Rigorous Movement of Convex Polygons on a Path Using Multiple Robots

Pierre Chamoun
School of Computer Science
Carleton University
Ottawa, Canada
pchamoun@connect.carleton.ca

Dr. Mark Lanthier
School of Computer Science
Carleton University
Ottawa, Canada
lanthier@scs.carleton.ca

***Abstract*—This paper describes an approach for pushing a convex polygonal object with rigor using multiple robots, along a desired rectilinear path in a two-dimensional polygonal environment. The goal is to rigorously push the object along the path while preserving its orientation and alignment, as well as precisely rotating it about its center when necessary. A path planning algorithm is presented which computes a shortest-path approximation between two points in the environment. In general, the path requires both translations and rotations of the object along the way. Robots are arranged into three groups, where each group is assigned a task of either pushing the object towards its goal or adjusting it as it veers off from the desired path. Each robot is computationally simple in that it merely moves towards a target point somewhere on the boundary of the object. As the robots move towards these target points, they cooperatively push the object with no interaction between one another. The robots rely on only three parameters to push the object: the orientation of the object, the current target point and the task they are required to perform. The target points are provided by a global control & monitoring system that monitors the progress and stability of the robots as they push the object along the path, providing direction to the robots in terms of tasks such as pushing, rotating, re-alignment, re-orientation or re-positioning commands. We verified our algorithm with a number of experiments that address the usefulness of the solution as well as the effects that an increase in the number of robots will have on the runtime and the data communication load.**

***Index Terms*—multi-robots, pushing, piano movers, cooperative transportation, swarm intelligence, path planning configuration space, central controller.**

I. Introduction

One of the ultimate goals in robotics is to design robots that are capable of planning and accomplishing tasks independent of human intervention. For example, multiple robots can be assigned to push heavy items in a factory, transport dangerous goods, or push a boat into a port. The use of multiple robots to accomplish such tasks allows the object to be pushed rigorously. In addition, the object may be too heavy for just one or two robots to push. The problem addressed in this paper is to design a strategy to enable multiple robots to rigorously push an object along a polygonal path in a two-dimensional environment containing polygonal obstacles. To be more precise, the problem statement is as follows:

Let W be the workspace (2D Euclidean space) containing a set $O=\{O_1,O_2,\ldots,O_n\}$ of stationary convex polygonal obstacles. Let M be a convex polygon that needs to be transported from a start point s to a destination point t in W. Assume that M is to be pushed along a piecewise linear path $P=\{p_1, p_2, \ldots, p_k\}$, where $p_1 = s$ and $p_k = t$ by a set of robots $R=\{R_1, R_2, \ldots, R_r\}$. Assume that the system is noise-free in that each robot is able to perform point-to-point travelling without error. The robots attempt to translate M along P such that the center of mass of M (i.e., defined as a reference point C) remains on P at all times. During translation from s to t, rotations of M (about point C) are also allowed at any point p_i, $\forall\, 1 \leq i \leq k$. It is possible that M may deviate from P since the robots do not move precisely in real life, however our algorithm is designed to minimize such path deviations by detecting and correcting misalignments along the way.

Our overall control system is shown in Figure 1. It consists of a *global controller (GC)* that monitors the motion of M and provides the robots with the necessary information in order to push M with rigor along each segment of P. We assume that all robots remain active and do not become incapacitated during the course of the operation. Communication between the robots and the *GC* is minimal and limited to providing (a) target points in W for the robots to move towards, (b) the orientation of M, and (c) the task to be performed. It is assumed that the *GC* has complete global knowledge of W at all times including the stationary obstacle set O, the trajectory path P, the position and orientation of M and the source/target points s and t.

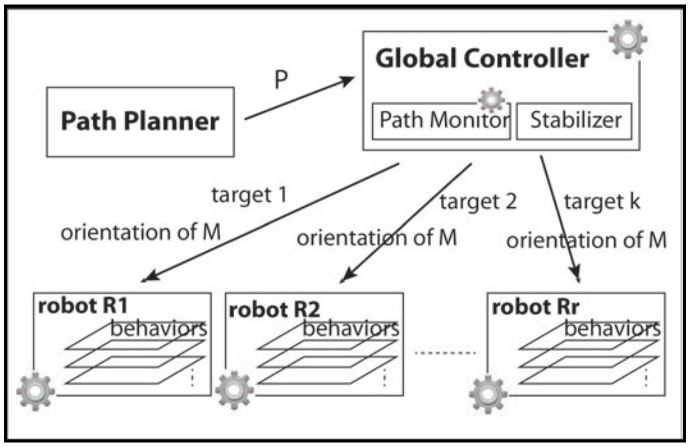


Figure 1: The overall system design.

