Object Orientation Tracking in 3D Space by Using Two Webcams

Do-Men Su
Bo-Hsun Chen

Outline

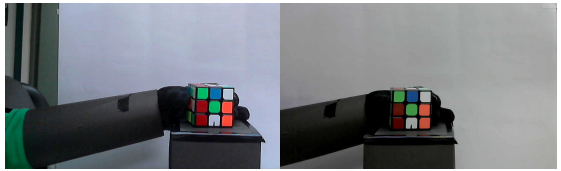
- Motivation
- Process
  - Experiment Setup
  - Building Disparity Maps with Two Webcams
  - Predict object orientation based on convolutional neural network (CNN)
- Experiment result

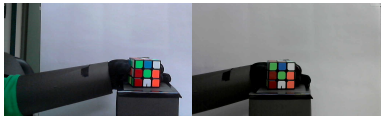
Motivation

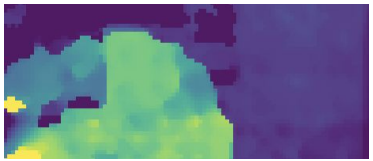
- Object orientation estimation in real time can be applied in many fields
  - Robotic manipulation, self-driving cars, etc
- Most of research utilized RGB-D or point-cloud cameras
  - Expensive and hard to access
- Build RGB-D camera system with two webcams and estimate object orientation
  - Cheaper solution

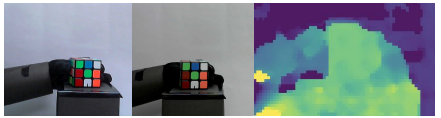
Training Process

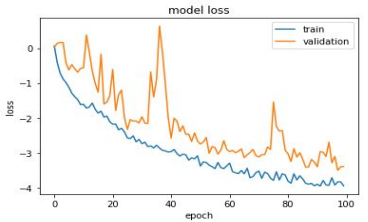
Taking Stereo Pictures → Building Disparity / Depth Maps → Training NN Model





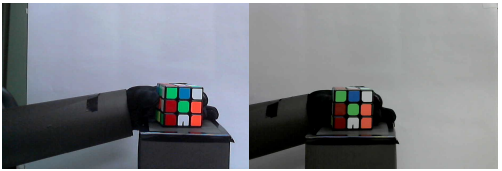


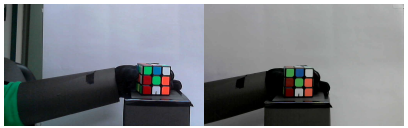


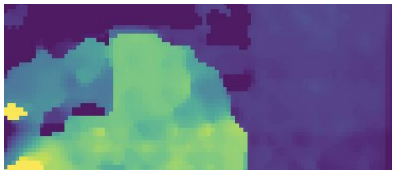


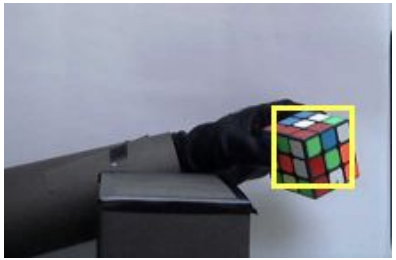
Testing Process

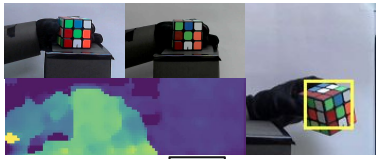
Recording Stereo Videos → Building Disparity / Depth Maps → Tracking Object Position → NN Prediction

