Robotics Benefiting From Deep Learning

By Hayden Gemeinhardt

Introduction

A robot is a machine that can sense the world around it and physically act upon those senses.

Often times, these senses are incomplete and robots execute actions based on uncertain knowledge, leading to potentially dangerous situations. As such, it is extremely important to narrow the margin of error to the smallest degree possible.

Robots also need to be flexible and versatile to adapt to a changing environment. This includes maneuvering around obstacles, identifying specific objects like humans or other animals, and responding to sudden complications.

Both of these capabilities are made possible in large part today due to deep learning. Identifying objects, for example, cannot be done effectively through a human programming certain details for the robot to look for. And this can also be applied to nearly every area of robotics, as designing features by hand is typically very difficult.

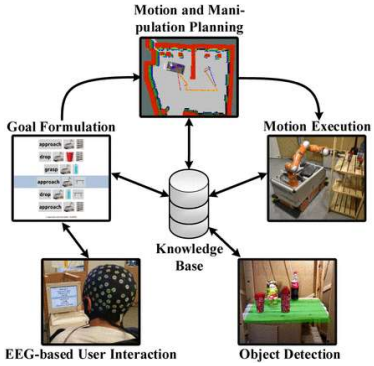
Instead, deep learning is utilized to enhance this process, for a multitude of reasons [1]. For example, deep neural networks can have hundreds of millions of parameters (allowing them to model complex functions for movement, etc.). Additionally, deep learning can form compact representations of sensor data and then create custom feature vectors (a vector containing multiple characteristics of something observed) from sensors using unsupervised learning. Combining the two, and you have a highly optimized method of planning and movement.

However, there are some downsides currently holding this area of research back due to the great challenge they present [2]. Learning the complex functions for movement is time consuming and often requires trade offing complexity for manageability. Interpreting human action for assistive robots (which we will look at) is both crucial and difficult, considering it is often hard even for humans to do so. And complicated control systems can sometimes be the greatest obstacle for certain robots, such as swarms or human hands, that prevents deep learning from being applied in the first place.

In this paper, we will explore the use of deep learning within manipulation planning to create practical and optimal solutions.

Manipulation Planning [3]

A robotic service assistant is a robot used to aid a human in either a specific task or for general purpose. This can include a shopping assistant to carry a basket or a robot who can get objects from across the room. This area of research looks a lot into disability/elderly assistance.



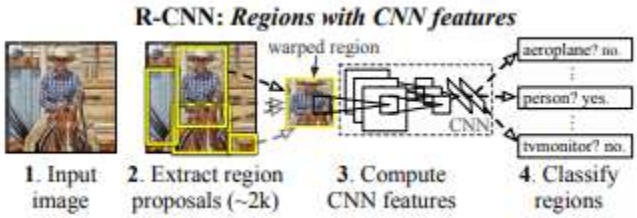
A baseline approach to achieving this goal is manipulation planning. Given an instruction from a user, the robot forms a goal, creates a path and plan to achieve that goal, then executes the plan. It also takes into account things like object detection and other senses to create that plan. The robot here is called a manipulator, interacting with a part. The manipulation task is moving that part to a specified location.

Object Detection [4]

One of the more difficult steps in this process is object detection. Creating a path can typically be done with mapping and localization, but in a real world scenario, this can be very limiting with unknown/moving objects. Deep learning opens this up with two general vision tasks:

Image Classification: given an image or sensor reading, the computer predicts the type of object

object localization: Upon locating an unknown object, the computer marks it with a bounding box



Combining the two allows the computer to achieve object detection. A modern approach to implementing this is with an R-CNN (Regions with CNN features). Given an input image, it first computes object localization proposals, placing bounding boxes around what it believes to be objects (there will be many different proposals). Then, using a CNN, it classifies the object inside the bounding box to say an apple or person or car.

This would require a hefty supervising training approach before applying it to the real world. Current examples of this is with companies like Tesla actively using information gathered from people using their products.