The *GC* initiates a *Path Monitor* process that acts as a sensor system which is capable of accurately measuring the location and orientation of the robots and *M* at all times while being pushed along *P*. The movement along *P* is broken down into individual translations of *M* along each segment of the path $p_i p_{i+1}$, $\forall 1 \leq i < k$ as well as potential rotations at each point p_i, $\forall 1 \leq i \leq k$ to re-orient *M* for the next segment. Part of the *GC* is the S*tabilizer Tool* which is a data structure that is capable of providing the robots with target points around the boundary of *M* to move towards as well as the orientation of *M*.

The *Path Planner* (*PP*) is used to generate an efficient (e.g., shortest) path *P*. It makes use of a graph that is constructed from all of the collision-free configurations of *M*. *Collision-free configurations* are a subset of what is called *configuration space* (*C-Space*), that was first proposed by Lozano-Perez et al. in 1979 [14][15], where each configuration is a set of all the rotational poses of *M* at a specific reference point *C*. Note as well, in Figure 1, that each robot has its own set of behaviors, which includes taxiing, avoiding collisions, pushing, rotating, repositioning, reorienting and realigning; each of which is activated based on certain conditions. The behavior design of the robots is based on the subsumption architecture model proposed by Brooks in 1986 [11].

II. Previous Work

The work presented here combines the area of computational geometry motion planning algorithms with multi-robots systems. The subject of path planning has been extensively researched, but the work most applicable to ours is that of Lozano-Perez et al. in 1979 [14]. They describe a collision avoidance algorithm for planning a safe path for a polygonal object moving among known polyhedral objects that are considered forbidden regions. Later Lozano-Perez [15] extended this idea and defined what is known as *Configuration Space* (C-Space). It was shown by Reif [25] that computing C-Space is PSPACE-hard, which implies NP-hard. The main problem is that the dimension of C-Space is unbounded. An excellent chapter by LaValle [13] on C-Space explains this notion in more detail. The work by Lozano-Perez et al. triggered additional work by Brooks et al. [16][17] to find the shortest path from a start position of an object to a goal point using C-Space. In 1982, Lozano-Perez and Brooks joined efforts to implement an algorithm based on C-Space for polygonal obstacles and a moving object with two degrees of freedom. Efforts in finding an accurate algorithm for C_{free} started with Schwartz et al. [18], who proposed to decompose C_{free} into a collection of non-overlapping cells and to represent cell connectivity using a graph. In 1988, Canny [19] proposed an algorithm that iteratively seeks a low-dimensional retraction of C_{free} by employing the techniques from differential topology. This algorithm was more efficient and less complex than that of Schwartz et al. A more recent paper by De Berg et al. [20] on computing push plans for an object by robots, makes use of C-Space to find the shortest path.

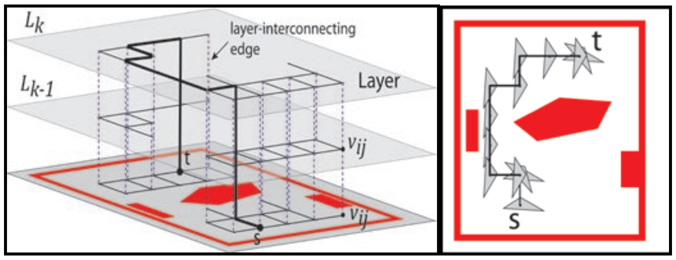
Several papers have been written, and simulations implemented, to study the underlying behavior of social insects, multi-robot and single-robot systems [11][12][21]. Social insects such as ants can transport a prey to their nest with minimal interaction with each other or the environment [23]. They self-organize to accomplish such tasks where their behavior is not directed by a central controller. A study done by Deneubourg et al. [8] investigating the transport of a worm in the ant Formica polyctena, shows that the transport became suddenly successful after a number of unsuccessful attempts. Deneubourg et al. attribute the outcome to the forces applied cooperatively by the ants which finally aligned and caused the worm to be transported in the right direction [23]. Bonabeau et al. [12] presented a decentralized system to solve cooperative transportation by a group of robots that act like ants. Chen et al. [3] used a swarm intelligence model where self-organized systems of homogeneous robots were built around simple behaviors, obtaining a decentralized and intelligent global behavior. Wang et al. [22] proposed a decentralized system where the object is surrounded with series of robots and the position of the object is controlled by the position of each robot that encloses and pushes the object.

