Adaptive Autonomous Navigation of Multiple Optoelectronic Microrobots in Dynamic Environments

Laurent Mennillo, Christopher Bendkowski, Mohamed Elsayed, Harrison Edwards, Shuailong Zhang, Vijay Pawar, Aaron R. Wheeler, Danail Stoyanov, *Senior Member, IEEE*, and Michael Shaw

***Abstract*—The optoelectronic microrobot is an advanced light-controlled micromanipulation technology which has particular promise for collecting and transporting sensitive microscopic objects such as biological cells. However, wider application of the technology is currently limited by a reliance on manual control and a lack of methods for simultaneous manipulation of multiple microrobotic actuators. In this letter, we present a computational framework for autonomous navigation of multiple optoelectronic microrobots in dynamic environments. Combining closed-loop visual-servoing, SLAM, real-time visual detection of microrobots and obstacles, dynamic path-finding and adaptive motion behaviors, this approach allows microrobots to avoid static and moving obstacles, and perform a range of tasks in real-world dynamic environments. The capabilities of the system are demonstrated through micromanipulation experiments in simulation and in real conditions using a custom built optoelectronic tweezer system.**

***Index Terms*—Micro/Nano robots, multi-robot systems, autonomous agents, visual servoing, motion and path planning.**

Manuscript received 24 February 2022; accepted 9 July 2022. Date of publication 27 July 2022; date of current version 25 August 2022. This letter was recommended for publication by Associate Editor D. Cappelleri and Editor X. Liu upon evaluation of the reviewers' comments. This work was supported in part by the U.K. Economic and Social Research Council under Grant ES/T011866/1 and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), under Grant ALLRP 548593-19. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any Author Accepted Manuscript version arising. *(Corresponding author: Laurent Mennillo.)*

Laurent Mennillo, Christopher Bendkowski, and Vijay Pawar are with University College London, WC1E 6BT London, U.K. (e-mail: l.mennillo@ucl.ac.uk; christopher.bendkowski.18@ucl.ac.uk; v.pawar@ucl.ac.uk).

Mohamed Elsayed, Harrison Edwards, and Aaron R. Wheeler are with the University of Toronto, Toronto, ON M5S 3H6, Canada (e-mail: mohammed.elsayed@mail.utoronto.ca; harrison.edwards@mail.utoronto.ca; aaron.wheeler@utoronto.ca).

Shuailong Zhang is with the School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 100081, China, and also with the Beijing Advanced Innovation Center for Intelligent Robots and Systems, Beijing Institute of Technology, Beijing 100081, China (e-mail: shuailong.zhang@bit.edu.cn).

Danail Stoyanov is with the Wellcome/EPSRC Centre for Interventional and Surgical Sciences (WEISS), University College London, W1W 7TS London, U.K. (e-mail: danail.stoyanov@ucl.ac.uk).

Michael Shaw is with University College London, WC1E 6BT London, U.K., and also with the National Physical Laboratory, TW11 0LW Teddington, U.K. (e-mail: mike.shaw@ucl.ac.uk).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3194308, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3194308

I. Introduction

Of THE variety of mobile microrobot technologies which have been proposed [1], light-driven approaches are particularly attractive for many applications due to their flexibility and precision. The Optoelectronic Microrobot (OEM) [2] is a recently proposed micromanipulation technology based on the application of optoelectronic tweezers (OET) to control microscopic actuators. OEMs offer several advantages over alternative light-based micromanipulation techniques, such as optical tweezers [3] and direct OET [4], particularly in the range of forces which can be exerted on target objects and their compatibility with sensitive biological samples such as live cells. Recent work [5] has shown that multiple OEMs can be combined to create microscopic motors and machines such as gear trains and valves. OEMs are typically controlled by a human operator who manually positions individual actuators using visual feedback. This limits their application to sequential, low throughput tasks. In this work, we describe a new adaptive approach to OEM control which enables real-time, closed-loop positioning of multiple actuators. Comprising a set of linked asynchronous software modules for detection of OEMs and obstacles, simultaneous localization and mapping (SLAM), task allocation, path-finding and adaptive motion behaviors, this system allows simultaneous autonomous manipulation of multiple microscopic objects in dynamic environments. The performance of the system for micromanipulation tasks is tested in simulation and under controlled experimental conditions using a custom-built OET system. We discuss the potential for wider applications of OEMs enabled by our work.

