Vision for Robotics in Optimal Control and Reinforcement Learning Systems

Riley Simmons-Edler
Princeton University
rileys@cs.princeton.edu

Abstract

Robotics is a large and multidiciplinary field that touches many topics, but chief among them is computer vision, as without visual information many robotic tasks would be infeasible. In this paper, we review recent work in robotics for three different but related task domains, pushing/grasping, navigation, and autonomous driving, and how computer vision techniques and representations are used in these systems. We discuss how vision interacts with planning systems such as reinforcement learning and optimal control, and how the field of robotics and vision techniques used in it are changing to meet the challenges encountered, and what future research in each domain might look like.

1. Introduction

Robotics as a field of computer science is innately multidisciplinary, combining mechanical and electrical engineering as well as computer systems and architecture, but is especially closely linked to vision and visual AI. Without visual understanding of a robot's environment, it is difficult if not impossible for the robot to make intelligent decisions and react to its environment. Curiously, while the field of computer vision has seen dramatic breakthroughs in performance and capability in recent years, these breakthroughs have not yet yielded similar breakthroughs in robotics. One reason for this is the complex link between motion planning and visual understanding- Our understanding of how to connect vision with motion planning lags behind performance on benchmark vision tasks, and there remains much disagreement as to what high-level approach is best.

Here, we review the landscape of methods and tasks considered in modern visual robotics. We describe the space of approaches in section 2, being broadly along two axes, corresponding to choice of *motion planning* algorithm, how robot actions are selected, and to the structural assumptions of the vision system it is combined with. We further discuss several major categories of tasks which are studied in robotics in section 3 and how methods for each fit into the landscape defined in section 2. We also discuss the challenges and limitations of existing technique for each domain. Lastly, we discuss some future directions for each area in section 4.

2. Method Overview

Here we provide an overview of the major categories of methods used in visual robotics research. Broadly speaking, there are two major method axes to consider: On the motion planning side, there is the choice of reinforcement learning(RL) versus optimal control(OC), which informs the design of the vision system for the robot. On the vision side, methods can broadly be grouped into model-based approaches that learn a structured model of the world, such as object pose prediction as used by Zeng *et al.* for robotic grasping[34], and model-free methods that map directly from pixels to motions using a neural network, such as Finn and Levine *et al.*'s work on grasping [22][11]. While optimal control and model-based vision are typically associated(and likewise for RL and model-free vision), reinforcement learning allows for both classes of vision system to be used. There also exist systems using aspects of both RL and OC, such as model-based RL using OC to select actions,[11] or which use imitation learning and RL to mimic and improve upon an optimal controller[5],[8],[20].

2.1. Reinforcement Learning versus Optimal Control

Reinforcement Learning First, we will describe RL and optimal control. *reinforcement learning* describes a category of algorithms for training machine learning models to solve Markov Decision Processes(MDPs).[3] An MDP is a process having some state s_t for timesteps $t \in \{1, 2, \ldots, T\}$. At each state s_t an action $a_t \in \{a_1, a_2, \ldots, a_n\}$ from among n possible actions is emitted, with some unknown function $p(s_{t+1}|s_t, a_t)$ determining the following state s_{t+1} from among some (typically large) state space. Actions a_t are selected by an agent $A(a_t|s_t, \theta)$ with learned parameters θ, commonly a neural network. This agent is trained to maximize the expected cumulative reward value $E(R_T)$ emitted by some reward function $r(s_t, a_t)$, with $R_T = \sum_{t=1}^{T} r(s_t, a_t)$. Critically for the training of neural networks, the reward function $r(s_t, a_t)$ does not need to be differentiable or convex. This property allows us to train neural networks to perform tasks in a real world environment where the gradient over action selection $a_i \in a_1, \ldots, a_n$(a.k.a. what would have happened had we selected some other action $a_{j \neq i}$) cannot be analytically determined. One challenge for robotics in particular is that the most general action spaces are often continuous, e.g. joint angles or motor impulses at a given time, which complicates the task. More broadly, generalization to new experiences and the number of experiences required to train the agent are major challenges in RL, and both are major concerns for practical real-world robotics.

