

# Priority coordination of fiber positioners in multi-objects spectrographs

Dominique Tao<sup>a</sup>, Laleh Makarem<sup>b</sup>, Mohamed Bouri<sup>c</sup>, Jean-Paul Kneib<sup>d</sup>, and Denis Gillet<sup>e</sup>

<sup>a,b,e</sup>REACT, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland

<sup>c</sup>LSRO, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland

<sup>d</sup>LASTRO, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland

## ABSTRACT

Projects such as "The Dark Energy Spectroscopic Instrument" (DESI) or "The Multi Object Optical and Near-infrared Spectrograph" (MOONS) are developing spectrographs, composed with more than thousand of optical fibers in a confined hexagonal focal plane, to study the evolution of the universe. Such systems make real time observation possible as each optical fiber is moved simultaneously to their pre-assigned target by a 2-arm positioner within an short interval of time to monitor a specific astronomical object. Moreover, astronomers prioritize the observation of some objects over those that hold less information, creating a hierarchy of importances or priorities. In a non-complete scenario where not all the positioners can reach their targets, being able to ensure the observation of the important ones is a desirable feature.

In previous works, a decentralized navigation function from the family of potential field was used for collision-free coordination. While it is complete for DESI [1, 2], it is different for MOONS [3] as the second arm of their positioners is two time the length of its first one. Covering a larger working space, they are prone to deadlocks, a situation where two or more positioners are blocked by each other and so unable to reach their targets.

In this paper and in the framework of MOONS project, as an extension of the decentralized navigation function, we present our new approach to integrate pre-assigned priority to positioners in order to coordinate them and to solve deadlocks situations. For this purpose, a finite-state machine combined with distance-based heuristics is used to regulate their movements. While their states dictate their behaviors with respect to each others, distance-based heuristics are used to limit their states transition only when interacting with their neighbours and to localize possible deadlock situations. With a local interaction, the advantage of such method lies in its simplicity as it does not burden that much the time complexity, being quasilinear. In addition, since it does not depend on the positioner's geometry, it is also scalable to other positioner's system.

A motion planning simulator with graphic interface, developed in python is used to validate the priority coordination of the system. The trajectories are first pre-computed before being sent in open-loop to the positioners. As a result, the positioners converging to their target improve from 60-70% to 80-95%. Although it was predictable and logic, the trajectories pre-computation time is longer than just using the decentralized navigation function since another layer of algorithm is added on top of it, which is/can be compensated with more performant hardwares and optimizations.

**Keywords:** MOONs, Optical fiber positioners, Priority coordination, Finite-state machine, distance-based heuristics, Deadlock problem

## 1. INTRODUCTION

Recently uncovered that not only the universe is expanding but also that its process is accelerating, massive spectroscopic surveys are used in many international projects such as "The Dark Energy Spectroscopic Instrument" (DESI) or "The Multi Object Optical and Near-infrared Spectrograph" (MOONS) to study its evolution. The used spectrographs are composed with more than thousand of optical fiber bundles, in a confined hexagonal focal plane. Each one are moving at the same time to a pre-assigned target by a two arms robotics positioner, making real time observation possible.

---

Further author information:

dominique.tao@alumni.epfl.ch, mohamed.bouri@epfl.ch, jean-paul.kneib@epfl.ch, denis.gillet@epfl.ch

Anti-collision movement between positioners is therefore essential for a well-functioning system. While the completeness of the system, ie convergence of all positioners to their targets, is ensured with the project DESI using a decentralized navigation function algorithm [1–3], the same approach is not valid for MOONS: its positioners second arm being two time longer as shown in figure 2 - 3, it is covering a larger workspace and hence creating possible collisions and deadlocks, a situation where positioners, blocked by one another, are unable to reach their targets.

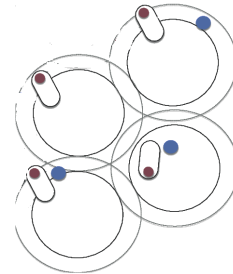
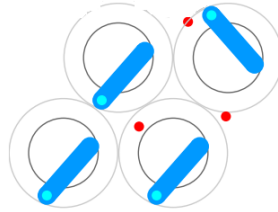


Figure 1. CAD positioner model Figure 2. MOONS positioner in simulation Figure 3. DESI positioner in simulation

Though desirable, having a non complete system is not a serious issue since not all astronomical objects contain the same valuable information to study the universe evolution. Astronomers prefer to prioritize the observation of the important targets even at the price of disregarding the less relevant ones.

With MOONS framework and as an extension of the previous work using a decentralized navigation function for collision avoidance [3], this paper explores an approach, through a finite state machine strategy, to take into account the importance/priority of each positioner’s target into the coordination of their movements. As it gives more chance for higher priority positioners to reach their targets, its other objective is also to increase the positioners convergence with the goal of having a complete system.

The organization of this paper is as followed: In section 2, a representation of the system and the finite state machine is described. Then we explain how deadlocks are localized and solved using an heuristic vector-distance based strategy. Section 3 then presents the result of our approach on various test cases of positioner-configuration. As a conclusion, future research and possible improvements are discussed in section 4.

## 2. PRIORITY BASED COORDINATION

Unlike many motion planning algorithms, the choice of using potential field to move simultaneously thousand of positioners stems from its time complexity and simplicity. It enables fast pre-computation of their anti-collision movements towards their targets. However, due to local minimum problems, using simple potential field to handle more complex systems which can exhibit deadlock situations prove to be complicated: it requires either a more complex development of the algorithm or an additional layer of complexity.

In this work, we design a decision layer using a finite state machine which is built on top of a decentralized navigation function algorithm. It’s role is to manage the interaction of positioners with their neighbour positioners having different orders of priorities.