II. Related Work

OET [4] relies on the application of light-controlled dielectrophoresis (DEP) to exert forces on microscopic objects suspended in solution between a pair of transparent parallel plate electrodes. When an external voltage is applied across the electrodes, the local electric field gradient between them can be controlled by projecting light patterns onto a photoconductive (PC) layer coated on top of the lower electrode. This in turn allows DEP forces to be exerted on microscopic objects suspended in the electrode gap, enabling micromanipulation using irradiance levels (typically $\sim 1\ \mathrm{Wcm}^{-2}$) which are several orders of magnitude lower than conventional optical

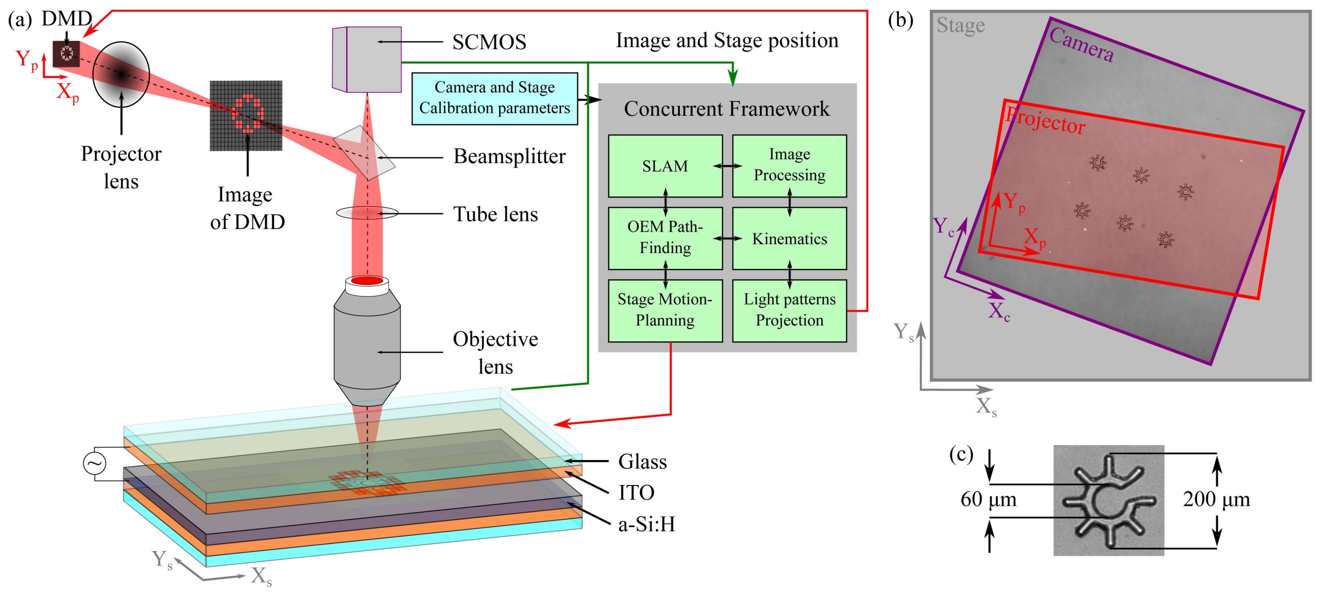

Fig. 1. System architecture. (a) Schematic of the OET system and software framework for OEM manipulation. (b) Reference frames for the sample positioning stage (grey), camera (magenta), and projector (red). (c) Brightfield image of a cogwheel-shaped OEM.

tweezers. The OEM [2] builds on this concept by using OET to control microrobotic actuators fabricated in bulk using SU-8 photolithography (Fig. 1(a) and (c)). By decoupling the magnitude of the DEP force from the size, shape, and composition of the manipulation target, OEMs can be used to exert larger forces (typically 100s of pN) on a wide range of secondary microscopic objects. Highly consistent OEM manufacture gives them a reproducible and predictable response to DEP forces, simplifying control. Results also suggest that the direct mechanical forces applied by OEMs are less damaging to sensitive cargoes, such as biological cells, than direct OET [2]. The wider application of OEM technology depends critically on increasing the flexibility and achievable throughput of micromanipulation tasks. To this end, we previously demonstrated automated open-loop control of multiple OEMs using computer vision and multi-agent path-planning [6]. Using this approach, OEMs were able to perform automated collect, transport, and release operations on microscopic particles within a lab-based OET microscope system. However, this open-loop approach relied on OEMs following fixed paths, identified during system initialization. As a result, it was unable to correct for any deviation from ideal OEM behavior or accommodate for moving obstacles and targets, limiting its application to well-characterized static environments. Here, we build on this previous work, presenting a new adaptive closed-loop method for the execution of OEM micromanipulation tasks. Using visual-servoing, dynamic path-finding and adaptive motion behaviors, this approach enables autonomous navigation of OEMs in real-world complex, dynamic environments.

