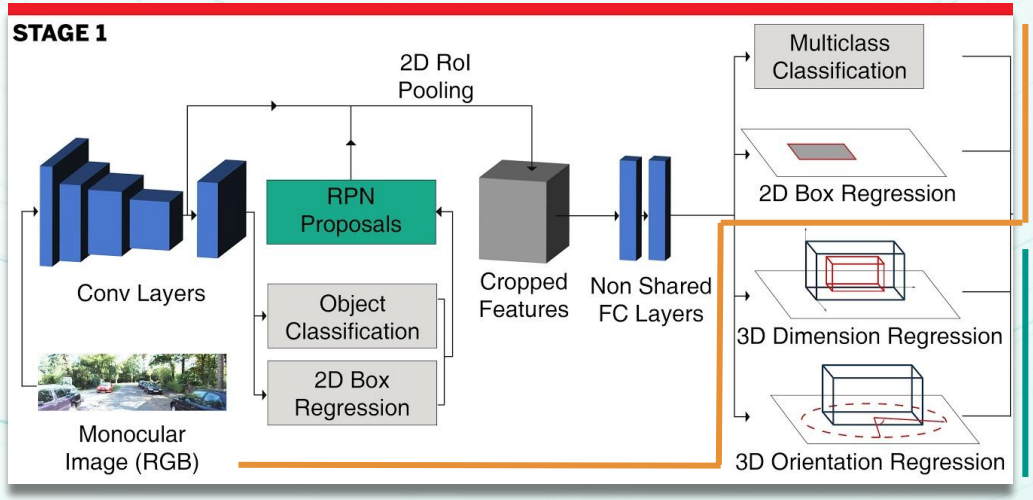
1. **2D Object Detection and 3D Intrinsics Estimation**
   a. Faster R-CNN architecture composed of 4 prediction heads
   b. Angle Regression Head - predicting local object angle
   c. 3D Dimension Regression Head - predicting 3D object height, width, length
2. **Estimating the 3D Object Translation**
   a. Local to global orientation
   b. Inverse 2D to 3D geometric mapping problem
3. **Refining the 3D Object Translation**
   a. Shift Network
   b. Volume Displacement Loss (VDL)



Architecture

2D Object Detection and 3D Intrinsics Estimation

- **Input** - Only one RGB image (monocular)
- Extract and score 2D region proposals by means of 2D anchors
- 2D ROI pooling for feature cropping and feature refinement
- Multiclass classification and box refinement, per class, only for top scored detections
- Angle and 3D dimensions regression
- **Output** - class, 2D bounding box, local orientation angle, 3D dimensions



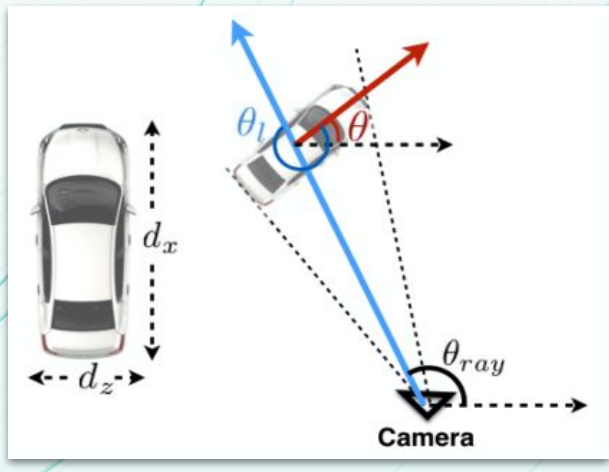
Faster R-CNN

Additional Regression Heads

Architecture

Angle Disentanglement

- A crucial assumption is made
  - **Objects are placed on the ground plane**
  - Only rotation around Oy axis is necessary
- An object's orientation angle is multifaceted
  - Θ_L - observed object's **local** orientation angle, around Oy axis
  - Θ_{RAY} - the ray from the camera origin to the center of the object
  - $\Theta_{(RY)}$ - real object's global orientation angle around Oy axis
    - **Ry** (global): $\Theta = \Theta_L - \Theta_{RY}$
- Faster R-CNN works with region proposal feature **crops**
  - Only the observed local angle Θ_L can be predicted



