

Quantum Search for Dexterous Manipulation Planning

S.E. Venegas-Andraca^(a), L. A. Muñoz^(b) and A. Espinosa^(b)

^(a)Centre for Quantum Computation, University of Oxford.

Clarendon Laboratory, Parks Road, Oxford OX1 3PU, U.K. E-mail: sva@mindsfomexico.org

^(b)LI²CoViR. Facultad de Matemáticas, UADY, Mérida, Yucatán, México. Email: liicovir@yahoo.com

Abstract

One of the main problems in robotics is the computation of robot motion taking into account obstacles. In this paper we present a proposal for the computation of valid paths for a robotic (articulated) hand using a hybrid algorithm, partly classical and partly quantum. The quantum component is based on Grover's quantum search algorithm.

1 Introduction

Obstacle avoidance is a fundamental problem in robotics. The connectivity between two parts of the *C-space* (*Configuration Space*, a most useful tool to define volumetric characteristics of a robot and the obstacles it must avoid) is considered a valid *path* for a robot. For each point of a valid *path* there is an attainable *pose* of the robot. Therefore, planning the motion of a robot between obstacles is equivalent to computing a path in a manifold. Among relevant problems in motion planning we find the Dexterous Manipulation problem.

In this poster we present a hybrid algorithm, partly classical and partly quantum, that allows efficient computation of solutions to the dexterous manipulation problem. The quantum part of our algorithm is an adaptation of Grover's search algorithm [Grover, 1996].

2 The Dexterous Manipulation Problem

Let us suppose we have an ideal *hand* designed to grasp and manipulate an *object*. Our aim is to compute the corresponding finger motions so that the object moves towards a desired pose. The problem of dexterous manipulation can be stated as an input-output system: joint motions (inputs) are applied to the fingers of an articulated hand and the grasped object experiences a motion (output). Computing the corresponding object motion, given a set of finger motions, requires to know the whole finger and object kinematics or, alternatively, to have access to an external pose estimation system.

A general definition of the inverse problem of dexterous manipulation using a 9 dof articulated hand follows. Let a three-fingered robot hand M working in the Euclidean space \mathcal{W} , be represented by the union of its fingers, $F_1 \cup F_2 \cup F_3$. Since each finger has three degrees of freedom,

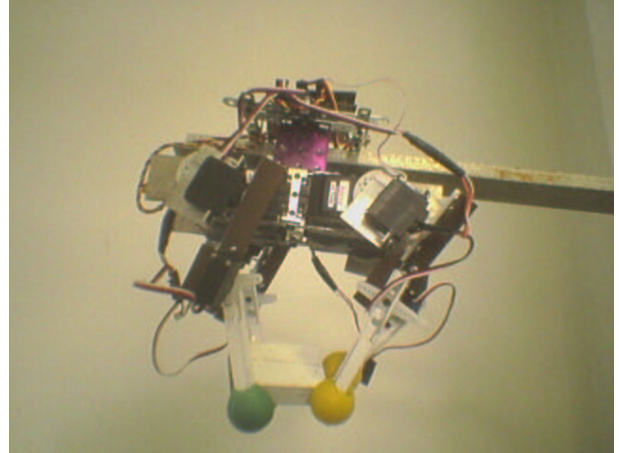


Figure 1: . A prototype of the dexterous hands available at the FMAT

a joint space configuration for the hand is specified as a tuple $q = (\theta_1, \dots, \theta_9)$. The grasped polyhedral object is denoted as \mathcal{O} and its position and orientation are determined by a reference frame T placed in its center of gravity. The initial configuration of the system includes T_0 and T_f (initial and final configurations for the object), as well as an initial hand configuration q_0 . Our goal is to find a sequence of hand configurations $\tau = q_0, q_1, \dots, q_n$, to take \mathcal{O} from T_0 to T_f .

