Sim2Real Training of Autonomous Vehicles

Deric Pang [1,2] Alexey Kamenev [2] Jeffrey Smith [2] Thang To [3]
Adrian Balanon [3] Nikolai Smolyanskiy [2]

[1] University of Washington

[2] NVIDIA Redtail

[3] NV Studios

June 17, 2018

NVIDIA Project Redtail: https://github.com/NVIDIA-Jetson/redtail.

Outline

Motivation

Autonomous robots have many applications.

- Self-driving cars
- Item deliveries
- Military vehicles

However...

- Labeled image data is expensive to collect.
- DNNs do well on vision tasks but they require a lot of data.

Could we train vision models on synthetic data generated by a simulator?

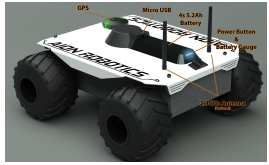
- In a simulation, we have full control of the environment.
- We have 100% accurate labels for things like image semantics, segmentation, and depth.
- It is very cheap to generate a ton of labeled synthetic data.

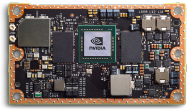
Our Goal

We want to train a rover to navigate an indoor office environment after only seeing synthetic data.

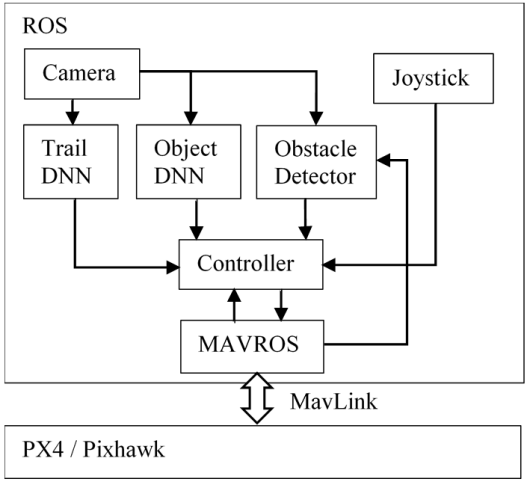
How do autonomous robots work?

- Hardware
  - AION Robotics R1
  - ZED stereo camera
  - NVIDIA Jeston TX2
  - PixHawk 2 flight controller
- Software
  - The Robot Operating System (ROS)
  - MAVROS
  - MAVLink
  - ArduPilot
  - Elbrus Visual Odometry





Fitting it all together



TrailNet

Smolyanskiy et al. (2017) train a neural net to predict orientation and translation of a robot along a trail.

Figure: A drone navigating a forest trail autonomously. Click for video.

TrailNet architecture

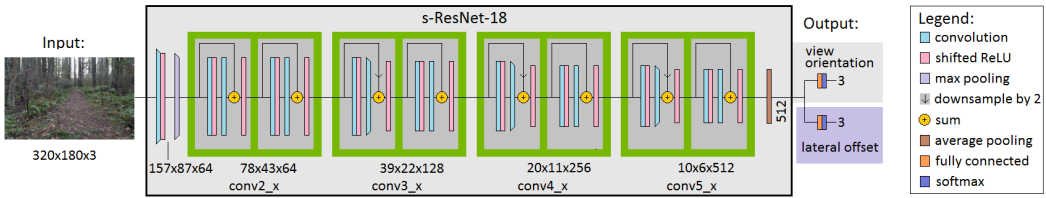


Figure: TrailNet architecture from Smolyanskiy et al. (2017).

For the most part, the network is identical to the standard ResNet-18 architecture.

- No batch norm.
- Shifted ReLU instead of ReLU.

TrailNet inference

- The network predicts a visual orientation vo and a lateral offset lo.
- A turning angle α is calculated with

$$\alpha = \beta_1(y^{vo}_{right} - y^{vo}_{left}) + \beta_2(y^{\ell o}_{right} - y^{\ell o}_{left}) \tag{1}$$

  where β_1 and β_2 are scalar angle parameters that control turning speed.
- α is transformed into a waypoint in the robot's local coordinate system.
- ArduPilot on the PixHawk moves the robot towards the waypoint.

