Combined Task and Motion Planing for Multi Robotic Arms in Pick/Place Problems

PhD Proposal

Nir Levi

A thesis presented for the degree of
Doctor of Philosophy

Mechanical Engineering Department
Ben Gurion University of the Negev
06/03/2016

Contents

List of Figures

1 Introduction

At the basis of this study we aim to express the potential of using number of robotic arms in parallel. Despite the usage of robotic arms in large applications at the industry (especially in the automotive industry) in most cases they are operating as separated units. As shown in figure 1 a car assembly line is built by a large number of robotic arms where some arms are sharing their workspaces. To avoid collision between a couple of arms a coordination method is applied, this method is based on timing principles such that the motions are priorly calculated. This means that the assembly line runs on open loop and any small changes of the sequence can cause a crush. On the contrary, having an autonomous method for coordinating motions that is based on the current state of the system may affect 3 aspects that directly connected to manufacturing costs:

1. The total time spent on manufacture a single vehicle may reduce.
2. The assembly line size may decrease.
3. A single assembly line can handle many models of vehicles.

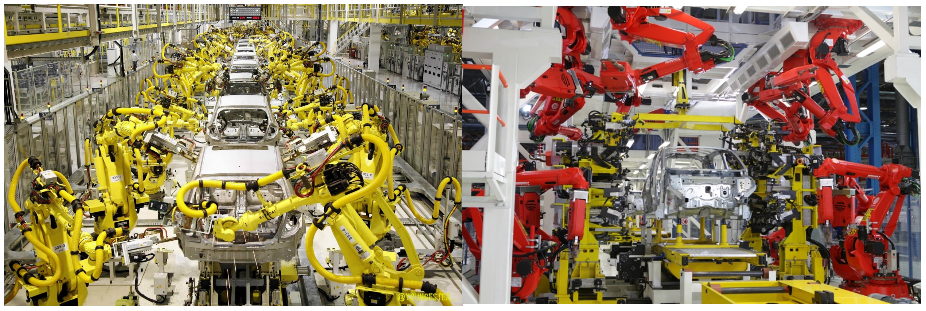


Figure 1: Assembly lines in the automotive industry

2 Literature Review

This chapter presents a literature review on integrated task an motion planning for multi robot manipulators. Although initial studies have begun since the late 80th, this issue is still a matter of research in academia due to its enormous potential. Moreover as computers getting better implementing a multi-robot planner may be a valuable opportunity for the industry. Our intention is: 1) to get the full picture of these research field 2) to know the current state of the art, and 3) formalizing a standard approach for such a problem.

2.1 Motion Planning

Todo: Motion planing for robotic arms, potential field, planning spaces, flocks

Todo: Collision Checking

We will start with a general description of the problem of planning for multi arms/agents. Lavalle [1] has showed that motion planning problem formulation is basically identical for single and multi robots, thus theoretically both have the same sampling based nor combinatorial search algorithms. Therefore scaling in the number of DoF is the only parameter affects the search process. For example, planning for 6 manipulators having 6 DoF each results in calculating motion plan for 36 DoF. Examine the most popular motion planners such as RRT [2], PRM [3] or their more recent variants, we found that it is computationally high to solve (exponential in time). There are two main approaches for calculating motion planning in systems consist of high number of DoF.

Centralized Motion Planning

The first (and tentative) is the Centralized Motion Planning in which the total DoF are taking into account when searching for solution. Although computational power is needed (i.e takes time), its pros comes with complete and optimal solution if any.

Decoupled Motion Planning

Alongside, the second approach Decoupled Motion Planning is more relevant in term of time consuming but lacks the completeness and optimality traits. As listed in [1, 4] decoupled planning is categorized into 3 main methods:

1. *Prioritized Planning* (PrP), In this approach we sort all the robots by priority and planning an individual path for each robot. Each path is calculated based on the hierarchy obtained by the sorting procedure, such that higher rank is being calculating first. The collision free element acquired by treating the higher ranking robots as a moving obstacles.

2. *Fixed Path Coordination* (FPC), This method is divided into 2 parts. At the first stage a path is planned for each individual robot as if its the only robot in the world so that in the second stage all of the paths a getting synchronize by means of timing in order to avoid collisions.

3. *Fixed Roadmap Coordination* (FRC), This strategy extends the FPC by assuming that each robot is guided by a roadmap. This yields a wider set

of routes to achieve a single robot goal (instead of one in the FPC). Thus timing of the overall coordination may be more accessible.

