Priority coordination of fiber positioners in multi-objects spectrographs

Dominique Tao[a], Laleh Makarem[b], Mohamed Bouri[c], Jean-Paul Kneib[d], and Denis Gillet[e]

[a,b,e]REACT, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland
[c]LSRO, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland
[d]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 developping spectographs, 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 space. 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 completness 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.

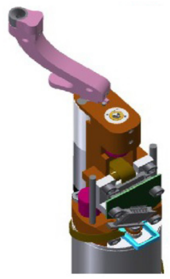
Figure 1. CAD positioner model

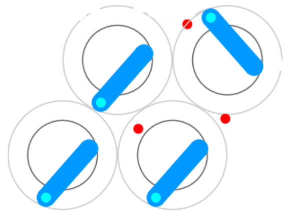
Figure 2. MOONS positioner in simulation

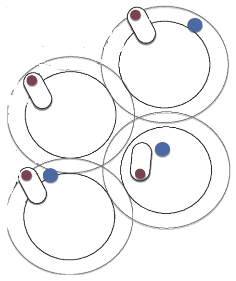
Figure 3. DESI positioner in simulation

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, requiring either a more complex developement 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.

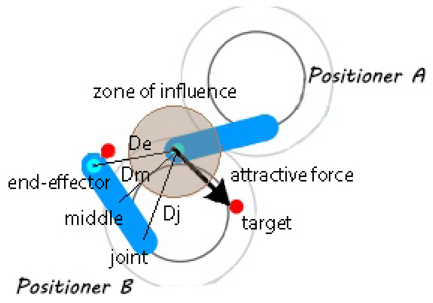

Figure 4. Variables representation of the system

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 i^{ieme} neighouring 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.

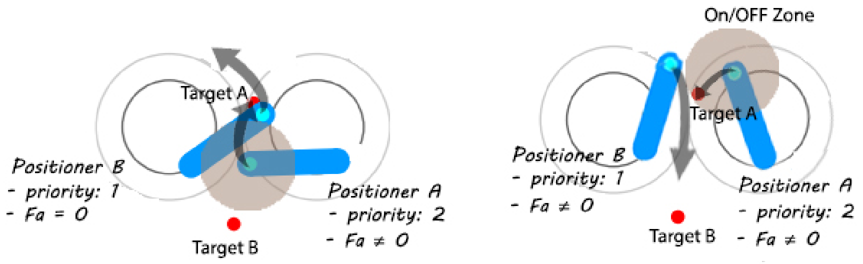

Figure 5. Principle of ON/OFF state (left to right image):
(left image) Positioner B inside $Z_{ON/OFF}$ of positioner A doesn't have any attrative force and is subject to repulsive force from positioner A, making it going to the opposite direction of its target B; (right image) The same positioner B, 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:

$$\begin{cases} y[t] = \dfrac{1}{M}\displaystyle\sum_{k=1}^{M} v[t-k] \\ y[t] < thres \Rightarrow noise \end{cases} \tag{1}$$

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 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 a matter of fact, deadlock situations can still occur when two ON high priority positioners are going against each other by propagating their oppposite movements through an intermediate low priority OFF positioner that is between them. This non-isolated scenario is illustrated in figure 6 (left image) where two positioners A and B would have been blocking each other through the OFF positioner C if without any regulation based on their states.

This problematic situation can be solved by having a better communication between positioners in order to allow a "chain reaction", which we defined as the process of one OFF positioner changing the first state of an other neighbouring positioner that is not the one that originally influenced it.

To do so and also for a general better coordination of the system, we use the principle of memorization by introducing two new states associated to each positioner:

- "MemPr":
  Initialize at 0, for an agent positioner, it is used to memorize either the pre-assigned priority or the MemPr of its neighbour positioner which was the one that changed the agent positioner states

- "MemId":
  Initialize at 0, for an agent positioner, it is used to memorize only the identity of the neighbour positioner which was the one that changed the agent positioner states

With this information, one OFF positioner, whose states was changed by an agent positioner, due to its higher priority, will be able to communicate with its other neighbour positioners with also the possibility to change their first state to OFF in order to allow the agent positioner to reach its targets more easily.