TrailNet training data

- TrailNet is trained on videos taken while traveling down a trail.
- Each image is given two labels:
  - *oriented* **left**, **straight**, or **right** with respect to the trail.
  - *translated* **left**, **center**, or **right** with respect to the trail.

Figure: Left, straight, and right oriented frames from the center translation.

Figure: Straight oriented frames from left, center, and right translations.

TrailNet data collection camera rig

Figure: Nikolai Smolyanskiy holds our camera rig on a forest trail.

TrailNet training data collection

Each fisheye camera captures a wide FOV image.

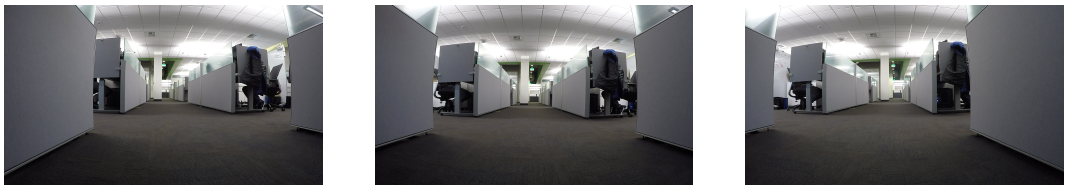


Figure: Left, center, and right camera frames captured from our camera rig.

We crop and undistort each frame into three 60° FOV frames which are offset by 25°. The crops provide visual orientation labels and the camera from which the frame was captured provides lateral offset labels.

A drive through the office

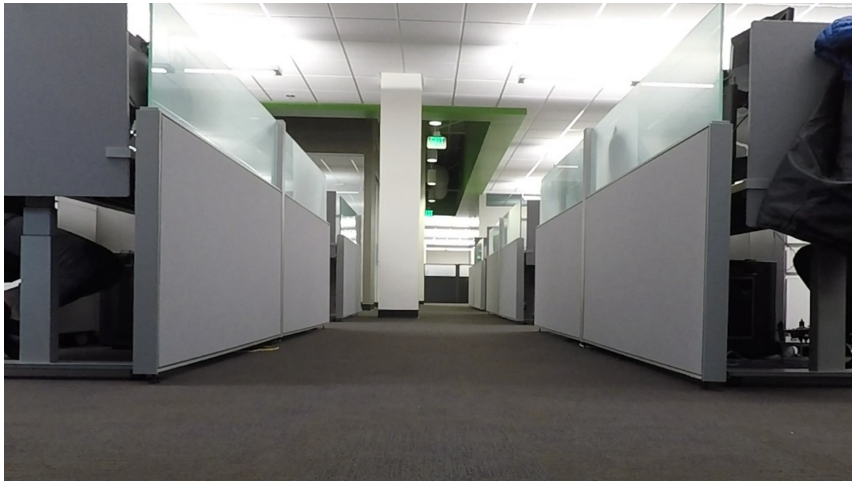


Figure: Inside NVIDIA's Redmond office. Click for video.

Simulator camera setup

Using Unreal Engine 4, we drive a camera rig through a simulated office environment to collect synthetic data.

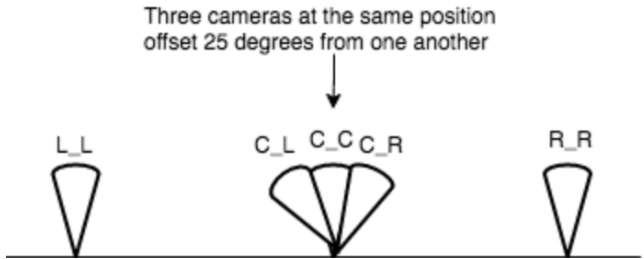


Figure: Simulator camera rig setup.

Simulated office environment

We created multiple office environments that are based on NVIDIA's Redmond office.