Works discussing motion planning for multiple robots often adopt one of the two approaches or else suggests a hybrid solution. A comparison between those approaches is given by Sanchez and Latombe [5] using the PRM planning scheme. In this work the authors examined a 36 DoF motion planning problem with each of the approaches described above and with the same path finding technique (SBL [6]). Experimental results have shown that the decoupled planning has failed in 30 to 75% of the trials and among the successful one the time differences had no significant change.

2.2 Task and Motion Planning

High and low level planning attributes are "must to have" when designing an autonomous behaviour in robotics. Each attribute standalone has no capability of getting insights from the big picture of the world: high level plan has to be based on physical properties where low level plan has to be guided by some rule. As of writing this paper the current *state of the art* has no standard formulation for such a binding. For this reason, we aim to give our focus and effort in the area of combining task and motion planning.

Task and Motion Planning for Multi Manipulators

Motion planning for multi-agent robots is an extensive research field consist of many variants. In this work we target to combine Task and Motion Planning for Multi Manipulators (TMPMM) mission. Maybe the main difficulty in such problems is planning for continuous and abstract configuration (states), this because motion planner are not comply with abstract parameters so as task planner don't comply with continuous parameters. TMPMM assignment are separated into 2: the ordinary pick and place task and the cooperative task.

An interesting work made by Koga and Latombe [7] describes a task of moving an object in a world built with obstacles using 3 arms having 6 DoF each. The solution suggested is based on calculating a path for the object while simultaneously check whether it can be grasped by one of the arms. If it reaches a configuration where the grasp is no more valid it search to grasp the object using a different arm and continues with the object's path. This way it's ensures the object continuous motion through a calculated cartesian path while being grasped.

3 Methodology

This chapter is about defining the research problems and the ways for solving it. Since the research target (as presented in chapter 1) is general, we have to define specifically what are the problems and how to find a solution. Throughout the chapter we will discuss over *a*) the definition that is used as a basis to define a research problems *b*) the system requirements for a solution *c*) the methods for analysing a solution and *d*) the simulation/experiments that will evaluate a solution performance.

3.1 Research Problem Description

In order to define more clearly what problems are we going to solve, first we will specify the research basic definition. We will use the fact that the majority of industrial robotic arms are statically based and so a group of arms have fixed distances between their bases. This fact leads us to the next definition:

Definition 1. A configuration set of robotic arms is given such that:

a) The arms links length, joint orientation and joint limitation is known.

b) The arms base position is known.

To complete a problem description, we will have to specify what objects are we going to use and the action which the arms will apply. The overall package of arms, objects and actions are the building stones of the next 4 problems:

Problem 1: Pick and Place of Single object by a Set of Robotic Arms

Here, the problem description is given by a set of robotic arms together with a single object such that the task is to move the object from its initial position to a predefined goal. Knowing that only arms are able to manipulate an object we can define the next assumptions:

- The initial configuration of the arms set is known.
- The initial and final position of the object in known and located at a workspace belongs to (at list) one of the arms.
- Only a single arm can grasp the object at a time.
- There are two types of obstacles: the arms themselves and the object.

In practice the difficulty arise when a single arm is not capable of completing the task alone and the solution gets a form of series of arms preforming pick and place operation (i.e transferring the object between themselves). This problem expose a number of open question:

1. What is the sequence of arms which will complete such a task.
2. What are the positions along the object path where a transfer happens.
3. What is the collision free trajectory of each arm throughout the sequence.

Problem 2: Pick and Place of Multiple objects Simultaneously

This problem expands problem 1 above such that the assignment is to move multiple objects from one place to a predefined goal using robotic arms only. This problem adds complexity to the overall system and expose the next assumptions:

- The initial configuration of the arms set is known.
- The initial and final position of all of the objects in known and located at a workspace belongs to (at list) one of the arms.
- The final configuration of all of the objects is defined without collisions.
- A single arm cannot manipulate more than one object simultaneously.
- Only a single arm can grasp an individual object at a time.
- There are two types of obstacles: the arms themselves and the objects.

At first look, this problem can be solved using *prioritize planning* principle; a solution for problem 1 is obtained for each object individually, next an execution scheduling process is maintained in order to avoid conflicts. Although it is an applicable solution it does not exploit the full capabilities of such a system which could handle a faster solution. Thus, the same open question as of problem 1 remained.

1. What is the sequence of arms that will manipulate each object.
2. What are the positions along the each object path where a transfer happens.
3. What is the collision free trajectory of each arm throughout the sequences.

Problem 3: Pick and Place of Single/Multiple objects in Environments with Obstacles

This problem is defined the same way as problem 1 and 2 (i.e as a single or multi object manipulation problem), except that it defined with a new type of obstacles (static nor dynamic). Thus, the last assumption in each problem list is changing as follows:

- The obstacles geometry, type and position are known.
- There are three types of obstacles: The arms themselves, the objects and the environmental obstacles.