Once the robot can detect obstacles and classify objects, it can then begin planning how to go about a specific task.

Planning [5]

The base foundation for planning can be derived through simple discrete mathematics.

Admissible Configurations

A manipulator, A, needs to have the capability to determine how to safely set a part, P, down. Let C refer to the configuration space to do so, and q ∈ C denote a part configuration.

First, we should remove all configurations where the manipulator collides with obstacles. An equation to portray the manipulator colliding with obstacles is $C_{obs}^{a} = \{(q^a, q^p) \in C | A(q^a) \cap O \neq \emptyset\}$, which means the set of configurations of the manipulator a and obstacles obs equals the set in which the manipulator a and part p is a part of C such that the set of manipulator configurations and obstacles O is not empty (meaning the manipulator exists in the same space as the obstacle). We can also denote the set where the part collides with an obstacle as C_{obs}^{p}, and the set where the manipulator and part collide with an obstacle as C_{obs}^{ap}.

Removing all of these configurations where objects collide gives us the equation $C_{ad} = C \backslash (C_{obs}^{a} \cup C_{obs}^{p} \cup C_{obs}^{ap})$, the set of admissible configurations, where C_{adm} is simply the set of C configurations excluding the collisions.

Stable And Grasped Configurations

The set C_{stable} is a subset of C_{adm} in which C_{stable} is all the stable part configurations for when the robot can safely set an object down. The set C_{grasp} is also a subset of C_{adm} for all grasped configurations. These two sets exclude any collisions due to being subsets of the admissible set. Thus, all safe configurations for an object exist within one of these two subsets, where it is either grasped or stable.

The importance of these two subsets is it specifies when the manipulator can change modes M between transit or transfer. Transit (manipulator not carrying the part) requires $q \in C_{free}$ where $C_{free} = C_{stable} \cup C_{grapsed}$, and transfer (manipulator carrying the part) requires $q \in C_{grasped}$. The modes can only change in the configuration space $C_{transition} = C_{stable} \cap C_{gr}$, or the manipulator cannot transition between grasped/ungrasped without the object in a stable position.

Creating A State Space and Action Trajectory Solution

From this we can create the state space X = C x M where the two modes (transfer and transit) M each contain the set C. The set $X_{free} = X \backslash X_{obs}$ is the set of states excluding states with a collision with an obstacle (similar to admissible configurations) and $X_{transition} = \{(q, m) \in X | q \in C_{transition}\}$.

We can now create a task. Setting an initial part configuration, $q_{initial}^{p} \in C_{stable}$ and a goal part configuration $q_{goal}^{p} \in C_{stable}$, the algorithm can compute a continuous path X = $\{X_{goal} \in X_{free} \mid q_{goal}^{p} \in C_{stable} \cap \tau(1) = q_{goal}^{p}\}$ and an action trajectory τ: [0,1] → X_{free} meaning the algorithm can decide if a combination of a path and action trajectory exist. We can also say the solution $\tau(s) \epsilon X_{transition}$, where the solution alternates between two layers of X (transit and transfer) when $x \in X_{transition}$.

Now, we have a robot capable of detecting objects and obstacles, and then being able to move an object from one place to another while avoiding obstacles.

Optimizing With Machine Learning

We have created a set of rules, but they remain unoptimized. It simply looks for a solution instead of the best solution.

As I stated in the introduction, a human designing features by hand is very difficult and inefficient compared to a computer. Humans would take a lot of time to custom program every different movement the robot can make. Instead, with these rules, we can create what is called a rule-based system where the manipulator can take this set of rules and find the best solution through deep learning.

