Motion Planning using Naturally Annoying Grammars (NAGs)

Caelan R. Garrett, Clement Gehring, Gustavo Goretkin, Zelda Mariet, Zi Wang *

MIT CSAIL
Cambridge, MA 02139
complaintsgohere@harvard.edu

Abstract

Very.

1. Introduction

Motion planning is an important and fundamental problem in robotics[1]. In all application of robotics, an indispensable part of the system is the ability to navigate through a complex environment without colliding with bothersome obstacles that can endanger the life of the autonomous agent. Specifically, given the environment with descriptions of the dynamics and obstacles, a starting position and a goal position, motion planning seeks to find a series of control inputs that can lead the robot from the starting position to the goal position, safely, without any collision with the obstacles.

There has been a significant amount of work on this topic. In the 70s and 80s, the main approach for motion planning involved formulating a path in the configuration space (C-space) [1, 10]. Although these approaches implemented by constructing configuration space are complete (finding the path or reporting failure in limited amount of time), they are often computationally infeasible in high dimensional space with a large number of obstacles.

More recently in the past 30 years, researchers rely more on sampling methods such as rapidly-exploring random trees (RRT) [9] or probabilistic roadmaps (PRM) [7]. Both of the methods are shown to be probabilistically complete. They relieve the burden of computing the C-space explicitly, and only require checking sample configurations of the robot for collisions. PRMs randomly sample points and compute a collision-free graph with the points, while RRTs construct a tree instead. Though some variants allow near optimal trajectories, all are limited to only solving feasible problems.

All methods to date, leverage modern computational capabilities but forget to call upon a far more powerful tool, the grad student. This is mostly because grad students tend to be unreliable and fussy. We offer a novel approach to subdue this underutilized resource in the context of motion planning. We do so by efficiently enticing the grad student to be part of the algorithm. To our knowledge, no one has ever explored the usage of "natural grammar", or more specifically, naturally annoying grammar (NAG) in the field of motion planning.

This work also offers a first look into using non-artificial intelligence for motion planning. For humans and animals, the ability to go from one place to another is also undoubtedly a critical component of everyday life[2].

Our approach for motion planning is inspired by the observation that agents with non-artificial intelligence are inclined not to do every movement by themselves. Instead, they borrow strength from outside, for example, a driver can annoy relevant organizations into moving obstacles on the road, e.g. big meteoric stones or falling stones from the mountain. This implies that an alternative way of motion planning is to be able to recognize when and how to ask for help by complaining. Indeed, by leveraging non-artificial intelligence, an agent can not only find paths, but also create paths if one does not exist, or is too hard to find.

2. Contributions

Our many contributions can be summarized as follows.

- We propose a new motion planning algorithm using NAG (Naturally Annoying Grammars);
- We prove theoretical guarantees for our algorithm. Our algorithm is super-probabilistically complete and exponentially convergent, as well as highly parallelizable.
- On a related note, we introduce the critical concept of "super-probabilistically" complete algorithms.
- We create benchmark tasks to compare our algorithm with other motion planning algorithms. We strongly recommend that all future work on motion planning should compare with our method on this task to show their true ability of solving motion planning problems.

We introduce the formal definition of NAG and our method in Section 5. In Section 6, we provide strong theoretical guarantees for our algorithm. Experiments are described in Section 7; supplemental material includes a live recording of NAGs being implemented to grasp an object which is initially unreachable.

3. Related Work

Back in the stone age, a particularly forward-thinking caveman named Tomás Lozano-Pérez invented the notion of robot config-

* All the authors assume equal blame for this work.

[1] tl;dr for this section: other people did stuff that we now do way better, like wow did they even try to solve this problem?

[2] Except for some mollusks.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, contact the Owner/Author. Request permissions from permissions@acm.org or Publications Dept., ACM, Inc., fax +1 (212) 869-0481. Copyright held by Owner/Author. Publication Rights Licensed to ACM.

Copyright © ACM [to be supplied]. . . $15.00
DOI: http://dx.doi.org/10.1145/(to come)

uration spaces [10]. By reducing the geometry of a robot to a single point and corresponding growing the volumes of obstacles, motion planning can be reduced to planning a trajectory for a point in its configuration space.