The problem of cluttered environment adds complexity to the system such that it limits the motions of arms nor the positions of the transfer points. Hence, it is mainly affects the motion planning part of the total planning scheme.

Problem 4: Pick and Place of Single/Multiple objects with Non-Homogeneous Team of Robotic Arms

In the case of non-homogeneous teams of robotic arms we depict a the same problem of moving an object by a series of arms, only now there are different kind of arm. A warehouse management robot team would be a good application where heavy lifting robotic arms are needed for manipulating plates and a light lifting robotic arms are needed for loading/dispersing boxes on top of the plate. In contrast to the scenarios of problem 1-3, here we let a single arm to manipulate number of objects at a time (as long as the objects are organized in a way that enables to grasp).

3.2 Algorithm Development

The problems described in section 3.1 have both decision and motion planning attributes. Each attribute standalone has a formal approach for solving it but lacks the other skills. Still, in order to solve these problems a 2 phases solution is required:

(phase 1) A high level task plan with a general purpose aimed to find a sequence of arms that will transfer an object from initial to goal positions. The idea behind this phase is to find a series of "arm indexes" where the object is being passed from one arm to the next such that the last arm will place the object at it's target.

(phase 2) A low level motion plan with a general purpose aimed to actually calculate a collision free trajectory for each step of the sequence. This is a more physical phase where the joint/work spaces of all of the arms are taken into account when calculating such a trajectory. The collision free refers for an arm that is transferring an object towards the next arm in the series (or toward the object's goal).

The two phases are coupled due to the fact that any logical sequence of arms is based on geometric properties, for example any 2 following arms should share a gripping position of an object (in global coordinates) in order to transfer it between themselves. Having this fact in mind we understand that our algorithm first objective is to combine the two phases above by searching for applicable sequence it terms of kinematic feasibility. Having a feasible sequence applies for being able to grasp an object at each step by a different arm, this is literally interpreted as a state where a configuration of a single arm was set. This notion leads us to our second algorithm objective which is to calculate a transition function between those states. Here, a transition function is a motion plan as detailed in phase 2 above and this is a key feature for success in such algorithm.

As for our third objective is to be able to use this algorithm for a maximal number of arms nor objects. This is a high priority goal which makes a major contribution for such a scenarios. The trade off for large scale planning is of-course linked to computation time of the algorithm. From this perspective, We expect the weakness of the algorithm at the motion planning phase. As shown in section 2.1 applying a *centralized* motion planning method on a system with high number of DoF will doomed to time-out before having a solution.

3.3 Algorithm Analysis

In order to validate the capabilities of any algorithm we should assess two major specifications:

Completeness This first parameter states for knowing whether a solution exist and as long as one exist if the algorithm can find it.

Complexity This parameter has a direct link to solutions computation time. In most cases this parameter estimates the maximal operation to do in order to find a solution (if any). In practice complexity tend to over estimate the computation time of a solution which actually much more faster. Thus, to get a realistic point of view about how good is our algorithm we should compare its performance against other techniques.

3.4 Simulation and Experiments

Any design of planning algorithm that implements phase 1 and 2 of section 3.2) will be tested under simulation and experiments, this in order to present its capabilities of displacing an object using multiple arms. The simulation code will be written under MATLAB environment which later on will be re-written with c++ in order to handle the experiments under GAZEBO/ROS framework. Although GAZEBO is a 3D simulation it holds behind a dynamics engine that calculates the physics of solid parts in terms of equation of motion and interacting forces (between two parts). This means GAZEBO can be used as a experiments environment where we can test scenarios consist of as many robot arms as we need.

The objective of these simulation is to preform a parametric inquiry which aimed to check the influence of a single parameter on the overall performance. The parameters that interesting us are as follows:

1. The number of robotic arms in a setup.
2. The total percentage of the shared workspace of an arm.
3. The number of joints/links having a single arm.

Throughout the research our target is to apply this inquiry over the problems listed in 3.1, thus for each of these problems we will have to build a different setup. Table 1 shows four different setups that will be tested on each problem. As detailed, column 2 represents the number of arms in a setup, column 3 represents the percentage of workspace shared between any couple of arms and column 4 represents the number of DoF in each arm. In fact, any change of a setup parameters will change the output of a planning algorithm such that we can estimate what are the factors that influencing on:

Setup Index	Number of Arms in a Setup	Percentage of Workspace Sharing	Number of DoF in a single arm
#1	3	20	TBD
#2	3	80	TBD
#3	10	20	TBD
#4	10	80	TBD

Table 1: 4 different setups to handle a parametric inquiry over the research problems

1. The total task operation time.
2. The overall configuration size.
3. The computation time.
4. How many robotic arms can be used.
5. What type of arms can be used.
6. How "dens" the arms are located.