The time overhead required for repositioning and realignment can be reduced by introducing the division of labor or what is called task allocation. Kube et al. [24] have devised a mechanism for repositioning and realignment inspired by social insects where the robots reposition randomly around a box. Our design is also similar to that of Lewis et al.[6] where the concept of a virtual structure is introduced. Using the virtual structure approach, a general control strategy was developed to force an ensemble of robots to behave as if they were particles embedded in a rigid structure. One of the most studied implementations is that of Liu et al. [7], where pusher robots are tightly coupled and a virtual robot is instructed to help the robots to keep a static distance and orientation when pushing.

Our work differs in a four main ways from the previous work just described. First, our work allows arbitrary convex polygons to be pushed, not just rectangular boxes [1][2][3][5][6][7][22][25]. Second, our work allows both translations and rotations of *M*, resulting in a more flexible and capable solution. Third, the objective of our work is to maintain accurate motion as *M* is being pushed. Hence we focus on monitoring the progress of the task to ensure that the overall solution provides smooth, rigorous motion of *M* throughout the task at hand by adjusting the robot-pushing strategy so that the robots "get back on track" when *M* begins to deviate from the desired path. Lastly, we provide some numerical results representing the deviation of *M* from the path *P* which can be used to compare the solution to future work.

III. Planning the object's path

The path planning algorithm presented in our approach is comprised of two main sub-sections: graph construction and path construction. First, multiple two-dimensional *grid subgraphs* are constructed where each graph denotes a *layer* (see Figure 2*a*). These layered subgraphs are interconnected with edges to create a final graph *G*. Within a layer, the vertices represent a translation of *M* about its center *C*, where *M* has a fixed orientation (i.e., rotation). Each successive layer represents a unique orientation of *M*. Once *G* is constructed and its edges are assigned weights, Dijkstra's algorithm is used to generate a shortest path approximation for *M*'s trajectory. Figure 2*a* shows a graph with only three layers. *M* starts at the bottom layer and then traverses *G* until it reaches the destination point. As *M* moves along *P*, it traverses edges in the



(a) (b)

Figure 2: (a) Graph with only three layers.(b) A fully generated P.

graph G. Travel along edges within the same layer of G represents a translation of M while travelling along edges of G that interconnect two layers represents a rotation of M about its reference point C. Figure 2*b* shows how M moves and rotates along P as the graph G is traversed.

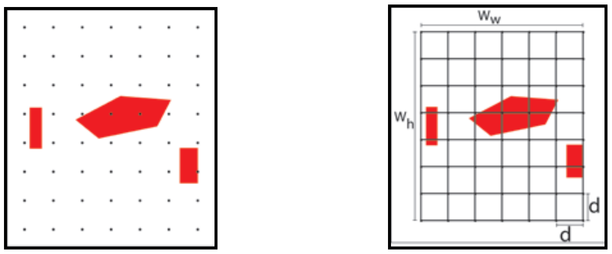
A. Graph and Path Construction

Let L_i, $1 \le i \le l$, be a grid graph with m×n vertices V_L and edges E_L such that $\{\forall\ u,v \in V_L,\ e \in E_L \mid u \neq v,\ |e|=d\}$ where d is an arbitrary positive non-zero constant length, $u=(x_u,y_u)$ and $v=(x_v,y_v)$ are adjacent and connected with an edge e iff $|uv|=d$ and $(x_u=x_v$ and $y_u \neq y_v)$ or $(x_u \neq x_v$ and $y_u = y_v)$. Let W_w and W_h be respectively the width and height of W where $W_w = d(m\text{-}1)$ and $W_h = d(n\text{-}1)$.

Figure 3*a* shows an upright square lattice of $m \times n$ vertices in W where each point represents the center point C of M and is positioned at equal distance d from vertically and horizontally adjacent vertices. The top-left vertex is located at the coordinate *(0,0)* relative to W. The initial step in constructing graph L_i is shown in Figure 3*b*, where the vertices are connected with edges. Note the length d of each edge in L_i and the width and height of W. Subsequently, consider translating M to each vertex in L_i. It is possible that some of the translated copies of M might overlap obstacles in W, as shown in Figure 4*a*. Such vertices are removed along with their incident edges to get the final form of the subgraph L_1 which would denote the first layer of G (see Figure 4*b*). The subsequent layers from L_2 to L_l are constructed in a similar manner to L_1 but with M rotated from layer to layer at a certain angle denoted by α_1 where $1 \le l \le 360$ (see Figure 4*b*-*c*). Moving up in the layers, the rotation of M increases such that $\alpha_1=0$ at L_1, $\alpha_2=a$ at L_2, $\alpha_3=2a$ at L_3, … , $\alpha_l=(l\text{-}1)a$ at L_l.