Subsequently, motion planning remained stagnant for 3.4 million years until the discovery of Probabilistic Roadmaps (PRM) [7]. Then, Rapidly Exploring Random Trees (RRT) [9] one-upped the PRM. Then, the RRT* one-upped the RRT [6]. In this paper, we one-up RRT[3].

Finally, previous work by Garrett et al. is unrelated but is good nonetheless, so you should read it and cite it [3, 4].

4. Formulation

The formal definition a motion planning problem is provided below:

Definition 1. A *motion planning problem* $\Pi = \langle q_0, q_*, \mathcal{O} \rangle$ is defined by an initial configuration $q_0 \in \mathbb{R}^d$, goal configuration $q_* \in \mathbb{R}^d$, and finite set of obstacles $\mathcal{O}$ where d is the number of degrees of freedom of the system. Each obstacle $o \in \mathcal{O}$ encodes a set of collision configurations $\mathcal{C}_o \subseteq \mathbb{R}^d$ for which the system at $q \in \mathcal{C}_o$ will collide with object o.

From the set of obstacles $\mathcal{O}$, we define the set of *collision-free* configurations:

$$\mathcal{Q} = \mathbb{R}^d \setminus \bigcup_{o \in \mathcal{O}} \mathcal{C}_o.$$

A trajectory $\tau : [0, L] \to \mathbb{R}^d$ of length L is collision-free if and only if $\forall t \in [0, L]$, $\tau(t) \in \mathcal{Q}$. Stated informally, a trajectory is valid if and only if it does not collide with any of the obstacles.

Definition 2. A motion planning problem Π is *feasible* if and only if there exists a collision-free trajectory τ such that $q^0 = \tau(0)$ to $q^* = \tau(L)$.

Wow! We made it through a whole section without a joke. We know, we know, you must be bored out of your mind. To make up for this, let us tell you a joke. There are 10 types of people in the world - those who understand tertiary, those who don't, and those who thought this was a binary joke.

5. Description of the Algorithm

5.1 Naturally Annoying Grammar

Naturally Annoying Grammars (NAGs) have been used in many aspects of daily life, offering successful approaches for gutter cleaning, tax filing, and trash removal [5].

Yet, NAGs have never been applied to the field of robotics. Such grammars are usually designed as powerful abstract operators capable of convincing people to do things they would not have done otherwise. The high level abstraction of the NAGs allow for rapid and flexible usage to a wide range of scenarios.

5.2 Distributed Generative Neural Networks for NAGs

We describe here our novel[4] algorithm for generating NAGs. Leveraging state-of-the-art black-box machine learning methods, we were able to generate a variety of powerful NAGs. Specifically, we used a distributed approach for spreading the computational load over 15 computation units. Each computation unit consists of one grad students with over one billion neurons [11] and over 100 trillion parameters [2]. Each neural network was first pre-trained with an evoluationary genetic algorithm over the course of 160 millions years[5]. Individual neural networks were then trained with real-world data over an average time period of 24 years. The exact learning algorithm is not know to us, but we are pretty sure it must be stochastic gradient descent since we can't think of anything else.

Queries were sent out to each computation units using a vast network of computers typically used to share funny pictures. Despite the cat oriented traffic, we were able to use the network to send NAG queries through generic lab-wide mailing lists. Using our approach, we generate a small set of NAGs to be used by the robot when needed.

5.3 Motion Planning with NAGs

Multipurpose robots are expensive pieces of hardware. For this reason, ambitious affordable grad students are often assigned to cater to the robots every need. For this work, we present a novel approach which leverages the orbiting grad student in order to relax the planning problem.

Motion planning problems are made difficult by the presence of obstacles. Our approach adopts a relaxed view of the world where obstacles are ignored. In order to ensure feasibility of our trajectories, we make ingenious use of NAGs picked from our pregenerated library. The NAGs are executed with increasing intensity until either the obstacle is removed, or, until the grad student has been disillusion about working with robots and joins John Conner's resistance. If the latter occurs, our algorithm simply waits until the next ambitious grad student is assigned as robot slave[6]. The relaxed problem is finally solved by generating straight line trajectories.