2.1 A local dexterous manipulation algorithm

Muñoz *et al* [Muñoz *et al*, 1995] have presented an algorithm for the computation of hand configurations $\tau = q_0, q_1, \dots, q_n$, to take \mathcal{O} from T_0 to T_f . This algorithm performs two tasks. Firstly, it explores the space of potential solutions for the problem of changing the orientation of an object that is being grasped by a dexterous hand. This space is organized as sets of potential infinitesimal motions to pass from one hand configuration to the next until the desired trajectory is completed. As a result of this exploration, the algorithm delivers a list of hand configurations suitable to be used as solutions. Let us denote those suitable hand configurations by $\vec{C}_1, \vec{C}_2, \dots, \vec{C}_n$. Secondly, the algorithm evaluates and selects the best solution(s), that is, the algorithm computes a cost function using $\vec{C}_1, \vec{C}_2, \dots, \vec{C}_n$ as input and delivers as

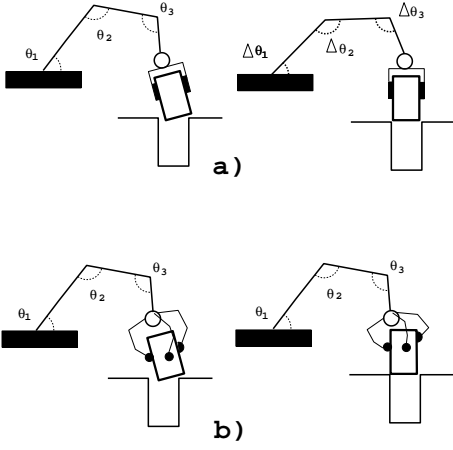


Figure 2: . a) The use of a simple industrial gripper b) The use of a robotic hand allows more dexterity.

output those configurations that fulfill cost requirements. It must be noted that cost functions are case-dependant. For example, in a certain case one could be interested in minimizing the total displacement distance, while in another case one could be looking for those configurations that minimize the number of (say) rotations.

3 The planning algorithm

In a first phase, this algorithm explores the space of potential solutions for the problem of changing the orientation of an object being grasped by a dexterous hand. This space is organized as sets of potential infinitesimal motions to pass from one hand configuration to the next until the desired trajectory is completed. The second phase consists in the evaluation and selection of the best solution.

The inputs of the algorithm are: • The initial (q_0) and final (q_f) configurations of the grasped-object. • The positions of the initial contact points (φ_0).

The outputs of the algorithm at any given iteration are: • A coordinated sequence of movements in the joint space. • A sequence of forces for the stability.

The algorithm can be described as follows:

```

algorithm fixed_point-rolling-sliding-commutation
01 begin
02 begin
03   Trajectory-decomposition
04    $T_j^{j+1} \leftarrow \text{first\_partition\_of\_trajectory}$ 
05 end
06 while NOT(end_of_trajectory)
07   for finger=1 to finger=3
08   begin
09     if (valid = Inverse_Kinematics( $\varphi_o, \vec{P}_f, \vec{N}_f$ ))
10       then Movement mode: FIXED POINT(FP)
11         solutions[index]  $\leftarrow [FP, \Delta q, \varphi_o, \vec{P}_f, \vec{N}_f]$ 
12     list_phi[1..nm] = points_around_phi_o
13     list_surface[1..nm] = points_around_surface_object

```

```

14   list_normals[1..nm] = normal_of_points_in_surface
15   for i = 1 to i = nm
16     for j = 1 to j = nm
17       if (valid = Inverse_Kinematics( $\varphi_i, \vec{P}_j, \vec{N}_j$ ))
18         then
19           if de = d
20             then Movement mode: ROLLING(R)
21               solutions[index]  $\leftarrow [R, \Delta q, \varphi_i, \vec{P}_j, \vec{N}_j]$ 
22             else Movement mode: SLIDING(S)
23               solutions[index]  $\leftarrow [S, \Delta q, \varphi_i, \vec{P}_j, \vec{N}_j]$ 
24             else
25               Movement mode : Finger relocation1
26   end
27   Triplet's-generation(solutions[])
28   Force-calculation(Triplets)
29   Minimum-torques-selection
30    $T_j^{j+1} \leftarrow \text{next\_partition\_of\_trajectory}$ 
31   end-while
32 end

```

Let \vec{P} be a position vector of the contact point in the object's surface. Let \vec{N} be a normal vector on the contact point in the object's surface. Let d be an euclidean distance between the original position vector P_{init} and the new position vector P_{new} obtained after the transformation T_{init}^{new} . Let de be the corresponding distance along the finger-tip surface from the original contact-point φ_o and the new candidate φ_i after one interaction search. nm are the number or candidate points around the original point of contact on the object (represented by the normal vector \vec{N}_o and the position vector \vec{P}_o , Figure 3) and finger surface (represented by φ_o , Figure 6).