### 2.1 System representation

Before explaining our approach, we need to establish how the system with priorities is defined. A depiction of the MOONS positioner and its equivalent in simulation are shown in figure 1 - 2 - 4, where only the second arm (corresponding to the curve segment of the CAD model) is represented. The red dot is the target pre-assigned

to the positioner's end-effector which holds the optic fiber represented by the dot at the extremity of the second arm.

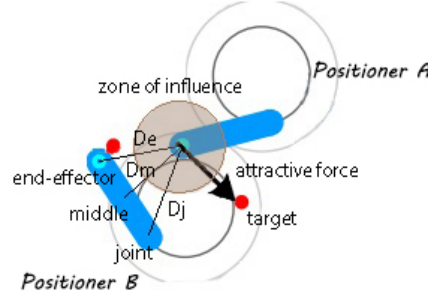


Figure 4. Variables representation of the system

Let  $p = \{1, 2, 3, 4\}$  be the pre-assigned priority of a positioner, with 4 for the most important target and 1 the less important one.

$D = \{D_e, D_m, D_j\}$  is the set of distance between respectively the end-effector, middle and joint part of one positioner to another one's part. In figure 4, these distances are represented from positioner B to only the end-effector part of positioner A.

As dictated by the principle of the potential field, each positioner is drawn to its target by an attractive force  $F_a$  and avoiding its *ieime* neighbouring positioners with a repulsive forces,  $F_{ri}$ . Further details on their implementations in the context of MOONS are given in [3].

With more than thousand positioners in a compact space and with each one having at most 6 neighbours, a "zone of influence",  $Z_{F_r}$ , is used for the repulsive forces. They are active only within this zone to limit and simplify the number of forces felt by one positioner from the others.

The zone  $Z_{F_r}$  can be of any shape, but in this work, it was decided to be a circle around each part (end-effector, middle and joint) in order to have a simple cover on the whole positioner. The source of the repulsive force comes from the part of the neighbour positioner that is inside the zone  $Z_{F_r}$ .

## 2.2 Finite-state machine

The finite state machine has a total of 5 states with their corresponding transitions. Designed to take into account different priority values, it regulates the positioners interactions for a better coordination of the system.

### 2.2.1 First State: ON/OFF

The first state is defined as the "ON/OFF" state. With the creation of a "ON/OFF zone",  $Z_{ON/OFF}$ , associated to a positioner, any neighbour positioners with a lower priority inside will have their attractive forces towards their targets disabled, ie  $F_a = 0$ . We refer to such a positioner as an OFF one in opposition to an ON one with its attractive force still active. Similarly to the influence zone  $Z_{F_r}$ , we use the same circle shape for  $Z_{ON/OFF}$ , centered on all the parts of the positioner in order to entirely cover it.

With only repulsive forces to prevent collision, it allows the ON positioner with the biggest priority to circulate through its OFF neighbours with less resistance, being the only one with an attractive force. This principle using priorities to solve the potential field local minima is illustrated in figure 5.

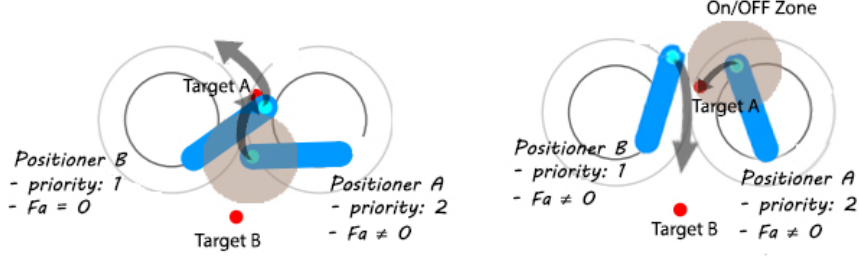


Figure 5. Principle of ON/OFF state (left to right image):

(left image) Positioner B inside  $Z_{ON/OFF}$  of positioner A doesn't have any attractive force and is subject to repulsive force from positioner A, making it going to the opposite direction of its target B; (right image) The same positioner B, once repulsed outside of the zone  $Z_{ON/OFF}$  by positioner A itself, regains its attractive force

A direct consequence of using only this state is the problem of deadlock due to oscillations at the limit of the zone  $Z_{ON/OFF}$ . Considering an ON positioner A and its OFF neighbour B with lower priority inside the zone  $Z_{ON/OFF}$  of A (figure 5 right image), when the positioner B is at the border of the zone, it can be subject to the problem of reactivation and deactivation of its attractive force. If the reactivated attractive force is opposing the movement of the positioner A, it will re-enter the zone  $Z_{ON/OFF}$  of A only to be repulsed again. This switching state process can block the positioner A as the neighbour positioner B will keep oscillating at the same place.

A deadlock can be interpreted as at least two positioners that stop moving towards their targets because of each other oscillating indefinitely at the same place in space. A moving average filter on the positioner's velocity can be used to localize and unblock this situation by adding noise from it. Though it might be tempting to modify the magnitude of the different forces or of the noise, one has to consider the risk of collisions due to positioners sudden increase of velocity.

Furthermore, only evaluating the velocity is necessary but not sufficient to detect a deadlock as explain later in the sub-chapter 2.2.4 since positioners can have a null velocity when arrived or when repulsing another one.

With  $v[t]$  the velocity of one positioner at time  $t$ ,  $y[t]$  is the output of the average moving filter of window  $M$  such as:

$$\begin{cases} y[t] = \frac{1}{M} \sum_{k=1}^M v[t-k] \\ y[t] < thres \Rightarrow noise \end{cases} \quad (1)$$

We define *thres*, which value is defined experimentally, as the threshold representing when the positioner can be considered as "stopped". The result  $y[t]$  is used to introduce noise once it is under this threshold.