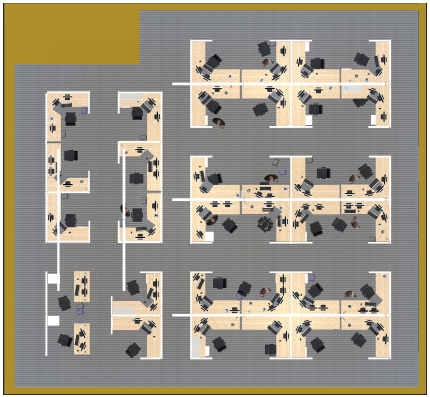


Figure: Bird's-eye view of a simulated office environment.

Figure: A view inside a simulated office environment.

A virtual drive through the office

Figure: A frame from the virtual office environment. Click for video.

Problems with synthetic data

- It is hard to create enough "noise" in the simulation.
  - A real camera gets knocked around and shakes while driving.
  - A real office space has random stuff lying around and is asymmetrical.
  - Real labeled data used for testing will not be perfect like in a simulation.
  - Real lighting can vary drastically around the office.
- Deep models like ResNet-18 will quickly overfit to training data that have little variety.
  - TrailNet converged in one or two epochs on this first synthetic dataset.
  - The model did not generalize to real data at all.

Let's add noise, then!

- The idea is to add so much noise to the training data in places where it "doesn't matter" that the model only learns the important bits.
- We want the model to learn the shape of walls and cubicles and not so much the color of the ground or walls.
- By doing this, the model may have lower training accuracy and test accuracy on synthetic data but will have higher test accuracy on real data.

Domain randomization

- In a simulator, we have full control of our environment.
- We can choose certain aspects of the domain that we don't want the model to learn or that will likely change in the real world.
- These items include:
  - Textures of the walls, ground, and ceiling.
  - Number of objects in the domain like chairs, desks, and people.
  - Light orientation and specular characteristics.

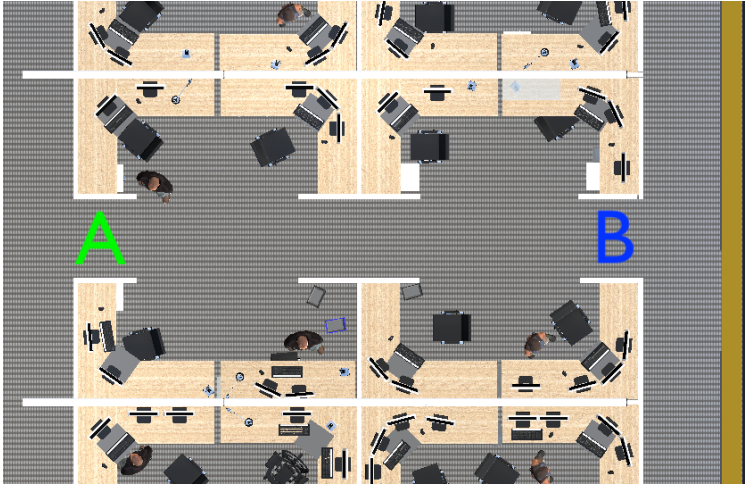
Adding noise to the labels

Consider how a human drives a car in a lane.

- Constantly correcting orientation error.
- There is a range of angles that can be considered as “straight” down the lane.
- We want to emulate this in the training data.