Now that the layers are constructed, the next step is to connect them with edges. Let I_k be the set of layer-interconnecting edges between layers $L_{(k\text{-}1)}$ and L_k where $1 < k \le 360/a$ and let vertex v_{ij} on L_k correspond to vertex v_{ij} on $L_{(k\text{-}1)}$ (see Figure 2*a*). Therefore, the vertices that join two adjacent layers have the same x and y coordinates in C-Space. If, because of collision with obstacles, vertex v_{ij} in layer $L_{(k\text{-}1)}$ or L_k is missing, then no interconnecting edge is added. The graph $G=\{L_1, L_2,\ldots, L_l\} \cup \{I_1, I_2,\ldots, I_l\}$ is therefore an undirected graph with at most lmn vertices and at most $l(2mn\text{–}m\text{–}n) + (l\text{-}1)mn$ edges.

The length of d could affect the possibility of finding a path P in G. Figure 4*b* shows that L_1 is disconnected where s lies in one component and t in another. Therefore, it is not possible to create a continuous path P from s to t in this single layer. This will force the algorithm to seek vertices in the adjacent layers of G (i.e., to require a rotation of M). However, by decreasing d, more configurations of M will be valid and a path P from s to t is more likely to be found.



(a) (b)

Figure 3: (a) Set of m×n vertices in W. (b) The grid graph.

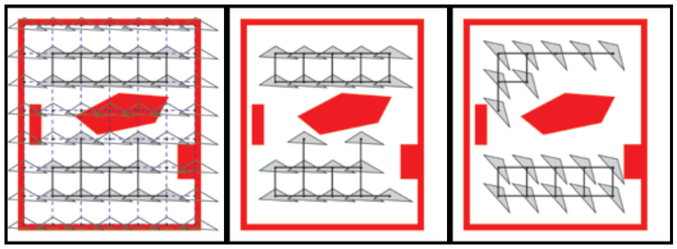
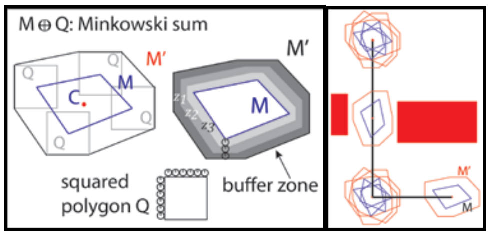

(a) (b) (c)

Figure 4: (a) Overlapped copies of M with the obstacles. (a) Removing the vertices along with their incident edges to get 1st layer. (c) Applying a rotation to M to construct the 2nd layer.

The length of d could affect the possibility of finding a path P in G. Figure 4*b* shows that L_1 is disconnected where s lies in one component and t in another. Therefore, it is not possible to create a continuous path P from s to t in this single layer. This will force the algorithm to seek vertices in the adjacent layers of G (i.e., to require a rotation of M). However, by decreasing d, more configurations of M will be valid and a path P from s to t is more likely to be found.

B. Extending The Algorithm to Accommodate Robot Bodies

We have shown how P is generated by the path planner. However, in order to maintain a rigorous balanced push by the robots, we need to expand the size of M such that the robots have enough space to maneuver around it. We expand M by an amount which is 6 times the diameter of the robots in order to leave enough space (i.e., defined as a *buffer zone*) so that the robots will stay away from M as much as possible while pushing or repositioning. The resulting expansion is denoted as M' (see Figure 5). It is this expanded shape M' which is used during the graph construction process to ensure that the translations and rotations of M can accommodate space for maneuverability. Once P has been generated, M' is no longer needed.



(a) (b)

Figure 5: (a) The buffer zone for M. Z_1,Z_2,Z_3 are areas in the buffer zone. (b) Growing M to M' where a rotation is required.

The buffer zone can be divided in three areas: (1) Area Z_1 is closer to the outer edges of the buffer zone. This area is a waiting zone for the robots when they are not needed. (2) Area Z_2 is a pathway for robots during a repositioning task. (3) Area Z_3 is a "stand clear" zone for robots in a repositioning task in order to avoid collisions with other robots.

