Brigham Young University

Practical High Speed Obstacle Avoidance in Unmanned Aerial Vehicles

James Jackson[*], Robert Pottorff [†]

July 7, 2016

1 Introduction

- Why obstacle avoidance?
  - Obstacle avoidance is an essential component of a safe operation of an autonomous agent.
  - At high speeds obstacle avoidance is hard, and hasn't worked well so far
- CS approach to Obstacle Avoidance
  - Obstacle avoidance is a mapping of a sequence of sensor inputs to an action space
  - Considering all the sensor data available at high speeds is intractable, but could become tractable given good abstraction
  - What is the correct high-level space to appropriately handle obstacle avoidance?
- Traditional Approach to Obstacle Avoidance

[*]James Jackson is with the Department of Mechanical Engineering, Brigham Young University jamesjackson@byu.edu
[†]Robert Pottorff is with the Department of Computer Science, Brigham Young University rpottorff@gmail.com

  - Historically, most obstacle avoidance algorithms have relied on hand-crafted, closed-form rules applied to hand-picked transformations of the sensor space such as SIFT [1], SURF [2], or ORB key points [3]
  - These methods use key points to transform sensor data into a 3-dimensional models and optimize potential trajectories to minimize likelihood of collision
  - Some methods have leveraged recent work in deep learning to minimize error between expert data and model data.
  - Few of these methods have shown robustness to sensor choice, particularly at high speeds, and none, to our knowledge, have studied the impact of specialized high level abstractions learned exclusively for obstacle avoidance.

- How to apply reinforcement learning to Obstacle Avoidance
  - Our motivation in this work is to show that the right high-dimensional feature space can enable an agent to perform with greater accuracy and at higher speeds than previous methods.
  - We apply advancements made in reinforcement learning to actively learn high-level representations that maximize performance on obstacle avoidance tasks, enabling us to achieve state-of-the-art performance at high speeds in simulated worlds that can successfully transfer to real-world scenarios.
  - The success of reinforcement learning to this domain points the way to a generalized algorithm for high speed control that is robust to sensor choice.

- Motivate Transfer learning applied to Obstacle Avoidance
  - Critical to reinforcement learning methods is the capacity to learn via trial and error, often a serious problem for real-world agents where failure is catastrophic.
  - To overcome this problem, we also introduce a method for combining real world and simulated data.
  - The success of reinforcement learning to this domain points the way to a generalized algorithm for high speed control that is robust to sensor choice.

2 Background

- Traditional Obstacle Avoidance
  - Most obstacle avoidance algorithms have focused on designing a closed-form avoidance rule given specific inputs.
  - FFLOA [4] - first ROAP, uses a heavy, specialized sensor, awesome results.
  - Schopferer [5] - closest to optimal ROAP. Considers kinematic feasibility of avoidance - What sensor did it use?
  - Oleynikova [6] - Stereo vision FFLOA

  - Saunders [7] - Local memory Voxel Grid-based planning
  - Richter et al. [8] - Path planning based on learned probabilities of collision and optimizing a dynamic model
  - Jackson [**?**] - aimed at reducing limitations imposed by traditional algorithms
  - none of these algorithms are fast enough to navigate a forest at 72km/h
  - Expensive map creation
  - dependence on SLAM
  - assume nominal forward velocity

- Reinforcement Learning
  - reinforcement learning is the process of learning a control policy for an agent using a reward signal
  - this is in contrast to supervised learning where a function is approximated using pairs of input and output signals
  - reinforcement learning allows agents to be trained on environments in which there is no expert pilot data available
  - it also allows agents to improve themselves over their lifetime [9]
  - formally reinforcement learning can be constructed as maximizing discounted total future reward over the execution of the policy
  -

    $$math$$

  - optimal action-value functions follow the form of Bellman equation
  -

    $$math$$

  - generally, reinforcement learning algorithms estimate $Q*$ using iterative update [math] [10]
  - this is impractical for high dimensional feature spaces, so recently deep neural networks have been used to approximate $Q*$ [11]
  - other applications of reinforcement learning to robotic motion in [12, 13, 14, 15, 16]
  - a more complete survey of reinforcement learning in robotics can be seen in [17]

- Deep Neural Networks
  - neural networks are a general function approximator that use alternating layers of linear and non-linear operations with parameters that are optimized with respect to a cost function

  - in context of obstacle avoidance, neural networks have been used in ALVINN [18, 19, 16]
  - recently, a special type of neural network called a convolutional neural networks [20] has vastly improved performance in many vision-based tasks
  - convolutional neural networks have layers that convolve learned filters across the input producing spatially-aware activations
  - because of their success, convnets are extremely common in vision applications
  - pertinent applications to obstacle avoidance include classifying obstacles [21]
  - and supervised learning to map images to action spaces decided by an expert pilot [22, 23]
  - most recently, [24] could follow an unknown trail in a forested area by training a DNN to minimize control error relative to an expert data set
  - in our work, we use DNNs to approximate directly a mapping from sensor input to actions that avoids obstacles while on course to a target
  - we train this network using reinforcement learning