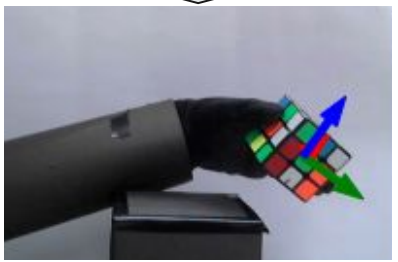




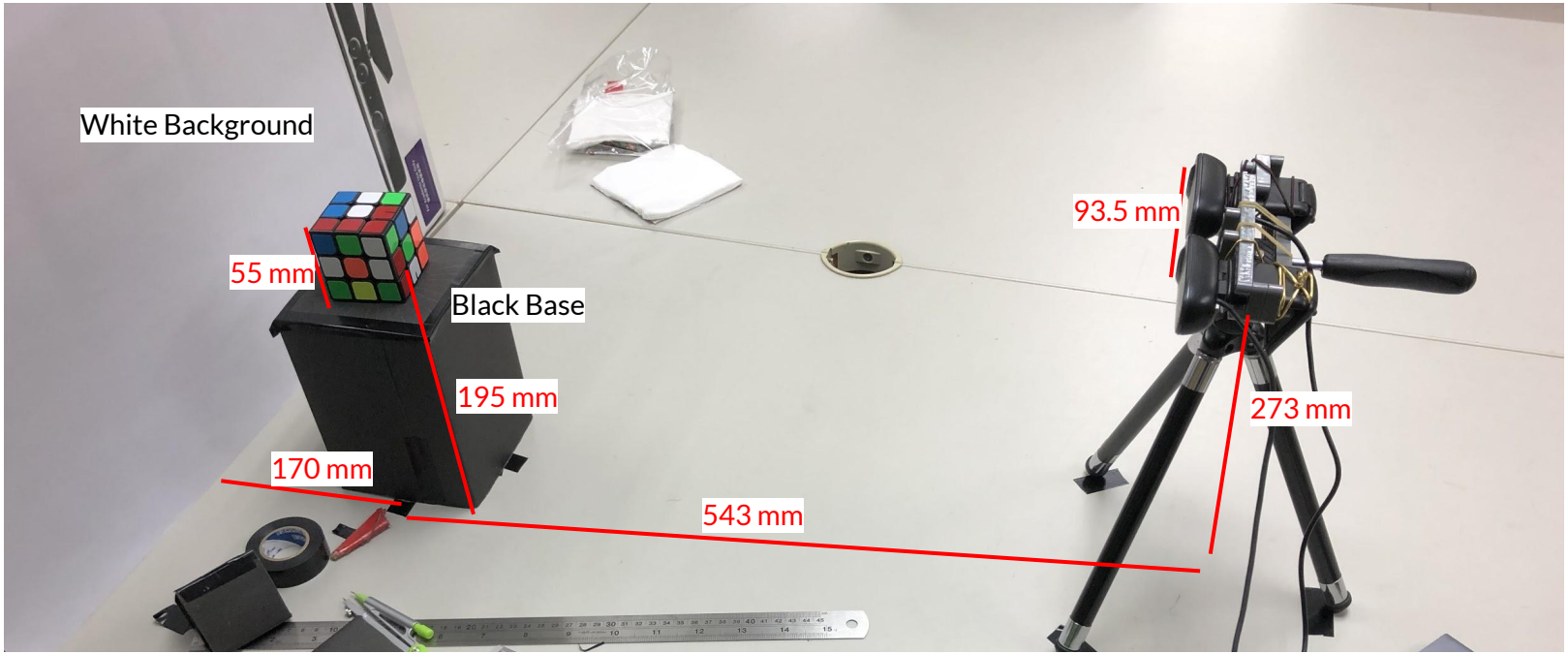






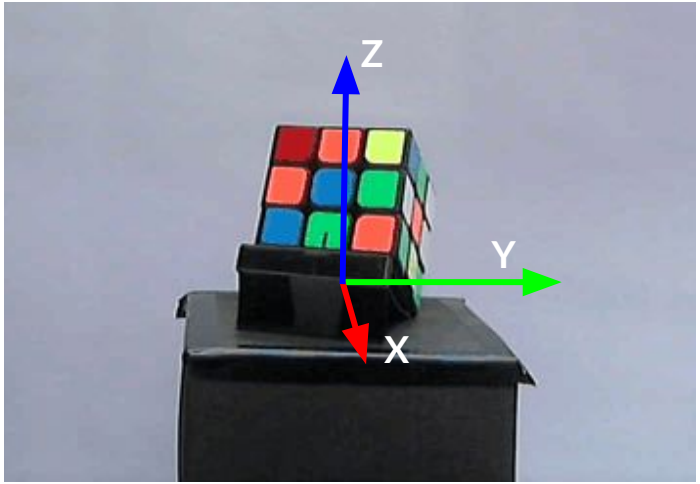


Experiment Setup



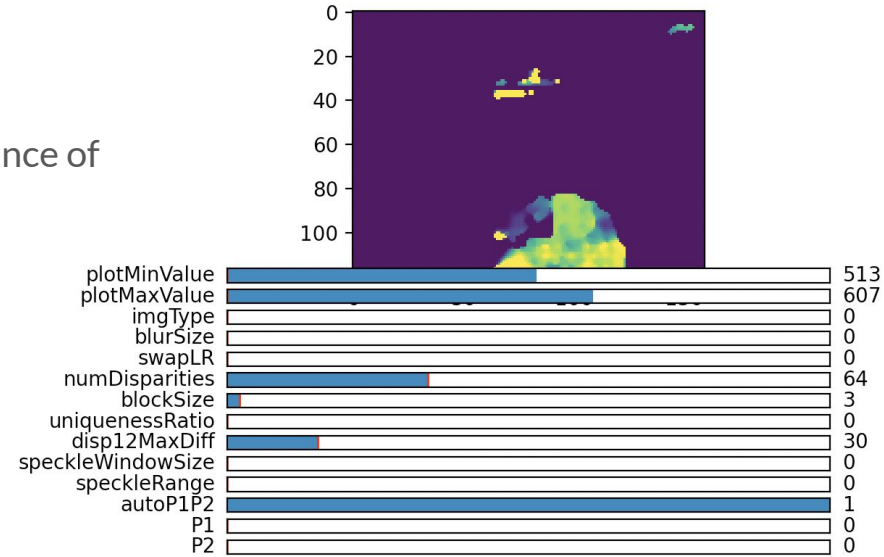
Experiment Setup - Building Datasets

- 197 images
  - Y = -23, -46, -60, -80, 0, 100, 23, 47, 60, 80
  - Z = 0, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300, 320, 340