IV. MULTI-ROBOT OBJECT PUSHING

As discussed, the *GC* uses the path planner to find a path *P* from *s* to *t*. In addition, the *GC* assigns multiple robots to push *M*. Each robot is able to push *M* by seeking a target point positioned at the boundary of *M*. As the robot reaches the target point, it collides with *M* causing it to move. The force applied to *M* by the robots while pushing could cause it to rotate or stray away from the original path. To avoid these issues, our algorithm creates and maintains a *stabilizer* and a *repositioning path*.

The path provided by the path planner is broken down into a set of *tasks* denoted by a task path list, *T*, that the robots are required to perform in order to push *M* along the path. A task has the following properties: task type, orientation of *M* and path segment. Three groups of tasks are used: “push”, “rotate” and “reposition” (see Figure 6). Define a *sub-task* to be a temporary task that has more priority than a task. When pushing *M*, three types of robots (i.e., denoted by robots specialties) are assigned to the task: “pushers”, “left adjusters” and “right adjusters”. Moreover, while pushing, the orientation and alignment of *M* could deviate from their true value. Therefore, two sub-tasks are used: an orientation sub-task and an alignment sub-task. In addition, when rotating *M*, left and right adjusters are used to perform the task where one group of robots push from the left side of *M* and another from the right side. The robots from both sides are positioned at the boundary of *M* based on the angle of the edge they need to push against to effectively perform an in-place rotation (see Figure 7*a*) .

While rotating, the robots unintentionally translate *M* from its center of rotation. Therefore, a push sub-task is used to translate *M* back to its original center of rotation. The reposition task is performed when the robots need to change positions in order to push in a different direction. The path monitor observes the progress of each individual task and upon task completion it informs the robots of their next task.

Consider a robot moving towards a target point provided by the stabilizer. A target point could be (1) a point p_i, $1 \leq i \leq k$, of path *P*, (2) a point on the boundary of *M* denoted as a *target-balance point* used for pushing or rotating *M* or (3) a point close to a target-balance point denoted as a *target-rest point*. A target-rest point keeps the robot away from *M* but at a close distance (i.e., *2.5d*, where d is the diameter of the robot) as a “standby” position (see Figure 7*b*), as well as away from other robots that are performing a repositioning task in order to avoid prolonged stagnation situations. The target-rest point falls in zone Z_3. The computation of these target points depends on the type of task to be performed.

The repositioning path is a path for the robots to follow while performing a repositioning task, where q_0 is the first point on the path such that angle $\square q_0\, p_1\, p_2 = 180^o$. Let $u_i\, u_{(i+1)}$ be a segment in an n-segment repositioning path, $1 \leq i < n$. Each segment $u_i u_{(i+1)}$ is then divided into $z = |u_i u_{(i+1)}| \,/\, d'$ equally-spaced sub-segments $q_{(i,j)} q_{(i,j+1)}$, $1 \leq j < z$, where d' is a constant.

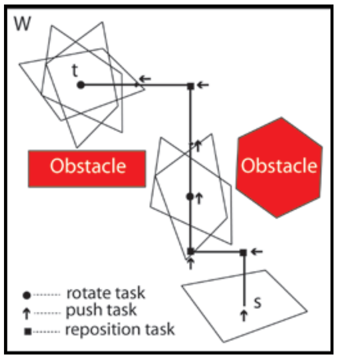
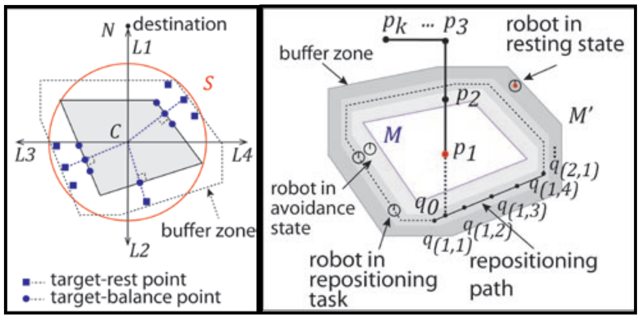


Figure 6: The path showing the various push, rotate and reposition tasks.



(*a*) (*b*)

Figure 7: (a)The stabilizer showing the target points and the buffer zone. (b) Repositioning path around M.

Figure 7*b* shows that the repositioning path falls in the Z_2 zone between the outer boundary of the buffer zone and the boundary of *M* to keep the robots away from *M* while repositioning. This allows enough space for a robot to pass beside a repositioning robot in between *M* and the repositioning path. If the robots touch *M* while repositioning they could cause it move unintentionally.