- Transfer Learning
  - Deep learning has so many parameters, needs tons of data AND reinforcement learning needs examples of failure therefore simulation is practically only option
  - Simulation is cheap, but comes at the expense of realism
  - Transferring from simulation to real-life is difficult
  - transfer learning is the process of transferring knowledge embedded in one network into another network
  - in our case we want to transfer learning from simulation to the real world
  - in contrast to [19] which simply constructs an ensemble of networks trained on different input spaces, true transfer learning augments a target with knowledge embedded from a source.
  - this vastly improves performance as parameters can be tuned to minimize error in the target input space in a way that leverages data from both source and target network tasks
  - recent work has been done with policy distillation [25] and progressive networks [26] to transfer policies between networks
  - A new method, which we introduce here, applies techniques originally developed for style transfer to agent policy [27]

- Application of Style Transfer to Policy Transfer
  - recent work has shown gram matrices to be an effective way of capturing style of images [27]

- style transfer works by looking at the inner product of all neurons, and adding an addition component to the cost function
-

$$math$$

- we apply a similar technique to transfer high-level abstractions learned in simulation to networks trained with expert data
- we do this by treating networks trained on simulated data as a source network, and networks trained on real data as a target network
- the additional constraint added during the training of the target network and is defined by the difference in the gram matrices between
- the two networks given the same input image
-

$$math$$

- this constrains the manifold of target network parameters to a stricter subspace defined by the source network, accelerating learning

3 Approach

- From a high level, we construct two simulators
- we train a combination of supervised and reinforcement learning on the first
- followed by fine-tuning using reinforcement learning in the second simulator
- what is simulator A?
  - built in gazebo
  - obstacles are pylons and walls randomly placed (See Figure 1)
  - second order dynamics for pitch/roll/yaw
  - first order dynamics for thrust
  - collision calculation
  - world is guaranteed to be traversable and meet constraints so that turning radius at n mph is at least x, and [8]-like
- what is simulator B?
  - same as A, but higher fidelity and slower
  - obstacles are simulated to look like indoor cluttered environments
- what does our network look like?

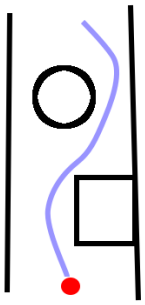

Figure 1: Sample Simulated Environment

  - identical structure to [28] using additional techniques of prioritized replay [citation needed], etc.
  - hyper-parameters
  - topology
  - picture

- how do we train an agent to succeed in supervised simulator A?
  - similar to [22] except with RRT in lieu of expert pilot
  - generate 10m training tuples of (sensor, goal trajectory, expected avoidance trajectory) over 10m (world, optimal trajectory)
  - account for orientation bias in training tuples and real-life noise in sensor data
  - cost function
  - network details

- how do we train an agent to succeed in unsupervised (RL) simulator A?
  - similar to [28] except with pre-trained network and UAV task instead of whatever they use
  - new worlds generated every episode similar to supervised worlds
  - cost function / q learning math
  - terminal state is defined as 5 meter radius to goal
  - discretized reward every 10 meters vs change in inverse distance
  - why discretized reward?
  - negative reward on crash / death

- how do we transfer knowledge from network A to network B
  - network A (trained for simulation) has identical structure to network B (trained for real-world)

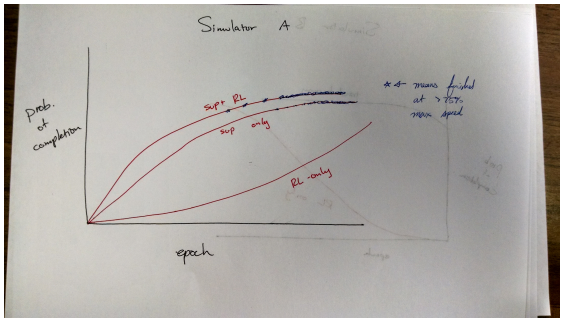

Figure 2: probability of completion versus training epoch

  - train network A using simulation
  - gather data for network B
  - train network B with additional cost of minimizing gram matrix similarity

4 Results

- Simulator A Results
  - How well does the network perform after supervised learning perform? (See Figure 2)
  - How well does the network perform after reinforcement learning? See figure 2
  - How well does sup+reinforce work? See figure 2
- Transfer Learning
  - What was the impact of the Reinforcement learning phase of Simulator B after transfer? (See Figure 3)
  - how many fewer epochs were required with transferred learning?
  - Convolution Filters Between Networks (See Figure 5)
  - How similar are the policies? What are the differences? (See Figure 4)
  - Compare style transfer to Policy Distillation and Progressive Networks
- Network Tuning
  - What did we have to do to make the network learn?
  - Comparison of Topologies
  - Comparison of Hyper-Parameters
  - Time to Convergence
- Simulator B Results
  - How well did it ultimately perform?