Optimal Control *Optimal Control*,[21][25] sometimes referred to under the broader category of *Control Theory*, is a class of algorithms for solving motion problems using direct optimization under a model of the world. Broadly, while RL attempts to directly learn to estimate the behavior of the world and take actions in the world, optimal control assumes some *a priori* model of the world to be true and then optimizes actions to fit the objective given that model. For many problems, OC is well-suited as given an accurate model of the world and a tractable action space to optimize it will perform robustly on new experiences and bounds on performance may be provable. However, given tasks where an accurate model of the environment is difficult to obtain or where the action space is large/highly non-convex and hard to optimize, OC performance will degrade rapidly.

2.2. Model-based versus Model-free Vision

Model-based *Model-based vision* here refers to robotic vision systems that adopt an explicit model of the world, such as trying to predict bounding boxes or 6D poses[34] or a map of the environment,[17] then planning actions using that model of the environment. The model assumed can range from highly prescriptive, such as a physics simulator that only requires a handful of values be predicted given visual observation, to very loose, such as reconstructing a spatial map of the environment directly from images.[16] These methods can be combined with either OC and RL for motion planning, and can be effective and efficient for problems where human prior knowledge can inform good models for the environment.

Model-free *Model-free vision*, conversely, refers to a system which makes no or minimal assumptions about the visual environment, and simply relies on a neural network to model the visual world implicitly as a result of end-to-end training. This class of methods is primarily associated with RL, which provides an objective that requires the network to learn implicitly about the world. Arguably methods relying on visual features transferred from a pretext task, such as image classification, would fall into this category as well. The critical differences among this class of methods are the action/task space used, which can range from continuous joint angles[22] to spatially-mapped action scores,[32] as well as variations in input, such as providing a visual description of the target[37] or multiple viewpoints.[5] These methods are flexible and potentially powerful due to making few assumptions about the structure of the task, but the lack of prior knowledge can make some tasks very slow to train or intractably complex.

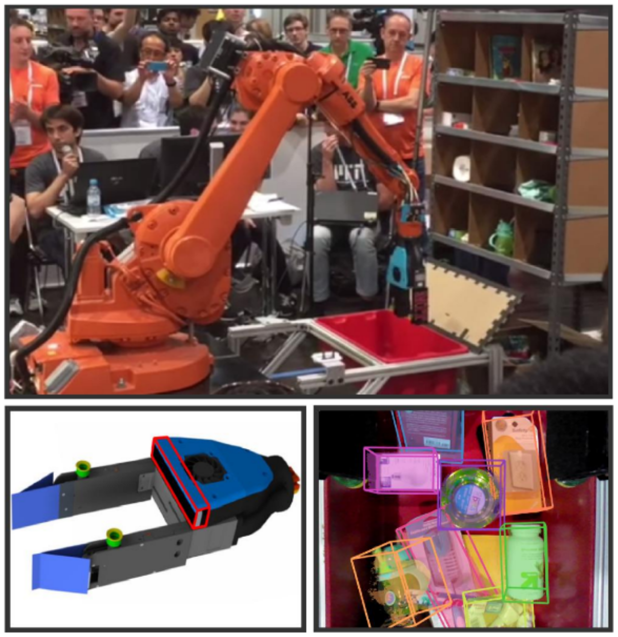


Figure 1. Example of the grasping task and a model-based OC-driven method predicting an object's pose and then optimizing a grasping trajectory to grab it, courtesy of [34].

3. Task Spaces

Here we describe each major category of robotics task we are exploring, including a discussion of each work and how it fits into the overall landscape of methods for the task.

3.1. Pushing and Grasping

One of the major tasks in robotics is to *grasp* or *push*(in the simpler case) objects to achieve some task. There are several notable benchmarks for this task, including the work of Yu *et al.* [31] on a benchmark of pushing tasks and the Amazon Picking/Robotics Challenge,[1] which attempts to provide a benchmark for the task.

Classic Approaches Traditionally, pushing and grasping was primarily considered a physics problem, with the main challenge being in modeling the (typically simplified) environment well enough for optimal control to plan robust movements and grasps[25][21]. Due to the (to this day significant) challenge in designing physical simulations accurate and fast enough for this task, these approaches have increasingly been challenged over the last decade. Goldfeder *et al.* [15] in 2009 produced a dataset of precomputed grasps for diverse object shapes that could be indexed using visual features, allowing closer itegration with vision than prevously. Yu *et al.* [31], developed a large experimental pushing benchmark dataset and demonstrated that many classical simplifications made, such as uniform friction, are not robust in practice. Bauza *et al.* [2] learned a Gaussian model for pushing and provided further evidence that many classic model assumptions used for non-learned systems don't hold in practice. More recently, however, grasping methods have moved towards more on-line approaches, with closer integration of vision and visual feedback into the grasping process.