### 2.2.2 Second & third state: ID and Priority memorization

Having introduced new variables with the first state, we need to determine how positioners interact with each others depending not only on their pre-assigned priority but also on their ON and OFF states.

As for the reason, problematic situations can also occur with the interaction of more than two positioners. Similarly to the positioner configuration illustrated in figure 6 (left image), deadlock can happen when two ON high priority positioners A and B are going against each other by propagating their opposite movements through an intermediate low priority OFF positioner C that is in-between them.

In order to solve this problem and also for a better coordination of the system, we use the principle of memorization by introducing two new states:

- "MemPr":  
Initialized at 0 for a positioner, it is used to memorize either the pre-assigned priority or the MemPr of its neighbour positioner which changed its states.
- "MemId":  
Initialized at 0 for a positioner, it is used to memorize only the identity of its neighbour positioner which changed its states.

With this information and in a similar situation than in figure 6, one OFF positioner C, whose states has been changed by an ON neighbour positioner A due to its higher priority, is now able to influence its other ON neighbours states if their pre-assigned priorities are smaller than "MemPr" of C. The positioner A which changed the state of positioner C has its pre-assigned priority memorized by C. It enables the positioner A, through C, to influence other positioners whose movements are opposing its one, allowing to have its convergence to its target with less resistance.

We define this process as a "chain reaction": one OFF positioner being able to change the states of its neighbour positioner that is not the one that originally influenced it.

To understand this principle and the assignments of values for "MemPr" and "MemId", the pseudo-code of algorithm 1 gives a concise overview on how all the states seen so far for an considered positioner, called agent positioner, are changed when interacting with one of its neighbour. A visual step-by-step example following the algorithm 1 is illustrated in figure 6.

Furthermore, these rules are only valid if the agent positioner has not yet arrived to its target and is inside the  $Z_{ON/OFF}$  zone of its considered neighbour. A condition related to another positioner state, the fourth one, that will be fully understand, as well as the whole condition in line 1, in the next sub-chapter 2.2.3.

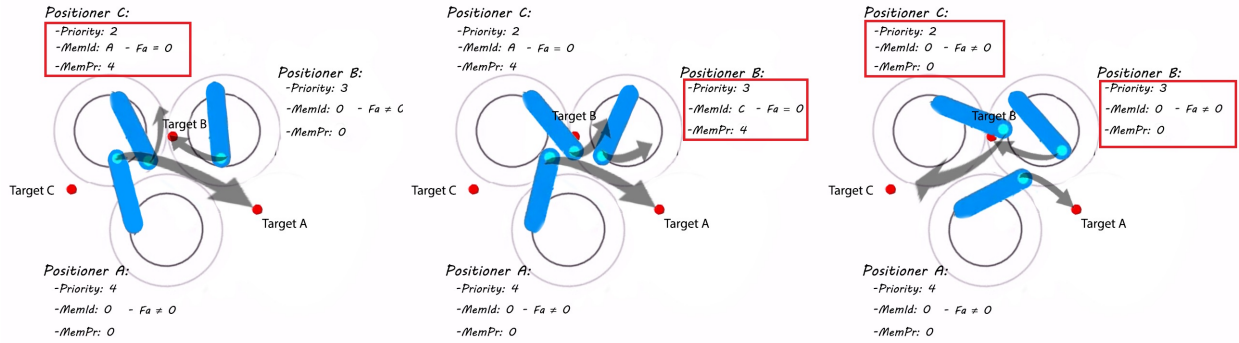


Figure 6. Positioner chain reaction using MemId and MemPr states, obeying algorithm 1 for interaction:

(left image) Positioner A with the biggest priority influences its neighbour positioner C which remembers the priority and id information of A when becoming OFF; (middle image) Even if positioner C pre-assigned priority is smaller than B one, because C is OFF, B has to look on the memorized information of C: B having a smaller pre-assigned priority, it becomes OFF and carry on the memorized priority of C; (right image) Once positioner A is out of the zone  $Z_{ON/OFF}$  of C, deep first search is used to reset all the states of all the positioners that are in the chain reaction because of A

The four main conditions in lines 3, 6, 9, 12, in the algorithm 1 correspond to all the possible interactions of two positioners based on their ON/OFF states. It shows how the neighbour positioner states is compared and is changing the agent positioner ones. Only the agent positioner states are modified. The reason is simply because algorithm 1 is applied to each positioner with respect to each of its neighbours in the system.

The general rule of comparison for the priorities in line 4, 7, 10, 13 depends on the ON/OFF state of the positioner: if the positioner is ON, the pre-assigned priority is used, otherwise it is the memorized priority "MemPr". Similarly, the possible values that can be assigned for the state "MemPr" of the agent positioner as shown in the lines after those conditions also depend on the first state ON/OFF of the neighbour positioner: the pre-assigned priority of the neighbour positioner if ON, else its "MemPr" priority. Therefore, all OFF positioners in the chain reaction remember the pre-assigned priority of the only ON positioner which is at the origin of this chain.

In all cases, the agent positioner always remember the ID of the neighbour positioner. It is used to track down all the positioners responsible for the chain reaction.

---

**Algorithm 1** *agent\_status(agent, neighbour)*

---

**output:** For a considered positioner, called agent positioners, modification of its 1st, 2nd and 3rd states with respect to its considered neighbour positioner

```

1 if agent.Arr == False or neighbour == OFF ;           // "Arr" state introduced on sub-chapter 2.2.3
2 then
3   if agent == OFF and neighbour == OFF then
4     if agent.MemPr < neighbour.MemPr then
5       agent.MemId = Neighbour Id
6       agent.MemPr = Neighbour.MemPr
7   else if agent == ON and neighbour == ON then
8     if Agent Priority < Neighbour Priority then
9       agent.ON/OFF = OFF
10      agent.MemId = Neighbour Id
11      agent.MemPr = Neighbour Priority
12  else if agent == OFF and neighbour == ON then
13    if agent.MemPr < Neighbour Priority then
14      agent.MemId = Neighbour Id
15      agent.MemPr = Neighbour Priority
16  else if agent == ON and neighbour == OFF then
17    if Agent Priority < neighbour.MemPr then
18      agent.ON/OFF = OFF
19      agent.MemId = Neighbour Id
20      agent.MemPr = Neighbour.MemPr
21  if agent.MemId == Neighbour ID then
22    Return indication Agent positioner states still being changed !