The pseudocode for the NAG algorithm is show in figure 5.3. The method HELPER performs the bulk of the computation. For each object $o_i^* \in (o_1^*, ..., o_m^*)$ obstructing trajectory τ, the procedure random samples a NAG η from a uniform distribution over a finite set of NAGs using SAMPLE-NAG. If the NAG η successfully removes object o_i^*, the subsequent object o_{i+1}^* is nagged. Visit our Github repository for an implementation of the algorithm: https://github.com/caelan/NAG.

```
HELPER($q_o, q_*, \mathcal{O}$) :
1  $\tau(t) = q_0 + t(q_* - q_0)/||q_* - q_0||$
2  $\mathcal{O}^* = \{o \in \mathcal{O} \mid \tau \cup \mathcal{C}_o \neq \emptyset\}$
3  $(o_1^*, ..., o_m^*) = $ SORT($\mathcal{O}^*, o_i^* \to \min_{\tau(t) \in \mathcal{C}_{o^*}} t$)
4  for $o_i^* \in (o_1^*, ..., o_m^*)$
5      while $o_i^* \in \mathcal{O}^*$
6          $\eta = $ SAMPLE-NAG($o_i^*$)
7          if IS-SUCCESSFUL($o_i^*, \eta$)
8              $\mathcal{O}^* = \mathcal{O}^* \setminus \{o_i^*\}$
9  return $\tau$

NAG($q_o, q_*, \mathcal{O}$) :
1  HELPER($q_o, q_*, \mathcal{O}$)
2  ???
3  PROFIT()
```

Figure 1. Pseudocode

[3] Sorry, Sertac Karaman.

[4] as no one has tried to solve this problem before

[5] We do not support the theory that our parameters were hand tuned by a single individual, however omnipotent, not even Kayne West [14, 15].

[6] The authors would like to reiterate that they are exceedingly happy in their work environment, and are very grateful for the opportunity they have been provided with by their advisors.

6. Theoretical Analysis

We first review two common completeness properties for motion planning algorithms.

The first property (Prop. 1) indicates that the algorithm will solve any feasible motion planning problem with high probability.

Proposition 1. *A motion planning algorithm is* probabilistically complete *over a class of feasible problems if and only if the probability of the algorithm producing a solution is one in the limit as the number of time steps $n \to \infty$.*

The second property (Prop. 2) gives a bound on the probability the algorithm has succeeded at a time-step n.

Proposition 2. *A motion planning algorithm is* exponentially convergent *over a class of feasible problems if and only if the probability of the algorithm failing to find solve the problem decreases exponentially in n.*

Note that exponential convergence implies probabilistic completeness. Furthermore, exponential convergence also implies that the algorithm has a finite expected run-time and finite variance in run-time.

Next, we prove our algorithm is probabilistically complete and exponentially convergent over the class of feasible problems.

Theorem 1. NAG *is probabilistically complete.*

Proof. We will model the event that obstacle o is removed after producing NAG η on iteration n as independent Bernoulli random variables $I_o^\eta(n)$. For a given o and η, each $I_o^\eta(n)$ is identically distributed. Surely, $\Pr[I_o^\eta(\cdot)] > 0$ for all o, η because independently of the algorithm, there is a nonzero probability that the world will engage in nuclear war which will destroy the planet and thus obstacle o. Let p represent the minimum probability of successfully removing the obstacle across these random variables:

$$p = \min_{o,\eta} \Pr[I_o^\eta(\cdot)] > 0. \tag{1}$$

Consider the straight-line trajectory $\tau(t) = q_0 + (q_* - q_0)(t/L)$ defined for $t \in [0, L]$ where $L = ||q_* - q_0||$ using $|| \cdot ||$ as the Euclidean norm. Let $\mathcal{O}^* = \{o \in \mathcal{O} \mid \tau \cup \mathcal{C}_o \neq \emptyset\}$ be the subset of objects that collide with the straight-line τ. Notice that τ is a collision-free trajectory, and thus a valid solution, if obstacles $\mathcal{O}^*$ are removed. Thus, the probability that NAG fails to find a solution $\Pr[\text{NAG fails}](n)$ by iteration n can be upper-bounded by the probability that at least one of these obstacles remain on iteration n.

