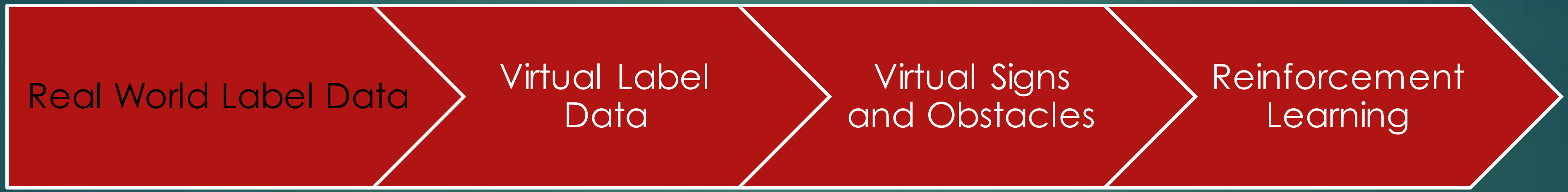
Autonomous Driving via Simulated Label Data

APPROACH AND LESSONS LEARNED

Overview

- Create images which 'replicate' the real-world (camera) collected images for use as labeled data.
- Train the same model used with real-world labeled data on the virtual world images.
- Move the model to the test vehicle and observe 'correct' steering predictions.