```

---

The lifetime of the memorization states, "MemId" and "MemPr", is bound to its agent positioner being OFF inside the  $Z_{ON/OFF}$  zone of the neighbour positioner that changed its states. Once outside, its states are re-initialized at their initial values, i.e. the first state at ON and the memorization states at 0. The states of all the other positioners that were influenced by the re-initilized agent positioner through the chain reaction process are also re-initialized. To do so, deep first search algorithm can be used with the "MemId" information to recognize which positioner is in the chain reaction.

When the algorithm 1 is reapplied again on the same two positioners, the agent positioner inside the  $Z_{ON/OFF}$

zone of its neighbour positioner won't enter any of the conditions mentionned above since its states were already changed previously by this same neighbour. The condition in line 15 is added to output that the agent positioner is still being affected by the same neighbour positioner, in order to avoid any complicated behaviors of the system.

### 2.2.3 Fourth state: Arrived

With the supposition that a positioner with the highest priority,  $p = 4$ , is already at destination, any lower priority positioners with their targets inside its zones  $Z_{ON/OFF}$  and  $Z_{F_r}$  are subject to repulsion forces and a changes of their states. The results are their attractive forces being disabled once inside the zones and the positioners being repulsed without resistance, hence never reaching their targets.

This particular oscillation scenario is shown in figure 7 with a the positioner A having a higher pre-assigned priority than a positioner B whose target is inside the zones of A. Once inside the  $Z_{ON/OFF}$  and  $Z_{F_r}$ , the positioner B, changing to OFF, will have its attractive force deactivate and repulsed.

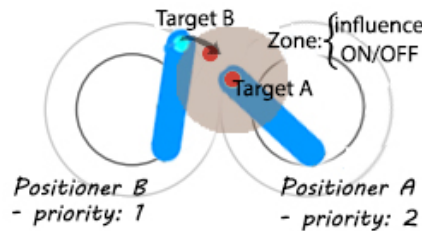


Figure 7. Principle of fourth state "Arr":

Problematic configuration with target inside a higher priority "already-converged" positioner zones  $Z_{ON/OFF}$  and  $Z_{F_r}$ .

In order to solve this problematic behavior, a fourth state, "Arr", is introduced to qualify a positioner that has converged to its target.

Coming with this new state is the inability for the already converged positioner A to affect any other positioners states which are inside its zone  $Z_{ON/OFF}$ , regardless of their priorities. Without having their attractive forces canceled, the neighbour positioners can converge to their targets that are inside the zone.

In addition, the zone  $Z_{F_r}$  has to be mitigated such that the repulsive forces felt from the neighbour positioners do not hinder their convergences. Although several complex methods can be used, we decided to simply reduce the radius of the repulsion zone  $Z_{F_r}$ . Though two very close targets can make two positioners oscillates a little bit as the one arriving might repulse the one already arrived, both can stabilize and converge surprisingly well by adding some dead-zone between the end-effector and its target or time delay before the positioner exit its "Arr" state.

### 2.2.4 Fifth state: Deadlock and pseudo-priority

In order to have a clearer and better explanation of the fifth state, this sub-chapter is divided in three part:

1. A description of the state and its principle
2. An explanation on how the states changes when interacting with other positioners
3. Lastly, a method to detect situations which require the activation of the state



## State & Principle

So far, the finite state machine only takes into account direct positioner interaction that have different priority values. When two positioners with the same priority but opposite movement direction enter in direct contact, deadlock situations is more highly to occur. This scenario is the reason why a fifth and last state is added to the finite state machine: to regulate the direct interaction between positioners with same priority value.

To better understand the design of this state and also of its principle, an analogy is first done with an example of the well-known traffic jam problem where two vehicles are being blocked by each other at one intersection. Of course, an ambulance in this scenario would have had the priority over the other vehicles, which is why we assume that they are two same vehicle of equal importance.

Usually in this kind of situation, the first goal of the drivers is not to reach their targets but instead to find a solution to the actual deadlock while respecting the code of the road. They also monitor if any more important vehicles such as an ambulance or a military tank are approaching (so they can quickly go off-route to let them pass).

To solve the deadlock, they just "isolate" themselves from the rest (involving more vehicles in the deadlock will only make the situation more complicated to solve) in order to better determine which one should have the priority over the other. To establish the best course of actions, their evaluations are usually based on the indications of direction (the targeted direction) given by the other driver.

With this example in mind, we define the fifth state, "lock\_sm", to designate if the positioner is in a deadlock situation with another one that has the same priority value. For the rest of this paper, we refer to this specific situation as a "lock\_sm" one.

In addition, a pseudo-priority variable, "psPr", is also associated to each positioner. It enables the system to obey the following principle, also illustrated in figure 8:

When in a "lock\_sm" situation, the first goal of the two positioners is not to reach their targets anymore but instead to solve their current problem by determining which one should have the priority over the other.

In order to solve this situation more quickly, both the distances from their end-effector to their respective targets are evaluated. The positioner with the smallest distance becomes more important.

The idea is to have the pseudo-priority, "psPr", used only for positioners in "lock\_sm" situation, in order to indicate which of the two positioners is more important while the interactions with the other positioners is regulated with the pre-assigned or memorized priority.