Architecture

Angle Regression Head

- Due to the periodicity of an angle's value α_L, regression requires a new encoding
  - **Sine** and **Cosine** encoding $\widehat{\sin_{\alpha_L}}$ and $\widehat{\cos_{\alpha_L}}$
  - Enforcement constraint used to promote angle consistency $\widehat{\sin_{\alpha_L}}^2 + \widehat{\cos_{\alpha_L}}^2 = 1$

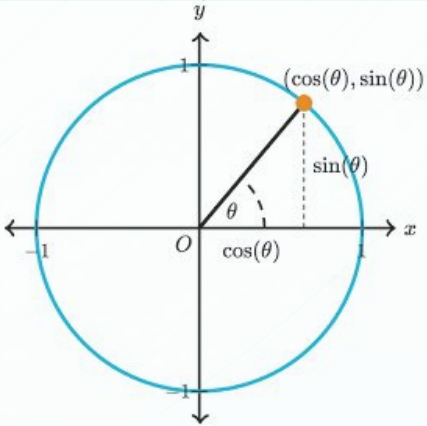
- For each object, C different angles are predicted, one for each class
- The loss is computed only for the winning class:

$$L_{\alpha_L} = || \sin \alpha_{L_{gt}} - \widehat{\sin_{\alpha_L}}||_2 + || \cos \alpha_{L_{gt}} - \widehat{\cos_{\alpha_L}}||_2 + L_{cnt}$$

$$L_{cnt} = ||1 - (\widehat{\sin_{\alpha_L}}^2 + \widehat{\cos_{\alpha_L}}^2)||_2$$

- At inference time, the local orientation angle is given by:

$$\alpha_L = atan2(\widehat{\sin_{\alpha_L}}, \widehat{\cos_{\alpha_L}}).$$



Architecture

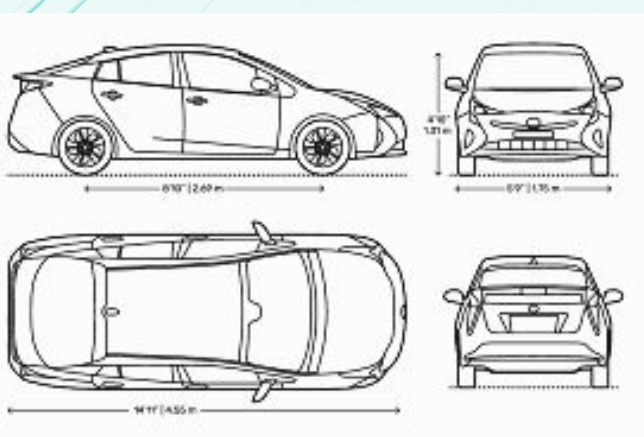
3D Dimension Regression Head

- In line with Faster R-CNN angle regression, the 3D object **height**, **width** and **length** are also regressed in C triplets, one per class $\mathbf{d} = [h, w, l]$
  - Because of low variance of the dimensions, we use predefined mean dimensions per class, that act as anchors $\overline{\mathbf{d}_c} = [\ \overline{h_c}, \overline{w_c}, \overline{l_c}\ ]$
  - Network predicts the logarithmic scale offsets to the mean dimensions, similar to bounding box regression in Faster R-CNN (log space) $\Delta\mathbf{d}_c = [\Delta h, \Delta w, \Delta l]$
  - The network will predict a **zero-centered distribution** around the mean dimension:

$$\Delta\mathbf{d}_c = \left[\ \ln \frac{h}{\overline{h_c}}, \ln \frac{w}{\overline{w_c}}, \ln \frac{l}{\overline{l_c}}\ \right]$$

- Loss is computed for the winning class using **L2 Loss**
- At inference time, the predicted dimension is given by:

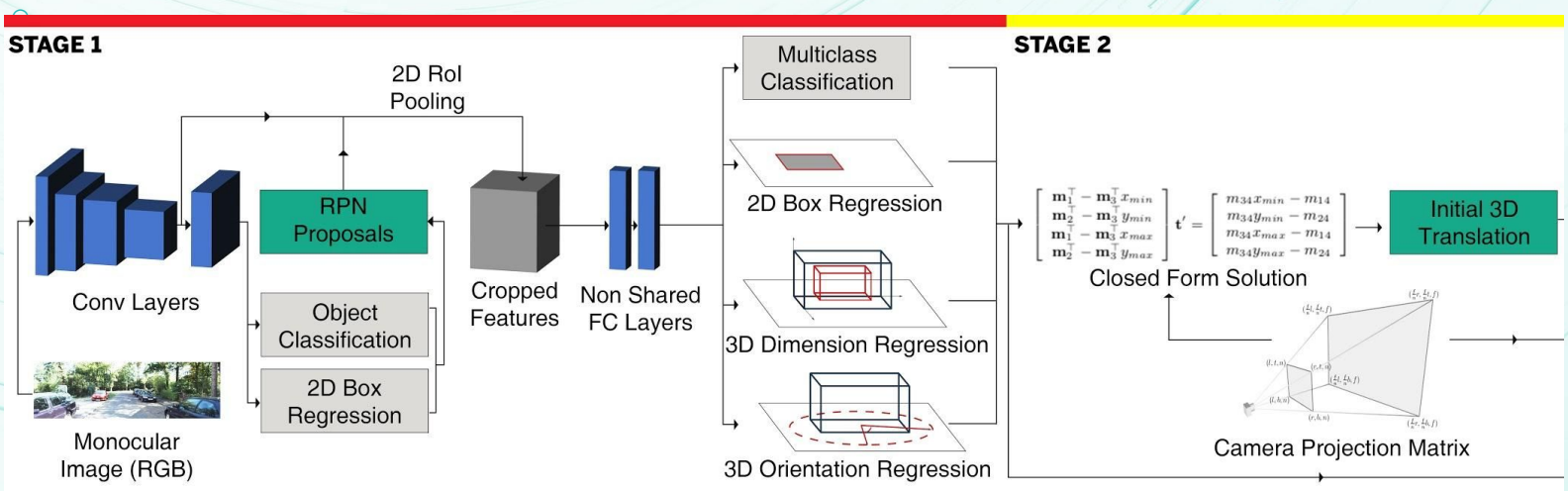
$$\mathbf{d} = [e^{\Delta h} \times \overline{h_c}, e^{\Delta w} \times \overline{w_c}, e^{\Delta l} \times \overline{l_c}]$$



Architecture

Estimating the 3D Object Translation

- Compute the **global angle** using the predicted **local angle** and **Theta ray** $\alpha_G = \alpha_L - \theta_{ray}$
- Having the Stage 1 predictions:
  - Bounding box $\mathrm{b}_{2D} = [x_{min}, y_{min}, x_{max}, y_{max}]$
  - Global angle α_G
  - Dimensions $\mathbf{d} = [h, w, l]$
  - Camera projection matrix **P**
- Estimate the **3D translation vector** $\mathrm{t'} = [\mathrm{t'}_x, t'_y, t'_z]^\top$ as a closed form, least squares solution