V. EXPERIMENTS

Our experiments were aimed at measuring the amount of deviation from the path as *M* was being pushed. We performed experiments with various shapes of M as well as with various numbers of robots, ranging from 3 to 48 robots divided into teams of three. We also performed experiments to investigate how data communication load and runtime change as more robots are assigned to the task. For the sake of brevity, we present here results from just one shape of *M*. See the work of Chamoun [4] for further experimental data.

The deviation distance is measured in pixels but can be converted into any appropriate units (e.g., cm, mm, inches). In addition, the units can be chosen with respect to the diameter of the robots or the size of *M*. Although the robots are simulated as separate threads, the time measurements for our experiments are based on the synchronous time-steps of the *GC* where each time-step represents a unit of time with respect to an iteration loop of the *GC*. Our experiments were performed in Java using Eclipse and has been tested in the following environment: Windows 7 Enterprise 64-bit, Intel ® Core ™ i7 CPU, Q820 @ 1.73GHz, 16GB RAM.

A. Convex Polygon Experiment

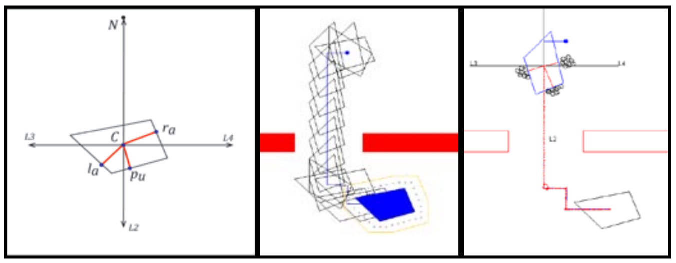
The shape of *M* for this experiment is shown in Fig. 8a. The target-balance points are calculated based on the angle of the edges that intersect with the stabilizer axis. In this experiment, the diameter of *M* was calculated as 120 pixels, which is the diameter of the smallest enclosing circle of *M*.

The charts in Figure 9 show the deviation distance during rotating and pushing tasks for a varying number of robots. Notice that when less robots are used, the deviation remains quite small (i.e., at most 2% of *M* 's diameter when pushing and around 9% during rotations). When 48 robots were used, the deviation increased to up to 19%. Such an increase was a recurring observation in our experiments since the interference between the robots and *M* becomes higher as more robots are used, which actually causes *M* to stray away from the path multiple times. This can be verified by examining Table 2, which shows the repositioning collisions during this particular experiment. This experiment actually produced the most deviation, when compared to the others experiments which typically ranged from about 9% to 15% deviation, with respect to the diameter of *M*.

Table 2 shows the average amount of time that each behavior was active during the experiment. Notice that as the number of robots increase, the amount of time spent pushing generally decreases, due to the increase in re-orientation, re-positioning and waiting behaviors, although the average time spent in each behavior is somewhat consistent.

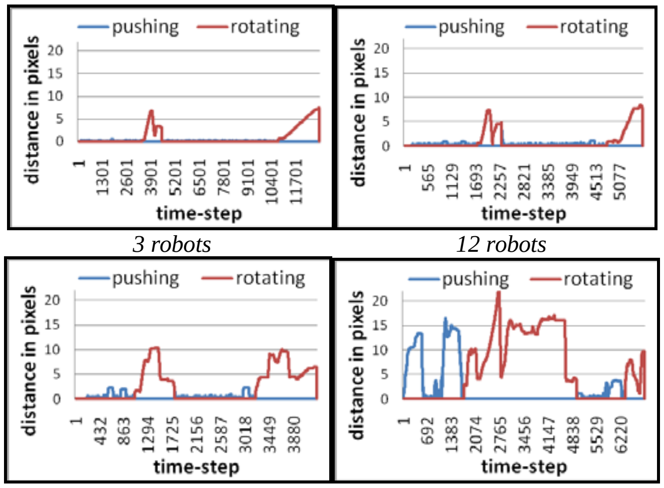
B. Data Communication Load

Throughout the task, the robots communicate with the GC to obtain updates regarding the position they should head towards, repositioning path coordinates and orientation of *M*. In our experiments, we assumed that the communication bandwidth between the GC and robots has no limit. However, we were interested in determining the data communication load between the robots and the GC. Table 3 shows the average number of bytes sent and received per robot and the number of bytes sent and received per robot per time step during the task. As the number of robots increase, the data sent/received per robot decreases (i.e., robots communicate less overall) because the task is completed sooner. A better indicating of the data communication load must take into account the completion time of the task. Hence, the table also shows the average amount of communication per robot per time-step (e.g., per millisecond).