Moreover, to facilitate their movements when unblocking the deadlock, the positioner with bigger pseudo-priority has an additional isotropic and constant repulsive force from the target of the other positioner. The force can be formulated as a direct weighted relation with the average of the recent past velocities values of the positioner in order to diminish the risk of collision due to sudden speed increase.



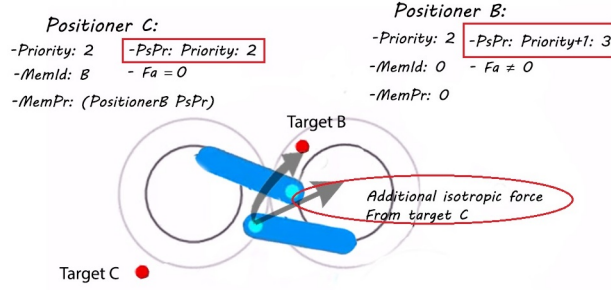


Figure 8. Principle of 5th state, "lock\_sm", and pseudo-priority, "psPr" :

With the same priority, inside their zone " $Z_{ON/OFF}$ " and in "lock\_sm" situation, the distance from positioner B end-effector to its target being smaller compared to the end-effector C to target C, the positioner B becomes more important symbolised by its pseudo-priority, "PsPr", being bigger and both fifth state equal to B

### Values & interactions

The fifth state, "lock\_sm", and the pseudo-priority, "psPr", are both respectively initialized at 0 and at the pre-assigned priority of the positioner. They are subject to a change of values when the considered positioner is inside the zone " $Z_{ON/OFF}$ " of one of its neighbour and the configuration of the two positioners is considered leading to a "lock\_sm" situation.

In case of a "lock\_sm" situation, the pseudo-priority of the positioner deemed the most important one is increased by 1, while both positioner's fifth states "lock\_sm" are being equaled to the ID of the most important positioner. The reason is to give an unique identification so the two positioners can recognize among their neighbours that they are both together in a "lock\_sm" situation.

The values of all the states depend not only on the ON/OFF state but also on the new introduced variables: "lock\_sm" and "psPr". The pseudo-code of algorithm 1 is then completed with conditions to take into account all these new possible interactions between positioners.

The reasoning on how to integrate these new cases of positioners interactions is shown with algorithm 2 which represents only the condition of line 9 of algorithm 1 with the agent and neighbour positioners being OFF. The same logic is applied to complete the rest of the conditions in lines 3, 6 and 12 of algorithm 1.

Because of the possible values of the fifth state, if it is bigger than 0, then its positioner is in a "lock\_sm" situation, not necessarily with its neighbour which can be in another "lock\_sm" situation with an other positioner. As a positioner can basically be either in a "lock\_sm" or not, four combinations for the fifth state are present as illustrated with the lines 2, 5, 8 and 11 of algorithm 2.

With a total of three priorities (pre-assigned, memorized and pseudo priority), their comparisons in the lines 3, 6, 9 and 12 of algorithm 2, and also in general for the system obey the following logic:

- If both positioners are not in a "lock\_sm" situation (line 2), their interactions follow the rules established in algorithm 1.
- If the positioners are both in a "lock\_sm" situation (line 11), the OFF positioner is still using its memorized priority "MemPr", but the pseudo-priority "PsPr" is used instead of the pre-assigned priority for the ON positioner. Consequently, the agent positioner memorizes the pseudo-priority of its neighbours.
- If one positioner is in a "lock\_sm" situation and the other is not, as shown in lines 5 and 8, one has to think back with the example of the traffic jam where two vehicles are blocked at an intersection. As mentioned, involving more vehicles into the deadlock situation is unwanted, which is why the pseudo-priority is not used at all for this mix-cases.

- If the two positioners are ON, then both use their pre-assigned priorities.
- If the two positioners are OFF, the one that is in a "lock\_sm" situation use its pre-assigned priority while the memorized priority "MemPr" is used by the other.
- For their ON/OFF mixed cases, we can push forward the reasoning with the vehicles problem to understand the reason of the priority comparison.  
 If we consider the two vehicles in the deadlock situation to be the most important one (an ambulance), then an external vehicle (a tourist vehicle) won't be able to influence them. As a matter of fact, even if one ambulance becomes less important than the other one in order to solve the deadlock, it is still an ambulance that cannot be less important than a simple tourist vehicle.  
 Following this logic, a positioner that is in a "lock\_sm" situation, regardless of being ON or OFF, will still use its pre-assigned priority. The other positioner not involved in the deadlock use either its "MemPr" if OFF, or its pre-assigned priority if ON.

For any combination of their ON/OFF states, when the agent positioner in a "lock\_sm" situation has been influenced by its neighbour that is not in a "lock\_sm" situation, it means that the neighbour was able to break out the deadlock situation because of its higher importance. Consequently, both the agent and the positioner in "lock\_sm" situation with it need to have their fifth states and pseudo-priority re-initialized.

The agent positioner will always memorized the priority that the neighbour positioner used for the priority comparison. Similarly, the agent always memorized the Id of the neighbour (not its MemId).

---

**Algorithm 2** Completion of the condition in line 9 in algorithm 1

---

**Data:** All possible interactions between a considered OFF positioner, called agent positioner, and its ON neighbours

```

1 if agent == OFF and neighbour == ON then
2   if agent.lock_sm > 0 and neighbour.lock_sm > 0 then
3     if agent.MemPr < neighbour.PsPr then
4       agent.MemId = Neighbour Id
4       agent.MemPr = Neighbour.PsPr
5   else if agent.lock_sm > 0 and neighbour.lock_sm == 0 then
6     if Agent Priority < Neighbour Priority then
7       Reset_Both_Positioners_FifthState()
7       Reset_Both_Positioners_PseudoPriority()
7       agent.MemId = Neighbour Id
7       agent.MemPr = Neighbour Priority
8   else if agent.lock_sm == 0 and neighbour.lock_sm > 0 then
9     if agent.MemPr < Neighbour Priority then
10      agent.MemId = Neighbour Id
10      agent.MemPr = Neighbour Priority
11  else if agent.lock_sm == 0 and neighbour.lock_sm == 0 then
12    if agent.MemPr < Neighbour Priority then
13      agent.MemId = Neighbour Id
13      agent.MemPr = Neighbour Priority

```

---

### Deadlock "lock\_sm" localization

In this paragraph, the objective is to describe how we consider the configuration between two positioners to be a "lock\_sm" situation.