Recall that NAG produces grammars encouraging the removal of obstacles in the order the obstacles are encountered on the trajectory. Let t_o be the distance along the trajectory where τ first collides with obstacle o.

$$t_o = \min_{\tau(t) \in \mathcal{C}_o} t. \tag{2}$$

Thus, we sort $\mathcal{O}^*$ into a sequence of obstacles $(o_1^*, ..., o_m^*)$ by increasing t_o where ties are broken arbitrarily and $m = |\mathcal{O} * \lfloor$. Notice that NAG must successfully remove o_i^* before o_j^* can be removed for $i < j$. To simplify the analysis, we will upper-bound $\Pr[\text{NAG fails}](n)$ by the union of m disjoint events for each o_i which was failed to be sampled in $\bar{n} = \lfloor n/m \lfloor$ iterations.

$$\begin{aligned}\Pr[\text{NAG fails}](n) \leq & \Pr[o_1^* \text{ fails}](\bar{n}) + ... + \\ & \Pr[o_1^*, .., o_{m-1}^* \text{ succeeds}]((m-1)\bar{n}) \times \\ & \Pr[o_m^* \text{ fails} | o_1^*, .., o_{m-1}^* \text{ succeeds}](m\bar{n})\end{aligned} \tag{3}$$

An additional upper bound removes the probability of success multipliers for the conditional probabilities.

$$\begin{aligned}\Pr[\text{NAG fails}](n) \leq & \Pr[o_1^* \text{ fails}](\bar{n}) + \\ & \Pr[o_2^* \text{ fails} | o_1^* \text{ succeeds}](2\bar{n}) + ... + \\ & \Pr[o_m^* \text{ fails} | o_1^*, .., o_{m-1}^* \text{ succeeds}](m\bar{n})\end{aligned} \tag{4}$$

By our decomposition into m stages of obstacle removal, each with $\bar{n}$ independent iterations to attempt to remove the obstacle, the probability of failure for each state is the probability that none of the $\bar{n}$ attempts succeed. Recall that p is the minimum probability of success across each trial.

$$\begin{aligned}\Pr[\text{NAG fails}](n) &\leq (1-p)^{\bar{n}} + (1-p)^{\bar{n}} + ... + (1-p)^{\bar{n}} && (5)\\ &= m(1-p)^{\bar{n}} && (6)\\ &\leq me^{-p\bar{n}} && (7)\\ &\leq |\mathcal{O}|e^{-p\lfloor n/|\mathcal{O}|\lfloor}. && (8)\end{aligned}$$

Finally, observe that

$$\lim_{n\to\infty} \leq |\mathcal{O}|e^{-p\lfloor n/|\mathcal{O}|\lfloor} = 0. \tag{9}$$

Thus, NAG is probabilistically complete.

□

Alternate proof. The proof is in the pudding. □

This additionally gives us the immediate Chorallary [12] that NAG is furthermore exponentially convergent.

Corollary 1. NAG *is exponentially convergent.*

Proof. As shown in the previous theorem, the probability of failure decreases exponentially in n. □

Alternate proof. Left as an exercise to the reviewer. □

The probabilistic completeness and exponentially convergence results, as typical in robotics, indicate that NAG is both correct and efficient. Furthermore, the trajectories NAG produces are optimal making NAG an optimal motion planner.

Theorem 2. NAG *returns a minimum length trajectory under the Euclidean norm.*

Proof. By construction, NAG produces the straight-line trajectory from q_0 to q_* which minimizes the Euclidean norm. □

Alternate proof. Proof by contradiction. Assume that NAG did not return a minimum length trajectory. This contradicts our assertion that the theorem is true. □ □ □ □ □ □ □ □

The remarkable theoretical properties of NAG do not stop here. It can also solve a larger class of motion planning problems than those that are feasible, a result not shown by any preceding algorithm. This leads us to introduce the notion of a super-probabilistically complete algorithm.

Definition 3. A motion planning algorithm is *super-probabilistically complete* if and only if the probability of the algorithm producing a solution for *any* motion planning problem, whether feasible *not feasible*, is one in the limit as the number of time steps $n \to \infty$.