3.1.1 OC-based methods

In addition to classical work, a large portion of recent work in grasping continues to use OC. As part of the Amazon Picking Challenge[1], Zeng *et al.* [34] learned to predict 6D object pose for use with OC planning and a sophisticated robot arm to simplify the task using suction as well as mechanical grasping. While this method was was effective in cluttered grasping environments and could grasp a wide range of objects, the OC and vision modules were largely independent and sequential, and each fundamentally assumed the other would work robustly. As a result, errors in pose prediction(such as due to a novel target object) usually resulted in grasping failures, and errors in the control module's physical simulations could not be corrected or adapted to using vision. The same authors subsequently developed a method with closer integration between grasping and vision, this time predicting grasp affordances directly from pixels, followed by using metric learning to recognize the grasped object once separated from background clutter(the task involved grasping specific objects from among an unordered collection of objects in a bin)[33]. This method worked better, but their ability to predict grasp affordances was limited by physical obstructions from surrounding clutter, as well as occlusions to the target. To resolve these issues, they developed an RL-based approach that can use pushing to clear obstructions and obtain a better view of a target before grasping, discussed in more detail in section 3.1.2[32].

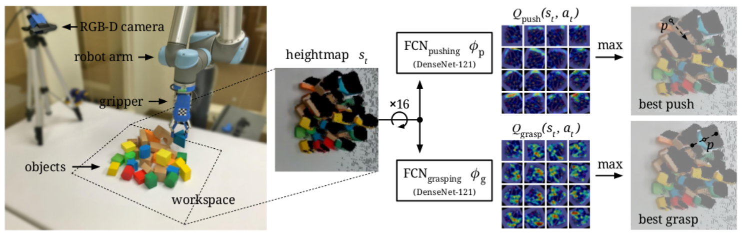


Figure 2. Example of an RL-based model-free approach to grasping, which seeks to address the issue of reactivity and challenging object arrangements. Figure courtesy of [32].

3.1.2 RL-based methods

In parallel with the development of more sophisticated/closely coupled vision models for use with OC, a separate branch of research using RL has developed. The first major work in this area was from Levine and Finn *et al.* in 2015[22], who learned from camera pixels to actions directly for several different grasping tasks. To make training tractable with a real robot, they had to pretrain their RL policy to match the output of OC controllers for motion primatives and their vision system using object detection to learn basic image features. Notably, this system had several major limitations- Robustness to perturbations was poor, and they did not demonstrate any transfer to tasks not trained on.

Another early work in this area was Pinto and Gupta 2015[26], who learned end-to-end grasping by randomly sampling a very large dataset of grasps and predicting the success or failure of sampled grasps directly. While this approach worked well for seen objects, generalization was mixed, and their approach relied on a fixed object detection and grasp proposal algorithm to propose candidate grasps given an uncluttered environment.

More recently, Finn and Levine[11] improved on their previous method through upscaled training with multiple robots, a larger and more complex neural network, and a next frame prediction task conditioned on a specified action produced by their RL policy. By predicting subsequent frames conditioned on a proposed action, they could use their predictions to optimize actions across multiple frames and learn to effectively pick up diverse objects. However, the challenges of future video frame prediction, itself a challenging task with its own body of research, was a limiting factor for their method, which could only function over short time horizons and as such was mostly limited to pushing rather than grasping to manipulate objects.

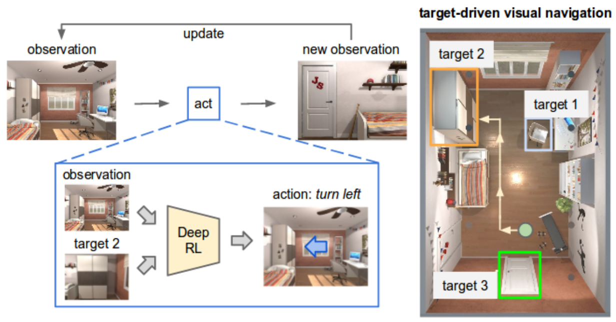


Figure 3. Example of the navigation task and a model-free RL approach to it, courtesy of [37].