With two positioners, called agent positioner and its neighbour, having an equal pre-assigned priority inside their zones " $Z_{ON/OFF}$ ", we use the "point of view" from both their closest parts (end-effector, middle or joint) to assess the possibility of deadlock.

The "point of view" can be understood as how the agent is seeing its target and its neighbours target with respect to its neighbours positioners closest part. Mathematically, we define it as the projection of the agent and neighbour targets on a line created by the vector between the closest parts of both positioners.

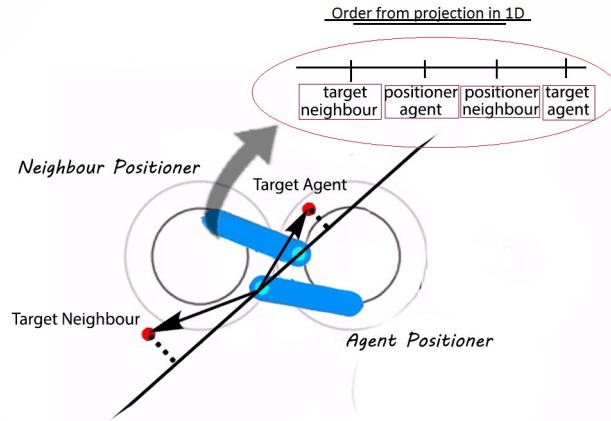


Figure 9. Point of view in 1D configuration:  
Projection of agent and neighbour target on the line created from closest part of both positioners

It basically projects everything into a line in order to evaluate in 1D if the agent target is on the same side as the agent positioner or is behind the neighbour positioner. For the latter case as shown in figure 9, the neighbour positioner becomes an obstacle to the movement of the agent one, possibly leading to a deadlock situation. We then compare the projection with a look-up table that contains all possible 1D configurations of the agent, neighbour positioners and their targets placed in 1 line (24 possible permutations).

The logic to distinguish which 1D configuration is considered a deadlock is simply to individually assess for each configuration if there is any collision with both positioners movements towards their target on the 1D line. The example as seen in figure 10 shows the reasoning with few examples which can be generalized to determine the rest of the 1D configurations.

Since the movement of the positioner is planar with 2 degree of freedom, having the point of view from both agent and neighbour positioners allows a better perspective of the current configuration. As such, the consequence is that if their configuration is considered as a deadlock from the point of view of one positioner, then both positioners are in deadlock situation. In other words, we only need one projections to be a 1D configuration deadlock in order to consider the whole situation as a "lock\_sm" one.

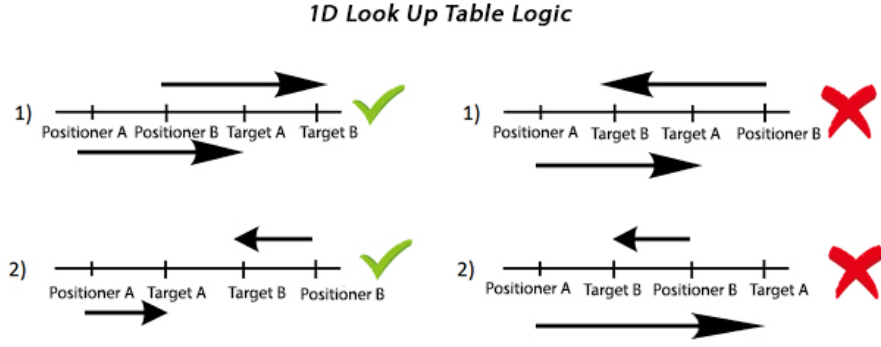


Figure 10. Configuration examples of look up table logic:

On the left column, positioner A and B movements are not opposing each other nor are the positioner on the way of the other movement; On the right column, the positioner B ending up at its target will hinder the movement of positioner A to its target, being in between them

### 3. SIMULATION & RESULTS

#### 3.1 Software & Hardware

To evaluate our approach, a simulator developed in python which recreates the shape and the behavior of the positioners is used. The trajectories of the positioners are first pre-computed and then directly given in open-loop to the electronics controlling the motors.

All simulation tests were conducted on a Lenovo thinkpad T450s with an Intel Core i7-5600U @ 2.69GHz x 4 processor, GeForce 940M SSE2 graphic card on an ubuntu 14.04 LTS version.

#### 3.2 Tests & results

Repeatability tests are done in order to assess the performance and efficiency of our approach. For this purpose, we compare the results obtained from our finite state machine (FSM) with the ones using only the decentralized navigation function algorithm (Without FSM). In the latter case, pre-assigned priorities are not taken into account as they are no mechanism to do so.

In addition to all the figures seen so far which are real working examples of different priorities positioner interaction, the goal is also to assess the robustness of the algorithm to coordinate a whole system of positioners with a hierarchy of importance.

##### 3.2.1 Repeatability: convergence and time performance

We run our algorithm on 8 different test files. Each represents one specific and unique initial configuration of thousand of positioners with their respective targets. All positioners have the same pre-assigned priority value.

Each configuration file is used to evaluate the following performances:

- Number of positioners successfully converging to their targets.
- Time in seconds necessary for the pre-computation of their movements.