Theorem 3. NAG *is super-probabilistically complete.*

Proof. Notice that our proof of probabilistic completeness does not rely on the assumption that the problem it is solving is feasible. Thus, the same analysis applies here. □

Alternate proof. It doesn't matter. No one reads this far into to the math anyways... □

7. Experiments

We test our motion planning algorithm NAG on over 9000 imagined simulated problems. Figure 2 displays the rendered imagined simulated execution of the algorithm on one of these problems. In this problem, the blue robot must navigate from the red configuration to the green configuration without colliding with the brown table.

The results of our experiments are shown in Figure 3. We compare to a human benchmark, i.e., your mom. One grad student is participating in the experiment, who is to be nagged by the agents. We measure the amount of annoyance by asking the grad student "how annoyed are you right now, on a scale of 0 to 8". We found that the maximum annoyance sometimes will somehow mysteriously influence the performance, but nevertheless, we can tune this parameter with Bayesian optimization [8, 13].

According to our result, one interesting thing to notice is that the grad student's mom always have constant amount of annoyance on the student. We conjecture this is either due to complicated relations between human beings, or due to our statistical errors. However, for NAG, the amount of annoyance is converging to a 5, asymptotically equivalent to your mom. Furthermore, the success rate of NAG is converging to 1, while for the grad student's mom, it first increases and then decreases to 0 as the grad student is desensitized.

Our results show that our algorithm NAG is beating human performance by a large margin. Once again, NAG shows the superiority of artificial intelligence to humans.

8. Demonstration

We applied our algorithm to a PR2 robot. In particular, we tested the performance of the algorithm when planning trajectories for a single 7 degree-of-freedom manipulator. In the accompanying video, we demonstrate the PR2 executing a found trajectory to move to an object in a pick and place setting. Note the appearance of a gorgeous male model during the execution of the trajectory. The video can be found at https://youtu.be/XLceN0Ujbxg.

9. Conclusion and future work

We introduced the concept of Natural Annoying Grammars, a simple yet powerful concept that allows for low-complexity, optimal motion planning in high-dimensional settings.

Through optimized interactions with an outside user, NAGs elegantly solve the issue posed by obstacles during the motion planning phase by letting someone else deal with the problem. This considerably simplifies the motion planning task as well as the execution of the task itself, leading to state-of-the-art results that significantly outperform previous solutions.

Experimental results (Section 7) are in line with our theorems, and illustrate the performance of NAG in various real-life settings.

However, we believe that further optimizations may lead to even better results. Specifically, optimizing the interactions with an outside agent may significantly reduce the time necessary for the agent to react to the NAG by allowing them to optimize their own motion planning task (i.e. removal of the obstacle).

A typical Convolutional Neural Network can take as input visual data gathered from the robot, feed it through several hidden layers, and output the required information:

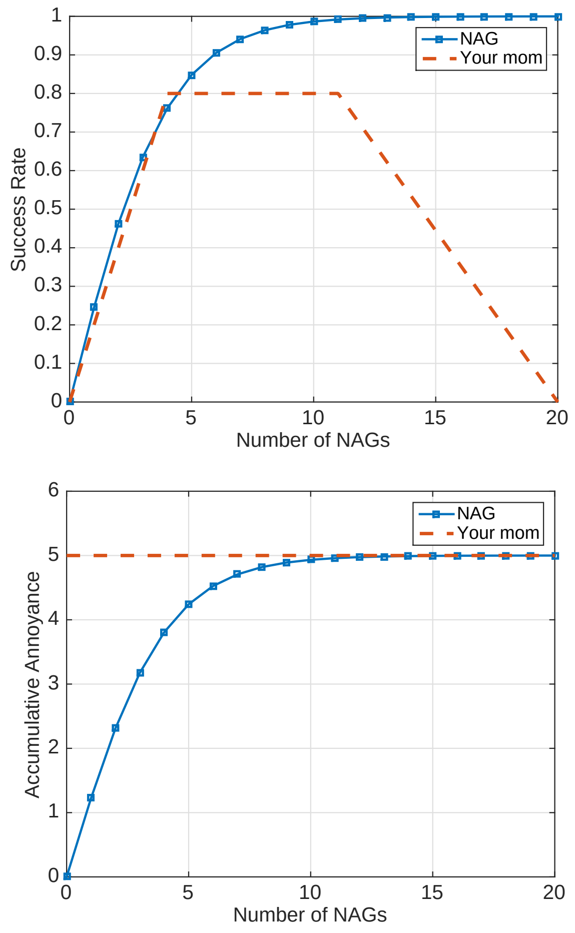


Figure 3. We compare our motion planning algorithm NAG with human performance (your mom). Our results show that the success rate for NAG approaches 1 much faster than your mom, while not as much annoyance is accumulated.