Another recent direction of research has used RL and vision to address the traditional weakness of OC-based control- incorrect or missing information about an object to be grasped. While not explicitly tied to robotic grasping, Jayaraman and Grauman[19] used RL to perform active object recognition, learning an RL policy to select new views of an object or scene to aid in object recognition. Such an approach could be used to help overcome situations where the initial view of an object to be grasped is insufficient for recognition or grasp affordance prediction. Zeng *et al.* [32] recently proposed a grasping system with a similar goal, using RL to learn directly from pixels how to combine pushing and grasping blocks to pick up a target block, which may be occluded or physically constrained by other blocks.

3.2. Navigation

Another major task for robots is navigating complex environments using visual information to reach a destination. Benchmarks on this task are more limited, as navigation in physical spaces is difficult to standardize outside of simulators, though fixed datasets such as KITTI do exist[13].

Classical Approaches Interestingly, the core challenges of navigation in a static environment given a reliable map were well solved decades ago, reducing to a tractable optimization problem[6]. Therefore, the major challenges of this field have been to handle cases where a map of the environment is either incomplete, inaccurate, or does not exist(and is not produced along the way). Generating high-fidelity maps of an environment, as part of the broader task of 3D reconstruction, remains generally unsolved, with some amount of error unavoidable. Classically active sensing, such as the work of Davison *et al.* [9] was used to help correct for errors in a pre-existing map, such as long-range drift. This direction of work(which is heavily used in self-driving cars today) is limited by the constraints of high-precision, high sample rate LIDAR sensors(chiefly size and cost), and is unsuitable for many tasks. Other works, such

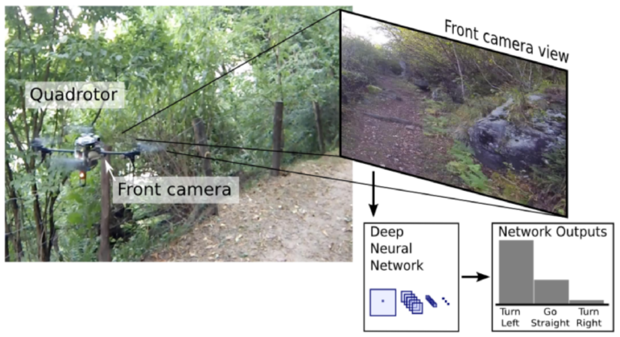


Figure 4. UAV navigation in a forest environment, with trinocular vision. Movement is represented as a single three-choice decision following a trail. Figure courtesy of [14].

as that of Fraundorfer *et al.* [12], produce maps while navigating, but are mostly concerned with producing a complete map of a static environment rather than performing directed navigation in a dynamic environment as humans do.

3.2.1 OC-based Methods

Optimal control for navigation is an approach extensively used in classic methods, and is closely coupled with Simultaneous Localization And Mapping(SLAM) techniques[10], with the goal of building a high quality map on-the-go and using that to optimize future movement(possibly with the goal of improving the map[12]). This body of work is fairly broad and not confined to robotics applications(being a subtask of 3D reconstruction, and referred to as Structure from Motion(SfM) in the non-robotics vision community), and as such will not be discussed in detail here, but is relatively effective for on-line navigation in a static environment. The major limitations of these methods come when environments are not static, such as for a vehicle driving on a crowded city street with moving vehicles and pedestrians, who must be separated from the map and handled by more intelligent vision systems. These challenges can be handled by various vision systems, and are heavily described by the autonomous driving literature, an overview of which can be found here[29].

3.2.2 RL-based Methods

Most recent research on navigation uses some form of RL to perform navigation(at least partly because most of the OC based research has shifted to working on autonomous driving in industrial labs). One major task within this space is UAV navigation. The first modern work in this category largely used imitation learning, the supervised learning approach for learning to solve MDP's, to directly learn from pixels to actions without an explicit mapping step. Giusti *et*

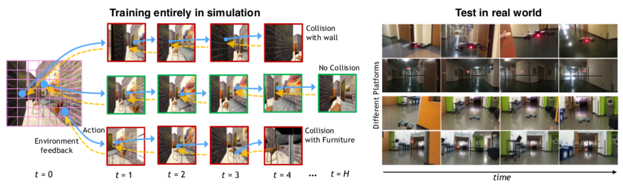


Figure 5. Example of UAV navigation and transfer from simulated to real environment, courtesy of [28].