Also, these setups are interpreted as both two or three dimensional problem, although in general a 2D problem may be more easy to solve it is not the often case here (See Appendix............). Still, the procedure for testing a problem in our research will be first simulation and experiments over a set of planner robotic arms and next simulation and experiments on a set of spatial robotic arms (The number of arms and the percentage of shared workspace remains the same).

ToDo: explain position controlled

4 Preliminary Results

Over this chapter we will describe our proposed solution for the multi arms manipulation planning problem with reference to the first problem in 3.1. A two dimensional illustration of such a problem is given in figure 2. As introduced, this scenario is an example of a setup built by 5 statically based robotic arms together with an object displacement assignment. As detailed in section 3.4 a similar scenarios varied by the number of arms, number of DoF and the percentage of shared workspace will be examined. Our algorithm objective is to give an automatic generation of a sequence of arms (and motions) for any given scenario/setup.

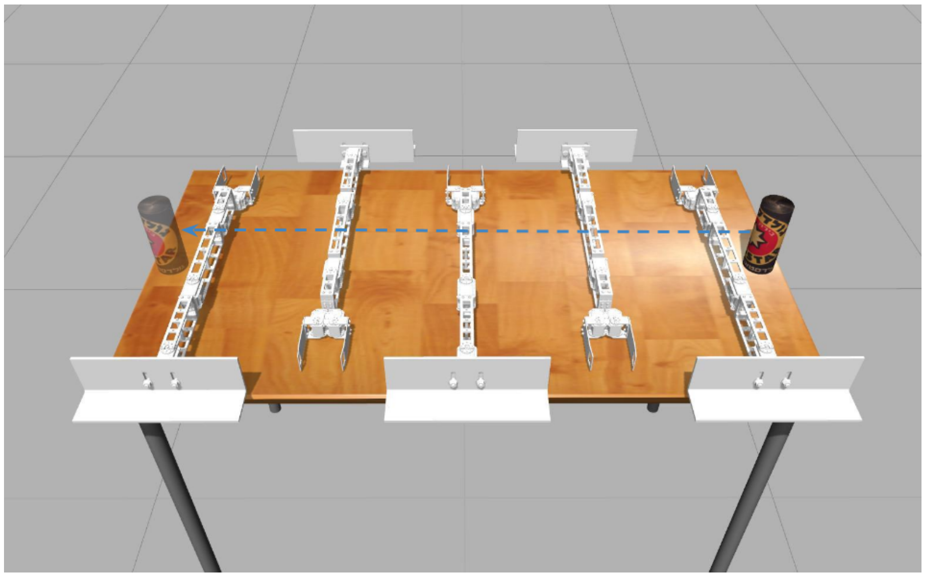


Figure 2: A setup illustration of 5 robotic arms together with a single object

4.1 Problem Formulation

In this work our intention is to engage a single object pick and place problem using multi manipulators. We will handle the next symbols description for our problem statement. A manipulator will denoted as A^m (the m'th parameter means the manipulator index) and an object as B. We assume to have an initial configuration given by M manipulators $\mathcal{A} = \{A^1, A^2, \ldots, A^m, \ldots, A^M\}$ together with a single movable object. Furthermore we assume that each A^m is given by a) configuration space C^m which represents the space of the arm joints value $c^m = \begin{bmatrix} q^{m,1} & q^{m,2} & \ldots & q^{m,n} & \ldots & q^{m,N} \end{bmatrix}^T$ where q expresses a single DoF, and b) a workspace W^m which represents the space of the arms gripper positions $w^m = \begin{bmatrix} x^m & y^m & \theta^m \end{bmatrix}^T$ in global coordinates. A mapping function of

W^m is given by the Forward Kinematics (FK) rule where $FK : C^m \to W^m$. Thus, the full state of the system is given by the configuration space of all of the arms $\mathcal{C}^A = \begin{bmatrix} C^1 & C^2 & \dots & C^m & \dots & C^M \end{bmatrix}^T$. Next, for keeping consistency with [7,8] we will presume the terminology of *transit*, *transfer* and *manipulation* paths as follows:

transit-path is a path that describes a motion of an empty arm (i.e not grasping any object). We will use this term to represent a path of an arm which avoids collision with other arms or actually moving toward an object in order to grasp it.

transfer-path is a path that describes a motion of an arm that is grasping an object within its gripper. We will use this term to represent a path of an arm that manipulates an object.

manipulation-path is a sequence of alternating *transmit* and *transfer* paths. We will use this term to represent a sequence of arms the moves an object from initial to goal positions.