III. Methods

A. Overview

Our OEM control approach is designed to allow adaptive, autonomous navigation of OEMs in dynamic environments through the use of real-time, closed-loop visual-servoing. It combines spatial calibration of system hardware, SLAM, object detection and tracking, task assignment, path-finding, and steering of physical and virtual objects. These methods are realized using a set of discrete modules working together within an asynchronous concurrent framework. Following off-line system calibration, images acquired by the camera are fed to a SLAM module along with the associated coordinates of the motorized sample stage to produce an up-to-date map of the observed device. An image processing module detects the objects present in the images (OEMs, static and moving obstacles) and labels the map. This labeled map is then used by a path-finding module which computes independent OEM trajectories. A motion-planning module positions the sample stage such that the sum of navigable trajectory path lengths within the projector image field is maximized. Finally, a kinematics module moves the detected OEMs along their trajectories by computing an updated image for display on the light projector.

B. Hardware Characteristics of the System

To control OEMs, we developed an OET system (Fig. 1(a)) by modifying an upright light microscope with a 4x/0.16 microscope objective lens, as described in our previous work [6]. A commercial digital light projector was coupled into the illumination path to project light patterns onto the PC layer of an OET device fabricated by coating glass microscope slides with indium tin oxide (ITO). The electric field (30 kHz, 30 V_{PP}) across the $\sim 150\ \mu\mathrm{m}$ gap between the electrodes was controlled using a function generator. Brightfield images were captured under transmitted light illumination with a blue LED using a scientific CMOS camera. A shortpass optical filter in the emission path of the microscope removed the specular reflection of the red light from the projector. The position of the sample was controlled using a motorized translation stage. Cogwheel-shaped OEMs (Fig. 1(c)), fabricated from SU-8 using photolithography were suspended in deionized water and added on top of the (lower) electrode prior to attachment of the upper electrode. The OET hardware has two particularly important characteristics that have influenced the design of our OEM control solution. Firstly, the camera and projector share the same optical train within the microscope (tube and objective lens). This means that both the observable area of the OET device substrate and the size of the image field from the projector are limited to a small fraction of the total device area. This field-of-view limitation is accommodated by jointly performing SLAM following system calibration and efficiently moving the stage using the computed map, to allow seamless microrobotic control over large distances. Secondly, despite their consistent manufacture, OEMs can exhibit inconsistent behavior due to *in situ* contamination and variability of the frictional forces with the PC substrate. A small axial (z) component of the DEP force can also cause OEMs to hover slightly above the PC layer, making them sensitive to the microfluidic forces within the device. The positions of targets and obstacles within the device can also be affected by fluid flows or by the self-propelled motion of live objects. As a result, closed-loop visual-servoing is employed for real-time OEM control, allowing for failure detection and recovery.

C. System Calibration

Spatial calibration of the camera, sample positioning stage and projector are essential to ensure accurate SLAM over large distances and allow precise light projection onto the computed device map (Fig. 1(b)). Firstly, the camera is calibrated to correct image distortion and measure the image pixel size. This is followed by a stage calibration procedure to determine the transformation matrix expressing relative translations of the stage in the camera reference frame. Finally, the projector is calibrated to allow direct mapping from camera to projector pixels.

1) Camera Calibration: A large number of photogrammetric and vision-based camera calibration techniques have been developed over the years, most notably [7], [8]. Whilst similar, microscope calibration presents some important differences due to the shallow depth-of-field, such as the use of a specific camera model, calibration patterns that are near parallel to the image plane and the restriction to single plane calibration [9]. We used the method proposed by Amni et al. [10], which is based on a modified version of Zhang's algorithm [8] for the parallel case, to correct geometric image distortion. This approach retrieves both the intrinsic and extrinsic camera parameters along with the distortion coefficients from a single calibration image. In place of the virtual target constructed in [10], a physical calibration target (R2L2S3P1, Thorlabs) is used. The real size $\mathtt{S}_\mathtt{p}$ of the image pixels, measured at $\sim 1.62\,\mu\mathrm{m}$, is computed by averaging the distance in pixels between each undistorted image point over the known distance between their corresponding 3-D points on the calibration pattern plane.