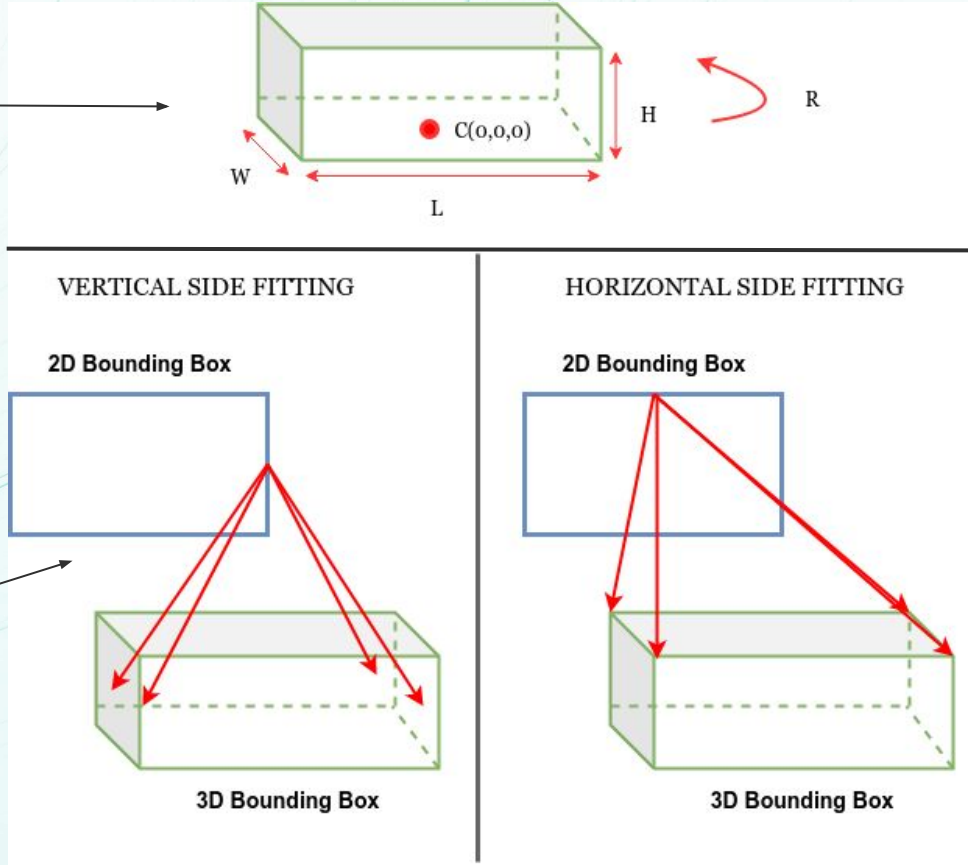
Architecture

- 3D bounding box reconstruction
  - Camera origin centered $\mathbf{x}_{3D_o}$
  - Shaped by dimensions
  - Rotated with the global angle about Oy
- Final 3D/2D corner points are given by:

$$\mathbf{x}_{3D} = \mathbf{R}_y(\alpha_G)\mathbf{x}_{3D_o} + \mathbf{t}'$$

$$\lambda \begin{bmatrix} \mathbf{x}_{2D} \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{x}_{3D} \\ 1 \end{bmatrix}.$$

- 3D BBox enforcement to fit 2D BBox
  - Each vertical 2D edge - 4 vertical 3D edges
  - Each horizontal 2D edge - 4 3D corners
  - **64 possible different configurations**



Architecture

- Each 2D BBox side fixed to a 3D BBox point
  - Unknown translation vector **t'**
  - For each 2D side we form the system