The goal is not only to verify if the performance of our algorithm is repeatable over different configurations of our system but also if it is acceptable to use it in real time.

To this end, for each configuration test file, the total number of around a thousand of positioners is partitioned into 12 subsets of sub-total positioners, on which we run our algorithm. The results are given in figures 11, 12, 13, 14, 15, 16, 17, 18.

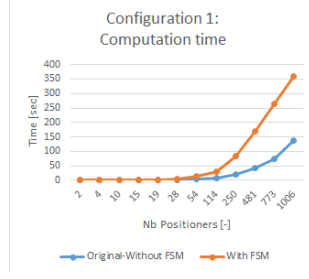
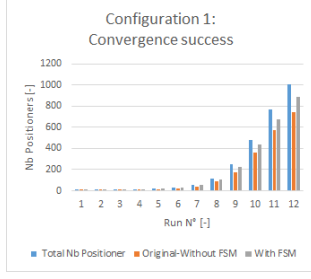


Figure 11. Configuration 1:

Convergence success and computation time data

- With FSM: Decentralized navigation function with finite state machine
- Without FSM: Decentralized navigation function without finite state machine

Total Positioners		54	114	250	481	773	1006
Conv.	With FSM	52	105	228	434	675	889
	Without FSM	41	88	176	358	577	747
Time (sec)	With FSM	13.8	30.9	84	169.7	265.2	360.1
	Without FSM	3.5	8.6	21.4	43.7	74.5	136.1

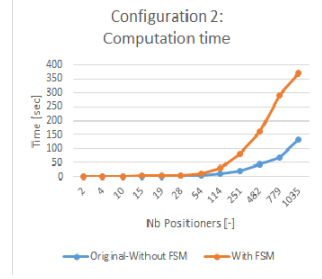
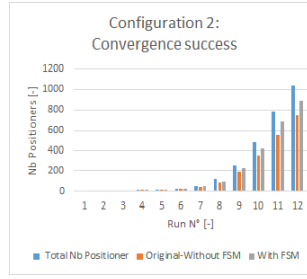


Figure 12. Configuration 2:

Convergence success and computation time data

- With FSM: Decentralized navigation function with finite state machine
- Without FSM: Decentralized navigation function without finite state machine

Total Positioners		54	114	251	482	779	1035
Conv.	With FSM	52	103	231	415	679	892
	Without FSM	45	91	189	350	555	750
Time (sec)	With FSM	11.8	31	81.2	159.2	291.7	369.7
	Without FSM	3.8	9.6	21.6	44	69.9	134.1

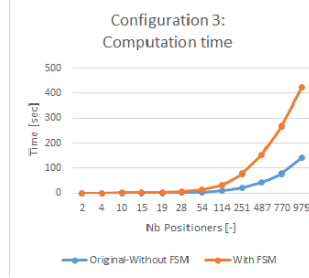
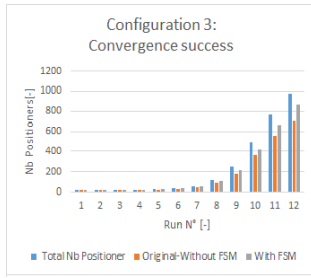


Figure 13. Configuration 3:

Convergence success and computation time data

Total Positioners		54	114	251	487	770	979
Conv.	With FSM	53	101	217	413	665	864
	Without FSM	41	86	181	358	554	704
Time (sec)	With FSM	14	32.3	79.9	155.1	267.2	425.2
	Without FSM	3.7	9.1	21	45.2	79.2	143.6

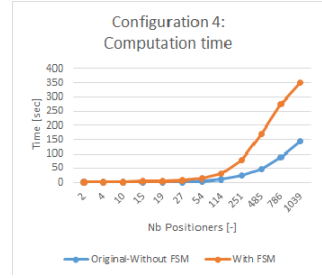
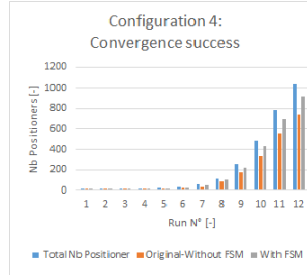


Figure 14. Configuration 4:

Convergence success and computation time data

Total Positioners		54	114	251	485	786	1039
Conv.	With FSM	48	103	222	436	698	914
	Without FSM	35	81	172	339	553	742
Time (sec)	With FSM	13.4	30.8	79.6	170.4	274.1	352.1
	Without FSM	3.8	9.3	23.9	47.3	89.5	144.9

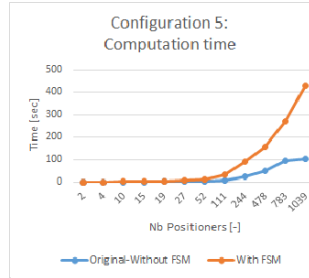
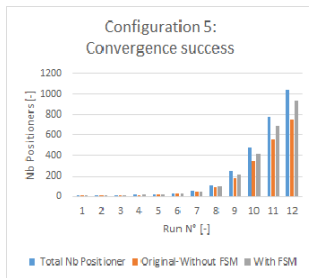


Figure 15. Configuration 5:

Convergence success and computation time data

Total Positioners		52	111	244	478	783	1039
Conv.	With FSM	50	105	219	421	689	931
	Without FSM	44	89	186	349	555	754
Time (sec)	With FSM	12.9	32.7	91.8	155.9	267.8	429.7
	Without FSM	3.8	8.7	24.1	47.2	93.3	103.4