Next, consider the following problem input:

- A set of m robotic arms.
- Initial and goal positions of an object B^{init}, B^{goal}

Our objective is to find a *manipulation-path* along which brings the object to its goal configuration. By definition each *manipulation path* is constructed by a sequence of arms $\mathcal{S} = \langle s^1, s^2, \dots, s^k, \dots, s^K \rangle$ preforming a *transfer paths*, where s^k represent an arm index and K is the number of arms in $\mathcal{S}$. As each arm of the sequence is identified with a $transfer$ action it holds a configuration space path noted as Q^k (where $Q^k \in C^{\mathcal{S}(k)}$). Now, we can define a sequence of paths $\mathcal{Q}$ for each step of $\mathcal{S}$ such that $\mathcal{Q} = \langle Q^1, Q^2 \dots, Q^k, \dots, Q^K \rangle$. Here, each Q^k represent a path for the corresponding arm s^k.

Practically, any execution of $\mathcal{S}$ "as is" may not be a accomplished due to the configuration of the other arms that possibly obstructing the path. As used in [7] a straight forward solution is obtained by moving the obstructing arms to a predefined non-obstructive location. This solution is less likely to succeed in 2D problem, especially on setups with high percentage value of the arms shared workspace. To overcome this issue any Q^k path should be coordinated with a *transmit* paths of the arms that are not transferring the object. Thus, in order to move a single arm a full state path is constructed as a subset of C^A. This eventually re-defines Q^k with a second index such that $Q^k = \begin{bmatrix} Q^{k,1} & \dots & Q^{k,m} & \dots & Q^{k,M} \end{bmatrix}^T$ where $Q^{k,m}$ represent the A^m path over the k step. In summary, a solution is formulated with the following parameters:

1. A sequence of K arms indexes $\mathcal{S}$ that represent which is the $transfer$ arm at each step.
2. A sequence of K paths $\mathcal{Q}$ that represent the motion of all the arms at each step of $\mathcal{S}$ (i.e the *manipulation* path).

4.2 Multi Arm Sequence Planning

As detailed, a *manipulation* path is built by a sequence of pick and place operations which in nature defines a sequence of subtasks. Thus, for convenience we will use the term subtask to denote the k'th procedure in $\mathcal{Q}$. Being able to solve each subtask separately simplifies the problem to a single arm pick and place solution. Still, this advantage makes no guaranties for solving the overall task unless a binding condition will be made between two following subtasks. Such a criteria is built by the *transfer point* terminology.

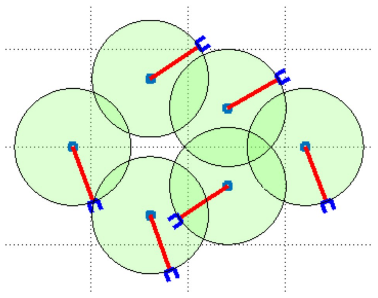
We let $\mathcal{T}^B$ be the object absolute path from its initial to goal position. By definition, each subtask moves the object along a portion of $\mathcal{T}^B$. Hence, we can reason that $\mathcal{T}^B$ is built up by a sequence of *work-space* paths each handled on the k'th subtask by a different arm, thus $\mathcal{T}^B = \langle T^{B,1}, ..., T^{B,k}, ..., T^{B,K}\rangle$. We will denote a *transfer point* as a position where any $T^{B,K}$ starts or ends. Tentatively any *transfer point* is located where an object is being picked or placed by a transferring arm. Any sequence S consist of K arms will have to pass through $K+1$ *Transfer Points Series* (TPS) where the first and the last points represents the initial and final state of the object respectively.

As stated, a solution of the whole problem is made possible by using a solution for a subtask. The first key feature for success in such a case is given by the initial and final state of each subtask. We will employ the continuous pick and place property of a sequence to determine that a final configuration of any k subtask will be the initial state of the $k+1$ subtask. This kind of property is perfectly match to a TPS definition where an initial and final goal of an object are determined for each subtask. Still, we cant generate a sequence of subtasks by having a TPS only. The second key feature that will fully define such a sequence is given by the definition of $\mathcal{S}$ that determine which is the transferring arm in any subtask. Thus, both TPS and $\mathcal{S}$ are fully defines a sequence of subtasks in terms of the transferring arm and its initial and final configurations.