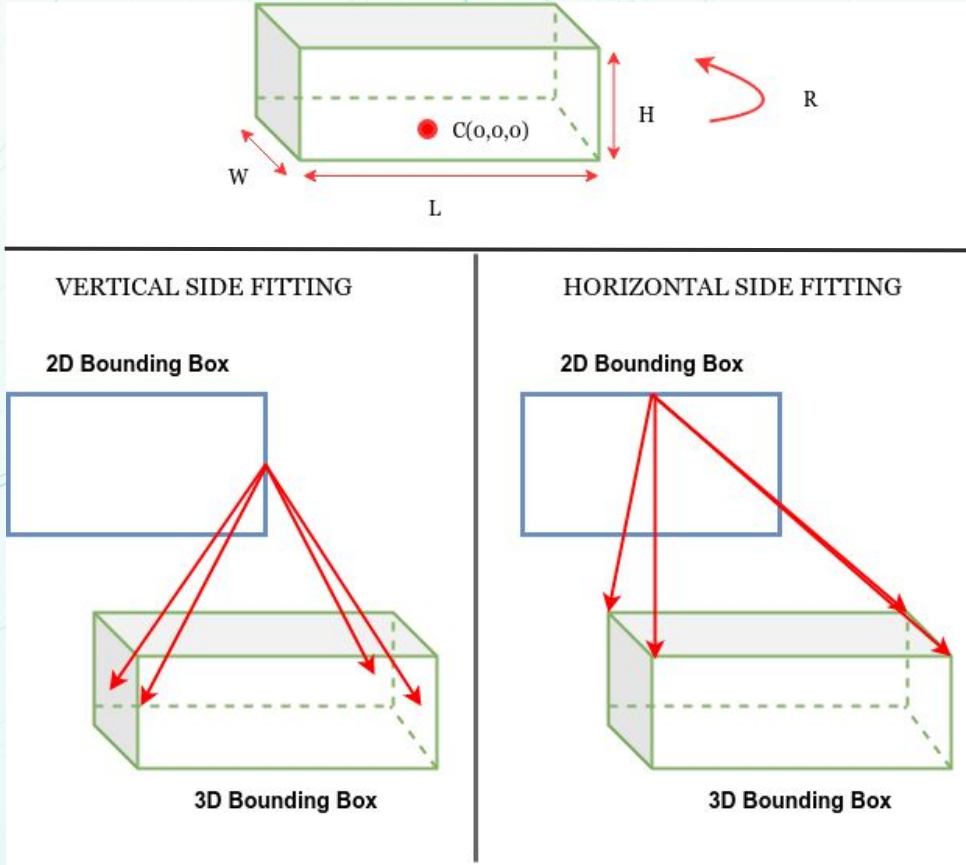
$$\underbrace{\mathbf{P}\begin{bmatrix} I & \mathbf{R}_y(\alpha_G)X_{3D_o} \\ 0 & 1 \end{bmatrix}}_{M}\begin{bmatrix} \mathbf{t}' \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} x_{2D_{side}} \\ y_{2D_{side}} \\ 1 \end{bmatrix}$$

- Taking all the 2D sides into consideration

$$\begin{bmatrix} \mathbf{m}_1^\top - \mathbf{m}_3^\top x_{min} \\ \mathbf{m}_2^\top - \mathbf{m}_3^\top y_{min} \\ \mathbf{m}_1^\top - \mathbf{m}_3^\top x_{max} \\ \mathbf{m}_2^\top - \mathbf{m}_3^\top y_{max} \end{bmatrix} \mathbf{t}' = \begin{bmatrix} m_{34}x_{min} - m_{14} \\ m_{34}y_{min} - m_{24} \\ m_{34}x_{max} - m_{14} \\ m_{34}y_{max} - m_{24} \end{bmatrix}$$

- Over-constrained system, solved by least squares method:

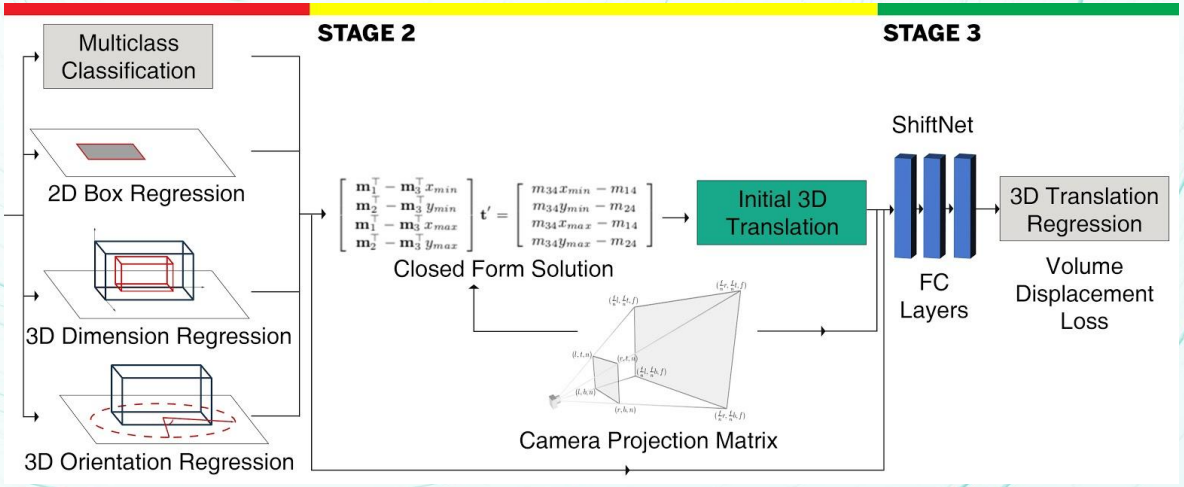
$$\mathrm{A\ t'} = \mathrm{b},\ \mathrm{b} \neq \mathbf{0} \longrightarrow \mathrm{t'} = (\mathrm{A}^\top \mathbf{A})^{-1}\mathbf{A}^\top \mathbf{b}$$