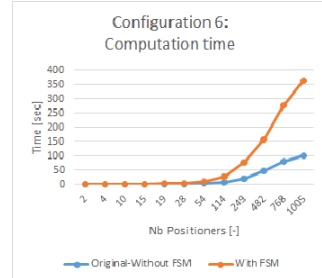
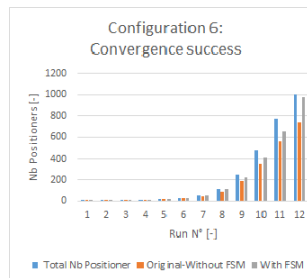


Figure 16. Configuration 6:

Convergence success and computation time data

Total Positioners		54	114	249	482	768	1005
Conv.	With FSM	54	110	225	415	651	973
	Without FSM	47	90	187	356	561	734
Time (sec)	With FSM	12	28.8	76.3	158.6	275.9	364.2
	Without FSM	3.9	8.6	22.6	46.4	79.8	101.2

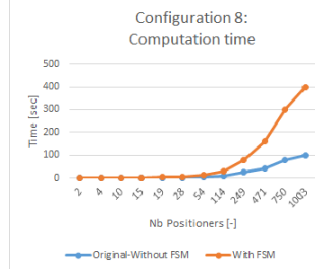
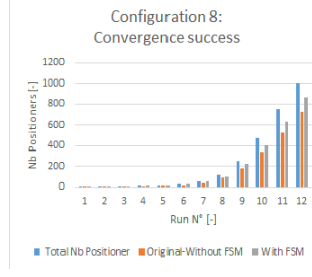
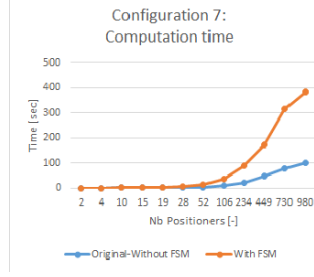
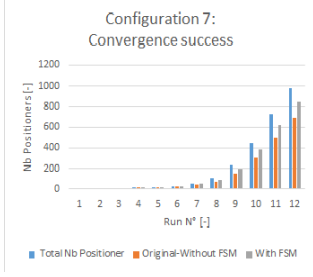


Figure 17. Configuration 7:  
Convergence success and computation time data

Total Positioners		52	106	234	449	730	980
Conv.	With FSM	50	90	196	382	621	844
	Without FSM	40	72	155	300	503	691
Time (sec)	With FSM	14.4	36.2	89	172.8	315.8	383
	Without FSM	3.9	8.6	21.1	46.8	76.5	102.1

Figure 18. Configuration 8:  
Convergence success and computation time data

Total Positioners	54	114	249	471	750	1003	
Conv.	With FSM	52	103	221	401	630	861
	Without FSM	40	86	179	342	525	723
Time (sec)	With FSM	12.3	29.2	78.2	161.9	301.2	399.5
	Without FSM	4	9.2	23.0	42.7	78.3	99.5

For all the configuration tests, the general behavior of the system remains similar: when using our finite state machine, an improvement of positioners convergence from 60-70% to 80-95% can be observed globally and for each test.

However, the computation time is bigger, growing into a quasilinear order. It is an expected result since we added an another layer of computation to the already existing algorithm that we use for comparison. With better optimization such as parallel computation and/or more performant hardwares, this burden can be compensated.

Though all their pre-assigned priorities to each positioner are the same value, the result of convergence improvement with our method also demonstrates indirectly the robustness of our algorithm to take into account different priorities. Due to the mechanism of the fifth state of the finite machine, interactions of positioners with similar priorities also involves all the other states as their priorities become different.

## 4. CONCLUSION

In the context of the project MOONS and as an extension of [3], an approach with a finite state machine is built on top of a decentralized navigation function from the potential field family. It enables better coordination of a multi-objects positioners inside a confined space. With its different states, it takes into account different priorities pre-assigned to each positioner, creating a hierarchy of importance where positioners with higher priority have more probabilities to reach their targets. Along with the possibility to manage deadlocks, it enables a better convergence of the system: an improvement from 60-70% to 80-95% of total positioners converging to their targets is observed from repeatability tests. The results are obtained from comparing our approach with the one using only the decentralized navigation function. Furthermore, the algorithm is scalable to other system configuration than MOONS as it doesn't depend on the positioner's geometry.

Since we added another layer of complexity to an already existing algorithm, the pre-computation time of each positioner trajectory is slower, becoming quasilinear. One possible work can be to optimize the software by either finding the right optimization method such as parallel computing or using more performant hardwares. Moreover, since our approach has no proof of completeness, more research on the optimization of the algorithm for a better positioner convergence can be consider.

Extending this approach to take into account other features, such as for the project SLOAN-SDSS5 that needs to select which (UV, visible or IR) optical fibers to use within one positioner is also a possibility. It is a subject which is particularly interesting for astronomers since not all objects in the universe are well observed in the same range of the optic spectrum.

## ACKNOWLEDGMENTS

I would like to thank my supervisor Dr. Denis Gillet for his patience and trust in me, and also to Dr. Mohammed Bouri and prof. Jean-Paul Kneib for their support. A lot of thanks to Ezequiel Gonzalez for our brainstorming

sessions and to Dr. Laleh Makarem for reviewing my work and her advices, without whom this paper and algorithm wouldn't have been possible; and finally, to all the friendly members of the REACT Lab who made my working place a nice and productive environment.

## REFERENCES

- [1] Makarem, L., Kneib, J.-P., Gillet, D., Bleuler, H., Bouri, M., Jenni, L., Prada, F., and Sanchez, J., "Collision avoidance in next-generation fiber positioner robotic systems for large survey spectrographs," *Astronomy & Astrophysics* **566**, A84 (2004).
- [2] Makarem, L. and Gillet, D., [*Decentralized Multi-robot Coordination in Crowded Workspaces*], EPFL (2015).
- [3] Makarem, L., Kneib, J.-P., and Gillet, D., "Collision-free coordination of fiber positioners in multi-object spectrographs," (2016).