Epsilon Optimal Path Planning for Active Vision for Grasping

by

Galen Brown

M.S. Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

November 2022

APPROVED:

Professor Berk Calli, Thesis Advisor

Professor Craig Shue, Reader

Professor Craig Shue, Department Head

Abstract

In this work I explore the use of active vision algorithms to improve robotic grasping. Robotic grasping algorithms aim to find a suitable grasp location on a given target object using visual data, i.e. images taken from a camera. Their performance depends on the camera viewpoint; some viewpoints are more suitable to detect a good grasp location than others. The role of active vision is to alter the camera viewpoint to help the robotic grasping algorithms. I provide an extensive overview of active vision strategies to find 'sufficiently good' grasps with as little camera movement as possible. I present several heuristic and data driven approaches to the problem in a constrained, discretized scenario and compare their performance. In addition, I demonstrate a method to find solutions in a more realistic, continuous scenario that are within a known error bound of optimal solutions. I outline the mathematical basis for this claim, and demonstrate its empirical characteristics in a number of simulated experiments. Using this information, I am able to show the limitations of current approaches and demonstrate that significant improvements in performance can be made by working in the continuous space rather than constraining the problem to the discrete space. This work provides novel information about the theoretical limits of active vision, which suggest directions for future research.

Acknowledgements

I would like to thank Sabhari Natarajan, my partner for the first year of research, for his invaluable assistance preparing the environment and implementing the discrete algorithms. The work presented in Chapter 4 was conducted together, the work presented in Chapter 5 was conducted after he graduated.

I would like to thank Albi Marini, the 2022 MER lab intern, for his help proof-reading and preparing graphics. I would also like to thank the other researchers in the MER lab; Abhinav Gandhi, Avnish Gupta, and Sreejani Chatterjee, for their advice and suggestions with the project.

I would like to thank Yunus Telliel for his advice and help with the broader impacts section.

Lastly, I would like to thank my advisor, Berk Calli, for his extensive help revising and editing the thesis. For as long as we have been working together, I have been privileged to be able to depend on his support and direction.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotic grasping is a vital capability for many tasks, particularly in service robotics. Most grasping algorithms use data from a single viewpoint to synthesize a grasp [2]. This approach attempts to create a single, master algorithm that is useful for all objects in all situations. Nevertheless, these algorithms tend to suffer when the viewpoint of the vision sensor is different than the images used in training [3]. Additionally, many graspable objects have observation angles that are "singular" from which no grasp can be synthesized. For example, if an object has only one graspable surface, which is self-occluded from the current viewpoint of the camera, the grasp synthesis algorithm would either fail to find any grasps or would need to rely on assumptions that might not always hold, and therefore lead to an unsuccessful grasp attempt.

The issues of these single viewpoint approaches can be addressed via active vision frameworks, i.e. by actively moving the camera and collecting more data about the task. There are several established methods for doing this. At one end of this

This thesis is partially based on a paper published in Frontiers in Robotics and AI [1]. The discrete work was presented in [1], the continuous work has not been published.

The author of this thesis is a trainee in Future of Robots in the Workplace – Research & Development (FORW-RD) NRT Program.

spectrum are methods that collect data to obtain a complete 3D model of the object. These approaches are slow, difficult to carry out in the real world, and vulnerable to misalignment if conditions change during or after data collection [4]. On the other end of the spectrum, there are algorithms designed to collect the minimal amount of data possible to produce a grasp. It has been shown in the grasping literature that even algorithms tailored for single viewpoints can see substantial performance boosts by applying very simple data collection procedures [3].

This motivated my research on active vision, partly presented in [1]. In it, we explored various heuristic and machine learning methods to find the shortest path along a viewsphere (an imaginary sphere surrounding the object that the camera is constrained to move along) that contained enough information to produce a grasp. We restricted the camera to moving in fixed steps in one of eight directions, discretizing the problem. By doing this, we were able to exhaustively explore the search space to find provably optimal solutions using Breadth First Search. This in turn allowed us to provide meaningful comparisons of our techniques to the best possible performance. That comparison is useful for two reasons: first, it shows where there is room for improvement and where current methods are satisfactory. A major conclusion of that work was that while the ratio of each method's results to one another was fairly consistent between objects, the ratio of those results to the optimal results varied quiet significantly between objects. This leads to the second use for optimal performance- to empirically determine how difficult the problem is for different objects. This cannot be done using relative results, and so optimal results allow qualitatively different research.

Given the utility of optimal results for the discrete problem, I then attempted to extend our work into continuous space. This necessitated some compromises. An exhaustive approach was impractical, but an approximation of true best performance proved tractable. My reasoning for the approximation was as follows: each point of the object is visible from a finite region of the viewsphere. Every potential grasp for a two fingered gripper consists of two object points. If a path along the viewsphere runs through the regions of visibility for both halves of a grasp, a camera following that path will collect enough information to synthesize a grasp. By approximating the region with a dense group of points, I can approximate the true distances between each region and make the approximation arbitrarily close to the true distance by increasing the density of the points. The path from the camera's start position to the points on the viewsphere can then be calculated, and the shortest combined path will be an approximation of the true shortest path, as shown in Figure 1.1.

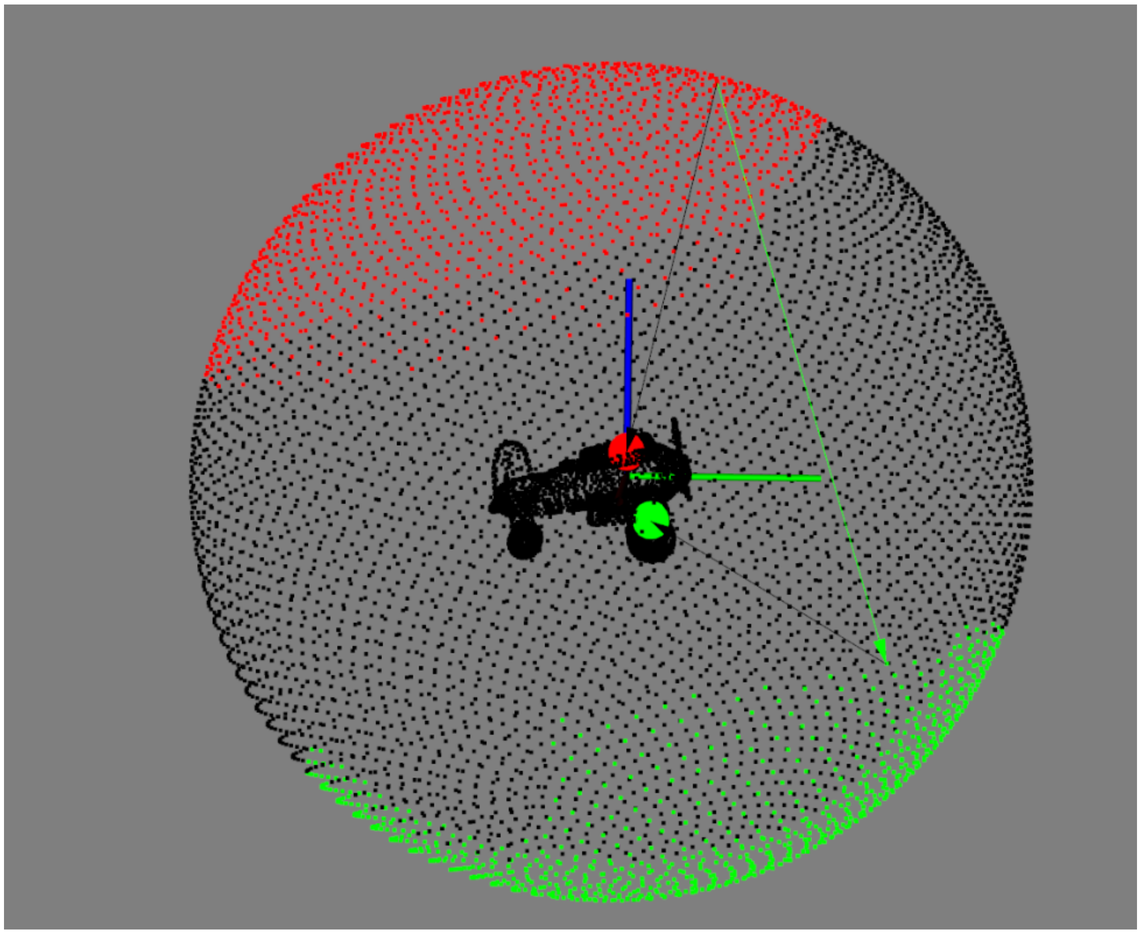

Figure 1.1: An example of the proposed method. The object in black has a pair of points on its surface (highlighted in green and red) which can form a grasp. Each point is visible from the region of the viewsphere shown in the corresponding color. The green arrow shows a potential path for the camera to follow from a start position which can view the red point to an end position which can view the green point. Note that while the green arrow directly connects the points for visual clarity, the camera is constrained to the viewsphere and all distances along it are measured by great circle distance.

This work presents the following contributions:

1. A selection of Heuristic and Machine Learning approaches for discrete search based on our work in [1]. As that work demonstrated, these represent approaches comparable to the state of the art for this problem.

2. Direct, empirical comparisons between the approaches and optimal, pessimal,

and naive approaches. We first presented these comparisons in [1], where we noted that the combination of these three benchmarks allowed us to make novel conclusions about the difficulty of the problem and the theoretical limits of improvement.

3. An algorithm to find epsilon-optimal paths for camera motion for grasping in continuous space, including runtime analysis and proof of correctness. This approach is novel in the literature, and allows us to extend the claims made about the discrete problem into the continuous space.

4. Extensive simulation testing of the epsilon-optimal algorithm to determine its typical, rather than worst case properties.

5. Comparison between continuous and discrete optimal paths, demonstrating and quantifying the trade offs between speed and accuracy both methods take, and providing concrete suggestions for future work.

Chapter 2

Related Work

2.1 Active Vision

Adapting robotic manipulation algorithms to work in an imperfect and uncertain world is a central concern of the robotics field, and an overview of modern approaches is given by [5]. At the same time, much of the research on robotic grasping does not attempt to move the vision sensor and focuses on single image grasp synthesis. In [6] a method is presented for grasping objects in cluttered environments that acknowledges the problem of accurately sensing a complex 3D environment. However, rather than collecting more information, their approach attempts to overcome the limitations of single viewpoints by storing prebuilt 3D models and using them to better analyze a single stereovision image. In a similar vein, [7] approaches industrial grasping by trying to more accurately map known features to objects instead of by trying to collect more data to resolve ambiguities in the images. A typical approach in literature is to train a neural network to produce grasps by annotating individual images with grasp candidates. This is the method used by [8, 9], among many others. Even in tasks peripheral to grasping, like shape and pose estimation, considerable work has gone into more refined algorithms and machine learning strategies for extracting information from single 2D images without attempting to capture more images or optimize the viewpoint [10, 11]. Most work assumes that the viewpoint is fixed or random, and so focuses on either processing the data (pose estimation, object localization and segmentation, etc.) or synthesizing a grasp from available data [12, 13].

Our research focuses on the problem of collecting new data to improve processing outcomes. Active vision has been applied to many aspects of machine vision, but often with the explicit goal of completely reconstructing the 3D model of an object [14, 15], rather than our objective of viewing just enough of the object to perform a grasp. Even in the narrower domain of active vision for grasp synthesis, not all work relates to our concerns. For instance [16]'s study on industrial grasping uses active vision to assist feature identification of known objects, but with the explicit goal of maximizing grasp precision rather than minimizing the information that needs to be collected to find a sufficiently good grasp. For the use of active vision to address grasping using incomplete information, there has been research into both algorithmic [17, 18] and data-driven methods [19, 3, 20, 21], with more recent works tending to favor data-driven approaches [2]. In particular, the work in [3] demonstrated that active vision algorithms have the potential to outperform state of the art single-shot grasping algorithms. [17] proposed an algorithmic active vision strategy for robotic grasping, extending 2D grasp stability metrics to 3D space. As an extension of that work [20], the authors utilized local optimizers for systematic viewpoint optimization using 2D images. [18] employs a probabilistic algorithm whose core approach is the most similar to our heuristics presented in Section 4.4. Our approaches differ in focus, since [18] selects viewpoints based on estimated information gain as a proxy for finding successful grasps, while we prioritize grasp success likelihood and minimizing distance traveled.

The data-driven approach presented in [3] avoided the problem of labeled data by automating data labeling using state of the art single shot grasp synthesis algorithms. They then used machine learning to estimate the direction of the nearest grasp along a view-sphere and performed gradient descent along the vector field of grasp directions. This has the advantage of being continuous and fast, but our discrete testing framework cannot be used for continuous algorithms [3].

One of our data-driven active vision algorithms utilizes a reinforcement learning framework. A similar strategy for active vision is used by [19] to estimate an information gain maximizing strategy for object recognition. We not only extend Q-learning to grasping, but do away with the intermediary information gain heuristic in reinforcement learning. Instead, we penalize our reinforcement approach for each step it takes that does not find a grasp, incentivizing short, efficient paths.

This work draws significant inspiration and code from our previous work [1], which was itself an extension of [22]. Here I have expanded the workspace of those papers from the discrete to the continuous space and provided more theoretical claims, but I have as much as possible retained their methods and settings to allow for comparable results.