Architecture

Translation

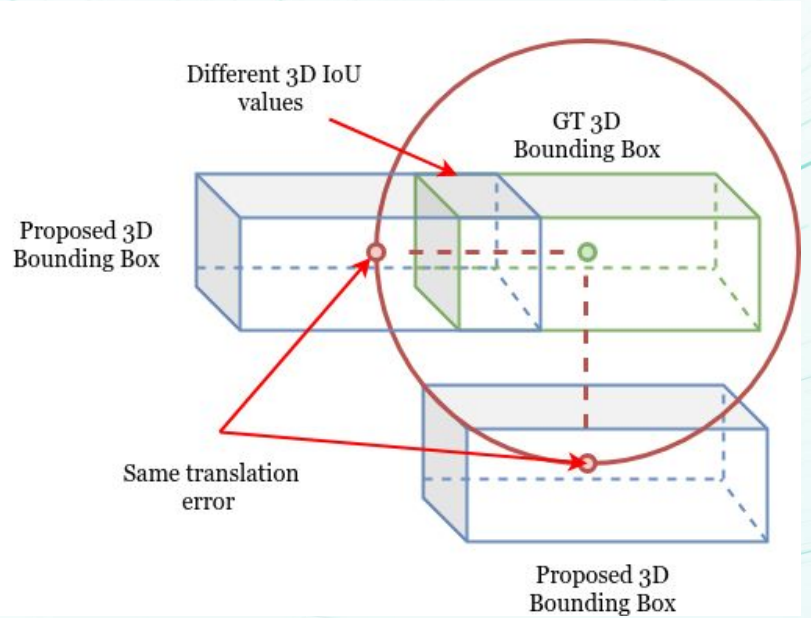
Refining the 3D Object



- Stage 2 estimated translation $\mathrm{t}' = [\mathrm{t}'_x, t'_y, t'_z]^\top$, is susceptible to noise
  - Accuracy drop from **80% to 14% 3D IoU** in the presence of noise from Stage 1
- **ShiftNet** - network that produces a refined translation $\mathrm{t}'' = [\mathrm{t}''_x, t''_y, t''_z]$
  - **Fully connected architecture**, 3 layers of 1024 neurons
  - **Purpose** - learns how to correct the Stage 2 translation using the Faster R-CNN error distribution
  - **Input** - Stage 1 2D BBox, local and global angles, Stage 2 translation and the projection matrix
  - **Output** - final refined translation

Architecture

Volume Displacement Loss



- **3D IoU** does not define a distance in metric space
- **Loss function** to approximate 3D IoU
  - L1, L2 loss only used for translation error
- **Translation error** alone does not suffice
  - Does not take into consideration shape variation
- **Two 3D boxes** on a sphere around the 3D GT box
  - Same translation error
  - Distinct 3D IoU values
- **The 3D IoU differences are huge** and and range between 0 and 50%

Architecture

Volume Displacement Loss

- **Volume Displacement Loss (VDL)**
  - Approximates 3D IoU metric
  - Represents a distance in metric space
  - Fully differentiable
  - Dimensions dependent
- **The quantity of volume** displaced by moving the 3D predicted box center alongside one local coordinate axis
- *Signed Translation Displacement Error*