2) Stage Calibration: To describe the relationship between a translation of the sample positioning stage and the observed translation in the calibrated camera image, let $R_s = (s; \mathbf{X}_s, \mathbf{Y}_s)$ be the stage reference frame on the same plane as the image reference frame $R_c = (c; \mathbf{u}, \mathbf{v})$. The transformation from a stage translation vector $\overline{\mathbf{t}}_s = [T_{\mathbf{X}s}, T_{\mathbf{Y}s}, 1]^T$ to the translation vector between two undistorted image positions $\overline{\mathbf{t}}_i = [T_\mathbf{u} \cdot \mathtt{S}_\mathtt{p}, T_\mathbf{v} \cdot \mathtt{S}_\mathtt{p}, 1]^T$ is $\overline{\mathbf{t}}_i = \overline{A\,\overline{\mathbf{t}}_s}$, where A is a 2-D transformation in homogeneous form with scaling factor s and rotation angle θ. To retrieve this transformation matrix, several images are acquired following known translations of the stage and visually correlated to determine the corresponding shifts in the camera frame. This procedure is performed using ZNCC, with sub-pixel accuracy achieved by fitting and retrieving the maximum of a cubic 2-D polynomial over the square region of correlation values centred on the coordinates of the highest correlation. These N stage and image translations in homogeneous form are then used to retrieve the orientation and scaling parameters $\Omega = \{s, \theta\}$ by solving the non-linear least squares minimization $J = \min_\Omega \sum_{n=0}^{N} [(\overline{A\,\overline{\mathbf{t}}_s}) - \overline{\mathbf{t}}_i]^2$. Once A is estimated, the relative image coordinates following a stage translation $\mathbf{t}_s$ are retrieved with $\overline{p}_i^1 = \overline{p}_i^0 + (\overline{A\,\overline{\mathbf{t}}_s})$, with $\overline{p}_i^0$ and $\overline{p}_i^1$ the image coordinates in $\mu\mathrm{m}$ before and after the stage translation, respectively. Similarly, the transformation matrix A is also used to retrieve the relative stage coordinates following an image translation.

3) Projector Calibration: The projector image frame is mapped to the camera reference frame by sequentially projecting light from small groups of DMD elements onto a sample and detecting them in the undistorted camera image. Interpolating between these measured points creates a full mapping of the projector image.

D. Simultaneous Localization And Mapping

Following hardware calibration, SLAM is performed using the camera and stage calibration parameters on undistorted camera images. While visual SLAM was considered to avoid relying on the stage's proprioceptive odometry, the high potential of observing only moving objects and a lack of static visual features on the devices made this approach unreliable. Instead, direct localization and mapping are performed by setting an origin point on the map, tied to the camera and stage reference frames, and operating from relative translations of either device. The displacement of the camera field-of-view from the stage proprioceptive coordinates is computed using $\overline{p}_i^1 = \overline{p}_i^0 + (\overline{A\,\overline{\mathbf{t}}_s})$ for direct stitching of the observed images, with the resulting map expressed in the camera reference frame.

E. Image Processing

Static and moving structures, such as the edge of the OET device, etched static structures, debris and OEMs are detected and labeled using classical image processing techniques, mostly implemented in the OpenCV library [11]. The map labeling procedure is used to detect obstacles in each brightfield camera image and produce a binary map representing the pixels occupied by each structure. A separate detector is used to identify the 2-D position and 1-D orientation of each OEM. The subsequent labeled map $\mathbb{M}$, along with all OEM poses, then acts as an abstraction layer for use by the other modules of the system.

1) Image Labeling: Brightfield images are smoothed using a bilateral filter to preserve sharp contrast changes, before a Laplacian filter is used to compute intensity gradients. The gradient image is thresholded to create a binary image, before

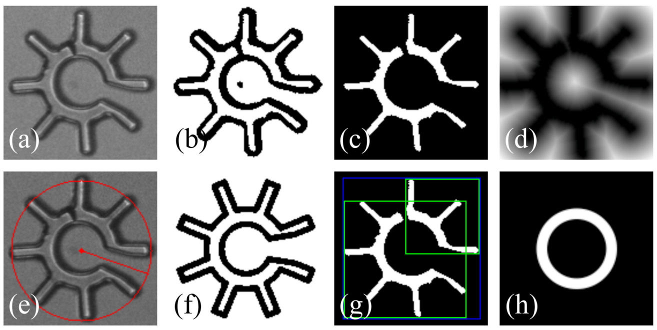

Fig. 2. OEM detection ((a), (b), (c), (e), and (g) are crops of full-resolution images). (a) Brightfield image of a cogwheel OEM. (b) Thresholded image $\mathcal{T}_t$ of (a). (c) Image of blobs $\mathcal{B}$ from (b). (d) Distance transform of $\mathcal{B}$. (e) Detected OEM. (f) Predefined template image $\mathcal{T}_m$. (g) Blobs bounding boxes (in green) in cluster $\mathbb{C}$ and valid combination $\mathcal{C}_i$ (in blue). (h) Pre-defined circular region corresponding to the circular chamber of the OEM.

a morphological opening is applied to discard small connected components.