2.2 The Use of Optimal or Near Optimal Solutions

[23], while focused on object classification rather than grasping, heavily influenced our theoretical concerns and experimental design. Their paper argues that contemporary machine learning based active vision techniques outperform random searches but that this is too low a bar to call them useful and demonstrates that none of the

methods they implemented could outperform the simple heuristic of choosing a direction and moving along it in large steps. Virtually all active vision literature (e.g. [24, 25]) compares active vision approaches to random approaches or single-shot state of the art algorithms. While there has been research on optimality comparison in machine vision [26], to the best of our knowledge, it has never been extended to 3D active vision, much less active vision for grasp synthesis. Our simulation benchmarks are an attempt to not only extend their approach to grasping, but to quantify how much improvement over the best performing algorithms remains possible.

The benefits of having a provable best solution extend beyond directing research. Recent studies by [27] and [28] show that optimal or near-optimal initialization allows Q-learning to produce near optimal results rapidly and for novel situations.

With these motivations in mind, there has been a great deal of work done on shortest path problems. Prominent examples include [29] and broad overviews of modern and historical techniques can be found in [30, 31]. Unfortunately, these approaches are based on graph theory, and can only interface with the problem if it can be reduced to a graph traversal; this is simple in the discrete case and difficult in the continuous case.

Fortunately, the value of approximate solutions to otherwise intractable problems has also been well established. Approximate solutions for NP-hard problems are an area of sustained and exciting development. It is generally believed that NP-hard problems are unsolvable in polynomial time, but a wide variety of problems can be approximated to varying standards of accuracy in polynomial time [32]. These solutions allow for practical applications which depend on "sufficiently good" approximations of optimality.

Even when exact solutions are possible, approximations may be preferable. Randomized primality testing is a classic example. Even though randomized algorithms can never prove a number is prime, they can produce an error bounded estimate of whether it is prime very quickly, enabling enormous practical applications [33]. Of particular relevance to this work is the development of “Distance Oracles” [34], algorithms that construct a data structure in preprocessing and then supply extremely fast distance approximations at runtime. In this work, I will present a distance oracle for the otherwise intractable problem of shortest continuous active vision paths.

Chapter 3

Broader Impacts

This work aims to directly address two academic problems: unguided research and the inaccessibility of research to outsiders. Metrics for comparing active vision algorithms will allow more focused research by eliminating unpromising or effectively solved problems. More broadly, by making the software pipeline I have used to perform this research freely available and well documented, I hope to lower the barrier to entry into robotics research. It can be extremely difficult and frustrating for prospective roboticists to find easy to use tools, which turns people away from the field and prevents broader engagement with the public. The use of open source tools in industry has documented economic benefits due to increased ease of use and decreased maintenance needs [35]. Beyond the economic benefits of increasing access to quality software, these factors disproportionately impact small institutions or individuals. By streamlining the process in this area, we can introduce younger and less experienced programmers to an interesting and ongoing area of interest in robotics and empower them to make substantial contributions to it.

Industrial and domestic robotics will be impacted by this work. Active vision is an important capacity for robots working in semi-controlled environments and will make them more flexible and better able to perform tasks in those environments. On a purely technical level, this will be a significant improvement and expand the potential uses of automation into regions that have been historically inaccessible. Nursing is of particular interest; it typically takes place in semi-controlled hospital environments and faces both acute and chronic personal problems [36]. For robots to work effectively in hospitals and care homes, they will need to be able to reliably interact with novel objects by grasping them. A 2017 study of teleoperated nursing robots identified fine motor control in general, and grasping in particular, as areas that their prototype struggled with. On top of the hardware difficulties the robot experienced, users reported difficulty adapting to the user interface to perform tasks such as camera location and grasping. A major direction for future work the study identified was increased low-level autonomy to assist the user in these tasks [37]. Until these abilities are developed, robotic nursing will be limited to extremely unusual, highly controlled settings.

Improved flexibility will benefit industry, but there is widespread and reasonable anxiety about the effects of increased automation on society. The evidence for this is mixed. There is cause for both optimism and concern, but overall the picture is less bleak than we might fear. A common and credible concern is that increased automation will displace industrial workers, but there is reason for optimism that this will not be the case. The clear trend in the United States since 1960 has been of a large and successful industrial sector that has steadily shed blue collar jobs [38]. However, there are positive use cases which suggest that increased automation may not strictly result in lowered employment. Modern spinning mills are nearly totally automated. The last decade has seen explosive growth in yarn production, but no change in employment [39]. This suggests that after a certain level of automation, increases to productivity do not displace workers. Empirical data from Spain [40], France [41], and the Netherlands [42] all suggest that overall automation can lead to increased employment, though the specifics remain an open question. All three 2020 reports agree that overall employment may be increased, though the Spanish investigation concluded that the field being automated saw depressed employment. The studies in France and the Netherlands saw within field employment rise with automation. Additional analysis of the French employment data suggested that automation did not exacerbate gender based pay disparities, a promising sign [43]. The exact mechanics remain an open question, but it is reasonable to say that increased industrial automation is a benefit to industry and appears to have positive social externalities.

Chapter 4

Discrete Active Vision

I will now present the formal problem statement in the discrete case, explain the goals and utility of our experiments, and outline the techniques investigated.

4.1 Problem Statement

Our goal in this work is to determine how to best move a camera on a robot arm around an object to collect information about how to pick it up. Further narrowing our focus, for this work we define "best" as the method that minimizes the distance traveled by the camera to find a "sufficiently good" grasp.

Active vision contains many nested optimization problems, and so minimizing distance traveled is not the only approach to take. One strategy is to minimize total time taken, as Li et al.'s and Namiki et al.'s work does [44, 45]. This encourages practical application of the work, but focuses research on the particulars of hardware and software for a single system. Other researchers, such as Arruda et al. or Kroemer et al., aim to maximize grasp quality or confidence without resorting to exhaustive exploration [18, 46]. This approach is very similar to the approach we selected. In all of these studies and our own work, quality and movement constraints are imposed to prevent the algorithm from running forever or wasting time searching for infinitesimally better grasps. The key difference is the focus of the optimization. We attempt to end the search as quickly as possible with a sufficiently good solution, while they attempt to return the highest quality solution possible that does not exceed a length limit.

We minimize distance for two reasons. First, it is useful in the real world; domestic and industrial robots that can find a sufficiently good grasp with minimal movement are more practical than ones that crane around their targets, exhaustively mapping them. Second, distance traveled is completely repeatable regardless of computer hardware. Every experiment can be exactly replicated from its starting position and random seed. Using system time as a metric would tie performance to the very low level details of how our methods interact with the CPU's caching. We restrict the arm's motion to a sphere surrounding the center of the object being examined. This is an abstraction designed to isolate the visual search component of the problem, and common in the literature. When the focus of the work is on perception or search, e.g. [18, 47], a viewsphere is used. In this work, this is referred to as the viewsphere V, an imaginary sphere of fixed radius r surrounding the object. The camera moves by sliding along this sphere, always pointing directly at the sphere's center. In the continuous case (addressed in Chapter 5) the camera can move from any point on V to any other point, but we will begin with the discrete case i.e. in addition to only moving along the viewsphere, the camera only moves in fixed intervals of 20^o, and only in one of the eight cardinal directions from its start position as shown in Figure 4.1. The alternative is a model where the camera is connected to a manipulator and moves with it as the active vision progresses. When the goal is to correct for sensor discrepancies as in Viereck et al.'s work, this alternative method is used [3].

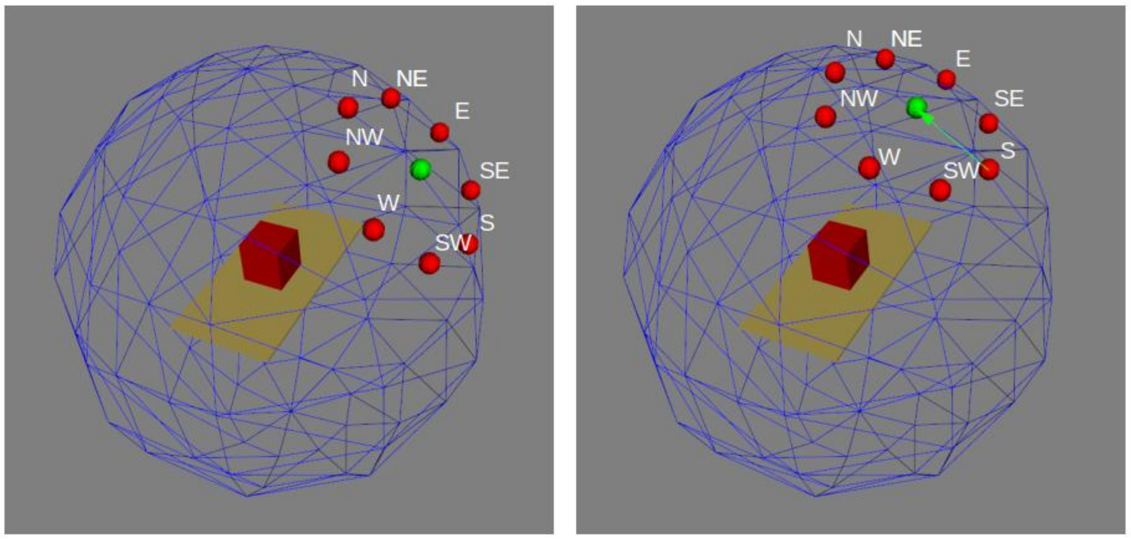

Figure 4.1: A demonstration of the discretized viewsphere. On the left the camera (represented as the green sphere) is in the starting position- 1 meter from the object's center, 45 degrees from the pole of the viewsphere. The eight red spheres surrounding it show the possible positions it can move to in its first step. On the right the camera has been moved to the North. The green arrow shows its past movement, the red spheres the possible next steps. The compass directions are always relative to the current position of the camera, not to any external reference frame.

Because we are not attempting to maximize the grasp quality, we consider the active vision successful if a "sufficiently good" grasp is found. The definitions of both grasp quality and sufficiently good are arbitrary and can be defined by the user. For simplicity, we use a force-closure grasping algorithm (see 4.5.1) to measure quality and consider a score of 150 out of 180 (higher scores are better) to be a sufficiently good grasp.

So, formally, the problem we are attempting to address is how to move a camera along V surrounding an object in the shortest possible path to find a sufficiently good grasp.

4.1.1 Research Goals

With this problem formulation, we had three main goals:

1. To provide a framework for objective comparisons of existing algorithms
2. To use that framework to understand the problem space being explored. Here, that requires answering three questions, one for each baseline in Section 4.2:
   (a) How much movement an optimal solution for each item requires. This establishes a ceiling on the gains which are possible by proposing new algorithms.
   (b) How much movement a random solution for each item requires. Combined with the optimal solution, this shows the **potential for improvement** that is possible. If the random solution is much worse than the optimal solution, significant progress can be made. If both are very close to one another in quality, the problem is already well solved by random motion and so further research is not warranted.
   (c) How much movement a naive solution for each item requires. This extends the previous item, by showing how much of that potential for improvement can be realized with simple algorithms. Only in cases where the optimal solution is significantly better than both the random and the naive solutions can improved algorithms lead to significant performance improvements.
3. To develop improved methods for solving the problem, as measured by the new baselines. This required gaining insight into the shortcomings of current approaches.

4.2 Comparison Framework

In literature, almost all active vision algorithms are compared to either random motion [18, 22], a static camera [48] or to other state of the art active vision strategies [3, 7, 49]. The first two approaches do not provide good baselines; random camera movement or no camera movement are such poor active vision strategies that even very poor algorithms outperform them by wide margins. State of the art comparisons can provide relative performance measures, but they cannot objectively measure performance. This means that if all current techniques share some underlying flaw, relative comparisons will never uncover it. To address these concerns, I developed three baseline methods for comparison. I will now present how each baseline approach attempts to solve the active vision problem.

4.2.1 Optimal: BFS

From a given start position, exhaustively explore all possible paths of 20^o steps using BFS. Continue until a “sufficiently good” grasp is visible or a depth threshold (5 steps in this work) is reached. This is the optimal possible performance in the discrete case. It is impossible for any other method to find a grasp in fewer steps, though other methods may perform equally well. By including this baseline, I am able to determine both the objective difficulty of the problem (i.e. how many steps it takes to find a “sufficiently good” grasp) and how close each proposed solution is to optimal performance, allowing us to compare algorithms absolutely instead of just relatively.

4.2.2 Pessimal: Random

From the start position, randomly select a direction and move there. Continue until a sufficiently good grasp is visible or the depth threshold is reached. Note that absolutely no weighting or constraints are placed on the random motion. If s is the camera's starting position, and p_1 is a valid position on the viewsphere to move to from s in one step, $s \to p_1 \to s \to p_1 \to s \to p_1$ is a perfectly acceptable length 5 path. This is a common standard in literature [18, 22], but it is very close to the *worst* active vision strategy that is not deliberately designed to perform poorly. Good active vision strategies tend to involve moving several consecutive steps in a given direction, which random motion very rarely does. As I will show in both empirical tests and by comparison to the final baseline, no algorithm should consistently under perform random motion.

4.2.3 Typical: Brick

From the start position, move Northeast (marked NE in Figure 4.1). Continue until a "sufficiently good" grasp is visible or the depth threshold is reached. Northeast was selected because it worked well in manual testing, but I make no claim that it is the best direction in most circumstances. This is a naive algorithm (named after leaving a brick on the accelerator of a car) that any method *should* outperform to be considered to have made a significant improvement over the baseline. Nonetheless, as [23] suggested and our work has confirmed, this is a reasonably effective solution which consistently outperforms the pessimal case and frequently equals or exceeds the results of other algorithms.

Now that I have described the baselines other methods will be compared to, I will describe those methods.

4.3 Machine Learning Methods

We began with two machine learning based approaches, since they are quick to implement and do not require extensive domain specific insights. Both approaches begin by collecting and compressing information from a simulated workspace, which I will now describe.