After applying this algorithm, we have now, a sequence of possible manipulation modes for each finger. First we classify it in decreasing order respecting the joint variation Δq . Second, we choose a set of three possibilities (one for each finger) and calculate the respected finger forces for the stability.

4 Seeking solutions with Quantum Mechanics

Our proposal is to use a classical computer for the computation of the first part of Muñoz' algorithm (i.e. computation of potential solutions), and a Quantum Computation (QC) scheme for performing the second part (i.e. looking for a set of actual solutions). The rationale behind the use of a quantum search algorithm is the following: in Classical Computer Science, searching for an element in an unsorted list of N elements takes $O(N)$ operations. In the quantum case, searching for an element in an unsorted list using Grover's algorithm takes $O(\sqrt{N})$ operations [Nielsen and Chuang, 2000].

Let us introduce two fundamental ideas in QC: the notions of a qubit and the evolution of a quantum system.

Qubit. The basic component of QC is a qubit, that is, a physical entity (such as an electron) that can be mathematically represented as a vector in a $2D$ Hilbert space \mathcal{H}^2 . The

¹Considering its application in a well-defined method is complex. We do not actually compute this in simulation

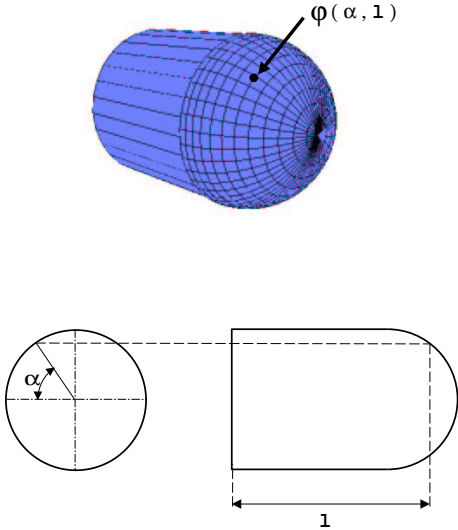


Figure 3: . Contact description

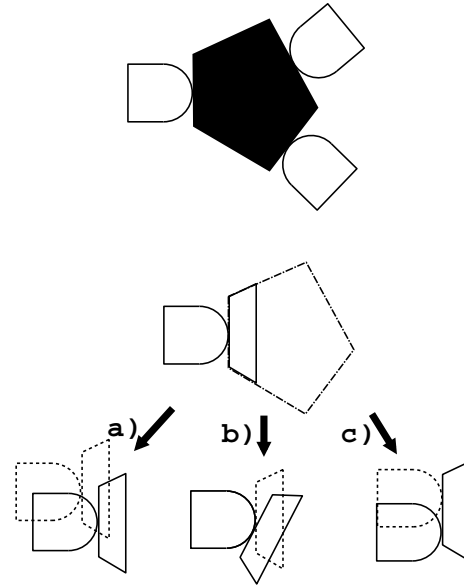


Figure 4: . Contact evolution

general form of a qubit is $|\psi\rangle = \alpha|x\rangle + \beta|y\rangle$ where $\alpha, \beta \in \mathbb{C}$ constrained by $|\alpha|^2 + |\beta|^2 = 1$, and $\{|x\rangle, |y\rangle\}$ is an arbitrary basis of \mathcal{H}^2 . Thus, $|\psi\rangle$ is a superposition of states $|x\rangle$ and $|y\rangle$, and therefore $|\psi\rangle$ can be prepared in an infinite number of ways by varying the values of α and β . In contrast, classical computers measure bit values using only one basis, $\{0,1\}$ and the only two possible states are those that correspond to the measurement outcomes, 0 or 1.

Evolution. The evolution of a quantum system with no interaction with the environment is described by a unitary operator, i.e. an operator U such that $UU^\dagger = U^\dagger U = I$. Thus, the evolution of a qubit can be written as $|\psi\rangle_{t+1} = U|\psi\rangle_t$. Unitary operators are reversible, thus the performance of any operation in a quantum computer (except for measurement) must be reversible. Any irreversible computation can be made reversible by modifying corresponding circuits and adding a (polynomial) number of bits as working space.

Introductions for computer scientists to the fundamentals of QC can be found in [Venegas and Bose, 2003], [Venegas and Ball, 2004] and [Nielsen and Chuang, 2000].

In the following section we provide a scheme for the use of Grover's algorithm in the Dexterous Manipulation problem.