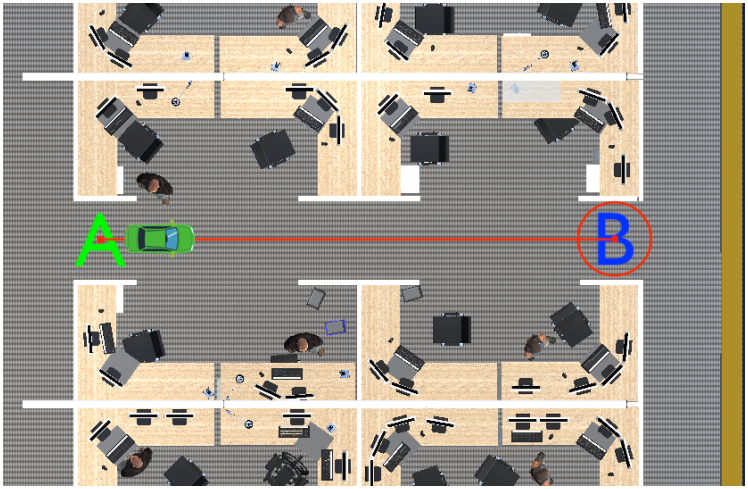
Noisy waypointing in simulation

Consider the task of driving the camera rig from A to B.



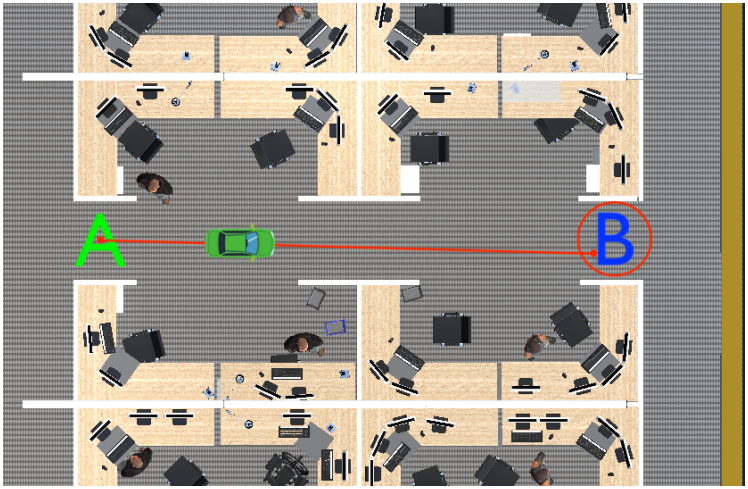
Noisy waypointing in simulation

We define the destination B as some point within a 5 cm radius of a center.