4.3.1 Data Collection

All of our experiments take input from a depth camera in the form of point clouds. For the discrete approaches, we needed to provide a representation of not just the current camera view, but past camera views. This requires two post-collection steps: calculating an unexplored point cloud, and merging the current point clouds with historical data.

To calculate the unexplored point cloud, a cube surrounding the workspace center is filled with evenly spaced points. Visibility is then estimated by projecting the object onto the image plane according to Equation 4.1.

$$X_p = KX/z_0 \tag{4.1}$$

where, X_p is the projected pixel co-ordinates, X is the point in the cube with coordinates $\begin{pmatrix} x_0 \; y_0 \; z_0 \end{pmatrix}^T$, and K is the camera intrinsic matrix described by Equation 4.2.

$$K = \begin{pmatrix} f_x & 0 & pp_x \\ 0 & f_y & pp_y \\ 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

The “depth value at X_p” and “z_0” are compared and if z_0 is greater than depth at X_p, a point in the object point cloud blocks X, so the point X is marked as occluded. These occluded points then form the unexplored point cloud.

Once both object and unexplored point clouds have been collected, they are merged with all past data for the current run. This is done by taking the union of object point clouds, and the difference of unexplored point clouds. These merged point clouds, with the addition of the camera position, will represent all of the information available to our active vision agent, and are ready to be interpreted.

4.3.2 Data Compression

In order to provide our machine learning models with a consistent input, we compressed the data supplied by the camera to a fixed sized vector and appended the camera position. The particular compression technique used was Height Accumulated Features (HAF), developed by [50] and used in [22] and [1] (See Figure 4.2).

In HAF, an nxn grid is laid over the camera view, and each square of the grid records the height of the tallest feature present in that grid. We tested 5x5 and 7x7 grids, found no appreciable difference between them, and so used 5x5 grids as they are slightly faster to compute and process.

The final feature vector is [52x1], created by appending a [25x1] flattened object HAF to a [25x1] flattened unexplored HAF to a [2x1] vector of camera polar and azimuthal angles (camera radius is constant).

4.3.3 Self Supervised Learning

We began with self supervised learning to replicate and extend the results of [22]. Because of this, we began by using that works' synthetic data generation technique. In it, training runs were constructed by taking a trial object and exploring one step in each legal direction. If a working path was found one step from the initial pose, that path was recorded. If not, a random walk was begun in each of the eight

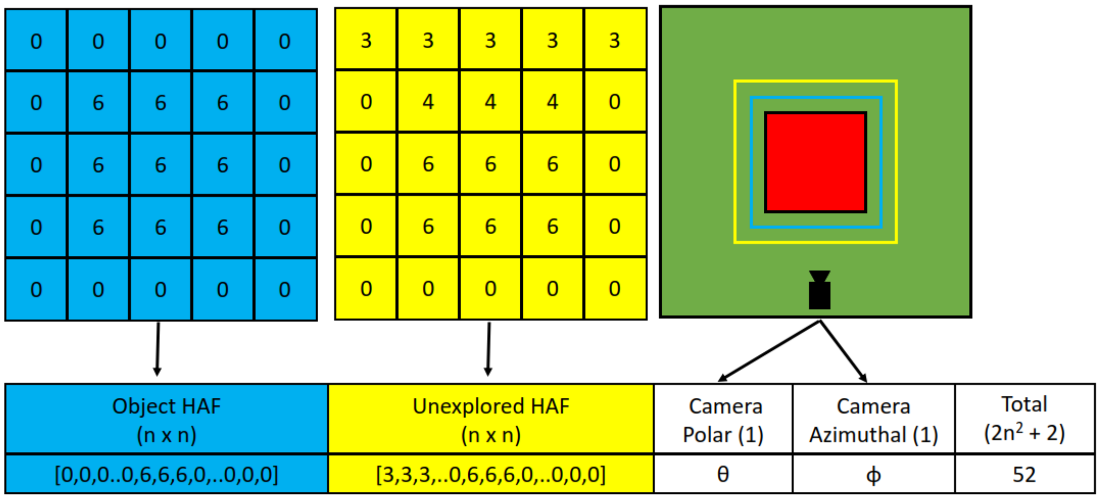

Figure 4.2: Demonstration of feature vector construction. Here a 6x6x6 cm cube is converted into a feature representation. The blue region shows the area the object HAF vector is based on and the yellow region shows the slightly larger area the unexplored HAF vector is based on. Note the "shadow" of lower values at the top of the unexplored region- these are the area directly across from the camera which are currently occluded by the cube. This also shows why the unexplored region is a necessary part of the feature vector. If the camera had completely observed the object the object data portion of the vector would be unchanged. Only the unexplored data region allows us and the algorithm to track what has already been seen.

directions and explored four steps. This was repeated three times for each direction, and the shortest working path out of the twenty four potential paths was recorded as the correct one. Algorithm 1 was then repeated for 1,000 initial poses for every object trained with. In our work, we collected these 1,000 training runs for the 10x8x4 cm and 20x6x5 cm prisms shown in Figure 4.3. More objects could have been used, but data collection was time consuming and initial results using only two training objects generalized very well to the test objects, so it was not pursued.

After the data had been collected, two self-supervised learning architectures were trained. Both took a feature vector of the current state and output an integer value $[1 \rightarrow 8]$ indicating the next direction to move (1=N, 2=NE, ..., 8=NW). Both begin

Algorithm 1 Data Generation Technique

```
Require: $start \leftarrow$ Initial viewpoint
Require: $exploration\_queue \leftarrow []$
Require: $shortest\_path \leftarrow$ None
  for all $viewpoint \in$ next possible viewpoints do
    $current\_path \leftarrow start + viewpoint$
    if $current\_path$ contains a grasp then
      return $current\_path$
    end if
    append $current\_path$ to $exploration\_queue$
  end for
  for all $start\_direction \in exploration\_queue$ do
    $iteration \leftarrow 0$
    while $iteration < 3$ do
      $depth \leftarrow 0$
      $current\_path \leftarrow start\_direction$
      while $depth < 4$ do
        $current\_path \leftarrow current\_path +$ random direction
        if $current\_path$ contains a grasp then
          $depth \leftarrow 4$
          if $current\_path$ shorter than $shortest\_path$ then
            $shortest\_path \leftarrow current\_path$
          end if
        end if
        $depth \leftarrow depth + 1$
      end while
      $iteration \leftarrow iteration + 1$
    end while
  end for
  return $shortest\_path$
```

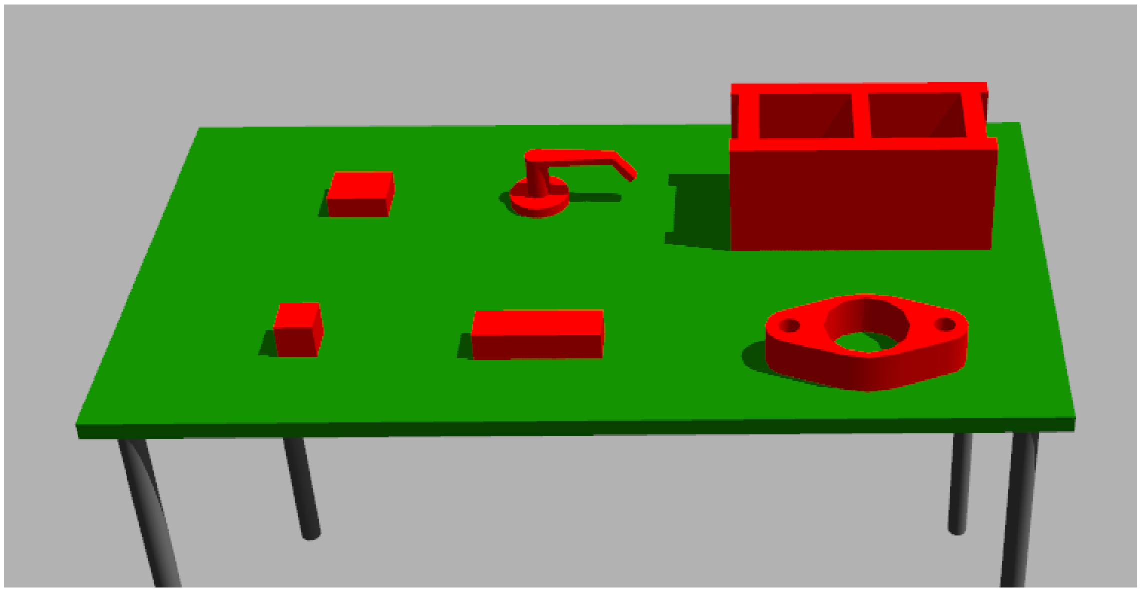
Figure 4.3: The simulated objects used in the discrete research. Objects are to scale with one another and the table. Back row, from left to right: 10x8x4 prism, handle, cinder block. Front row, left to right: 6x6x6 prism, 20x6x5 prism, gasket

by compressing the [52x1] feature vector to a [26x1] vector using PCA. The methods then diverge; one runs logistic regression on the compressed vector and the other runs Latent Dirichlet Allocation (LDA) on the same input to produce the same effect. The shared architecture of both methods can be seen in Figure 4.4 (a). Both methods were implemented using the scikit-learn library [51].

4.3.4 Deep Q Learning

We constructed a deep Q Learning model. Reinforcement learning was chosen for the second model because the problem lends itself well to interactive exploration. Supervised learning was impractical because the optimal solutions were unknown, unsupervised learning was impractical because of the size of the search space, but reinforcement learning could find good strategies in that large search space. The network was built using the Keras library [52]. It consists of four dense, fully connected [128x1] layers connected by ReLu transitions. These feed into a [8x1] softmax

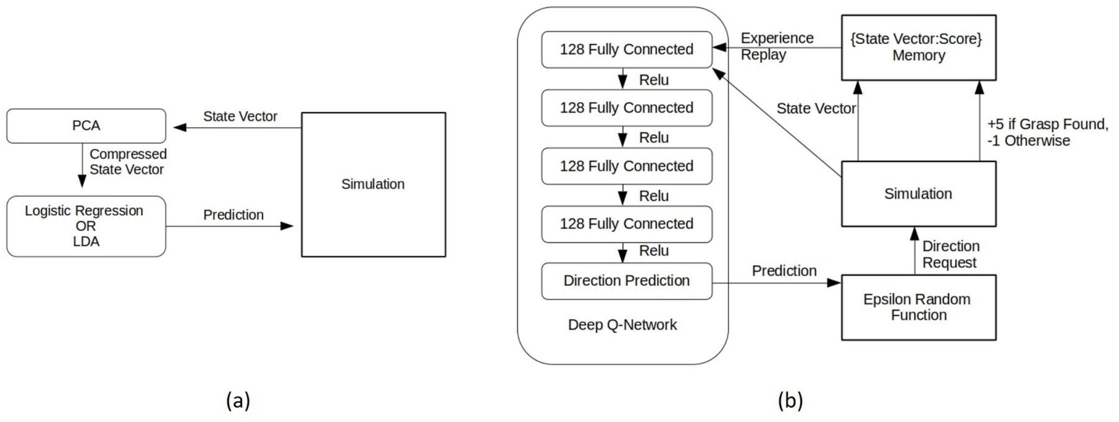

Figure 4.4: The machine learning architectures used. (a) shows the self-supervised learning approach, (b) the deep Q learning approach

layer, which predicts the next direction the camera should move. During training, an epsilon random gate was used to add noise. After each prediction, a random value was generated, and if it exceeded a threshold that decreased towards 0 as the training progressed a random value was substituted for the network's prediction. After a prediction was made, the camera was moved in the requested direction, and a new feature vector was captured and sent to the network, as shown in Figure 4.4 (b). Each iteration of training continued until a sufficiently good grasp was found or five unsuccessful steps had been taken, whichever happened first. The Q Learning model was trained until convergence on the objects in Figure 4.3, taking roughly 1,300 iterations. Ideally, both methods would have received the same training set, but the neural network required much more initial training than the dimensionality reduction based approaches.

4.4 Heuristic Methods

After developing and testing the machine learning based approaches, we built and tested two heuristic algorithms based on our observations of the machine learning's behaviors and existing methods in literature.

4.4.1 2D Heuristic

To begin, we built a simplified procedure designed for our specific problem statement and arrived at Algorithm 2. It begins with the same feature vector as the machine learning based approaches, and then estimates the potential useful information that can be gained by moving in each direction. To consider the impact of moving a direction, e.g. North, the algorithm takes the current known object and simulates moving the camera 20^o North of the current position. It projects the known object and the unexplored region onto a plane perpendicular to the camera's viewpoint. In this projection, every overlapping pixel between the current unexplored region and the potential exploration region represents information that moving the camera North could reveal. Running this count for each direction, we arrive at an *optimistic* estimate of how much information moving in this direction can reveal, under the (often incorrect) assumption that no new object points will be revealed by moving the camera. When new object points are revealed, they may occlude the unexplored region, and so reduce the information gained. The direction in which the highest best-case estimate lies is then selected. The resulting algorithm is simple, very fast, and generally robust, though it is easily mislead by objects where the optimistic and actual information gains are significantly different from each other.