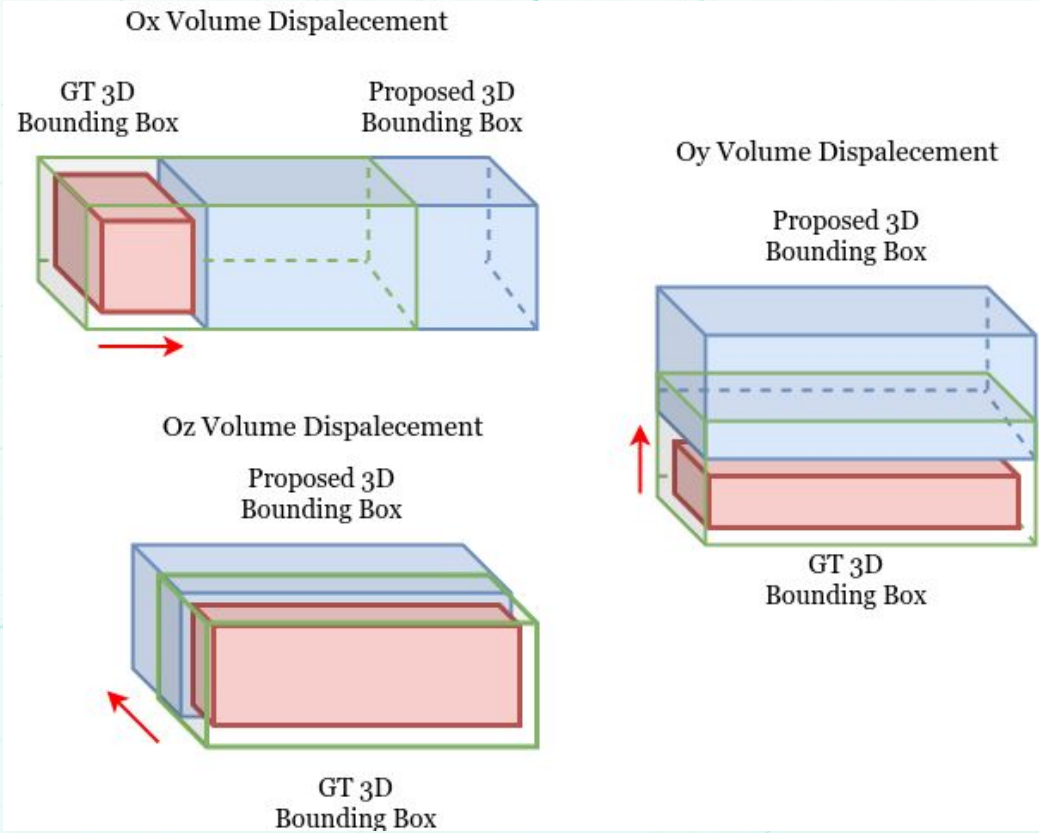
$$\Delta \mathbf{t} = [t''_x - t_x, t''_y - t_y, t''_z - t_z]^\top$$

- *Relative to local object's coordinate axis*

$$\Delta \mathbf{t}_{\alpha_G} = \mathbf{R}_y(\alpha_G)\Delta \mathbf{t}$$

- **Final VDL Loss**

$$L = \underbrace{w \times h}_{A_{yOz}} \times |\Delta x_{\alpha_G}| + \underbrace{w \times l}_{A_{xOz}} \times |\Delta y_{\alpha_G}| + \underbrace{h \times l}_{A_{xOy}} \times |\Delta z_{\alpha_G}|$$



Training Procedure

Dataset. KITTI 3D Object Detection : train/val Chen split

Stage 1 Setup

- Faster R-CNN architecture, ResNet-101 backbone, COCO pretrained
- SGD, 30 epochs, momentum 0.9, learning rate 3×10^{-4} and 2 decays of 0.1 at ⅓ , ⅔
- Loss weighting strategy: the uncertainty of a task
  - 1.0, 2.0, 5.0 and 100.0 for the classification, localization, orientation and dimension regression

ShiftNet Setup (Stage 3 refinement)

- Pretrain using Stage 1 ground-truth and Stage 2 3D translation estimate
  - Fine-tune on real estimates produced by Stage 1
- 6.6 AP improvement (13.8 vs 7.2 car AP@0.7 3D IOU) on real image estimates from Stage 1
- 3D Synthetic data augmentation improves the AP (0.5 3D IoU) for the classes with fewer examples (22% vs. 20% for Pedestrian, 73% vs. 47% for Cyclist) for GT input in Stage 1 and 3D translation estimate in Stage 2

Results

Stage 3 Refinement

Fig. 2: Stage 2 (top) and Stage 3 (bottom) results comparison. Note that Stage 3 improves the 3D estimation due to its noise robustness. Turquoise boxes denote objects with the same orientation and magenta color the opposite orientation. Best viewed in color.