(a) (b) (c)

Figure 8: (a)The shape used for M, showing target-balance points being non-perpendicular to edges of M. (b) Translations of M along P. (c) Trace along P while M was being pushed by 24 robots.



3 robots 12 robots

24 robots 48 robots

Figure 9: Deviation distance per time-step during the experiment.

The average number of bytes per time-step is calculated by multiplying the average amount of bytes sent and received from all the behaviors by the number of robots and then dividing that by the number of time-steps for the simulation. Notice that the average number of sent/received bytes per robot per time-step decreases as more robots are used. This occurs because during repositioning, some robots might exchange more data with the *GC* than other robots since it depends on their distance from the next target point they are required to reach. Also, the number of points to follow on the repositioning path could vary from one robot to another, hence when averaged, the amount of data being exchanged per robot is smaller.

Table 1: Average amount of data sent & received (in bytes) between the robots and the GC.

Bytes Sent/Received	**3 robots**	**12 robots**	**24 robots**	**48 robots**
per robot	659319	244338.05	168266.78	114746.32
per robot per time-step	41.33	38.74	34.28	30.41

Table 2: Number of collisions during the repositioning tasks.

Number of collisions	**3 robots**	**12 robots**	**24 robots**	**48 robots**
Robot-to-Robot	19.00	417.00	860.71	2516.88
Robot-to-M	11.67	12.83	31.50	191.94

Table 3: Average amount of time a behavior was active, for various team sizes during the experiment.

Behaviors	**3 robots**	**12 robots**	**24 robots**	**48 robots**
Pushing	25.62%	26.20%	24.64%	22.64%
Rotating	15.35%	12.61%	15.50%	10.39%
Realignment	25.62%	26.20%	24.64%	23.61%
Reorientation	0.00%	0.82%	0.76%	7.51%
Repositioning	0.06%	0.15%	0.38%	0.97%
Waiting	33.34%	34.02%	34.07%	34.88%

I. CONCLUSION

In this work, we presented a strategy for pushing, with rigor, a convex polygonal object on a path using multiple robots in a *2D* environment with polygonal obstacles. Our strategy is based on previous research in three areas: path planning and configuration space as described by Lozano-Perez [14][15], the design of behavioral models as described by Brooks [10][11], and cooperative transport and swarm intelligence as described by Bonabeau et al. [12][23].

When compared to previous work, our work differs in that we allow the pushing of arbitrary convex polygons as opposed to just simple rectangular boxes. Also, our work makes use of computational geometry techniques to compute an efficient rectilinear path on which to push the object, which allows both translations and rotations of the object. Our work differs as well in that we aim to push the object rigorously, attempting to minimized the amount of deviation from the path throughout the task. To our knowledge, previous work has not focused on transporting objects using rigorous movement.

Through experimentation, we were able to verify that our algorithm was successful in keeping the deviation from the path below 20% of the diameter of the object being pushed. We believe that this can be improved further by modifying the algorithm so that it defines additional target-balance points along the perimeter of the object being pushed instead of confining them to three unique groups.

Regarding future work, we are considering alterations to the algorithm that allow robots to switch specialties on-the-fly, likely allowing the task to be performed with just one or two robots that perform multiple tasks in sequence, such as pushing, adjusting, and rotating. This could reduce the repositioning requirements of the robots and as a result decrease the number of unnecessary collisions with the object being pushed. Also, although our algorithm has been implemented for convex polygons, we believe that it can be easily extended to handle curved-shaped objects because the stabilizer design is flexible enough to work with any object shape requiring simply the computations of tangents to the curved edges. Lastly, in regards to data load, an improvement can be made by implementing a mechanism to enable robots within the same team to share the same information so as to reduce the data communication between the robots and the *GC*.

REFERENCES

[1] Trojanek, P., Szynkiewicz, W., and Zieliński, C., “Definition and composition of individual robot behaviours in cooperative box pushing,” In Proceedings of the 13th IEEE IFAC International Conference on Methods and Models in Automation and Robotics. Technical University of Szczecin, pp. 29-30, 2007.

[2] Parra González, E.F., Ramírez-torres, J., and Toscano-Pulid, G., “A New Object Path Planner for the Box Pushing Problem,” Electronics, Robotics and Automotive Mechanics Conference, pp. 119-124, 2009.