4.4.2 3D Heuristic

Our main insight gained from observing the simulation testing up to this point was that *not all information is equally valuable.* There are large regions of most objects which cannot possibly contain a grasp, but in Algorithm 2 or a standard information gain framework like the one found in [18] will be marked as having

Algorithm 2 2D Heuristic policy

```
Require: $obj \leftarrow$ Object point cloud
Require: $unexp \leftarrow$ Unexplored point cloud
  for all $viewpoint \in$ next possible viewpoints do
    if viewpoint within manipulator workspace then
      $obj\_trf \leftarrow$ Transform $obj$ to viewpoint
      $obj\_proj \leftarrow$ Project $obj\_trf$ onto image plane (B/W image) and dilate
      $unexp\_trf \leftarrow$ Transform $unexp$ to viewpoint
      $unexp\_proj \leftarrow$ Project $unexp\_trf$ onto image plane (B/W image) and dilate
      $non\_occ\_unexp\_proj \leftarrow unexp\_proj - obj\_proj$
    end if
    Record the number of white pixels in $non\_occ\_unexp\_proj$
  end for
  Choose the direction with maximum white pixels
```

large amounts of potential information simply because they are unexplored. To address these weaknesses, we developed Algorithm 3. It runs in much the same way as Algorithm 2, but with two key differences. First, ray-tracing rather than projection is used to estimate the potentially revealed area. This is much more computationally expensive, but allows for a much more accurate determination of potentially revealed space, since it measures the full area being revealed rather than just its surface area. Second, the criteria outlined in 4.5.1 are applied to the known points and the regions in which complimentary grasps could lie are overlaid onto the unexplored region. Only the portions of the unexplored region which could form a grasp with the explored region are then considered. The direction which has the potential to reveal the largest number of grasp containing points is then selected. This algorithm is significantly slower and more complicated than the 2D Heuristic, but addresses many of its shortcomings.

Algorithm 3 3D Heuristic policy

```
Require: $obj \leftarrow$ Object point cloud
Require: $unexp \leftarrow$ Unexplored point cloud
Require: $points\_threshold \leftarrow$ Minimum number of non-occluded unexplored
    points needed for a new viewpoint to be considered useful
  $useful\_unexp\_trf \leftarrow$ Unexplored points with potential for a successful grasp
  for all $viewpoint \in$ next possible viewpoints do
    if viewpoint within manipulator workspace then
      $obj\_trf \leftarrow$ Transform $obj$ to viewpoint
      $useful\_unexp\_trf \leftarrow$ Transform $useful_unexp$ to viewpoint
      $non\_occ\_useful\_unexp \leftarrow$ Check occlusion for each $useful\_unexp\_trf$ using
        local surface reconstruction and ray-tracing.
    end if
    Record the number of points in $non\_occ\_useful\_unexp$
  end for
  $max\_points \leftarrow$ Maximum points seen across the possible viewpoints
  if $max\_points \leq points\_threshold$ then
    Run the previous for loop with twice the step-size
  end if
  $max\_points \leftarrow$ Maximum points seen across the possible viewpoints
  Choose the direction which has $max\_points$
```

4.5 Discrete Testing

In order to compare the three baselines, two heuristics, and three machine learning approaches, extensive simulation testing was needed. Our goal was to provide as much diversity of objects and poses as possible, in order to locate any situations which exposed algorithm specific weaknesses. First, we selected the objects in Figure 4.5 from the YCB dataset, ensuring that none of these objects had been included in the training sets for any of the machine learning algorithms. Next, we ran each approach on every object 100 times in different poses and recorded the results. The camera always began at the same position in each test, and the object was rotated around its z-axis to represent different starting orientations. Since testing of the discrete algorithms was concerned with relative algorithmic performance, care was taken to ensure that every algorithm had as close to identical conditions as possible.

To do this, the start positions for each run were seeded. We randomly generated 100 values from 0 to 359, and every test ran through each of those same 100 values in order, ensuring that each algorithm had the same objects in the same configuration.

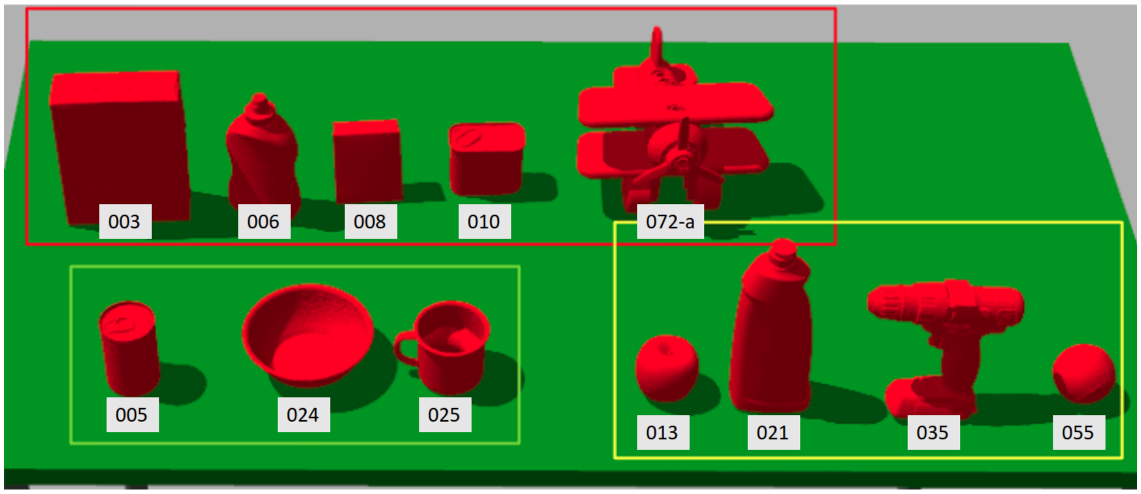

Figure 4.5: The objects used for simulation testing for both discrete and continuous experiments. Each object has been labeled with its YCB ID, and outlined according to its difficulty; red for hard, yellow for medium, and green for easy.

4.5.1 Grasp Synthesis

These methods are largely agnostic to the specific method of grasp synthesis, so we used a force-closure algorithm based on [22], since it is reliable and simple. In it, grasp quality is defined by how closely the surface normals of two object points are opposed to each other, formally:

$$GQ = 180 - (min(\angle(\overrightarrow{p_1p_2}, \overrightarrow{n_1}), \angle(\overrightarrow{p_2p_1}, \overrightarrow{n_1})) + min(\angle(\overrightarrow{p_1p_2}, \overrightarrow{n_2}), \angle(\overrightarrow{p_2p_1}, \overrightarrow{n_2}))) \quad (4.3)$$

where, GQ is the grasp quality, p_1 and p_2 are the contact points, n_1 and n_2 are the surface normal vectors at each contact point. Grasp quality cannot fall below 0 (if both surface normals are parallel to each other and perpendicular to the axis between p_1 and p_2) or exceed 180 (if both surface normals are exactly opposed along this axis). We defined a grasp as any point pair whose grasp quality, as evaluated by Equation 4.3 exceeded 150. Additionally, we added three constraints for increase realism

1. Curvature- to be graspable, neither point may exceed the mean object curvature by a factor of more than 1.3 times.

2. Area- to be graspable, each point must be surrounded by at least 1.5cm of relatively flat material. This is based on the Franka Emika's gripper, and calculated by projecting the surrounding object points onto a plane perpendicular to each point's surface normal and then measuring the radius of the minimum inscribed circle on that plane.

3. Collisions- Once the final grasp has been found, the gripper model is loaded into simulation and spun around the grasp axis. If no position can be found which does not intersect with the object, the grasp is rejected.

More conditions could be added for increased realism, but in [1] these gave good results and the choice of grasp algorithm is not of critical importance for this work, so we limited ourselves to these for simplicity.

4.5.2 Discrete Results

After the simulation testing was complete, we were able to empirically compare the various algorithms' performance, as shown in Figure 4.6. Here the BFS policy forms an absolute ceiling, which no other policy ever exceeds, and the Random policy forms a floor which indicates a significant problem if another policy falls below it. This does happen for several steps in some objects (e.g. step 1 of objects 013 and

055), but none of the algorithms tested underperformed Random after the full five steps. For many objects, all methods performed very similarly. For example, for object 025 (mug), only difference between algorithms occurs in step one. The BFS immediately finds a working grasp 100% of the time, while the other algorithms succeed only some of the time, with the "Brick" strategy performing most poorly and only finding a grasp in 80% of the trials. By step 2, every algorithm has found a working grasp, and so every one scores 100% for the remainder of the trial.

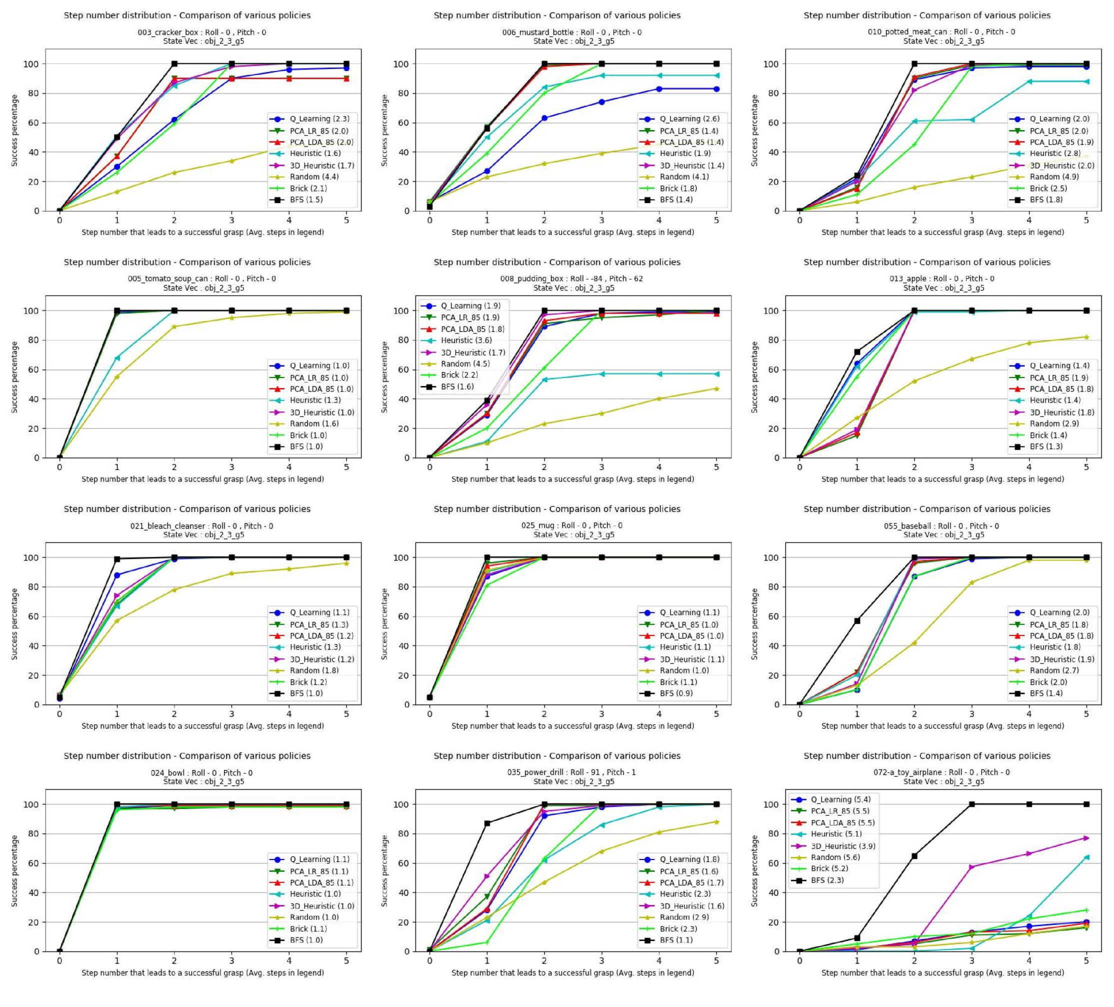

Figure 4.6: The results of applying each policy to each object in simulation. Units are number of runs out of 100 which succeeded after N steps. If a method fails to reach 100/100 by step 5, it never finds a successful grasp after 5 steps for some initial poses.

Based on this, we were able to define a "difficulty score" based on the ratio between the Random performance and the BFS performance. If, as in the case of object 024, BFS and Random performance are equal, the object is very easy, since any path is as good as any other. On the other hand, for objects like 072-a, BFS performs quite well but Random does terribly, suggesting that good performance is possible but difficult to arrive at by random motion, so the object is hard to plan an active vision path for. Formally, we defined the difficulty of the object as the ratio between the number of successes after two steps of Random motion and the number of successes after two steps of BFS. For example, when analyzing object 055, after two steps 41 of the Random trials had succeeded, but 100 of the BFS trials had succeeded. This gives 055 a difficulty of 0.41. We used the ratio after two steps, rather than five, because it spreads the distribution out more, but note that this ratio doesn't capture the full complexity of what is happening in the trials. However, it gives a good rule of thumb for what fraction of optimal performance to expect as a baseline for each object.