al. [14] trained UAV's to follow forest trails using a three-camera RGB setup mounted on a human's head as a proxy for ground truth drone trajectories(shown in figure 4). The task of their system was then to predict which direction the path would go(left, right, forward) for each frame of the video, which could then be used to fly a physical drone along a previously-unseen trail with near-human accuracy. While impressive, a constrained action space(only three options), a clearly defined path free of obstructions, and abundant real world training data were required. Similar to [5], the use of three cameras to add robustness to distribution drift by sampling slightly different views was likely important for making this method work robustly.

This method was then extended by other work to improve robustness and reduce the amount of real-world training data required through domain transfer from a simulator. Daftry *et al*. [8] used a simulator along with limited real world data to train a drone capable of navigating with only a single monocular camera with a continuous range of left-right movement options while dealing with obstacles and demonstrating robustness to environment variations such as weather. The use of synthetic pretraining and the large amount of data it provides made this possible, but the method still required fine-tuning on real data to get good performance, and the actual task was relatively simple(go that way as far as you can).

More recently Sadeghi and Levine[28] demonstrated a method for UAV navigation which removes the need for training on real images, with the bold and amusing title of "Real Single-Image Flight Without a Single Real Image." Compared to previous works, they use extensive simulated data along with a Q-learning RL algorithm to learn indoor navigation, a harder task than outdoor navigation in open forest terrain. While their simulator was not particularly realistic in content or rendering quality, they argue that extensive variation of objects and textures in simulated scenes overcomes this limitation. While better than previous work, their crash rate on real environments remains much higher than in simulated environments.

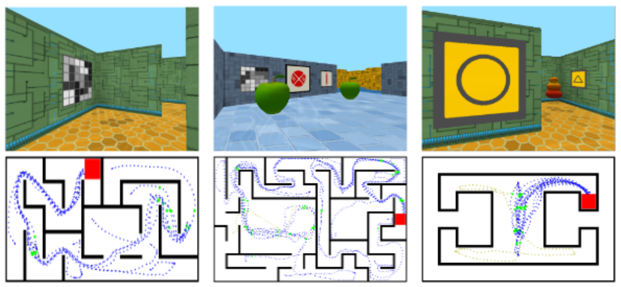


Figure 6. Example of navigation in simulated mazes. While not directly analogous to a robotics task, techniques developed here could be used with physical robots. Figure courtesy of [23].

In addition to UAV navigation, terrestrial robot navigation is also studied with both real and simulated robots. Compared to most UAV navigation, this branch of navigation usually requires handling both more complex and cluttered environments as well as harder navigation tasks, such as moving through mazes[23][4][36] or human CS departments[17][16][37]. Navigation in complex spaces also often requires longer-ranged reasoning than the proximal obstacle avoidance that is handled by UAVs, needing to plan a path to a target over a longer time horizon.

As with the UAV task, simulated training is often necessary due to the sample complexity required and the physical difficulties/hazards of training a robot from scratch in a real environment. The work of Zhu *et al*. [37] is the closest analog to the UAV works previously discussed, using simulated training from pixels to actions, then fine-tuning on small amounts of real data, but adding the additional constraint of needing to locate and navigate to a target specified by an image taken in the scene.

There also exists a significant body of work on navigation purely in simulation. While much of this is outside the scope of our discussion, a few works are worth noting for their implications on future work. Kahn *et al*. [20] propose PLATO, an approach for training an RL policy using an optimal controller as a tutor to speed up training and allow for safety guarantees with an incompletely trained policy. While they only demonstrate it for a simulated UAV task, this approach could help with data collection/training on navigation tasks in the real world without putting humans and fragile objects at risk from misbehaved robots. Mirowski *et al*. [23] and Bhatti *et al*. [4] perform navigation in simulated mazes(example shown in figure 6), but rather than use a model-free approach mapping pixels directly to actions, these methods perform intermediate vision tasks(predicting depth and egomotion loop closures in [23] and performing SLAM and using it as an input to the policy