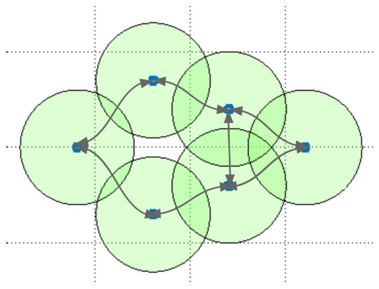
Our planning approach will be guided by the principle of integrating $\mathcal{S}$ and TPS and their interpretation for subtasks. Hence, the algorithm will be structured into two phases where phase 1 will be aimed to find a logical combination of $\mathcal{S}$ and TPS, and phase 2 will be aimed to generate the corresponding subtasks and motions. It is noteworthy that this approach has the attributes as detailed in 3.2.

Planning Process

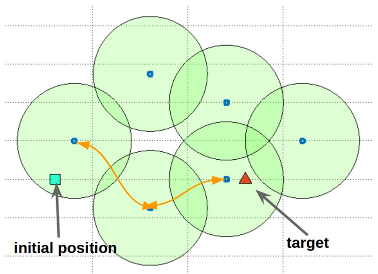
Recalling the work of Koga and Latombe [7], it is shown that the object's global trajectory is the governing law when it comes to determine which is the grasping arm at any time. On the contrary we wish to give another perspective for the problem. Our guiding rule comes from shared workspaces of the arms. We are using the fact that any *transfer point* is located only in one of the shared workspaces and consequently $\mathcal{T}^B$ must pass through them.



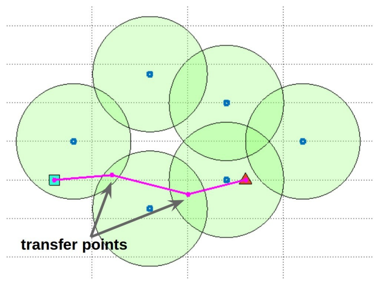
(a) general configuration



(b) phase 1: generated connectivity graph



(c) phase 2: Graph shortest path based on task properties



(d) phase 3: calculating transfer points

Figure 3: Generating a sequence of arms and transfer points

Search for Arm Sequence and Transfer Points

One of the key component of our work is find any pair of arms sharing together a part of their workspace such that $W^i \cap W^j \neq \emptyset$. This fact gives an insight for any transfer that could appear. Representing this data is made by using a graph where nodes represents an arm workspace and arcs represents a shared workspace between two arms. It means that a setup consist of M arms is interpreted by a graph with M nodes. Adding the graph arcs is applied by going throw any possible combination of two arms and check whether together they share a part of their workspace. A generation of this graph will tentatively be called the *connectivity graph* (example illustration is given in figure 3b).

By attributing the initial and goal position of the object (B^{init} and B^{goal}) to one of the arms work space we can reason about the starting and final node in the *connectivity graph*. At this point, any graph search would give a solution with a form of a sequence of arms index $\mathcal{S} = \langle s^1, ..., s^k, ..., s^K \rangle$ connected by their shared workspaces (as shown in figure 3c) such that:

$$B^{init} \in W^{\mathcal{S}(1)} \tag{1}$$

$$B^{goal} \in W^{\mathcal{S}(K)} \tag{2}$$

$$W^{\mathcal{S}(k)} \cap W^{\mathcal{S}(k+1)} \neq \emptyset \tag{3}$$

To complete the algorithm first phase we still have to determine the positions of the *transfer point series*. As shown in figure 3d we can detect that any *transfer point* is located in the shared work space of any two successive arms in $\mathcal{S}$. Still, we have to determine the exact position in the shared work space. One option is to use a linear programming method to minimize the total length accepted by connecting any pair of successive *transfer point* by a straight line. In our implementation we choose a more straight forward method and selected the center of area of the shared workspace.

MASP Algorithm

Algorithm 1 shows the main procedure for generating a solution. Here, lines 1-5 represents the first phase of the algorithm for calculating the transfer arms and transfer points where line 6 represent the motion planing phase.

Algorithm 1 Multi Arm Sequence Planning

```
Input: $B^{init}, B^{goal}, \mathcal{A}$
Output: $\mathcal{Q}$
1: $cg \leftarrow$ Generate $ConnectivityGraph\,(\mathcal{A})$
2: $W^{first} \leftarrow$ find($W^{first}$ such that $B^{init} \in W^{first}$)
3: $W^{last} \leftarrow$ find($W^{last}$ such that $B^{goal} \in W^{last}$)
4: $\mathcal{S} \leftarrow$ BFS($cg, W^{first}, W^{last}$) // generate ws sequence
5: $TPS \leftarrow$ Calc $TransferPointsSeries\left(\mathcal{S}, B^{init}, B^{goal}\right)$
6: Subtask_Sequence $\leftarrow$ GenerateSubtasksSequence(TPS,S)
```

4.3 Multi Arm Motion Planning

This section, as a part of the whole process will explain our result in calculating a collision free *manipulation* path for each of the arms. As declared in section 4.2 using the MASP algorithm our problem is formalized as a sequence of single arms subtasks. Still, a single arm motion planner is not applicable on a multi arm scenario and multi arm motion planner is highly complex to solve with large number of arms. This conclusion focused our efforts to find a solution for the multi arm path finding problem planned with a reasonable amount of time.