We then somewhat arbitrarily split the difficulties into three groups- Easy objects, where the difficulty ratio is > 0.8, Medium objects where the difficulty ratio falls between 0.4 and 0.8, and Hard objects where the ratio ≤ 0.4. Based on this, the twelve objects tested can be classified as follows:

1. Easy: Tomato soup can (005), Bowl (024), Mug (025)
2. Medium: Apple (013), Bleach cleanser (021), Power drill (035), Baseball (055)
3. Hard: Cracker box (003), Mustard Bottle (006), Pudding box (008), Potted meat can (010), Toy airplane (072-a)

The reason for these classifications can be seen in Figure 4.7. In it we can clearly see how not all object results are comparable; in the first step almost all methods

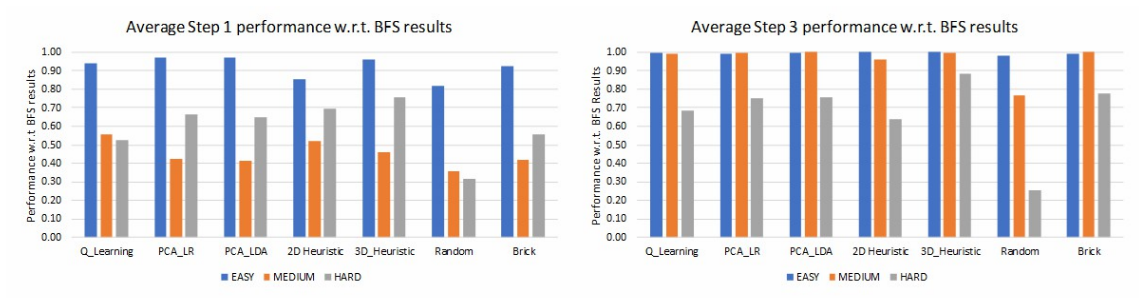

Figure 4.7: Performance of each method (other than BFS) averaged over all objects for step 1 (on the left) and step 3 (on the right). Performance represents the fraction of optimal performance achieved and has been calculated by taking the number of successes for each method and dividing by the number of BFS successes.

produce nearly optimal results for easy objects, and by step three every method but the 2D Heuristic and Random is indistinguishable from optimal performance for easy and medium objects. But that isn't very impressive, since Random is also doing well for those. It is only for the hard objects that significant improvements are possible, and so only for the hard objects that meaningful conclusions can be drawn about algorithm performance. And while there are variations in performance, only the 3D Heuristic significantly outperforms the Brick baseline for hard objects. This suggests that for most objects the active vision strategy is simply not very important, since even the totally mindless strategy of picking a direction and traveling in a straight line will get good results. However, there are some objects where the strategy is significant and can provide useful improvement over this baseline.

Chapter 5

Epsilon-Optimal Path Planning

Having introduced the common strategies of active vision and demonstrated the usefulness of an optimal solution in the discrete case, I will now formally define that problem in the continuous case and show how an epsilon bounded approximation of its optimal solution can be found.

5.1 Problem Statement

In the continuous case, I use the same goal as in the discrete case: how to move a camera along the viewsphere surrounding an object in the shortest possible path to find a sufficiently good grasp. To approach this in the continuous space, I densely pre-populate the viewsphere with viewpoints using a Fibonacci Lattice [53] (see Section 5.3.1 for details). This allows me to exhaustively search the viewsphere to a known resolution. For clarity, I introduce the following terms:

- Let V be a viewsphere of radius r. Let v_i be an arbitrary point generated according to Section 5.3.1$\in V$.
- Let x_i be an arbitrary point of radius $r \notin V$. Let X be the set of all x_i.

- All distances along V and X are great-sphere distances.
- Let s be the point on the viewsphere V the camera begins at.
- Let d be the number of points on the viewsphere before s is added, and let m be the maximum distance any point $\in V$ and it's nearest neighbor $\in V$ before s is added.
- Let C be a 3D point cloud representing the object being grasped, and let c_i be an arbitrary point $\in C$.
- Let G be an arbitrary two finger grasping algorithm G(c_1, c_2) which takes two points and returns a True/False decision if they are suitable for grasping. Two points c_1, c_2 are *graspable* iff G(c_1, c_2) == True. Our earlier concept of a "sufficiently good" grasp is a specific case of G.
- Let L be an arbitrary visibility algorithm L(v_1, c_1) which takes two points and returns a True/False decision if they have line of sight to one another. Two points v_1, c_1 are *visible* iff L(v_1, c_1) == True.
- A *valid grasp* is a path from which two *graspable* points are *visible*, that is, c_i and $c_j \in C$ $(i \neq j)$ such that
    1. G(c_i, c_j) == True, and
    2. L(p_a, c_i) == True, L(p_b, c_j) == True where $p_a, p_b \in P$.

With these terms defined, the problem in the continuous case becomes the following: What is the shortest path $P = \{s, p_1, ...p_n\}$ along the viewsphere V such that it contains a *valid grasp*.

Further, in the continuous space I make the following assumptions

1. P exists

2. V is sufficiently dense that $\forall$ points $c_i \in C$, $x_i \notin V$ at radius r such that L(x_i, c_i) == True $\implies \exists v_i \in V \mid$ L(v_i, c_i) == True within at most m of x_i.

The first assumption is necessary for the problem to be defined at all (how to determine if an object is impossible to grasp is not addressed here).

The second assumption is necessary to provide a guarantee of closeness to optimality- if some points are viewable from r but not viewable from any point on V, they could be part of a solution that is arbitrarily shorter than the one that can be found by exclusively considering the points on V. However, V can be made as dense as necessary, and as long as at least one point on V can view each visible point on the object, I will show how a solution within a known error of the true optimal solution can be found.

5.2 Proposed Algorithm

5.2.1 Setup

Setup consists of three stages: visibility checking, grasp pair checking, and distance calculation. This constitutes the significant bulk of the computational load of the problem, and can be reused for new calculations for the same object beginning at a new start point. Thus, after the very expensive setup, any number of solutions can be found in short order (see 5.2.3 for exact run time analysis).

Visibility

The first step of setup is to calculate all regions of visibility using Algorithm 4. For each relevant object point, we calculate the region of the viewsphere which can view it and store it in vis_pairs. If a point is ineligible for grasping, it is irrelevant and so marked as being totally occluded.

Algorithm 4 Check Visibility

```
Require: $obj \leftarrow$ Object point cloud
Require: $view\_sphere \leftarrow$ Viewsphere point cloud
  $vis\_pairs \leftarrow [\,]$
  for all $object\_point \in obj$ do
    if not Eligible for grasping then
      Add ($object\_point$, [ ]) to $vis\_pairs$
    else
      $region\_of\_visibility \leftarrow [\,]$
      for all $view\_point \in view\_sphere$ do
        raytrace $object\_point \rightarrow view\_point$
        if not raytrace occluded and $view\_point$ satisfies angle criteria then
          Add $view\_point$ to $region\_of\_visibility$
        end if
      end for
      Add ($object\_point$, $region\_of\_visibility$) to $vis\_pairs$
    end if
  end for
```

Grasp Pairs

The second step of setup is to identify which object pairs can form grasps using Algorithm 5. Iterating over every visible pair of grasp points, I compare them to the criteria described in 4.5.1. Points which satisfy all the criteria are marked as being able to form grasps with one another.

Point Distances

Finally, the distances between viewpoints are calculated using Algorithm 6. For each visible object point, every point in its region of visibility is compared to every point in each complimentary object point's region of visibility. The shortest distance to a viewpoint which can view a complimentary point is then recorded. Once this is finished, each viewpoint that can see a viable grasp point has been marked with the shortest distance from itself to a viewpoint that can see one of the points needed to complete that grasp. With this step done, we are ready to calculate the optimal

Algorithm 5 Find Grasp Pairs

```
Require: $obj \leftarrow$ Object point cloud
Require: $view\_sphere \leftarrow$ Viewsphere point cloud
Require: $vis\_pairs \leftarrow$ List of $(object\_point,[view\_point])$ pairs
  $grasp\_pairs \leftarrow [\,]$
  for all $object\_point \in obj$ do
    if not $vis\_pairs[object\_point] == [\,]$ then
      for all $later\_object\_point \in obj$ do
        if $later\_object\_point$ and $object\_point$ form a grasp then
          Add $object\_point$ to $grasp\_pairs[later\_object\_point]$
          Add $later\_object\_point$ to $grasp\_pairs[object\_point]$
        end if
      end for
    end if
  end for
```

path from any start point on the viewsphere.

5.2.2 Optimal Path Finding

After Algorithms 4, 5, and 6, Algorithm 7 produces a triple consisting of (Shortest Path Length, First point in path after s, Second point in path after s). This is done with brute force search. Beginning at the camera's start position, the distance to each viewpoint which can see at least one valid grasp point in the sphere is calculated. Then, the distance calculated in Algorithm 6 is added, to find the shortest full path that connects the start to the first viewpoint to a second viewpoint, with a valid grasp being visible along the path. This is repeated for each point, and the shortest total distance and the two viewpoints defining that shortest path are returned.

Algorithm 6 Find Point Distance

```
Require: $obj \leftarrow$ Object point cloud
Require: $vis\_pairs \leftarrow$ List of ($object\_point$,[$view\_point$]) pairs
Require: $grasp\_pairs \leftarrow$ List of ($object\_point$, $object\_grasp\_partner\_point$) pairs
  $partner\_dist\_list \leftarrow [\ ]$
  $partner\_list \leftarrow [\ ]$
  for all $object\_point \in obj$ do
    for all $view\_point \in vis\_pairs[object\_point]$ do
      $best\_dist \leftarrow$ infinity
      $current\_dist \leftarrow$ infinity
      $partner \leftarrow$ -1
      if not $partner\_list[view\_point]$ == [ ] then
        $best\_dist \leftarrow partner\_dist\_list[view\_point]$
        $current\_dist \leftarrow partner\_list[view\_point]$
      end if
      for all $partner\_object\_point \in grasp\_pairs[object\_point]$ do
        for all $partner\_view\_point \in vis\_pairs[partner\_object\_point]$ do
          $current\_dist \leftarrow$ spherical distance between $view\_point$ and
          $partner\_view\_point$
          if $current\_dist < best\_dist$ then
            $best\_dist \leftarrow current\_dist$
            $partner \leftarrow partner\_view\_point$
          end if
        end for
      end for
      $partner\_dist\_list[view\_point] \leftarrow best\_dist$
      $partner\_list[view\_point] \leftarrow partner$
    end for
  end for
```

Algorithm 7 Find Shortest Path From Start

```
Require: $start\_point \leftarrow$ Point at radius r that the camera begins at
Require: $obj \leftarrow$ Object point cloud
Require: $view\_sphere \leftarrow$ Viewsphere point cloud
  insert $start\_point$ into $view\_sphere$
  insert $start\_point$ into the $vis\_pairs$ of any $object\_point$ it can view, using the
  criteria in Algorithm 4
  $best\_dist \leftarrow$ infinity
  $current\_dist \leftarrow$ -1
  $partner \leftarrow$ -1
  for all $[view\_point, distance\_to\_partner] \in partner\_dist\_list$ do
    $current\_dist \leftarrow$ spherical distance between $start\_point$ and $view\_point$ +
    $distance\_to\_partner$
    if $current\_dist < best\_dist$ then
      $best\_dist \leftarrow current\_dist$
      $partner \leftarrow view\_point$
    end if
  end for
  return $[best\_dist, partner, partner\_list[partner]]$
```

5.2.3 Runtime Analysis

Let d be the number of points in the viewsphere, and n be the number of points in the object.

Algorithm 4 iterates over all points in the viewsphere for each point in the object, and so runs in $O(d * n)$ time.

Algorithm 5 iterates over all later points in the object for each point in the region of visibility for each point in the object, and so runs in $O(n * d * n) = O(n^2 * d)$ time (though in practice the region of visibility is much smaller than d).

Algorithm 6 iterates over all viewpoints ($O(d)$) in the complimentary points' ($O(n)$) region of visibility for each point in the region of visibility ($O(d)$) for each point in the object ($O(n)$), and so runs in $O(d * n * d * n) = O(n^2 * d^2)$ time (though again this is an extreme worst case scenario).

Thus the setup for Algorithm 7 runs in $O(n^2 * d^2)$ since none of the above algorithms are called more than once or call each other. Algorithm 7 itself runs in O(max(n, d)) time, since testing the visibility of the *start_point* takes O(n) time and comparing the distances takes O(d) time. At useful levels of viewsphere density $d >> n$, so Algorithm 7 runs in O(d) time.

5.2.4 Proof of Correctness

I claim:

Theorem 1. *There exists a truly shortest path P=s$\rightarrow x_1 \rightarrow x_2$ which views c_1 and c_2, where x_1 and x_2 are viewpoints of distance r from the object center. x_1 and x_2 need not be present in V.*

Remark. *Please note that s, x_1, and x_2 do not need to be separate points- I have empirically observed situations in which the camera's start position can view both grasp points immediately, so $s = x_1 = x_2$!*

Proof of Theorem 1. Consider the sets of all points which can view c_1 and c_2 from distance r, A and B respectively.

Now, consider each point $a_i \in A$. Since we are operating in a great-circle distance space, there is a shortest path from each a_i to B. Take that path length and add the distance $x_i \rightarrow a_i$.

Repeat $\forall a_i \in A$. One of these paths must be shortest or tied for shortest, so the claim holds.

□

Theorem 2. *Algorithm 7 produces a path containing a valid grasp, that is, P'=s$\rightarrow v_1 \rightarrow v_2$ which views c_1 and c_2.*

Proof of Theorem 2. Because all object points considered by Algorithm 7 satisfy G, if the algorithm produces any path, that path will contain a *valid grasp*.

By assumption 2, if a point satisfies G, it will be visible from V.

By assumption 1, there are at least two points on V which can produce a working path.

Since Algorithm 7 is exhaustive, every point on V will be considered.

Thus, Algorithm 7 will produce an output containing a *valid grasp*, and so the claim holds.

□

Theorem 3. *P' is the shortest possible path that contains a valid grasp considering only the points in V, satisfying the problem statement.*

Proof of Theorem 3. Assume to the contrary that there exists a shorter path P”=s$\rightarrow v_a \rightarrow v_b$.

The shortest path connecting any three points on a sphere will consist of two great circle distances, so P” must consist of two great circle distances whose sum is smaller than the sum of P's distances.