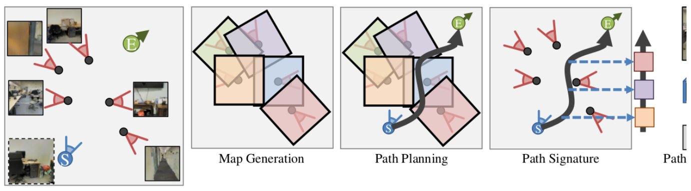


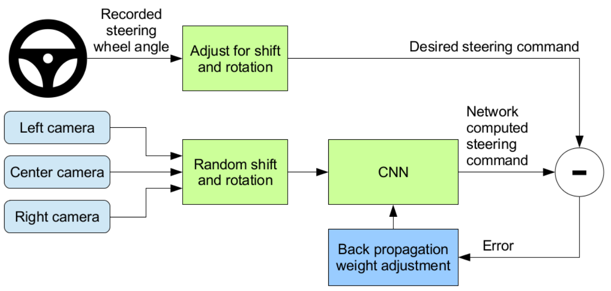
Figure 7. Example of a model-based approach to navigation in CS departments. Figure courtesy of [17].

network in [4]) and show improved performance as a result.

In line with these simulated results, Gupta *et al.* [17][16] propose a heavily modularized approach to RL-based navigation, shown in figure 7. Their system takes as input several landmark snapshots of the environment and a destination, produces a partial map of the environment from the given information, plans a path through the map, recognizes landmarks for correcting egomotion drift from the plan, and finally executes the plan via an RL policy. Notably, each component of this system is learned via deep neural network, although they are not trained in combination. While this work is impressive in its use of explicit vision modules, and tackles the more challenging task of navigation in (simulated scans of) real human CS buildings with only limited data to build a map from, performance remains relatively low, and much work remains to be done on this task.

3.2.3 Autonomous Driving

One additional task within navigation that bares special mention is autonomous driving, which we distinguish from broader navigation due to the specific constraints and simplifications encountered. While a full review of autonomous driving methods is beyond the scope of this work, it is an interesting domain to touch on as it has attained much more real-world traction than most robotics tasks. Most driving systems rely on vision systems to perform vision primitives such as object tracking or segmentation, which are then fed into optimal controllers to handle motion planning. To date, this general approach has been far more successful than RL-based alternatives. As most research of this type is now done in industrial research groups, we will focus mainly on RL-based approaches here that are of more academic interest. [29] provides a review of neural networks for autonomous driving in general for the interested reader. One interesting OC-based work worth noting is DeepDriving[7], which seeks to minimize the amount of explicit world modeling and sensing required for effective OC planning and unlike most such systems uses only front facing cameras to navigate.



Recorded steering wheel angle
Adjust for shift and rotation
Desired steering command
Left camera
Center camera
Right camera
Random shift and rotation
CNN
Network computed steering command
-
Back propagation weight adjustment
Error

Figure 8. Example of the Bojarski *et al.* imitation learning method. Note the use of three cameras and augmentation of inputs. Figure courtesy of [5].

A notable landmark paper in RL-based self-driving is Bojarski *et al.* 2016[5], who use imitation learning to learn directly from pixels to driving directions in the real world. In contrast, most practical self-driving systems use a heavily modularized approach, with discrete vision modules for tasks such as vehicle tracking, drivable surface segmentation, and pedestrian detection, which this method handles implicitly. A schematic of their system is shown in figure 8 One challenge in training directly from human data, as this work does, is that humans typically do not deviate from correct behaviour, and thus when the autonous vehicle eventually does it would not be expected to behave well. By using multiple cameras set at angles from the correct driving direction to provide a “buffer” of driving states that represent deviations from desired behavour and can be trained against, Bojarski *et al.* manage to train a robust system that can self-correct and achieve good performance on real roads(albeit worse performance than publicly reported numbers for state of the art OC-based approaches).

Other work on the topic also exists. Zhang and Cho[35] propose an extension of the Dagger algorithm[27] to make this iterative training and data collection algorithm viable for autonomous vehicles. Xu, Gao *et al.* [30] phrase the problem of driving as one of egomotion prediction, and learn to predict future motion given a current image state and a previous motion trace, which makes the task learnable from crowdsourced vehicle camera data. Recently, Hecker *et al.* [18] proposed to improve existing self-driving systems with a failure prediction module to predict situations where the driving module(whether learned or OC based) is likely to fail so that a human can take over.