[3] Chen, X., and Li, Y., “ Modeling and simulation of a swarm of robots for box-pushing task,” 12th Mediterranean Conference on Control and Automation, Kusadasi, Aydin, Turkey, 2004.

[4] Chamoun, P., “Rigorous Movement of Convex Polygons on a Path Using Multiple Robots,” Master's Thesis, School of Computer Science, Carleton University, Ottawa, Canada, 2012.

[5] Yamada, S., and Saito, J., “Adaptive action selection without explicit communication for multi-robot box-pushing,” Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 31(3). pp. 398–404, 2001.

[6] Lewis, M.A., and Tan, K.H., “High Precision Formation Control of Mobile Robots Using Virtual Structures,” Journal Autonomous Robots archive, vol 4 Issue 4, 1997.

[7] Liu, S., Liu, F. and Tang, F., “Cooperative transport strategy for formation control of multiple mobile robots,” Journal of Zhejiang University, Science C, vol 11, pp. 1-13, 2010.

[8] Deneubourg, J.L., Dorigo, M., and Labella, T.H., “Self-Organised Task Allocation in a Group of Robots,” In Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems, Tokyo, Japan, 2004.

[9] Karigiannis, J.N., Rekatsinas, T.I., and Tzafestas, C.S., “Fuzzy Rule Based Neuro-Dynamic Programming for Mobile Robot Skill Acquisition on the basis of a Nested Multi-Agent Architecture,” 2010 IEEE International Conference on Robotics and Biomimetics – RoBio, 2010.

[10] Brooks, R.A., “A robot that walks: Emergent behaviors from a carefully evolved network,” In Beer, R., et. al. (eds.), Biological Neural Networks in Invertebrate Neuroethology and Robotics, Academic Press, 1993.

[11] Brooks, R.A., “A Robust Layered Control System for a Mobile Robot,” IEEE Journal of Robotics and Automation, vol 1 RA-2, pp. 14-23, 1986.

[12] Bonabeau, E, and Kube, C., “Cooperative transport by ants and robots,” Journal of Robotics and Autonomous System, vol 30, pp. 85–101, 2000.

[13] LaValle, S.M., “Planning Algorithms,” University of Illinois, “The Configuration Space,” Chapter 4, 2006.

[14] Lozano-Perez, T., and Wesley, M.A., “An algorithm for planning collision-free paths among polyhedral obstacles,” Communication of ACM, vol 22, pp. 560-570, 1979.

[15] Lozano-Perez, T., “Spatial Planning: A Configuration Space Approach,” IEEE Transactions on Computers, vol C-32-2, pp. 108-120, 1983.

[16] Brooks, R.A., “Solving the Find-Path problem by representing free space as generalized cones,” M.I.T. Artificial Intelligence Lab., Rep. AIM-674, 1982.

[17] Brooks, R.A. and Lozano-Perez, T., “A subdivision algorithm in configuration space for Find path with rotation,” M.I.T. Artificial Intelligence. Lab., Rep. AIM-684, 1982.

[18] Schwartz, J.T., and Sharir, M., “On the piano movers II. General techniques for computing topological properties on real algebraic manifolds,” Adv. in Applied Mathematics, vol 4, pp. 298-351, 1983.

[19] Canny, J.F., “The complexity of robot motion planning,” Institute of Technology Cambridge, Massachusetts, MIT Press, 1988.

[20] De Berg, M., and Gerrits, D.H.P., “Computing push plans for disk-shaped robots,” In Proceedings of 2010 IEEE International Conference on Robotics and Automation , pp. 4487-4492, 2010.

[21] Kube, C. R., and Zhang, H., “Collective Robotics: From Social Insects to Robots,” Adaptive Behavior, vol 2, 189-218, 1993.

[22] Wang, Z., and Kumar, V., “Object closure and manipulation by multiple cooperating mobile robots,” In Proceedings of IEEE International Conference on Robotics and Automation, vol 1, pp. 394–399, 2002.

[23] Bonabeau, E., Dorigo, M., Guy Theraulaz, G., “Swarm Intelligence: From Natural to Artificial Systems,” Oxford, 1999.

[24] Kube, R.C., and Zhang, H., “Stagnation Recovery Behaviors for Collective Robotics,” In Proceedings 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, Los Alamitos, pp. 1883-1890, 1995.

[25] Reif, J. H., “Complexity of the mover’s problem and generalizations,” Annual IEEE Symposium on Foundations of Computer Science, pp. 421–427, 1979