- Top row: Projective system fitting (prone to error)
- ShiftNet refinement (error adaptive)
    - Improves occluded objects (most notable)

Results

- The model was trained on the KITTI 3D Detection Dataset
- Evaluation was done both on the Chen split set and KITTI 3D Detection Test set
- The evaluation metric used is 3D IoU and Birds Eye View IoU
- For monocular-only based methods, we obtained state of the art results (at that time)

			AP_{3D} (%)			AP_{BEV} (%)		
Method	Setup	Class	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [1]	Mono	Car (IoU≥0.7)	2.53 / -	2.31 / -	2.31 / -	5.22 / -	5.19 / -	4.13 / -
DeepBox3D [16]	Mono		- / 5.85	- / 4.10	- / 3.84	- / 9.99	- / 7.71	- / 5.30
OFT-Net [20]	Mono		4.07 / 2.50	3.27 / 3.28	3.29 / 2.27	11.06 / 9.50	8.79 / 7.99	8.91 / **7.51**
MLF-Mono [23]*	Mono+PD		10.53 / 7.85	5.69 / 5.39	5.39 / 4.73	22.03 / 19.20	13.63 / 12.17	11.60 / 10.89
ROI-10D [15]*	Mono+PD		10.25 / 12.30	6.39 / 10.30	6.18 / 9.39	14.76 / 16.77	9.55 / 12.40	7.57 / 11.39
Linear System (Ours)	Mono		7.24 / 6.80	5.98 / 4.14	5.54 / 3.50	14.74 / 11.75	12.48 / 8.34	11.22 / 6.80
ShiftNet (Ours)	Mono		**13.84** / **8.13***	**11.29** / **5.22**	**11.08** / **4.78***	**18.61*** / **13.32**	**14.71** / **8,49**	**13.57** / 6.40
OFT-Net [20]	Mono	Ped. (IoU≥0.5)	- / 1.11	- / 1.06	- / 1.06	- / 1.55	- / 1.93	- / 1.65
Linear System (Ours)	Mono		1.51 / 0.53	1.51 / 0.53	1.51 / 0.53	1.51 / 0.53	1.51 / 0.53	1.51 / 0.53
ShiftNet (Ours)	Mono		**7.55 / 13.36**	**6.80 / 10.59**	**6.12 / 10.59**	**8.24 / 13.81**	**7.50 / 11.44**	**6.73 / 10.76**
OFT-Net [20]	Mono	Cyc. (IoU≥0.5)	- / 0.43	- / 0.43	- / 0.43	- / 0.43	- / 0.79	- / 0.43
Linear System (Ours)	Mono		1.38 / 0.73	0.90 / 0.43	0.90 / 0.43	1.42 / 0.53	0.90 / 0.53	0.90 / 0.53
ShiftNet (Ours)	Mono		**1.85** / **3.03**	**1.08** / **3.03**	**1.10** / **3.03**	**2.30** / **3.58**	**2.00** / **3.03**	**2.11** / **3.03**

Table 1: 3D object detection results on KITTI Chen/Test splits. We report $AP_{3D}(\%)$ and $AP_{BEV}(\%)$ for Car, Pedestrian and Cyclist classes. Methods that use a pre-trained monocular depth network (Mono+PD) are in blue. We denote with **bold black** monocular state of the art, **bold black*** cases when we outperform one Mono+PD method and **bold blue**, cases where we outperform all Mono+PD. Best viewed in color.

Demo (KITTI)


Legend
Same Direction
Opposite Direction
Object From Orientation

Conclusion

- With a much lighter architecture and without a pre-trained depth network we obtained state of the art results at the time of paper submission
- On the Car class, but also on the Pedestrian and Cyclist classes, we outperformed the main monocular image 3D detectors at paper submission time on all difficulty levels on KITTI val, test splits.
- Kitti submission

Thank you!