For example, in a study looking at optimizing grasps in motion planning for e-commerce warehouse robotic arms [6], they used deep learning to reduce the overall amount of jerking caused by the robot. A bottleneck around planning and execution of robot motion was addressed with a Grasp-Optimized Motion Planner (GMOP) but resulted in jerking (fast acceleration/deceleration) and thus reducing the service-life of a robot with constant wear on the gears. The study first looked at introducing a "jerk limit", though that resulted in computations requiring tens of seconds. Using a DNN, they were able to learn trajectories, then "warm start" the GMOP with approximations from the DNN, reducing the computation time to milliseconds and therefore making the jerk limits practical, increasing the life-span of the robots.

Personal Remarks On Further Optimization

I think it would be interesting to look into further improving upon this by putting it through a rule-based machine learning algorithm (RBML) (of which you can find in my paper here) to optimize the rules.

RBML systems have been used in the past to train a computer to manage a wind-energy farm. So say we instead have a warehouse completely managed by a computer and a fleet of robots (a high possibility in the future). How would we go about creating rules for all of these robots to follow in the most efficient manner?

Simply put, there is no good way for humans to do so. Instead, we could use a learning classifier system trained through a simulation, then apply that learned set of rules to real world robots.

In an Amazon warehouse, items are not sorted logically. Instead, they embrace chaos and randomly store things so one item may be able to be found in a hundred different places instead of one and thus make it easier to find. This made sense in the past. Now, with a fleet of robots carrying the shelves around for us, can we optimize it beyond randomly storing?

One of the main goals of RBML is to find rules humans would have never thought of. Humans have currently thought of the most optimal system of randomly storing things, but what if an RBML system could find a better way?

Say the day comes when Amazon uses robotic arms to store shipped in items instead of humans. Given past training, the robot could learn to strategically store items using a complex set of rules without skipping a second. Humans do not have this capability, but robots do.

In computer science, we learn of different data structures for different situations, such as a skiplist using a randomized algorithm (similar to Amazon's warehouse). However, what if we learn to model the warehouse more like a binary heap for a priority queue? You may have a worse worst case scenario, but in trade off your best case scenarios are more common.

We could run an algorithm through association rule learning (an RBML technique) and the computer would then be able to predict certain rules, such as winter implies Christmas decorations have high priority and vice versa. Or as I said, a learning classifier system to completely overhaul the management process.

I would not be surprised if they are currently working towards that, though I could not find a paper talking about it. Perhaps first they would need to invest in another fleet of state-of-the-art robots.

This was simply a bare-bones example of how RBML could be applied to a manipulation planning algorithm on a large scale.

Conclusion

In this paper, we looked how deep learning can be applied to make a simple retrieval task for a robot practical and successful in a real world scenario. We can use object detection to find objects and avoid obstacles, then using the detection method, find a path based on a set of rules. Further optimizing this process can be done with deep learning, and possibly rule based machine learning.

Sources

[1] Lenz, I. (2016). *Deep Learning For Robotics.* Cornell. doi:https://cs.stanford.edu/people/asaxena/papers/ianlenz_phdthesis.pdf

[2] Pierson, H., & Gashler, M. (2017, July 22). Deep Learning in Robotics: A Review of Recent Research. Retrieved November 21, 2020, from https://arxiv.org/abs/1707.07217

[3] Kuhner, D., Fiederer, L., Aldinger, J., Burget, F., Völker, M., Schirrmeister, R., . . . Burgard, W. (2019, March 20). A service assistant combining autonomous robotics, flexible goal formulation, and deep-learning-based brain–computer interfacing. Retrieved November 21, 2020, from https://www.sciencedirect.com/science/article/pii/S0921889018302227

[4] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014, October 22). Rich feature hierarchies for accurate object detection and semantic segmentation. Retrieved November 21, 2020, from https://arxiv.org/abs/1311.2524

[5] LaValle, S. M. (2014). *Planning algorithms*. New York (NY): Cambridge University Press.

[6] Ichnowski, J., Avigal, Y., Satish, V., & Goldberg, K. (2020, November 18). Deep learning can accelerate grasp-optimized motion planning. Retrieved November 21, 2020, from https://robotics.sciencemag.org/content/5/48/eabd7710/tab-pdf