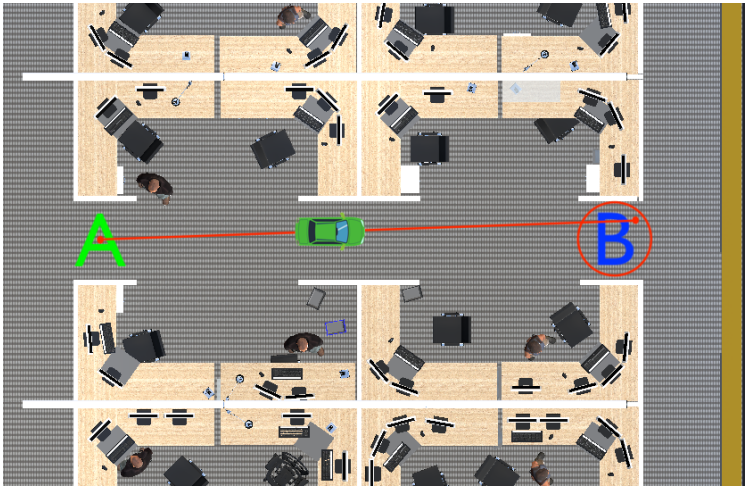
Noisy waypointing in simulation

We sample a new endpoint at every frame. **This makes the data more realistic!**



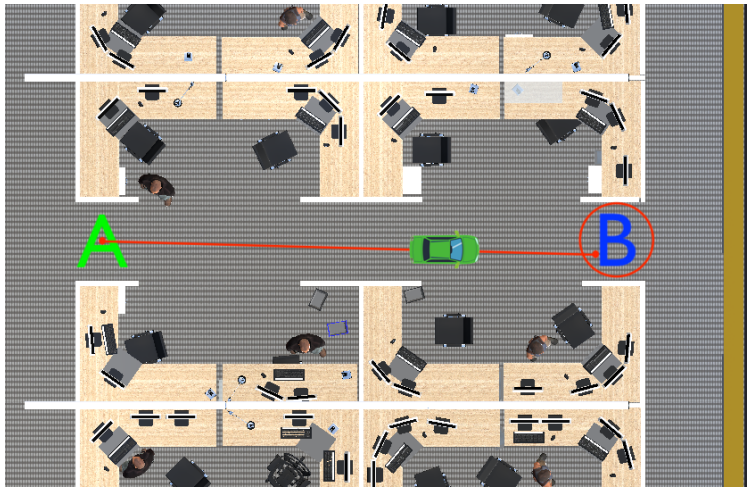
Noisy waypointing in simulation

We sample a new endpoint at every frame. **This makes the data more realistic!**



Noisy waypointing in simulation

We sample a new endpoint at every frame. **This makes the data more realistic!**



A virtual drive through the domain randomized office

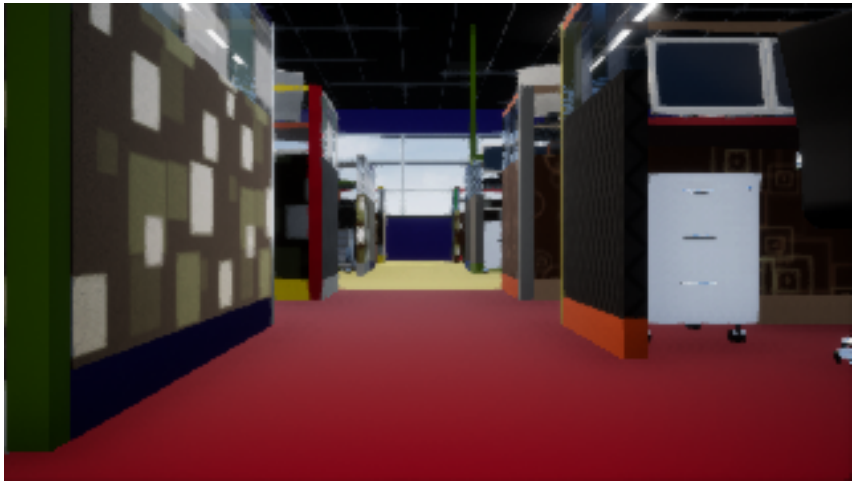


Figure: A domain randomized frame. Click for video.

What are the effects of DR?

Figure: Mean **synthetic** orientation frame.

Figure: Mean **DR** orientation frame.

Figure: Mean **synthetic** translation frame.

Figure: Mean **DR** translation frame.

What are the effects of WPR?

Figure: Mean **synthetic** orientation frame.

Figure: Mean **WPR** orientation frame.

Figure: Mean **synthetic** translation frame.

Figure: Mean **WPR** translation frame.

DR + WPR?

Figure: Mean **synthetic** orientation frame.

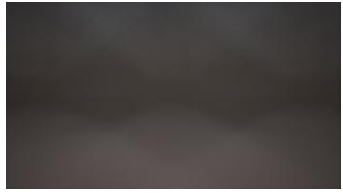


Figure: Mean **DR + WPR** orientation frame.

Figure: Mean **synthetic** translation frame.

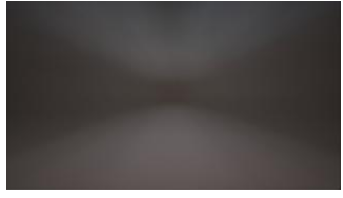


Figure: Mean **DR + WPR** translation frame.

Ablation study

Train Dataset	OTA	TTA
Penta-cam Sim v4	46.61%	35.06%
Penta-cam Sim v4 + WPR	43.67%	36.30%
Penta-cam Sim v4 + DR	53.74%	39.50%
Penta-cam Sim v4 + WPR + DR	50.63%	44.19%

Table: Reporting orientation test accuracy (OTA) and translation test accuracy (TTA).