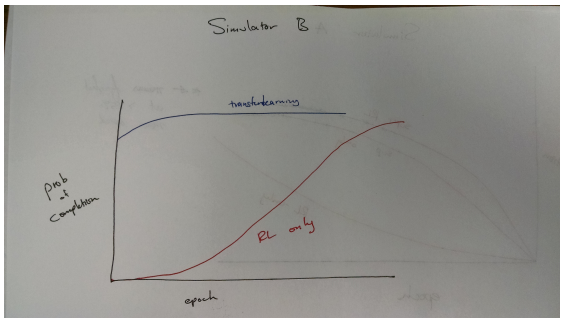

Figure 3: Probability of Completion for the network using style transfer verses a standard reinforcement learner. The network using transfer learning did better

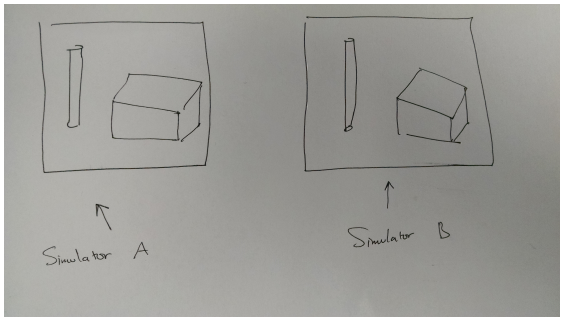

Figure 4: Difference in policy between network trained in Simulator A vs Simulator B

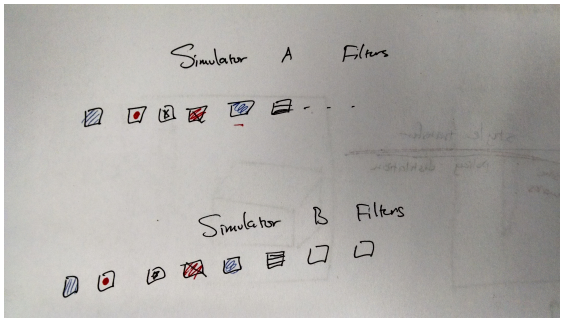

Figure 5: Comparison of Convolutional Filters learned by network trained in Simulator A vs Simulator B

Figure 6: Quadcopter in Simulation A

Figure 7: Quadcopter in Simulation B

– Discussion of Speed (we went really fast)

5 Conclusion

- It works
- We went fast
- We learned some stuff about how style transfer can be used to transfer learning
- We learned how to transfer learning on robotic simulations
- We want to try it in real life
- There are other things we want to try too

6 Appendix

References

[1] J Andrew Bagnell and Jeff G Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1615–1620. IEEE, 2001.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[4] Tao Geng, Bernd Porr, and Florentin Wörgötter. Fast biped walking with a reflexive controller and real-time policy searching. In *Advances in Neural Information Processing Systems*, pages 427–434, 2005.

[5] Circesan Dan C. He Fang-Lin Rodriguez Juan P. Fontana Flavio Faessler Matthias Forster Christian Schmidhuber Jurgen Di Caro Gianni Scaramuzza Davide Gambardella Luca M. Guisti Alessandro, Guzzi Jerome. A machine learning approach to visual perception of forest trails for mobile robots. *Robotics and Automation Letters*, 1(2):661–667, 2016.

[6] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[7] Dong Ki Kim and Tsuhan Chen. Deep Neural Network for Real-Time Autonomous Indoor Navigation. 2015.

[8] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, page 0278364913495721, 2013.

[9] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, pages 2619–2624, 2004.

[10] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[11] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[12] Jeff Michels, Ashutosh Saxena, and Andrew Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. *Proceedings of the 22nd international conference on Machine learning - ICML '05*, 3(Suppl 1):593–600, 2005.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[14] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys. Reactive Avoidance Using Embedded Stereo Vision for MAV Flight. 2015.

[15] Dean a Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems 1*, pages 305–313, 1989.

[16] Charles Richter, John Ware, and Nicholas Roy. High-speed autonomous navigation of unknown environments using learned probabilities of collision. *Proceedings - IEEE International Conference on Robotics and Automation*, (Icra):6114–6121, 2014.

[17] Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.

[18] Stephane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. Learning monocular reactive UAV control in cluttered natural environments. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1765–1772, 2013.

[19] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. IEEE, 2011.

[20] Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *CoRR*, abs/1511.06295, 2015.

[21] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[22] J Saunders, R Beard, and J Byrne. Vision-based Reactive Multiple Obstacle Avoidance for Micro Air Vehicles. *2009 American Control Conference, Vols 1-9*, pages 5253–5258, 2009.

[23] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Srikanth Saripalli. Flying fast and low among obstacles. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2023–2029, 2007.

[24] Simon Schopferer and Florian Michael Adolf. Rapid trajectory time reduction for unmanned rotorcraft navigating in unknown terrain. In *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, pages 305–316, 2014.

[25] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[27] Russ Tedrake, Teresa Weirui Zhang, and H. Sebastian Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, September 28 - October 2, 2004*, pages 2849–2854, 2004.

[28] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. In *The biology and technology of intelligent autonomous agents*, pages 165–196. Springer, 1995.