In our problem the overall goal is to find a multi arm trajectory for each of the subtasks extracted by the MASP algorithm. We will start by formulate a general subtask given by the next input definition:

- A set of arms given by its initial configuration.
- An index of the transferring arm.
- Initial and final positions of the movable object.

Next, we will use the assumptions as detailed in 3.1 problem 1 to denote that

- There are two types of obstacles: the arms themselves and the objects.

This all means that we will expect for collision only between a pair of arms nor a pair of arm and object (exclude the transfer arm). As previously defined any subtask is obligated to a single arm pick and place operation no matter the number of arms in the set. An interesting cases emerge when only few arms obstructs the *transfer* arm way, this notion simplifies the solution by planning with less DoF which drastically reduce computation time.

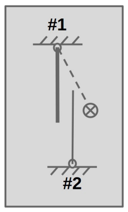
An illustration of the problem is given in figure 4 where a the same task is given by three different setups distinct by the number and position of the arms. Examine figure 4a we find that the target is to move arm #1 to the object (final configuration is given by the dashed line) where arm #2 obstruct its direct path. Furthermore, the obstruction happens as arm #2 is stuck between arm #1 on the left and the object on the right which makes it impossible to clear arm #1 path. To overcome the obstruction arm #1 should move backward (i.e away from the target) in order to clear a space for a new position of arm #2 from which arm #1 has a clear path towards the object. Next, examine figure 4b we can reason that the solution applied on figure 4a is not valid unless arm #4 will clear the space where arm #1 is moving backward. In addition, arm #3 needs to clear a space for arm #4, arm #6 needs to clear space for arm #3 and so on until arm #9 clears a space for arm #7. These two examples demonstration visualized the drawbacks of using *decoupled* methods. For example by using *prioritize planning* we might associate arm #1 with the highest priority in the set which will obviously end up with no solution. Next, figure 4c expose the the drawbacks of using *centralized* methods that will plan a path for 9 arms instead for only two (which seriously increase computation time).

These three examples spans the drawback of the two planning approaches with application on multi robot arms. Over the next section we will show our Multi Arm Motion Planning (MAMP) scheme which attributed by yhe next two requirements:

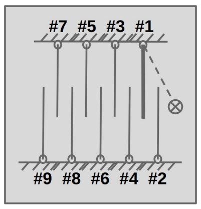
- Changing priority when needed.
- Plan a path for moving arms only.

Planning logic

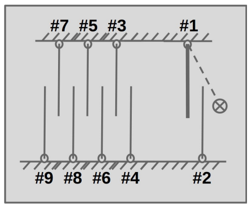
Here, we will analyse the logic stands behind our planning scheme. As we will see it is build up by the same principles of *decoupled* planning approach with the essence of *prioritized* method. By resolving each subtask we find that it associate with only a single arm (e.g *transfer arm*). In fact this is the key feature that guides our search, as we can assume that only a single arm gets the highest priority while all the other are non-prioritized. As shown in the examples of figure 4a associate only the transfer arm with the highest priority wont find a solution. Still, changing the priority to the obstruct arm means nothing without having a feasible target. Hence, in order to depict a procedure of changing priority it has to come with adding a target.



(a) A setup consist of two arms with single DoF each.



(b) A setup consist of 9 arms with single DoF each.



(c) A setup consist of 9 arms with single DoF each.

Figure 4: Two setups for complexity demonstration. On both cases arm #1 is the *transfer* arm and the object is marked by $\otimes$.

Clearly, any new target associated to a non-transfer arms will be subjected to "clear the path of the transfer arm". Such a scheme expands hierarchy of targets belongs each time to a different prioritized arm. For example, the hierarchy of targets in figure 4a will be then:

1. priority: arm #2, target: clear arm #1 path.
2. priority: arm #1, target: go to object.

As we will see, expanding the hierarchy of targets is useless unless each target is represented by position. This means that besides of changing priority we have to reason about the positions that is actually applies for the target meanings.

As prioritized hierarchy principle stands for acquiring valid solutions we still have to generate the arms motion. recall that one of the planner requirements is to generate motion for the moving arms only, but how can we know which is the moving arms before applying the motions. Again, to overcome, we will use the same key feature that gives priority for a single arm only to classify each arm with two rules: a *master* and a *slave*. Contrary to flocks behaviour, here a *slave* arm implies for avoiding any collision with the *master* arm. Naturally, such a description may resolved by artificial potential field methods which will be discussed in the MAMP algorithm section.