This example of chain reaction, also illustrated in figure 6 with a step by step explanation, is part of a more general set of rules that is used to regulate positioner's movements. It is represented by the pseudo-code of algorithm 1 which gives a concise overview on how all the states seen so far of an agent positioner are changing when interacting with a neighbour positioner.
Furthermore, these rules are only valid if, inside their $Z_{ON/OFF}$ zones, the considered agent positioner has not yet arrived to its target, A condition that will be which will be another positioner state later explain, as well as the whole condition in line 1, in the next sub-chapter 2.2.3.

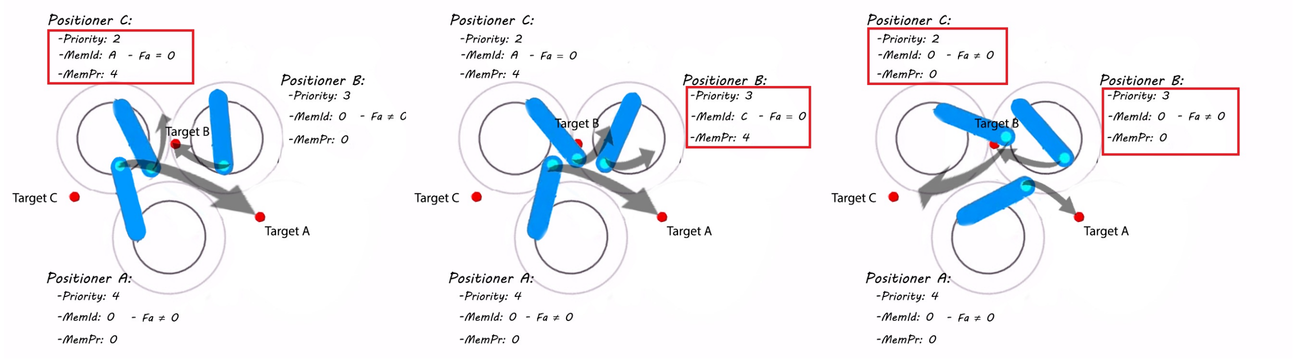

Figure 6. Positioner Chain Reaction using MemId and MemPr states, following algorithm 1 for interaction (left image) Positioner A with the biggest priority influences its neighbour positioner C which remembers the priority and id information of A when becoming OFF as shown in the red rectangle; (middle image) Even if positioner C priority is smaller than B, because C is OFF, B has to look on the information of C to find how that the movement of the positioner that influenced C is more important due to its priority, B becomes OFF and carry on the information that C has;(right image) Once positioner A is out of the zone $Z_{ON/OFF}$ with C, deep first search is used to reset the information carried by the chain reaction

As one can see with the lines 3, 6, 9, 12 in the algorithm 1, the four main conditions correspond to the possibilities for the neighbour positioner to affect the states of the agent positioner. Only the agent positioner states is changed, the reason is simply because algorithm 1 is applied to each positioner with respect to each of its neighours in the system.

As a general rule for the comparison in line 4, 7, 10, 13, the priority used 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: Pre-assigned priority of the neighbour positioner if ON, else its "MemPr" priority.