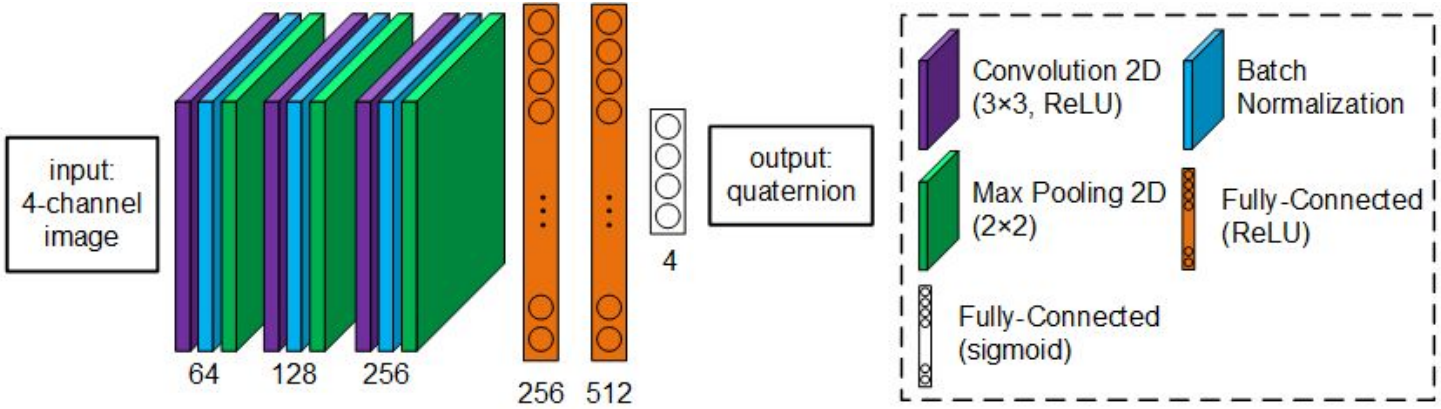


Building Disparity Maps with Two Webcams

- OpenCV: StereoSGBM_create
- Tuning Variables
  - numDisparities: The maximum distance of the same point on two images
  - blockSize
  - disp12MaxDiff: Maximum allowed difference
  - P1, P2: disparity smoothness



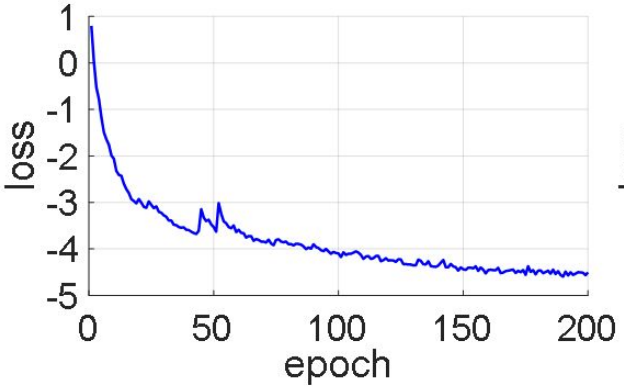
CNN structure



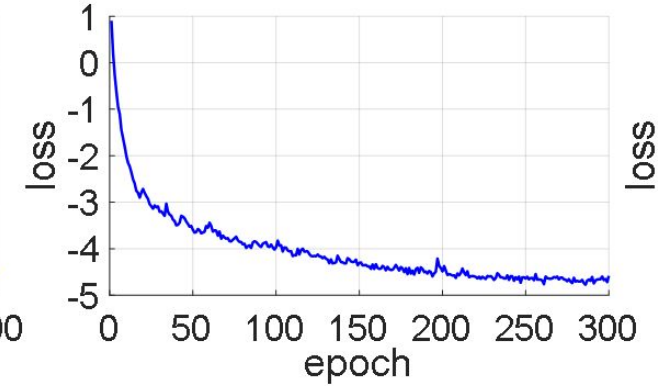
- Augment dataset by rotating images (rotating along X-axis of World frame)
- Reference PoseCNN in [1]
- 2D convolutional layers: extract features from RGB-D images
- Fully-connected layers: infer relationships between features and quaternions
- Loss function: $L = \log\left(\sum_{i=1}^{N}(y_i - \hat{y}_i)^2\right)$

Comparison

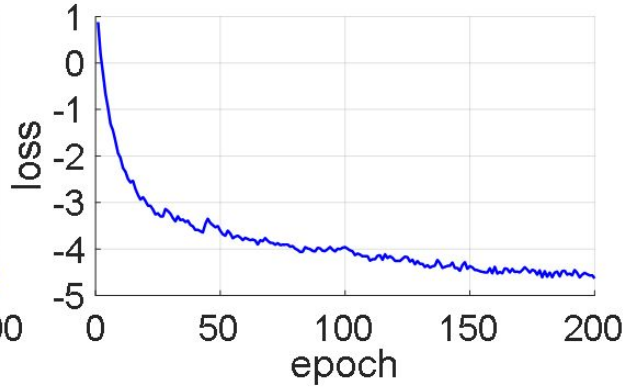
RGB (3 channels)



RGB-D (4 channels)