Real data vs. synthetic data

Figure: Mean **real** orientation frame.

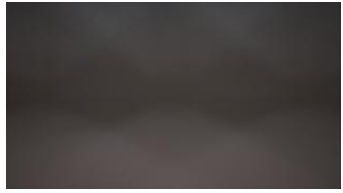


Figure: Mean **synthetic DR + WPR** orientation frame.

Figure: Mean **real** translation frame.

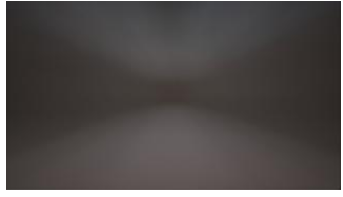


Figure: Mean **synthetic DR + WPR** translation frame.

Datasets that we use

Tri-cam v2

- 365,893 frames.
- Collected by driving camera rig around NVIDIA's Redmond office.

Penta-cam Sim v4

- 20,000 frames.
- Domain randomization of textures and lighting.
- Collected using 4 different paths in simulated office environment.
- Waypoint randomization used along paths.

Model test performances

Train Dataset	Tune Dataset	Tune Epochs	OTA	TTA
Tri-cam v2	None	n/a	83.83%	73.56%
Penta-cam Sim v4 + WPR + DR	None	n/a	50.63%	44.19%
Mixed Penta-cam Sim v4 + WPR + DR + Tri-cam 2	None	n/a	82.14%	74.72%
Balanced Penta-cam Sim v4 + WPR + DR + Tri-cam v2	None	n/a	78.21%	72.92%
Penta-cam Sim v4 + WPR + DR	Tri-cam v2	1	75.61%	69.56%
Penta-cam Sim v4 + WPR + DR	Tri-cam v2	5	82.88%	77.31%
Penta-cam Sim v4 + WPR + DR	Tri-cam v2	10	83.38%	78.02%
Penta-cam Sim v4 + WPR + DR	Tri-cam v2	15	83.60%	75.26%
Penta-cam Sim v4 + WPR + DR	Tri-cam v2	20	83.40%	75.77%

Table: Reporting orientation test accuracy (OTA) and translation test accuracy (TTA).

Model autonomy scores

Using the models with the best test performances from each domain (real, synthetic, real + synthetic), we calculate an autonomy score:

$$a = \frac{\#\text{ commands issued by DNN}}{\text{total }\#\text{ commands issued}}$$

This score is also used in DriveNet (Bojarski et al., 2016).

Train Dataset	Tune Dataset	Tune Epochs	Autonomy
Tri-cam v2	None	n/a	89.83%
Penta-cam Sim v4 + WPR & DR	None	n/a	93.50%
Penta-cam Sim v4 + WPR & DR	Tri-cam v2	10	96.25%

Table: Autonomy scores of the best models in each domain.

Videos!

- Tri-cam v2 in NVIDIA Redmond office
- Penta-cam Sim v4 + WPR + DR tuned 10 epochs on Tri-cam v2 in NVIDIA Redmond office
- Fails

Lessons learned

- The cost to create realistic environments in considerable.
  - Just because it's cheap to collect synthetic data does not mean it will be the most economical overall.
  - Requires people with specialized skills.
- Test score is not necessarily representative of performance on the task.
  - The model with the highest autonomy did not have the highest test scores.
- It is necessary to go beyond just making the data look like real data.
  - Artifacts which appear in real data should be simulated.
- If possible, combining synthetic data with some real data is the best approach.

Acknowledgements

Thanks to the whole Redtail team for supporting me throughout my internship.

Big thanks to Duncan McKay, Thang To, Adrian Balanon and Kirby Leung from NV Studios for gathering requirements, building the virtual office environments and collecting synthetic data for the project.

References

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Nikolai Smolyanskiy, Alexey Kamenev, Jeffrey Smith, and Stan Birchfield. 2017. Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness. *arXiv preprint arXiv:1705.02550*.