But Algorithm 7 is exhaustive and checks for this condition, so if P” existed, Algorithm 7 would have selected it instead of P', a contradiction. □

Theorem 4. *L(P') $\geq$ L(P).*

Proof of Theorem 4. Let A be the region of visibility for c_1, B be the region of visibility for c_2.

Algorithm 4 cannot mark points outside of A or B as viewing c_1 and c_2, so v_1 and v_2 must be elements of A and B respectively.

Since P is the shortest path between A and B, P' must be at least as long as it, so the claim holds. □

Theorem 5. *L(P') - 3*m $\leq$ L(P), where m is the maximum separation between a point in V and it's nearest neighbor in V.*

Proof of Theorem 5. Consider the optimal path P connecting s, v_1, and v_2. By definition of m, $\exists$ at least one point $v_i \in V$ within m distance of any other point $\in V$.

By assumption 2, there is guaranteed to be at least one point $\in V$ which is also $\in A$. So, for any points x_a and x_b, $\exists v_a, v_b \in V$ such that the distance from x_a to $v_a \leq m$. Thus, within V, there must be a path $P' = s \to v_a \to v_b$ which satisfies Condition 1.

Now consider the subsections of that path, $s \to v_a$ and $v_a \to v_b$.

If v_a and v_b are displaced by m away from x_a and x_b in opposite directions of one another, $L(v_a \to v_b) = L(x_a \to x_b) + 2m$.

Likewise, if v_a is displaced from x_a by m in the opposite direction of s, $L(s \to v_a) = L(s \to x_a) + m$. s cannot be displaced because it is an artificial point.

Both conditions can occur iff v_a, s, and v_b are colinear with s between v_a and v_b. In this case, $L(P') = L(P) + 3m$.

In any other case, $L(P')$ must be smaller, since otherwise you would need to increase the length of at least one of the subsections of P'. But each subsection is displaced as far as possible- if v_a is more than m from x_a, by assumption 2 there is another point closer to x_a which can view it, and so v_a would not be selected. Likewise for v_b.

Thus, P' can be at most $3m$ longer than P, and so the claim holds.

$\square$

5.3 Simulation Experiments

I used YCB Dataset object models to empirically measure the performance of this approach [54]. Each object was loaded as a point cloud from the provided .ply file and then the shortest path was estimated using Algorithm 7 at varying point cloud densities and by BFS taking 20^o steps as in the discrete study.

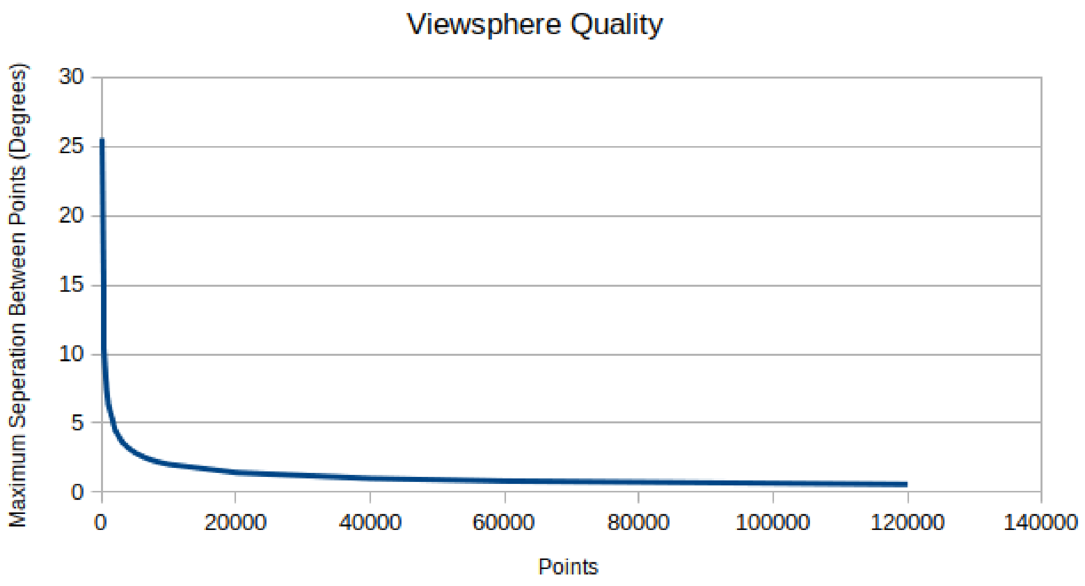

Figure 5.1: Empirical measurement of m, the maximum separation between closest viewpoints on a viewsphere of given density.

5.3.1 Viewsphere Generation

Generating evenly spaced points on a viewsphere is a subject of significant interest, and true even spacing is an open question in mathematics. Fortunately, the Fibonacci Lattice provides a good approximation [53], which will place points arbitrarily close to any point on the viewsphere as the density increases. Note, however, that the distance between an arbitrary point and the closest point on the viewsphere will decrease on average, but it will **not** decrease monotonically- in many cases increasing the viewsphere density will move the closest point farther from it. I can guarantee that this distance will not increase by more than 1/2 the maximum empirical distance between two points on the viewsphere, which I have measured for a sample of densities and present in Figure 5.1.

5.3.2 Visibility Determination

In this work, visibility is defined by ray-tracing- two points are visible from each other if no other object point occupies a voxel in a straight line between the two of them. I added the additional requirement that to count as being visible, the object's normal must be within 60^o of the viewpoint as shown in Figure 5.2. This was done to improve realism; without this constraint the camera can “view” objects that are directly perpendicular to it.

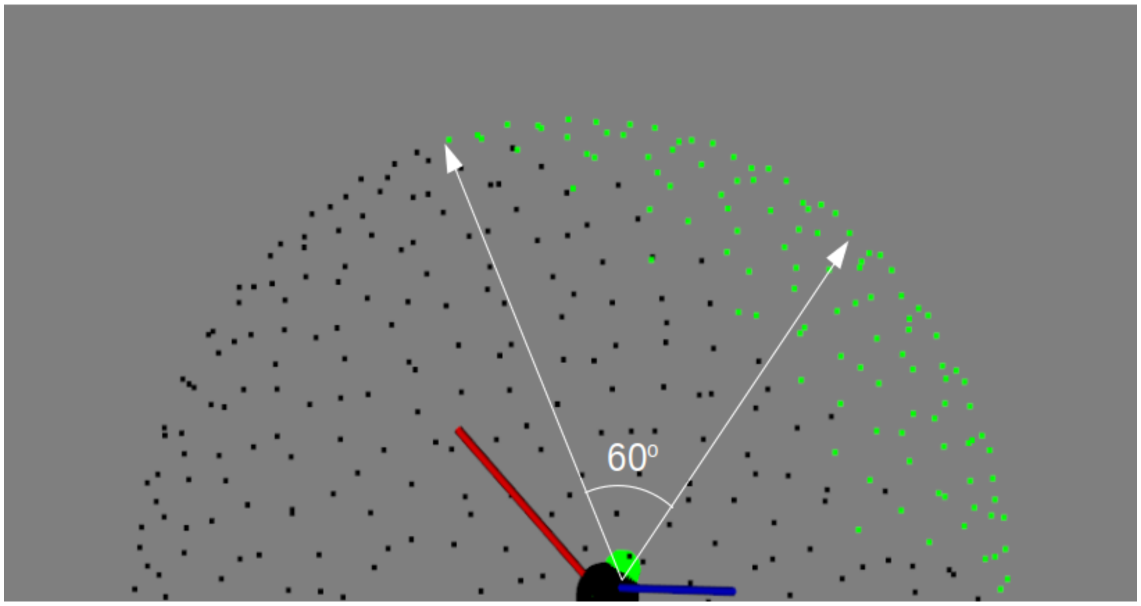

Figure 5.2: Demonstration of the visibility constraints imposed in this project. Potential viewpoints (in green) of the object point (shown by the large green sphere) must lie within 60 degrees of the intersection of the object point's normal with the viewsphere.

5.3.3 Comparison of Continuous to Discrete Searches

I initially expected continuous searches to be strictly better than discrete searches, but at very low viewsphere densities the discrete *can* produce shorter paths than the continuous search. Even at very high densities, it is possible for the discrete path to be coincidentally very close to the optimal path, and so outperform the continuous search. However, in practice, at even very low densities the continuous search quickly outperforms the discrete search in both the average and worse case scenarios. When the viewsphere contains at least 2,000 points, the average result for continuous search on all objects tested outperforms the discrete average by more than the pessimal margin of error, showing that the continuous average shortest path must be shorter than the discrete average shortest path. Distances are measured in degrees along the viewsphere, according to great circle distance. Once the density is increased to 6,000 points, the average improvement is at least 3.09^o (a 7.73% decrease from the continuous average) and ranges as high as 14.8^o (a 19.55% decrease). Higher densities lead to continued improvement.

Additionally, the runtime of the continuous search is dependent on the density of the viewsphere and object, but totally independent of the length of the optimal path. The discrete search on the other hand becomes exponentially more expensive with the number of steps in the path, making finding either long paths of finely detailed paths prohibitively expensive. Thus, for many cases, the continuous search is not only higher fidelity but faster to run than the discrete case. As Figure 5.5 shows, continuous searches on viewspheres less than 2,000 points in size are faster across all metrics than their discrete counterparts, but offer higher quality solutions as shown in Figure 5.3.

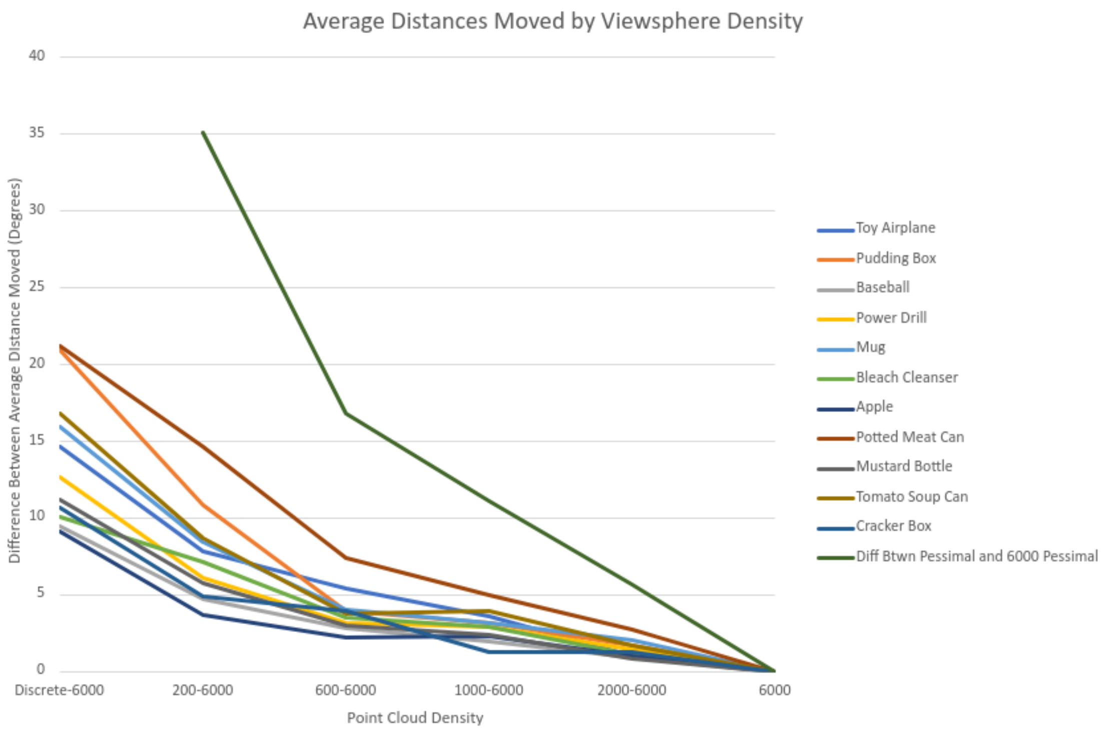

Figure 5.3: Comparison between average solutions for each object at various viewsphere densities and the pessimal possible solution at 6,000 points. That is, I assumed that the 6,000 point average was off by as much as possible, then subtracted it from all lower density solutions to find the maximum possible improvement. The green "difference" class shows the absolute difference between epsilons for each density.