Roadmap



Real World Label Data

Virtual Label Data

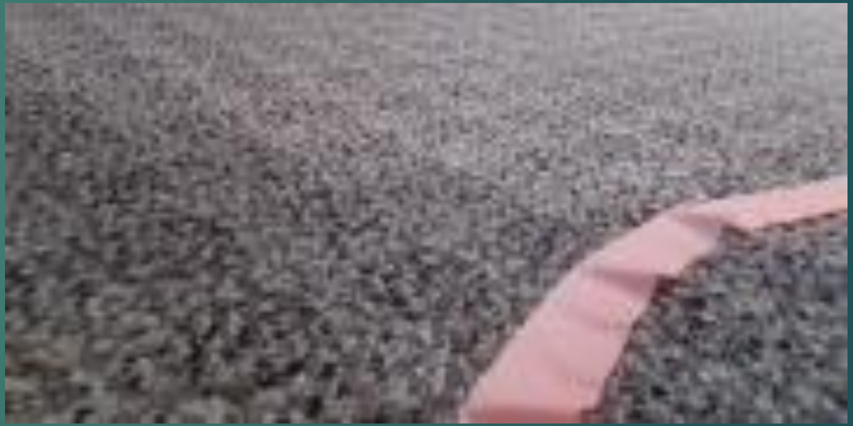
Virtual Signs and Obstacles

Reinforcement Learning

Real World

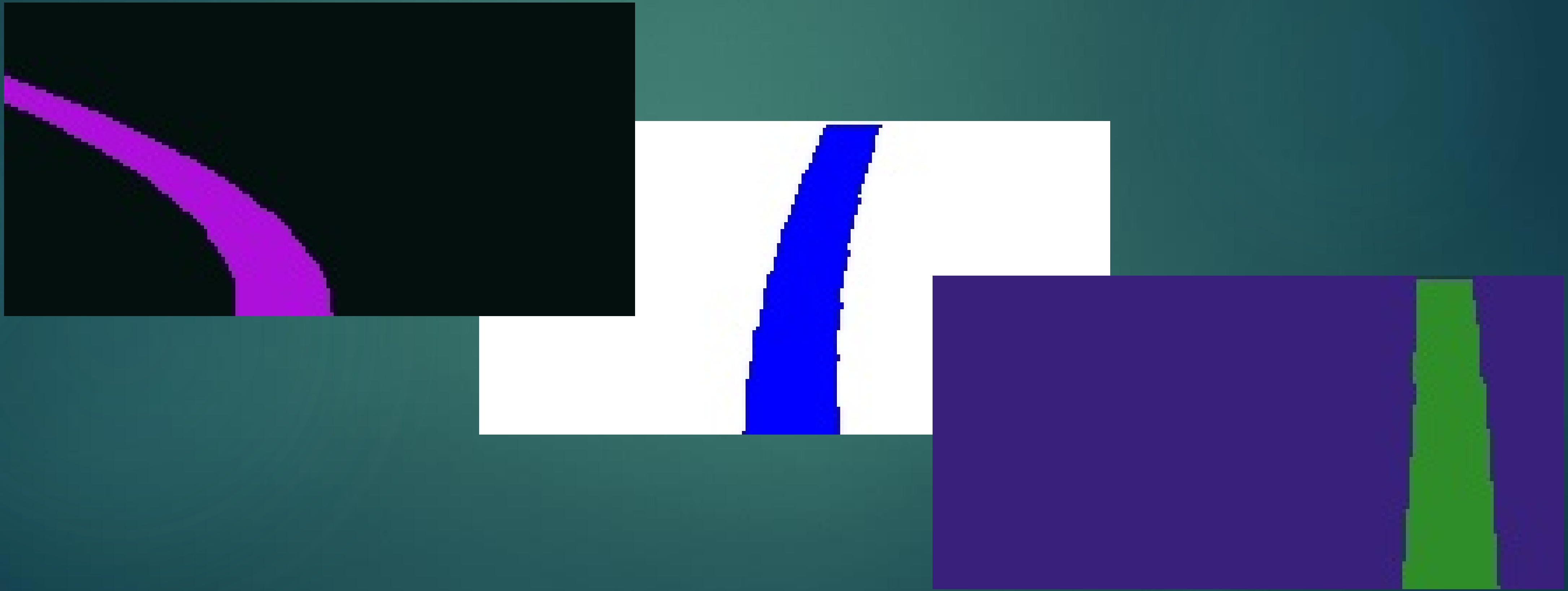
- Sample Labeled Data

- Steering Angle: 463



Simulated Image Label Data

- Virtual Image and Steering Data generation

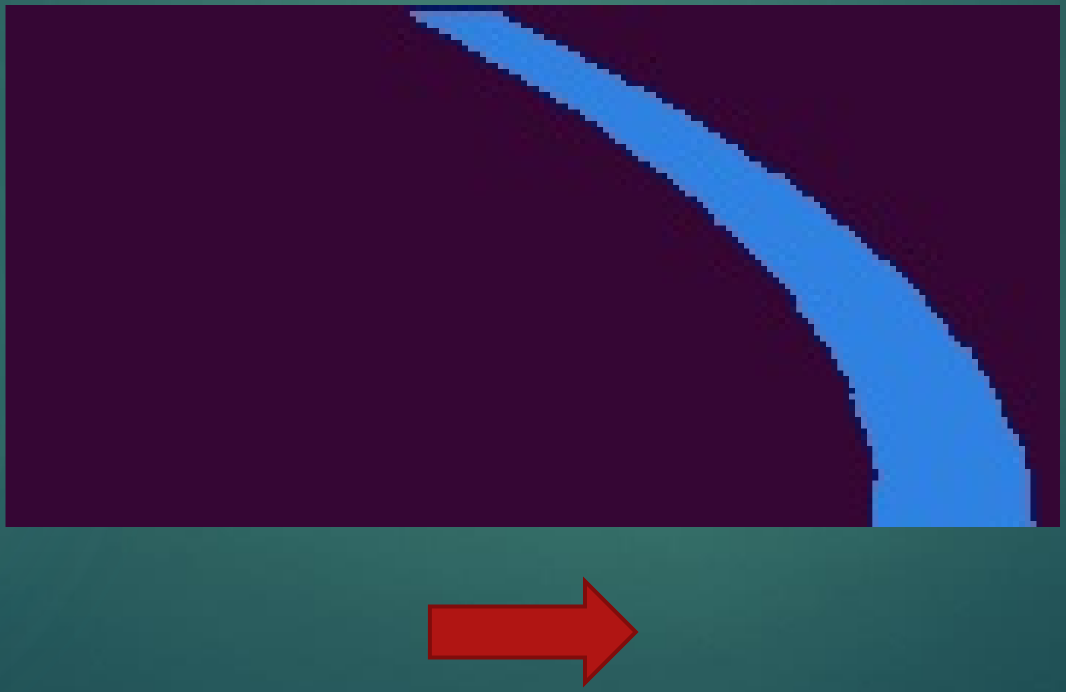


Random Color Background, Random Color Line

- Vary Background and Line

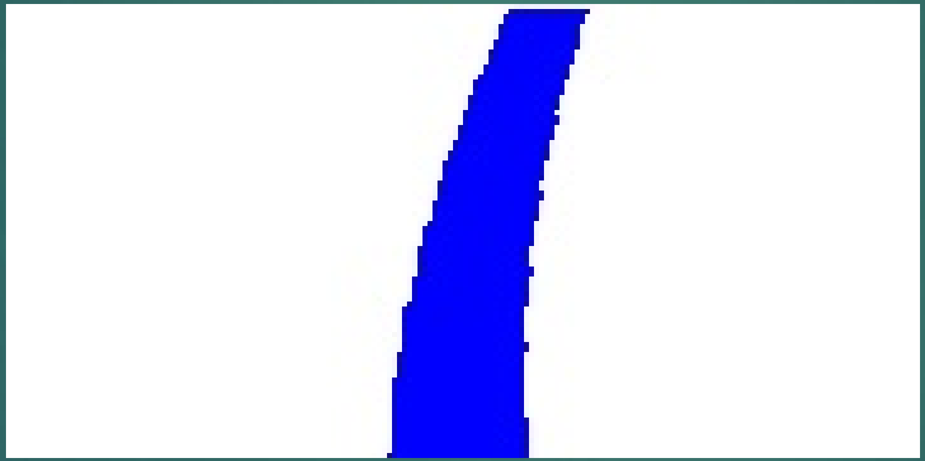
Off-Center Examples

- Steering label data 'compensates' by turning the car *towards* the line.