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

Algorithm 1 agent_status(*agent*, *neighbour*)

output: Modification of agent positioners 1st, 2nd and 3rd state 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
               agent.MemPr = Neighbour.MemPr
6      else if agent == ON and neighbour == ON then
7          if Agent Priority < Neighbour Priority then
8              agent.ON/OFF = OFF
               agent.MemId = Neighbour Id
               agent.MemPr = Neighbour Priority
9      else if agent == OFF and neighbour == ON then
10         if agent.MemPr < Neighbour Priority then
11             agent.MemId = Neighbour Id
               agent.MemPr = Neighbour Priority
12     else if agent == ON and neighbour == OFF then
13         if Agent Priority < neighbour.MemPr then
14             agent.ON/OFF = OFF
               agent.MemId = Neighbour Id
               agent.MemPr = Neighbour.MemPr
15     if agent.MemId == Neighbour ID then
16         Return indication Agent positioner states still being changed !
```

The lifetime of the memorization states, "MemId" and "MemPr", is bound to its positioner being OFF inside the $Z_{ON/OFF}$ zone of the other positioner that changed its states. Once it is outside of it, not only all its states are reinitialized at their initial value, i.e. ON/OFF state at ON and the memorization state at 0, but also all the states of the other positioners that were influence by the reinitiliazed positioner through the process chain reaction. To do so, the deep first search algorithm can be used with the MemId information to recognized which positioners need to have its states re-initiliazed.

Since algorithm 1 is constantly used on every positioner of the system, it has to take into consideration the case where an agent positioner with all its states already changed by a neighbour positioner: when the algorithm 1 is reapplied again on the same two positioners, the agent positioner inside the $Z_{ON/OFF}$ zone won't enter any of the conditions mentionned above since its states were already changed previously by the same neighbour positioner. The condition in line 15 is added to ouptut that the agent positioner is still being affected by the same neighbour positioner, in order to avoid any complicated behaviors of the system due to reinitialization.

2.2.3 Fourth state: Arrived

With the supposition that a positioner with the highest priority, $p = 4$, has already arrived at destination, any lower priority positioners with their target inside its $Z_{ON/OFF}$ and Z_{F_r} zone will be subject to repulsion forces and a change of their states, resulting in having their attractive forces disable once inside the zone and hence never reaching their targets.

This particular oscillation scenario is shown in figure 7, where the positioner A with higher priority than positioner B, whose target is inside the zone of A, will desactivate the attractive force of B and repulse it. Like a similar problem already seen on sub-chapter 2.2.1 with the ON/OFF state, B is ON outside and OFF inside and hence will never reach its target.

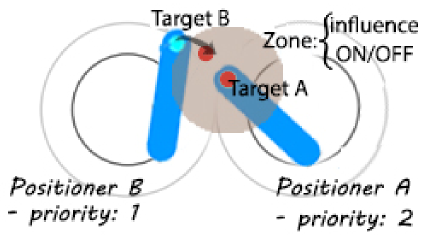

Figure 7. Problematic situation with target inside higher priority "already-arrived" positioner zone

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