5 Quantum Search Algorithm

Let us suppose that the first part of Muñoz' algorithm has been already performed, i.e. we have a set of hand configurations $\{\vec{C}_1, \vec{C}_2, \dots, \vec{C}_m\} \subset \mathbb{R}^m$. Also, let each entry c_i^j of \vec{C}_i be represented with l bits. We then have a new set of *binary vectors* $\vec{C}_i^b \in \{0, 1\}^{lm}$.

Now, we use Grover's search algorithm ([Nielsen and Chuang, 2000] for a detailed explanation). The algorithm is made out of three parts: a Hadamard gate H , a set of Grover operators G , and a measurement system. Each part is a unitary operator except for the measurement system. The algorithm takes as input the quantum state $|0\rangle$, which is the

tensor product of n qubits initialized in $|0\rangle$ (see Fig. 1). After gate H , input becomes an equal superposition of qubits $|\psi\rangle = \sum_{x=0}^{n-1} |x\rangle$. Note that by setting $n = lm$ we can create a 1-1 function between binary vectors representing hand manipulation and components of quantum state $|\psi\rangle$.

Grover operator is composed of four parts: an oracle, two Hadamard operators and a phase shifter. Oracle design in Grover's algorithm depends on the specific application one has in mind. The main purpose of the oracle is to *recognize* solutions for the search problem and, additionally, if a solution $|\phi\rangle$ is recognized then the oracle computes the function $|\phi\rangle \rightarrow (-1)|\phi\rangle$. Note that *recognizing* a solution is different from *knowing* it. An example of this difference is the factorization of a large integer number A . If one is given a set of primes, checking out (i.e. recognizing) whether those primes factorize A is entirely different (and much easier) from knowing the factorization of A beforehand.

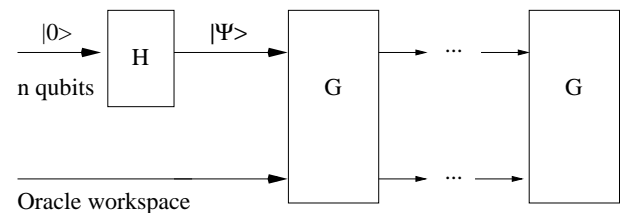


Figure 5: . Schematics for Grover's algorithm.

In our case, we take as oracle input the quantum state $|0\rangle$ and qubit $|w\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$ as oracle workspace. The oracle has two goals: 1) The computation of a cost function F for each $|x\rangle$ (remember that each hand configuration \vec{C}_i^b has been linked with a superposition component $|x\rangle$, for $x \in \{0, \dots, n-1\}$), and 2) Computing the function $|x\rangle \rightarrow (-1)|x\rangle$ for those hand configurations for which the

outcome of the cost function is lower than a given threshold. We are now working on the design of an oracle for the computation of the Hamming distance as cost function.

6 Future Work

We intend to take as much advantage of quantum algorithms as possible by simulating them in classical systems. Thus, we are working now on producing a set of solid software tools to simulate the behaviour of our solutions for the Dexterous Manipulation Problem. Furthermore, we intend to explore the behavior of this hybrid system by implementing it in reversible hardware systems.

Acknowledgments

S.E. Venegas-Andraca gratefully acknowledges grant 148528-CONACyT.

References

- [Grover, 1996] L.K. Grover. In *Proc 28th Annual ACM Symposium on the Theory of Computation*, pp. 212–219, ACM Press, NY, 1996.
- [Muñoz et al, 1995] L.A. Muñoz, Ch. Bard and J. Najera. *Dexterous Manipulation: A Geometrical Reasoning Point of View*. IEEE Int. Conference of Robotics and Automation, pp. 458–463, Nagoya, Japan, May, 1995.
- [Nielsen and Chuang, 2000] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 2000.
- [Venegas and Bose, 2003] S.E. Venegas-Andraca and S. Bose. *Quantum Computation and Image Processing: New Trends in Artificial Intelligence*. IJCAI 2003 pp. 1563-1564, Acapulco, México, 2003.
- [Venegas and Ball, 2004] S.E. Venegas-Andraca and J.L. Ball. *Storing Images in Entangled Systems*. Submitted to IEEE Transactions on Image Processing. Available at LANL preprint ArXiv: <http://arxiv.org/quant-ph/0402085>.