Both *prioritized hierarchy* and *master/slave* attributes have inspired the logic behind the MAMP algorithm. Over the next sections we will show how together those attributes interact and forms the algorithm design.

Motion Generation

Recall that one of our major goals was being able to find fast solution. With this sense, the trade off for planning fast is to apply a *decoupled* methods. In fact, being able to plan fast was our driven motor for choosing a motion planner (generate a new technique was never our goal). Thus, any motion was generated using a specified artificial potential field together with steepest decent concept, this way we could plan fast with high number of DoF.

As already been said any *slave arm* is subjected to the motion of the *master* arm. By this assumption we define two motion modes: *attractive* and *repulsive*. An *attractive* mode is dedicated to attract the *master* arm to the target and *repulsive* mode is dedicated to remove a *slave arm* from a *master arm* path. Respectively, an *attractive* potential field is calculated as distance from the object to the *master arm* gripper and *repulsive* potential field is calculated as a function of the minimal distance between a pair of *master* and *slave* arms.

MAMP Algorithm

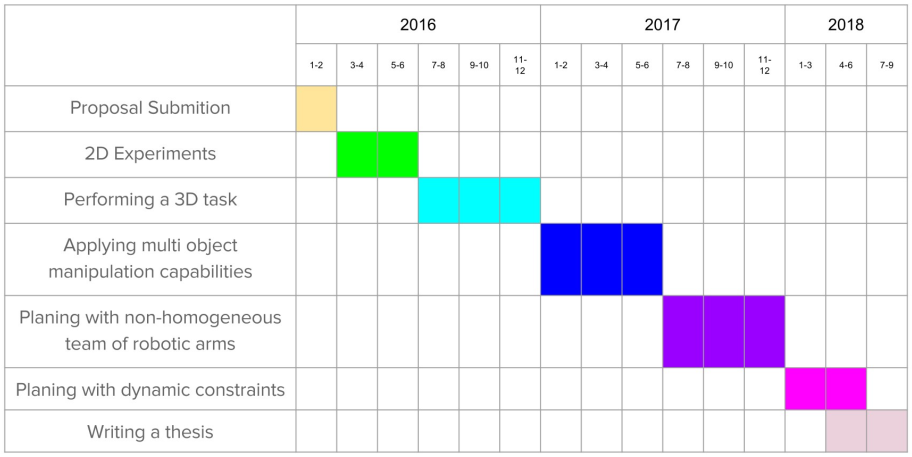
Here we will show how to u

Algorithm 2 Multi Arm Motion Planning

```
Input: subtask, $\mathcal{A}$
Output: $\mathcal{Q}$
 1: target_queue ← init_target_queue(subtask)
 2: while size(target_queue) ≥ 0 do
 3:     $P$ ← Find attractive Path(target_queue(1))
 4:     $O$ ← Verify obstruct Arms($P$)
 5:     if No Arms Obstructs then
 6:         $\mathcal{Q}$.push_back(P)
 7:         target_queue.remove(1)
 8:     else
 9:         $P$ ← reverse($P$)
10:         $O$ ← Verify obstruct Arms($P$)
11:         target_queue.push_front($O$)
12:     end if
13: end while
```

mamp algorithm move algorithm verify algorithm

5 Research Program



	2016						2017						2018		
	1-2	3-4	5-6	7-8	9-10	11-12	1-2	3-4	5-6	7-8	9-10	11-12	1-3	4-6	7-9
Proposal Submition	🟨														
2D Experiments		🟩	🟩												
Performing a 3D task				🟦	🟦	🟦									
Applying multi object manipulation capabilities							🟦	🟦	🟦						
Planing with non-homogeneous team of robotic arms										🟪	🟪	🟪			
Planing with dynamic constraints													🟪	🟪	
Writing a thesis														🟪	🟪

Figure 5: Time table

Immediate goals for first paper

future goals

References

[1] S. M. LaValle, *Planning algorithms.* Cambridge university press, 2006.

[2] ——, "Rapidly-exploring random trees a ew tool for path planning," 1998.

[3] 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*, vol. 12, no. 4, pp. 566–580, 1996.

[4] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulationa survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[5] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 2, 2002, pp. 2112–2119 vol.2.

[6] ——, "On delaying collision checking in prm planning: Application to multi-robot coordination," *The International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[7] Y. Koga and J.-C. Latombe, "On multi-arm manipulation planning," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on.* IEEE, 1994, pp. 945–952.

[8] ——, "Experiments in dual-arm manipulation planning," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, May 1992, pp. 2238–2245 vol.3.