5.3.4 Approximations of True Optimal Paths

In addition to the ceiling on error Theorem 5 gives us, Theorem 4 also provides an absolute floor on the difference between the true optimal path and the results of Algorithm 7. From this, we can calculate the range of possible lengths of the optimal path P_d for a viewsphere of density d as falling between

$$P'_d \geq P_d \geq P'_d - (3 * m_d) \tag{5.1}$$

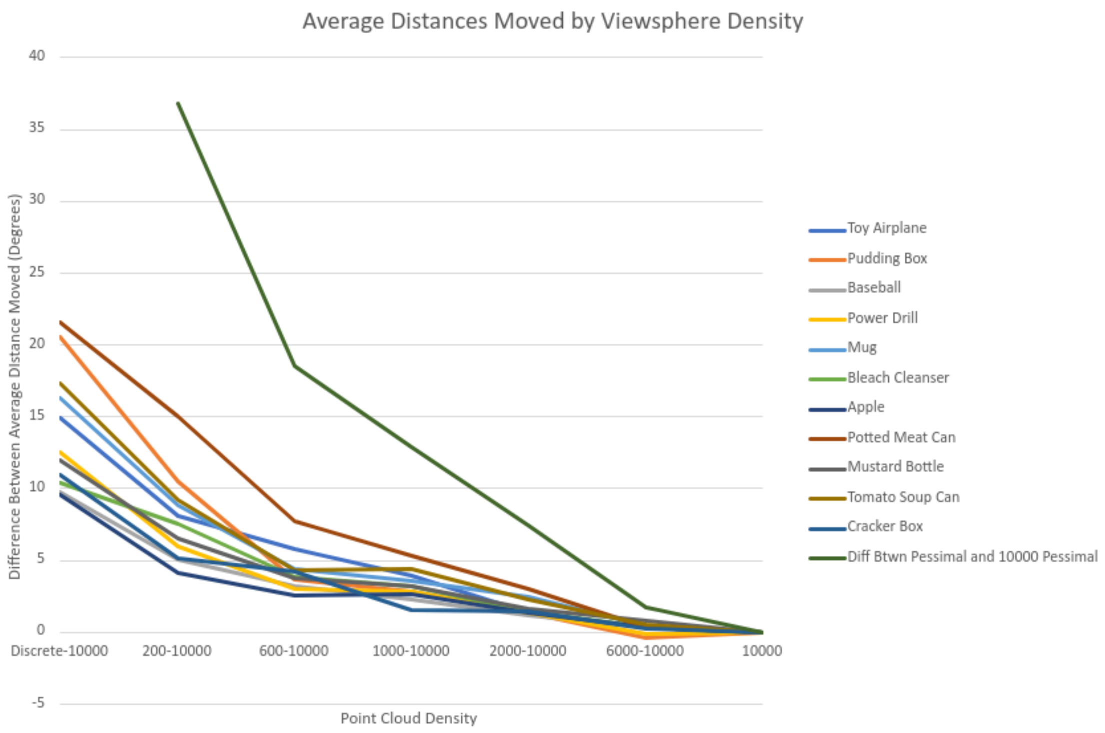

Figure 5.4: Comparison between average solutions for each object at various viewsphere densities and the pessimal possible solution at 10,000 points. As you can see, in both cases the typical performance improves much faster than the worst case performance.

If we have data for multiple densities $D = d_1, d_2, ...d_n$, P_D is bounded by

$$max_{d \in D}(P'_d) \geq P_D \geq min_{d \in D}(P'_d - (3 * m_d)) \tag{5.2}$$

This means that while the optimal value is always within $(3 * m_d)$ of the returned value, it is possible for even lower density results to further restrict this range. In the best case in my testing, I was able to restrict the optimal value to a range of 2.84^o by combining data from 200 to 10,000 points, which only guarantees an optimal range of 6.08^o. More typical results are displayed in Figures 5.6, 5.7, and 5.8 for a sample of objects identified as easy, medium, and hard to grasp in our previous work.

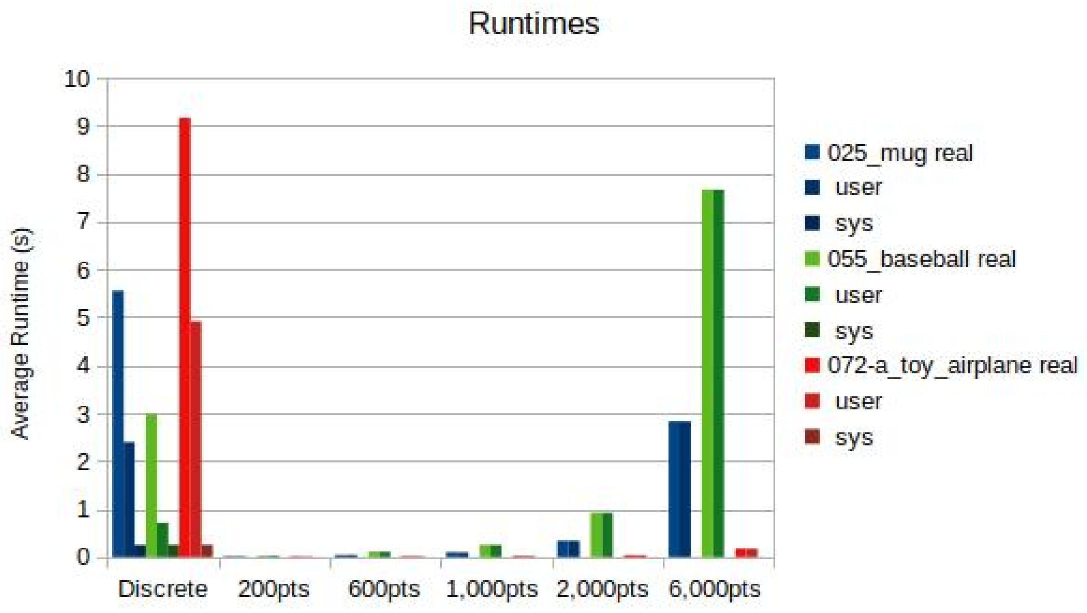

Figure 5.5: Comparison between observed runtimes for discrete searches and continuous searches at different resolutions. All values were found by running 100 searches and averaging the time taken over all of them.

5.3.5 Empirical Performance of Continuous Search

From Theorem 5 we know an absolute upper bound on the amount of error Algorithm 7 can return for a given viewsphere density. But in practice, I expected the error to be lower- on average the true optimal path point will be off by 1/2 m, not the worst case of 1 m. Measuring this directly is complicated by the fact that we never find the true optimal path, only the epsilon range around it. As a proxy, we can measure the difference between returned distances at various point densities and the returned distance at the highest density, assuming that the highest density's values are off by as much as possible. This assumption is clearly wrong, but useful for visualizing the rate of approximation refinement. The results can be seen in Figures 5.3 and 5.4. Empirically, the average path length returned is far lower than the expected worst-case path length.

Object	Discrete Difficulty	200 pts	600 pts	1000 pts	2000 pts	6000 pts	Average
005_tomato_soup_can	Easy	0.25	0.23	0.35	0.30	0.29	0.28
025_mug	Easy	0.24	0.24	0.28	0.33	0.24	0.27
013_apple	Medium	0.11	0.14	0.21	0.19	0.22	0.17
021_bleach_cleanser	Medium	0.20	0.21	0.25	0.20	0.20	0.21
035_power_drill	Medium	0.16	0.16	0.22	0.18	-0.07	0.13
055_baseball	Medium	0.14	0.17	0.18	0.16	0.19	0.17
010_potted_meat_can	Hard	0.41	0.42	0.41	0.41	0.18	0.36
008_pudding_box	Hard	0.29	0.20	0.22	0.18	-0.21	0.14
003_cracker_box	Hard	0.14	0.23	0.12	0.20	0.15	0.17
006_mustard_bottle	Hard	0.18	0.20	0.25	0.22	0.45	0.26
072-a_toy_airplane	Hard	0.22	0.31	0.31	0.18	0.20	0.24
Average	N/A	0.21	0.23	0.25	0.23	0.17	0.22

Table 5.1: Calculated ratios between the results achieved at each point density and the pessimal expectation. Pessimal values are based on the 10,000 point results in the same way Figures 5.3 and 5.4; the 10,000 point result is assumed to be off by $3m_{10,000}$ and the resulting value is the "true" optimum. Pessimal expectations are calculated by adding $3m_d$-$3m_{10,000}$ to this result for each value of m_d at each point cloud density. Negative ratios are possible, and indicate a better result was achieved at the current point cloud density than the 10,000 point cloud.

Table 5.1 quantifies much better the average results are than the pessimal results. This was calculated by taking the data from Figure 5.4 and expressing it as a ratio, rather than a raw value. For example, the 200 point sample for object 005 has a ratio of 0.25, indicating that average distance achieved for that object using only 200 points was 25% the length it would be expected to in the worst case scenario. If every result were the worst case, every entry in the table would be a 1. Entries above 1 would show sub-worst case performance, and should be impossible. Interestingly, there is no clear connection between object difficulty and improvement with increased density. The object with the worst empirical performance was the potted meat can (010), a hard object, but the best empirical performance was found for the power drill (035), a medium object.

We can observe from Table 5.1 that the average result returned is only 22% the length of the worst case result that could be returned, confirming that the typical behavior is much closer to the true values than the worst case suggests. In fact, it is even closer than I expected- none of the samples exceed half the length of the expected worst case. This suggests that while the algorithm can only guarantee a path $3m$ longer than optimal, in practice it typically returns a path $0.7m$ longer than optimal.

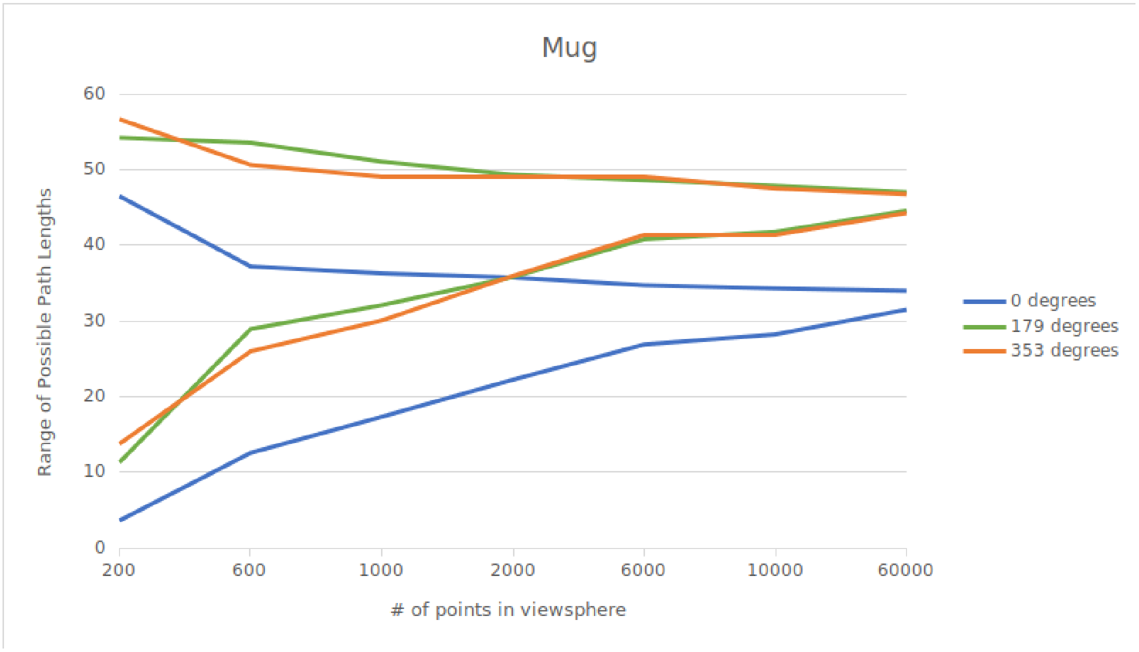

Figure 5.6: Calculated possible ranges of the optimal path length for an “Easy” object. Three initial viewpoint angles were selected for demonstration. Here, and in the following images, the range of possible optimal path lengths narrows as more information is acquired with each run.

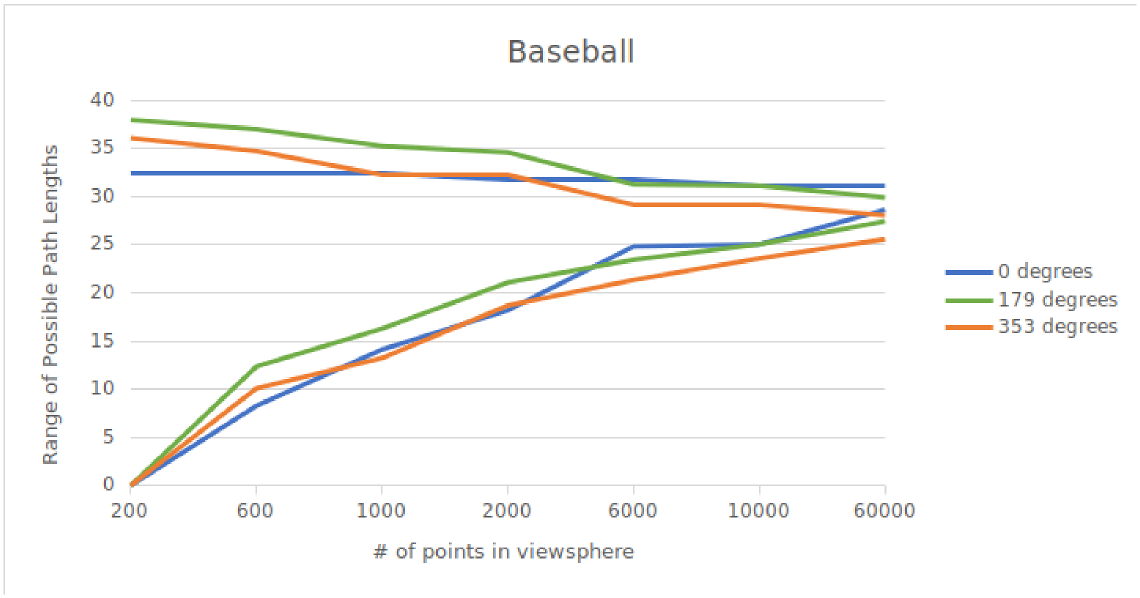

Figure 5.7: Calculated possible ranges of the optimal path length for a "Medium" object. Three initial viewpoint angles were selected for demonstration.