Simplify

- Single color line and single color background, always centered

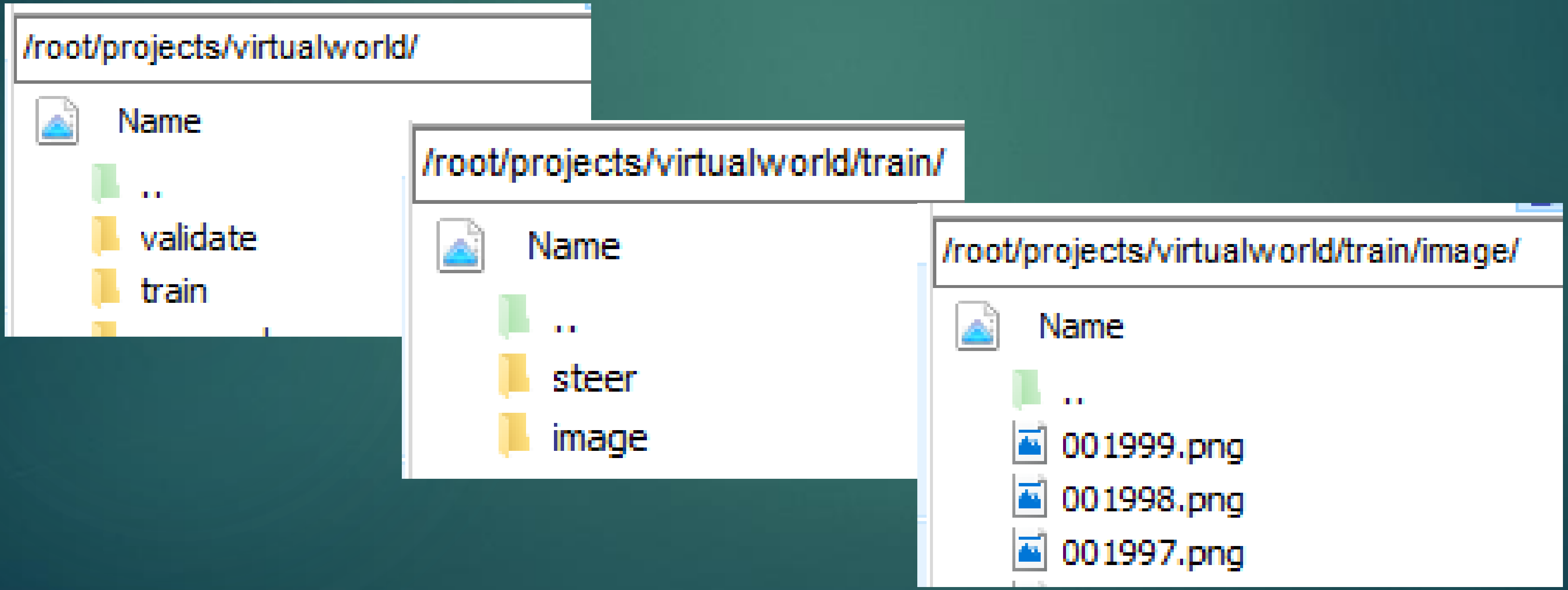


Train with Data Generation In-situ

```
# ==============================================================================
# Yield function
# ==============================================================================
def get_gen(self):
    while True:
        for x in range(l_bound, r_bound):
            image, steer = virtual.generate_frame(True) #indicate training mode is turned on
```

Train with Pre-rendered Data

- Thousands of images where pre-rendered to 'Image' and 'Steer' directories.
- The camera-capture labeled data training program was 'pointed at' the simulated data directories.



/root/projects/virtualworld/

Name

..

validate

train

/root/projects/virtualworld/train/

Name

..

steer

image

/root/projects/virtualworld/train/image/

Name

..

001999.png

001998.png

001997.png

Pre-render Code

- batch_num = 32
- session_t = ['train']
- session_v = ['validate']

- gen_train = BatchGenerator(session_t, batch_num)
- gen_valid = BatchGenerator(session_v, batch_num, jitter = False)

Training Results

- Model never converges.
  - Image generation on-the-fly
  - Pre-rendered images
  - Images with consistent background, line and position
  - Images that only display curves (similar to earlier real-world capture)
  - Images with dropout, blurring, etc.
  - Changed the Loss Function
  - Changed the Optimizer

Adding Noise

- sometime = lambda aug: iaa.Sometimes(0.7, aug)
- sequence = iaa.Sequential([sometime(iaa.GaussianBlur((0, 1.5))),
- # blur images with a sigma between 0 and 3.0
- sometime(iaa.Sharpen(alpha=(0, 1.0), lightness=(0.75, 1.5))),
- # sharpen images
- sometime(iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 3.), per_channel=0.5)),
- # add gaussian noise to images
- sometime(iaa.Dropout((0.0, 0.1))),
- # randomly remove up to 10% of the pixels
- sometime(iaa.CoarseDropout((0.10, 0.30), size_percent=(0.02, 0.05), per_channel=0.2)),
- sometime(iaa.Add((-10, 10), per_channel=0.5)),
- # change brightness of images (by -10 to 10 of original value)
-],
- random_order=True # do all of the above in random order
-)

Image Too Sterile?

- My thoughts are that the computer rendered images do not have anything 'interesting' for the CNN to latch on to.

Next Steps

- Use 'real' background image
- Suggestions?