Coming with this new state is the inability of the already converged positioner to affect any other positioners states within its zone $Z_{ON/OFF}$, regardless of their priorities, such that they can also converge to their targets which are inside its zone.

In addition, the zone Z_{F_r} or the repulsive forces needs to be mitigated such that the repulsion forces felt from the other positioners do not hinder their convergences. Although several complex methods can be used for this purpose such as force magnitude reduction following a mathematical model for a better forces distribution, we decided to simply reduce the radius of our 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, it works 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 this fifth state, this sub-chapter will be divided in three part:

1. A description of its state and principle by first illustrating it with an example on with the real world traffic circulation. The objective of this analogy is to enable a better understanding on how this state was designed.

2. An explanation on how the state changes when interacting with other positioners
3. Finally how to detect situations which require its activation

State and Principle

So far, the finite state machine only takes into account direct positioner interaction with different priorities. When two positioners with same priority and opposite movement direction enter in direct contact, deadlock situation 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.

To better understand the design of this state introduced below and also to have a better understanding of it principles, we can take as an example the well known problem of trafic jam 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 equally important.
Usually in this kind of situation, the first goal of the drivers is not anymore to reach their targets but instead to find a solution to the actual problem, while monitoring if any important vehicle such as an ambulance or a military tank is approching (so they can quickly go off-route into the sidewalk for example to let them pass).
To solve this problem, they just "isolate" themselves from the rest (as involving more vehicles will only make the situation more complicated to solve) to better find a solution by dertermining which one should have the priority over the other. To do so, each one is assessing the situation based on their own perspective and the code road rules to determine which one should go first, usually depending on which direction they wanted to go at the beginning.

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

In addition, a pseudo-priority variable, "psPr", is also associated to each positioner to enable the system to obey the following principle, as illustrated in figure 8:
When in a "lock_sm" situation, the first priority 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. To do so, both their distances from their end-effector to their respective targets are evaluated such that, in orded to solve this situation more quickly, the positioner with the smallest distance becomes more important. The idea is to have the pseudo-priority, "psPr", for only the "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 priority.