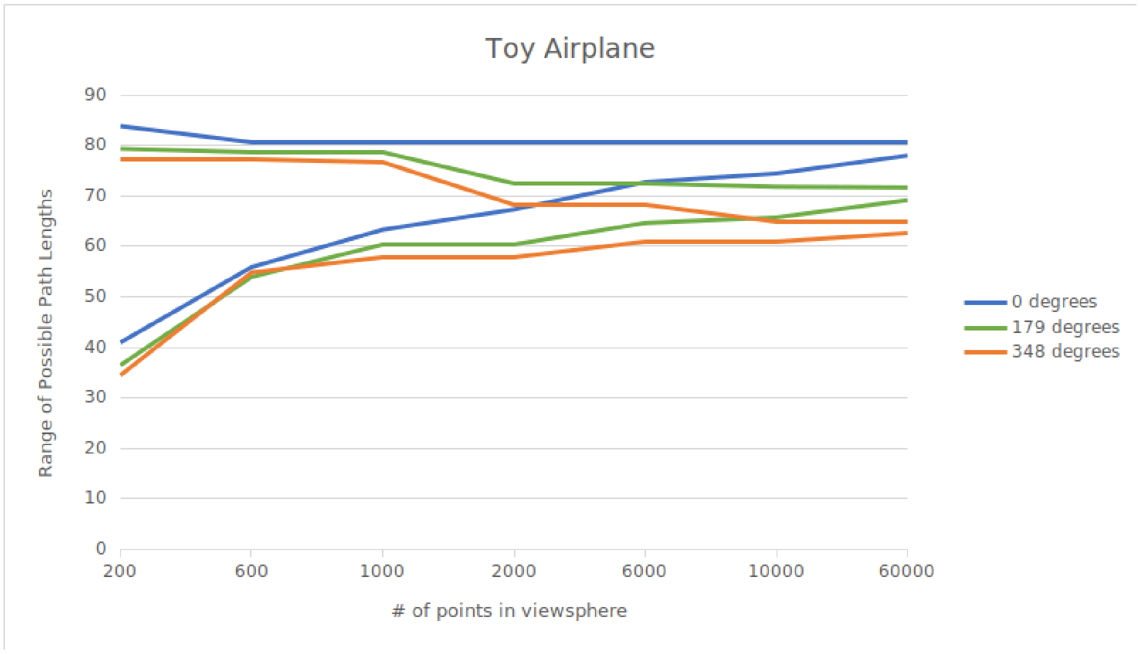

Figure 5.8: Calculated possible ranges of the optimal path length for a "Hard" object. Three initial viewpoint angles were selected for demonstration.

Chapter 6

Conclusion and Future Work

I have demonstrated a technique to approximate true optimal paths for active vision along a continuous viewsphere, and proved that this method can be made arbitrarily close to the true paths by increasing the viewsphere density. I have used simulations to explore the typical, rather than worst case, behavior of this approach on a variety of YCB objects. From this, we can conclude

1. There are significant gains to be had by moving in continuous rather than discrete space for this active vision task.

2. Searching the continuous space to a known margin of error can be less computationally expensive than searching the discrete space.

3. The estimates for optimal path length continuous searching produces are more useful for classifying object difficulty than the discrete path estimates, since they consistently fall closer to the theoretical optimum.

Epsilon-optimal continuous path planning has limitations. The setup step is computationally expensive, and so the technique is most useful for repeated distance queries. More significantly, the setup requires complete foreknowledge of the object, making it unhelpful for path planning “in the field”. Both of these limitations are inherent to my problem approach, and cannot be solved by iterative improvement. Fortunately, there are two promising use cases for the technique: as a benchmark to compare algorithms, and as an expert demonstration for machine learning techniques.

When we were restricted to the discrete case, we could not fairly benchmark continuous algorithms because they could find shorter solutions than the best solutions discrete approaches could produce. By using continuous optimal paths, we can now directly compare any active vision algorithm for grasping by measuring its deviation from the optimal length.This will allow for better benchmarking by comparing absolute performance instead of relative performance, which will clarify how much future improvement is possible. A follow up study should be performed to compare typical active vision strategies and determine how much each one can improve.

Furthermore, the technique for finding epsilon-optimal continuous paths is deliberately flexible and open to expansion. Other researchers will be able to add constraints or alternate grasping standards to adapt the technique to their own work. Adapting this approach to grippers with more than two fingers will require more work; nonetheless, the same framework of dividing the viewsphere into regions of visibility and finding paths between enough regions of visibility to construct a grasp should be applicable.

Finally, the existence of epsilon-optimal paths for this active vision problem enables new approaches. Because it is difficult for humans to produce good camera paths, past machine learning approaches have needed to develop their policies from naive starting policies. In the future, I plan to investigate using epsilon-optimal paths as an expert demonstration for reinforcement learning.

Bibliography

[1] S. Natarajan, G. Brown, and B. Calli, “Aiding grasp synthesis for novel objects using heuristic-based and data-driven active vision methods,” *Frontiers in Robotics and AI*, vol. 8, 2021.

[2] S. Caldera, A. Rassau, and D. Chai, “Review of deep learning methods in robotic grasp detection,” *Multimodal Technologies and Interaction*, vol. 2, p. 57, Sep 2018.

[3] U. Viereck, A. Pas, K. Saenko, and R. Platt, “Learning a visuomotor controller for real world robotic grasping using simulated depth images,” in *Proceedings of the 1st Annual Conference on Robot Learning* (S. Levine, V. Vanhoucke, and K. Goldberg, eds.), vol. 78 of *Proceedings of Machine Learning Research*, pp. 291–300, 2017.

[4] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, p. 6405–6416, Curran Associates Inc., 2017.

[5] C. Wang, X. Zhang, X. Zang, Y. Liu, G. Ding, W. Yin, and J. Zhao, “Feature sensing and robotic grasping of objects with uncertain information: A review,” *Sensors*, vol. 20, p. 3707, Jul 2020.

[6] A. Saxena, L. Wong, M. Quigley, and A. Y. Ng, “A vision-based system for grasping novel objects in cluttered environments,” *Springer Tracts in Advanced Robotics Robotics Research*, p. 337–348, 2010.

[7] Z. Zheng, Y. Ma, H. Zheng, Y. Gu, and M. Lin, “Industrial part localization and grasping using a robotic arm guided by 2d monocular vision,” *Industrial Robot: An International Journal*, vol. 45, no. 6, p. 794–804, 2018.

[8] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 3406–3413, Institute of Electrical and Electronics Engineers Inc., jun 2016.

[9] F. J. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3355–3362, oct 2018.

[10] A. Kurenkov, J. Ji, A. Garg, V. Mehta, J. Gwak, C. Choy, and S. Savarese, "DeformNet: Free-form deformation network for 3D shape reconstruction from a single image," in *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, vol. 2018-January, pp. 858–866, Institute of Electrical and Electronics Engineers Inc., may 2018.

[11] H. Zhang and Q. Cao, "Fast 6D object pose refinement in depth images," *Applied Intelligence*, vol. 49, pp. 2287–2300, jun 2019.

[12] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, pp. 1677–1734, mar 2021.

[13] M. Salganicoff, L. H. Ungar, and R. Bajcsy, "Active learning for vision-based robot grasping," *Machine Learning*, vol. 23, no. 2-3, p. 251–278, 1996.

[14] S. Khalfaoui, R. Seulin, Y. Fougerolle, and D. Fofi, "View planning approach for automatic 3D digitization of unknown objects," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7585 LNCS, pp. 496–505, Springer Verlag, 2012.

[15] J. Daudelin and M. Campbell, "An Adaptable, Probabilistic, Next-Best View Algorithm for Reconstruction of Unknown 3-D Objects," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1540–1547, jul 2017.

[16] X. Fu, Y. Liu, and Z. Wang, "Active Learning-Based Grasp for Accurate Industrial Manipulation," *IEEE Transactions on Automation Science and Engineering*, vol. 16, pp. 1610–1618, oct 2019.

[17] B. Calli, M. Wisse, and P. Jonker, "Grasping of unknown objects via curvature maximization using active vision," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 995–1001, 2011.

[18] E. Arruda, J. Wyatt, and M. Kopicki, "Active vision for dexterous grasping of novel objects," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2881–2888, 2016.

[19] L. Paletta and A. Pinz, "Active object recognition by view integration and reinforcement learning," *Robotics and Autonomous Systems*, vol. 31, pp. 71–86, 2000.

[20] B. Calli, W. Caarls, M. Wisse, and P. P. Jonker, “Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation,” *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1810–1822, 2018.

[21] B. Rasolzadeh, M. Björkman, K. Huebner, and D. Kragic, “An active vision system for detecting, fixating and manipulating objects in the real world,” *The International Journal of Robotics Research*, vol. 29, pp. 133 – 154, 2010.

[22] B. Calli, W. Caarls, M. Wisse, and P. Jonker, “Viewpoint optimization for aiding grasp synthesis algorithms using reinforcement learning,” *Advanced Robotics*, vol. 32, no. 20, pp. 1077–1089, 2018.

[23] D. Gallos and F. Ferrie, “Active vision in the era of convolutional neural networks,” in *2019 16th Conference on Computer and Robot Vision (CRV)*, pp. 81–88, 2019.

[24] G. de Croon, I. Sprinkhuizen-Kuyper, and E. Postma, “Comparing active vision models,” *Image and Vision Computing*, vol. 27, no. 4, pp. 374–384, 2009.

[25] P. Ammirato, P. Poirson, E. Park, J. Košecká, and A. C. Berg, “A dataset for developing and benchmarking active vision,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1378–1385, 2017.

[26] V. Karasev, A. Chiuso, and S. Soatto, “Control recognition bounds for visual learning and exploration,” in *2013 Information Theory and Applications Workshop (ITA)*, pp. 1–8, 2013.

[27] E. S. Low, P. Ong, and K. C. Cheah, “Solving the optimal path planning of a mobile robot using improved Q-learning,” *Robotics and Autonomous Systems*, vol. 115, pp. 143–161, may 2019.

[28] M. J. Bency, A. H. Qureshi, and M. C. Yip, “Neural Path Planning: Fixed Time, Near-Optimal Path Generation via Oracle Imitation,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 3965–3972, nov 2019.

[29] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[30] K. Magzhan and H. M. Jani, “A Review And Evaluations Of Shortest Path Algorithms,” *INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH*, vol. 2, no. 6, pp. 99–104, 2013.

[31] S. E. Drayfus, “AN APPRAISAL OF SOME SHORTEST-PATH ALGORITHMS,” 1963.

[32] V. Paschos, "An overview on polynomial approximation of np-hard problems," *Yugoslav Journal of Operations Research*, vol. 19, no. 1, 2016.

[33] R. Solovay and V. Strassen, "A fast monte-carlo test for primality," *SIAM Journal on Computing*, vol. 6, no. 1, pp. 84–85, 1977.

[34] M. Thorup and U. Zwick, "Approximate Distance Oracles," *Journal of the ACM*, vol. 52, no. 1, pp. 1–24, 2005.

[35] K. Blind, M. Böhm, P. Grzegorzewska, A. Katz, S. Muto, S. Pätsch, and T. Schubert, "Study about the impact of open source software and hardware on technological independence, competitiveness and innovation in the eu economy," 2021.

[36] L. M. Haddad, P. Annamaraju, and T. J. Toney-Butler, "Nursing shortage," *British Medical Journal*, vol. 3, pp. 534–535, 2 2022.

[37] C. N. Agraz, M. Pfingsthorn, P. Gliesche, M. Eichelberg, and A. Hein, "A survey of robotic systems for nursing care," *Frontiers in Robotics and AI*, vol. 9, 4 2022.

[38] R. Z. Lawrence and L. Z. E. R. Lawrence, "Policy brief us employment deindustrialization: Insights from history and the international experience," *Peterson Institute for International Economics*, 2013.

[39] V. I. Postrel, *Thread.* Basic Books, 2021.

[40] E. Camiña, Ángel Díaz-Chao, and J. Torrent-Sellens, "Automation technologies: Long-term effects for spanish industrial firms," *Technological Forecasting and Social Change*, vol. 151, p. 119828, 2 2020.

[41] P. Aghion and C. Antonin, "What are the labor and product market effects of automation? new evidence from france," *CEPR Discussion*, 2020.

[42] J. Bessen, M. Goos, A. Salomons, and W. van den Berge, "Firm-level automation: Evidence from the netherlands," *AEA Papers and Proceedings*, vol. 110, pp. 389–93, 5 2020.

[43] G. Domini, M. Grazzi, D. Moschella, and T. Treibich, "For whom the bell tolls: The effects of automation on wage and gender inequality within firms," *SSRN Electronic Journal*, 10 2020.

[44] A. Namiki, Y. Nakabo, I. Ishii, and M. Ishikawa, "High speed grasping using visual and force feedback," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4, pp. 3195–3200.

[45] X. Li, X. Su, Y. Gao, and Y.-h. Liu, "Vision-Based Robotic Grasping and Manipulation of USB Wires," vol. 1, pp. 3482–3487, 2018.

[46] O. B. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1105–1116, 2010.

[47] S. J. Dickinson, H. I. Christensen, J. K. Tsotsos, and G. Olofsson, "Active object recognition integrating attention and viewpoint control," *Computer Vision and Image Understanding*, vol. 67, pp. 239–260, sep 1997.

[48] R. Cheng, A. Agarwal, and K. Fragkiadaki, "Reinforcement Learning of Active Vision for Manipulating Objects under Occlusions," nov 2018.

[49] D. Rakita, B. Mutlu, and M. Gleicher, "Remote Telemanipulation with Adapting Viewpoints in Visually Complex Environments," in *Proceedings of Robotics: Science and Systems*, 2019.

[50] D. Fischinger and M. Vincze, "Empty the basket - A shape based learning approach for grasping piles of unknown objects," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 2051–2057, 2012.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2015.

[53] M. Roberts, "How to evenly distribute points on a sphere more effectively than the canonical fibonacci lattice — extreme learning," 6 2020.

[54] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research:," *http://dx.doi.org/10.1177/0278364917700714*, vol. 36, pp. 261–268, 4 2017.