4. Discussion

In the beginning of this review, we asked the question, “why do robotic vision systems lag behind traditional vision tasks in performance?” Considering all major domains discussed above, some major unifying challenges compared to computer vision as a whole are:

- Real world data availability. A constant across all domains is the need for simulated data in training, which is largely not the case for traditional vision tasks. This stems in each case from the challenges of collecting large amounts of real world data that can be used for training robotics systems. The main exception to this trend is autonomous driving, where many methods are able to collect extensive real world data, and, partially because of this, display much higher real-world performance than other tasks. To make tasks like grasping and generalized navigation practical it will likely ultimately be necessary to "bite the bullet" and collect large amounts of real-world data.

- Problematic failure modes. For a lot of vision tasks, such as classification or detection, low percentage failure cases are considered acceptable and par for the course. In most robotics applications in the wild even a 1% failure rate would be considered unacceptable, as failures often have severe consequences, from damaged robots to risk of human injury. Even when failure isn't catastrophic, failures can cause significant aggrevation and delay, such as grasping failures in a warehouse causing incorrect product sorting, or navigation failures resulting in robots getting lost or stuck in corners and requiring human intervention.

- The complexity of the task. Most robotic vision systems are holistic in nature, and must implicitly or explicitly perform multiple nuclear vision tasks in combination to succeed. Whereas it is relatively simple to achieve some nontrivial performance on most nuclear vision tasks by throwing convolutional networks at them, the relative scarcity of multitask and multistage vision success stories outside of robotics is telling. Success stories for RL such as Atari games and file system management either require no vision or only very simple visual understanding and can perform fine using a 90's-style LeNet architecture such as in [24].

- Standardization of benchmarks. Unlike vision-only tasks, where benchmark datasets can easily be distributed and compared against, robots are physical and often purpose built to a very high degree. Continued improvements in hardware also make physical standardization undesirable for performance reasons. Regardless of the causes, the lack of benchmarks means not only that it is hard to compare methods apples-to-apples, but that a large amount of the actual work of robotics is technically redundant with previous work. While some progress has been made in this direction in recent years, it remains a fundamental issue for the field and slows down the development of new methods.

However, the field of robotics is not without hope- most of the tasks discussed here can be performed by humans or simpler animals at a very high level of performance, which can be seen as a long-term proof of concept for the field. For example, fruit flies have only 250,000 neurons in total, and less than 1 million synapses(network connections/weights), yet can function on a level that no existing algorithm for autonomous navigation can. More proximally, modularization of tasks and transfer of techniques used in other areas of vision and AI are likely to help move the field forward. We discuss some potential future directions for each domain below.

Grasping While this review only covers a few directions within the field of robotic manipulation(a major omission being development of better optimized grasping arms and structured environments to simplify the problem space for industrial applications), For grasping tasks in relatively unstructured environments, the largest extant challenge seems to be in reactivity and robustness- both vision systems and planning algorithms are prone to failure cases, including situations that are hard to resolve simply by adding additional training data(such the limits of most vision cameras on transparent or mesh-structured materials, or the challenges of planning to pick up amorphous objects such as cloth). Our opinion is that the key to solving this will be through either RL-based systems that learn how to adjust behaviour in response to new data(such as a grasped object slipping out of the grasper), or (likely more proximally) OC-based systems that can plan on-line and use additional vision information to change its trajectory during execution. On this latter point, the transfer of techniques from navigation and self-driving may prove useful, as these domains make heavy use of reactivity to function.

Navigation The main factor in performance within the broader domain of navigation seems to be how constrained the task being addressed is. More constrained tasks, such as UAV navigation or autonomous driving, where environments can be made relatively regular, with well constrained and largely local action and decision spaces, can demonstrate very good performance, while less constrained tasks such as navigating a CS building remain challenging. While for the immediate future OC-based systems are likely to remain dominant for well-constrained tasks, less constrained tasks remain hard for both approaches. Based on recent work, it is likely that regularizing the task via model-based vision systems to improve generalization and training sample efficiency will be key. In addition, effective simulated-to-real transfer will be essential. While not discussed here, we believe that developments in meta learning technique would help maintain performance on real domains relative to simulated ones.