Moreover, to facilitate the movement into solving the deadlock problem, the positioner with bigger pseudo-priority is having an additional isotropic and constant repulsive force from the target of the other positioner that is in a "lock_sm" situation with it, which can be formulated as a direct weighted relation with the current velocity of the positioner.

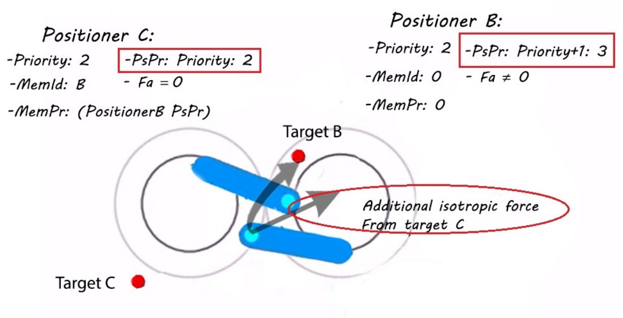

Figure 8. Principle of 5th state, "lock_sm", and pseudo-priority, "psPr" :
With the same priority, inside their zone "$Z_{ON/OFF}$" and in "lock_sm" situation, the distance from positioner B end-effector to its target being smaller compared to the end-effector C to target C, the positioner B becomes more important symbolised by its pseudo-priority, "PsPr", being bigger and both fifth state equal to B

Initialize at 0 and at the pre-assigned priority of the positioner for respectively the "lock_sm" state and the pseudo-priority, "psPr", these states change when being inside the zone "$Z_{ON/OFF}$" and the configuration of the two positioners is considered leading to a deadlock situation.
In case of a "lock_sm" situation, the pseudo-priority of the positioner deemed the most important one is increase by 1, while both positioner's fifth state "lock_sm" is equal to the ID of the most important positioner. The reason is to give an unique identification to each "lock_sm" problem between two positioners.

Deadlock localization
In order to localize a deadlock, we use the point of view from both the closest parts (end-effector, middle or joint) of the two positioners with equal pre-assigned priority inside the zone " $Z_{ON/OFF}$". The objectif is to evaluate if they could possibly create a "lock_sm" situation.
Considering an agent and neighbour positioner with equal priority, 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 target on the line created by the vector between the closest parts of both positioners that could possibly be responsible for the deadlock situation.

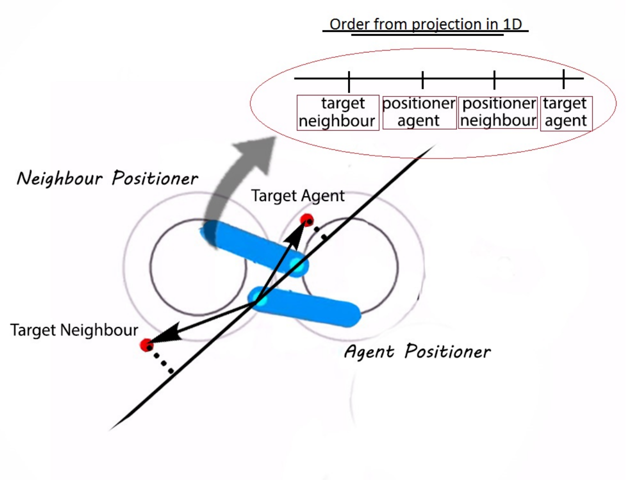

Figure 9. 1D configuration:
Projection of agent and neighbour target on line created from closest part of both positioners

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 then becomes an obstacle to the movement of the agent one, possible leading to a deadlock situation. Since the movement of the positioner is planar with 2 degree of freedom, having the point of view from both positioners allows a better perspective of the current configuration.

In order to assess if the 1D configuration obtained from projection is considered a deadlock situation, we compare the resulting 1D projection result with a look-up table that corresponds to all possible 1D configurations of the agent, neighbour positioners and their targets placed in 1 line, so all 24 possible permutations. The logic behind to distinguish which configuration is considered a deadlock is simply to individually assess for each configuration if they are any collision in both positioners movements towards their target on the 1D line. The example as seen in figure 10 shows few reasoning examples that can be generalize to determine the rest of the 1D configurations.

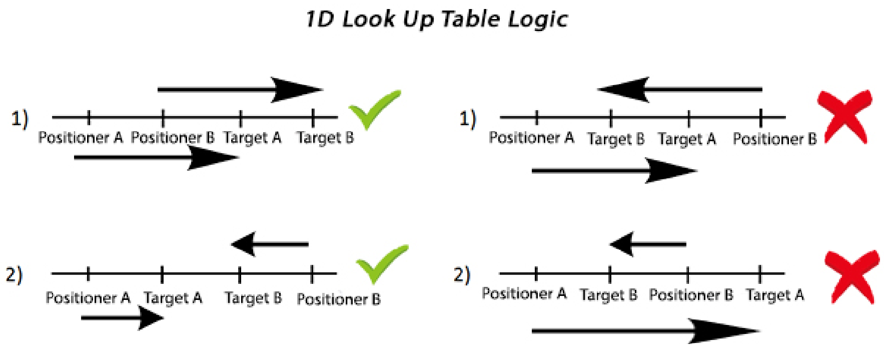

Figure 10. Configuration examples of look up table logic:
On the left column, positioner A and B movements are not opposing each other nor are the positioner on the way of the other movement; On the right column, the positioner B ending up at its target will hinder the movement of positioner A to its target, being in between them

3. SIMULATION & RESULTS

3.1 Software & Hardware

To evaluate our approach, a simulator developed in python that recreates the shape and the behavior of the positioners is used. The trajectories of the positioners are first pre-computed and then given directly in open-loop to the electronics controlling the motors.
The simulation tests were conducted on a Lenovo thinkpad T450s with a Intel Core i7-5600U @ 2.69GHz x 4 processor, GeForce 940M SSE2 graphic card on a ubuntu 14.04 LTS version.

3.2 Test cases

A first test on the repeatability of our approach is done by assessing its performance and efficiency on different test case files, each composed of several total number of positioners with unique initial configurations.
The results of both methods,i.e. using the decentralized navigation function algorithm with and without our finite state machine approach, are compared by measuring the number of positioners successfuly converging to their targets and the time necessary for the pre-computation of their movements.
Indirectly, the improvement of the positioner convergence results from this test is also a good indication on how well our finite state machine methodology takes into account priority and deadlock solving using .

3.2.1 Priority and Repeatability: convergence and time performance

We run our algorithm on 8 different test files, where each represents one specific initial configuration of thousand of positioners, with their respective targets. Though all their pre-assigned priorities for each positioner are the same, the convergence success with our method also demonstrate the robustness of our algorithm when dealing with positioners having the same priorities, and hence indirectly having different order of priorities due to the mechanism of the fifth state of the finite machine.

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 our algorithm is repeatable over different configurations of our system but also if it is acceptable to use such an approach in real time by evaluating the pre-computation time.
To this end, for each configuration test files, the total number of around thousand of positioners are partitioned into a 12 subset of total positioners, on which we run our algorithm. In order to assess how the positioners convergence rate improved, we compare the result of our algorithm with the one collected by only running the decentralized navigation function without our finite state machine, and thus without taking into account any priorities pre-assigned to each positioners.

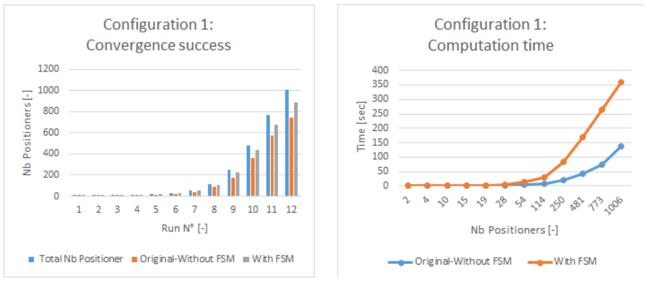

Figure 11. Configuration 1:
Convergence success and computation time data

- 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

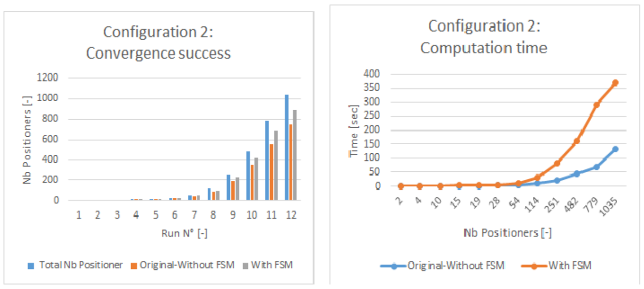

Figure 12. Configuration 2:
Convergence success and computation time data

- 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

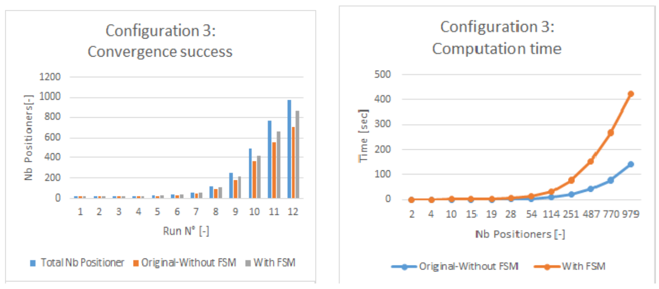

Figure 13. Configuration 3:
Convergence success and computation time data

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

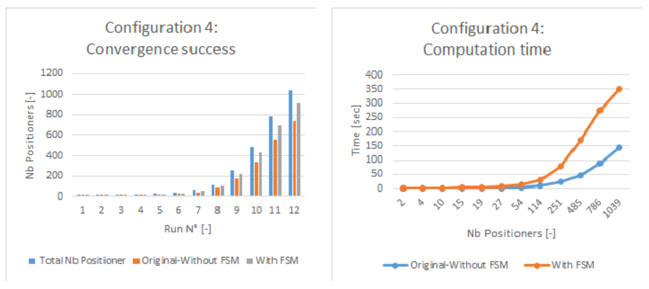

Figure 14. Configuration 4:
Convergence success and computation time data

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

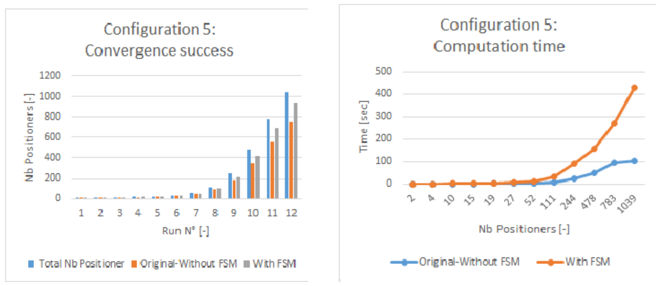

Figure 15. Configuration 5:
Convergence success and computation time data

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

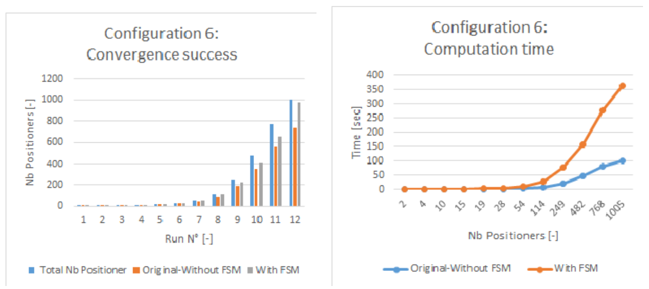

Figure 16. Configuration 6:
Convergence success and computation time data

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

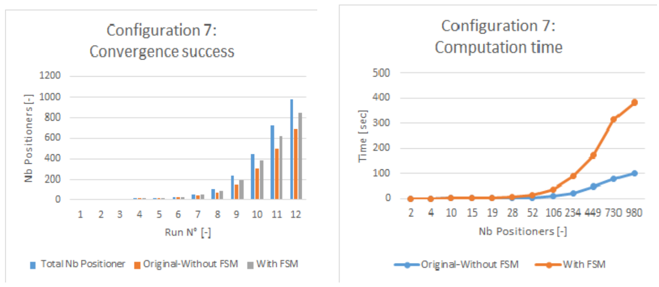

Figure 17. Configuration 7:
Convergence success and computation time data

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

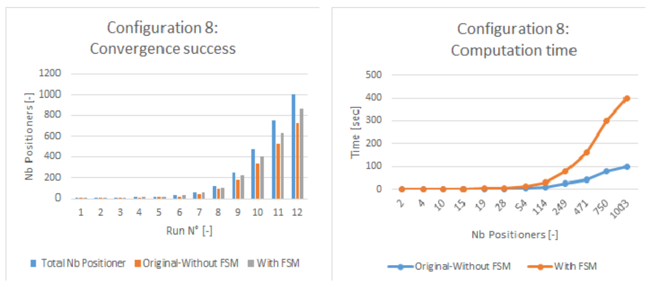

Figure 18. Configuration 8:
Convergence success and computation time data

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 test, 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 for each test. However, as we expected, the computation time became bigger, growing into a quasilinear order, since we added an another layer of computation. With better optimization such as parallel computation and/or more performant hardwares, this burden can be compensated.

4. CONCLUSION

In the context of MOONS project, an approach with a finite state machine on top of a decentralized navigation function from the potential field family is used to enable better coordination of a multi-objects positioner inside a confined space. With its different states, it takes into account pre-assigned priority to each positioner, creating a hierarchy of importance where positioners with the highest priority have more probability to reach their targets than lower priority ones, enabling a better coordination of the positioners movements and also the possibility to better manage deadlocks situations. An improvement from 60-70% to 80-95% of total positioner converging to their target is observed from repeatability tests, results taken from comparing our approach of adding a finite state machine layer with the one using only the decentralized navigation function.
Since we added another layer of complexity to an already existing algorithm, the precomputation time of each positioner trajectory is slower, becoming quasilinear. One possible work can be to optimize either the code by finding the right optimization method such as parallel computing or using more performant hardware. Furthermore, since our approach has no proof of being complete, more research on the optimization of the algorithm for a better positioner convergence or even a proof for completness can be consider.
Extending this approach to take into account other features, such as in the project SLOAN-SDSS5 for selecting which (UV, visible or IR) optical fibers to use withing one positioner, can be a subject which astronomers are very interested since not all objects in the universe are well observed in the same range of optic spectrum.

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr. Denis Gillet and also Dr. Mohammed Bouri and prof. Jean-Paul Kneib, without whom this paper and project about priority coordination would not have been possible. A lot of thanks to Ezequiel Gonzazel for our brainstorming sessions and also for the friendly members of the REACT Lab for making my work environment a nice and productive place.

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).