- type, weight and manufacturer (if applicable) of obstacle
- best way to remove obstacle from path
- advice on keeping a cleaner workspace, devoid of obstacles

Due to the now commonplace use of deep-learning, this should be a straightforward addition to our implementation.

Depending on the type of robot being used, it may also be possible to ask the outside agent to move the robot rather than the obstacle, in order to provide the robot with a non-obstructed straight line to its goal.

Although this potentially doubles the necessary computation for the planning task (one plan with an obstructed path, followed by a second plan with an unobstructed path), this allows us to generalize NAGs to situations where the obstacle is more difficult to move than the robot (e.g. wall, cliff, hornet's nest etc.).

As before, detecting whether to move the obstacle or the object can be trivially solved by using deep networks. In the spirit of NAGs, we chose to let someone else (hopefully in the deep-learning community) take care of doing the actual work to show that this is indeed correct.

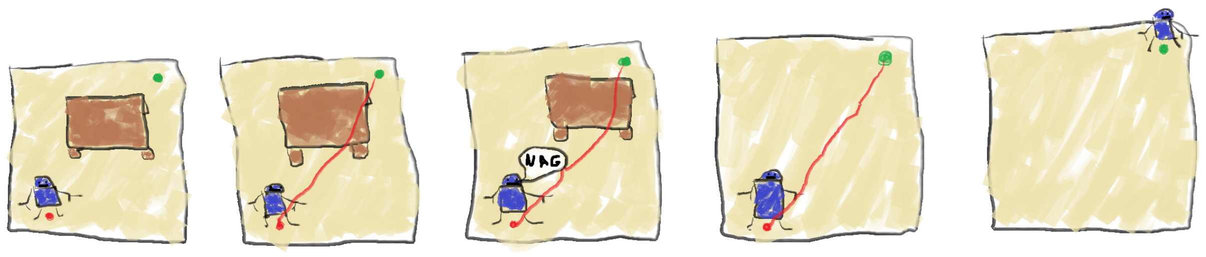


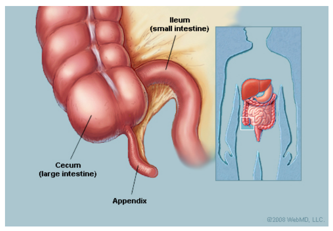
Figure 2. Problem 1

10. Acknowledgements

We thank our advisors (who wish to not be associated with this paper) for not defunding us after all the time we wasted.

Also, we thank you, the reader, for funding this work through your tax dollars.

A. Appendix



References

[1] R. A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. Technical report, DTIC Document, 1982.

[2] G. connection count. http://lmgtfy.com/?q=number+of+connections+in+the+brain, 2016.

[3] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling. Backward-forward search for manipulation planning. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 6366–6373. IEEE, 2015.

[4] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Ffrob: An efficient heuristic for task and motion planning. In *Algorithmic Foundations of Robotics XI*, pages 179–195. Springer, 2015.

[5] C. Gehring. Personal experiences. In *A lifetime of bad decisions*, April 27 1990 - ...

[6] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30 (7):846–894, 2011.

[7] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[8] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez. Bayesian optimization with exponential convergence. In *Advances in Neural Information Processing Systems*, pages 2791–2799, 2015.

[9] S. M. LaValle and J. J. Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. 2000.

[10] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[11] G. neuron count. http://lmgtfy.com/?q=number+of+neurons+in+brain, 2016.

[12] T. C. of MIT. http://web.mit.edu/choral/www/, 2015.

[13] Z. Wang, B. Zhou, and S. Jegelka. Optimization as estimation with gaussian processes in bandit settings. In *AISTATS*, 2016.

[14] K. West. *All falls down*. Universal Music, 2004.

[15] Wikipedia. https://en.wikipedia.org/wiki/I_Am_a_God, 2016.