Overall, the problem facing most RL-based driving methods is one of robustness- collecting enough data to train discrete vision modules is already considered an industrial-scale task, and end-to-end learning is likely to increase this need for data. In addition, practical concerns remain, as while some guarantees of sane behaviour can be made for OC-based systems, the lack of interpretability in most deep neural networks means that currently no guarantee can be made that an RL system will not suddenly swerve off the road into a ditch given some noisy input or adversarial image.

4.1. Acknowledgements

The author would like to thank caffeine, without which this manuscript would not exist.

5. References

References

[1] Amazon robotics challenge results 2017. http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-newsArticle&ID=2290376. Accessed: 2018-04-14.

[2] M. Bauzá and A. Rodriguez. A probabilistic data-driven model for planar pushing. *CoRR*, abs/1704.03033, 2017.

[3] R. Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[4] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. S. Torr. Playing doom with slam-augmented deep reinforcement learning. *CoRR*, abs/1612.00380, 2016.

[5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[6] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. In B. Prasad, editor, *CAD/CAM Robotics and Factories of the Future*, pages 144–148, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.

[7] C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. *CoRR*, abs/1505.00256, 2015.

[8] S. Daftry, J. A. Bagnell, and M. Hebert. Learning transferable policies for monocular reactive MAV control. *CoRR*, abs/1608.00627, 2016.

[9] A. J. Davison and D. W. Murray. Mobile robot localisation using active vision. In H. Burkhardt and B. Neumann, editors, *Computer Vision — ECCV'98*, pages 809–825, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[10] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, Jun 2001.

[11] C. Finn and S. Levine. Deep visual foresight for planning robot motion. *CoRR*, abs/1610.00696, 2016.

[12] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. pages 4557–4564, 11 2012.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[14] A. Giusti, J. Guzzi, D. C. Ciresan, F.-L. He, J. Pablo Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, and L. M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1:1–1, 01 2015.

[15] C. Goldfeder, M. T. Ciocarlie, H. Dang, and P. Allen. The columbia grasp database, 05 2009.

[16] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik. Cognitive mapping and planning for visual navigation. *CoRR*, abs/1702.03920, 2017.

[17] S. Gupta, D. F. Fouhey, S. Levine, and J. Malik. Unifying map and landmark based representations for visual navigation. *CoRR*, abs/1712.08125, 2017.

[18] S. Hecker, D. Dai, and L. Van Gool. Failure Prediction for Autonomous Driving. *ArXiv e-prints*, May 2018.

[19] D. Jayaraman and K. Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. *CoRR*, abs/1605.00164, 2016.

[20] G. Kahn, T. Zhang, S. Levine, and P. Abbeel. PLATO: policy learning using adaptive trajectory optimization. *CoRR*, abs/1603.00622, 2016.

[21] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, February 1987.

[22] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015.

[23] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell. Learning to navigate in complex environments. *CoRR*, abs/1611.03673, 2016.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[25] R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[26] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *CoRR*, abs/1509.06825, 2015.

[27] S. Ross, G. Gordon, and J. A. D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artifical Intelligence and Statistics (AISTATS)*, April 2011.

[28] F. Sadeghi and S. Levine. (cad)$ˆ2$rl: Real single-image flight without a single real image. *CoRR*, abs/1611.04201, 2016.

[29] G. von Zitzewitz. ”survey of neural networks in autonomous driving”. 2017.

[30] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. *CoRR*, abs/1612.01079, 2016.

[31] K. Yu, M. Bauzá, N. Fazeli, and A. Rodriguez. More than a million ways to be pushed: A high-fidelity experimental data set of planar pushing. *CoRR*, abs/1604.04038, 2016.

[32] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning. *ArXiv e-prints*, Mar. 2018.

[33] A. Zeng, S. Song, K. Yu, E. Donlon, F. R. Hogan, M. Bauzá, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. A. Funkhouser, and A. Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *CoRR*, abs/1710.01330, 2017.

[34] A. Zeng, K. Yu, S. Song, D. Suo, E. W. Jr., A. Rodriguez, and J. Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. *CoRR*, abs/1609.09475, 2016.

[35] J. Zhang and K. Cho. Query-efficient imitation learning for end-to-end autonomous driving. *CoRR*, abs/1605.06450, 2016.

[36] J. Zhang, J. Tobias Springenberg, J. Boedecker, and W. Burgard. Deep reinforcement learning with successor features for navigation across similar environments. 12 2016.

[37] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *CoRR*, abs/1609.05143, 2016.