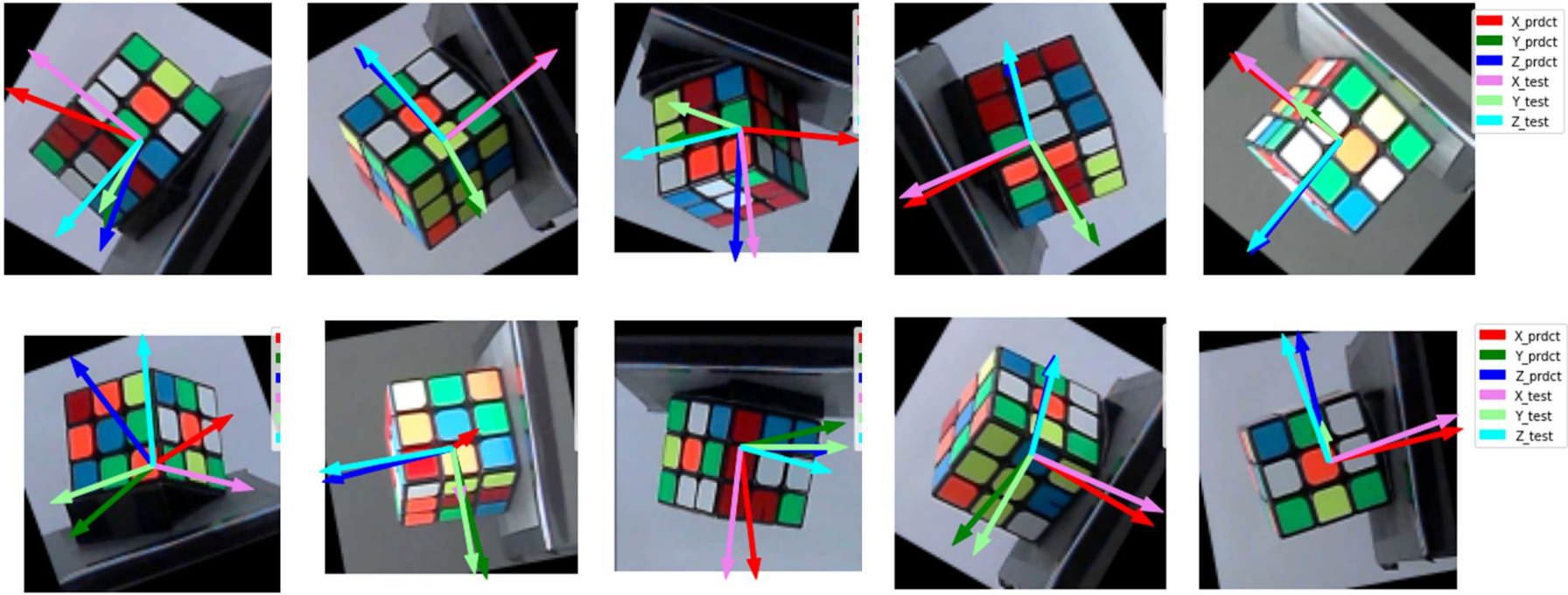
RGB-RGB (6 channels)



	RGB	RGB-D	RGB-RGB
final training loss	-4.5035	-4.5839	-4.6329
final valid loss	-4.7390	-4.7038	-4.9640

- All three are trainable, RGB-D not better, two-image best

Valid of RGB-RGB



- Some align well, some have small rotation bias, some fail

Test on Video

fixed-body frame: Red: X-axis , Green: Y-axis, Blue: Z-axis

RGB (3 channels)

RGB-D (4 channels)

RGB-RGB (6 channels)

- Only-Left-Image performs best
  - disparity map and right-image may be distractors

Future Work

- Predict object position
- Use robotic arm to automate building datatsets
- Use CNN to roughly predictly at first, use traditional method to predict precisely

Thank you for your listening, any question?

References

[1] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes, " arXiv:1711.00199v3, 2018.