2) OEM Detection: We developed a new OEM detector, which improves on our previous work [6] allowing robust real-time detection of both touching and partially damaged OEMs in brightfield images (Fig. 2(e)). A pre-detection operation is first performed to find the combinations of distinct connected components satisfying several OEM shape characteristics. Raw camera images (Fig. 2(a)) are smoothed using a bilateral filter, followed by adaptive thresholding with a Gaussian kernel. The resulting binary image $\mathcal{T}_t$ (Fig. 2(b)) is morphologically opened to detach all loosely connected structures before all connected components significantly smaller or larger than the expected area of an OEM m_a are removed to produce an image of blobs $\mathcal{B}$ (Fig. 2(c)). Each blob $b \in \mathbb{B}$ is then assigned to a cluster $\mathbb{C} \subseteq \mathbb{B}$, in which all blobs have a neighbor whose L_2 intra-centroid distance is less than half the expected diameter of an OEM m_s, so that $\forall\ b_x \subsetneq \mathbb{C}, \exists\ b_y \neq b_x,\ b_y \subsetneq \mathbb{C},\ d(b_x, b_y) < \frac{m_s}{2}$. In the case of an isolated blob b, $\mathbb{C} = \{b\}$. All combinations $\mathcal{C}_i,\ i \in \{1,\ \ldots \sum_{k=1}^{n} \binom{n}{k}\}$ of $k \leqslant n$ blobs in each cluster $\mathbb{C}$, with $card(\mathbb{C}) = n$, are then evaluated so that their combined area $m_a^- \leqslant area(\mathcal{C}_i) \leqslant m_a^+$, and the maximum side length of their combined bounding boxes satisfies $m_s^- \leqslant size(\mathcal{C}_i) \leqslant m_s^+$, with $m_a^\pm$ and $m_s^\pm$ arbitrary factored values of m_a and m_s, respectively (Fig. 2(g)). Finally, the valid combinations with the same combined bounding box are removed to eliminate duplicates. While this approach becomes prohibitively expensive as clusters grow larger, clusters rarely exceeded five connected components during our experiments in real-world conditions. The detection phase then uses a square template image $\mathcal{T}_m$ (Fig. 2(f)) derived from the CAD model of the OEMs, slightly modified to resemble their appearance in the binary image $\mathcal{T}_t$. A template matching method using the APT image feature descriptor proposed in [12] is used to identify OEMs in the image. APT vectors are simple 1-D arrays of mean image intensity values in concentric regions of circular annuli that have the same area. Their rotational invariance and low computational complexity make them particularly well-suited to real-time detection of OEMs. A single APT vector $\mathtt{APT}_m$ with maximum radius $r = \frac{size(\mathcal{T}_m)}{2}$ is computed on the square OEM template image $\mathcal{T}_m$. For each valid combination $\mathcal{C}_i$ of each cluster $\mathbb{C}$, APT vectors $\mathtt{APT}_i$ of the same radius are computed following a sliding-window approach on the region covering the centroid of the combined bounding box of $\mathcal{C}_i$ in image $\mathcal{T}_t$. $\mathtt{APT}_i$ are then compared to $\mathtt{APT}_m$ by computing their mean-squared error, and the coordinates of the minimal MSE value in the region satisfying $\mathrm{MSE}(\mathtt{APT}_i, \mathtt{APT}_m) \leqslant t_{\mathtt{APT}_d}$ is considered as an OEM candidate location, with $t_{\mathtt{APT}_d}$ an arbitrary threshold value. Finally, all OEM candidates are filtered by eliminating other surrounding candidates with a larger MSE value. Next, the orientation of each OEM candidate is measured by detecting the location of its aperture, which is estimated as the highest value of the distance transform computed on a pre-defined circular region (Fig. 2(d) and (h)) of the OEM bounding box in the image of blobs $\mathcal{B}$.

3) OEM Tracking: Each detected OEM is linked to its observation in the previous image frame in order to update its position and orientation on the device map. The Euclidean distance between a newly detected OEM m_d and all the latest known OEM observations m_i is measured in increasing order. If the nearest previous observation $d(m_d, m_i) \leqslant \frac{m_s}{2}$, this observation is updated with the newly detected OEM values. If no close match is found, an image inset corresponding to both observations is compared using APT descriptors in increasing order of distance below a maximum radius. If the minimal MSE value of their respective APT vectors $\mathrm{MSE}(\mathtt{APT}_d, \mathtt{APT}_i) \leqslant t_{\mathtt{APT}_r}$, with $t_{\mathtt{APT}_r}$ an arbitrary threshold value, the previous observation is updated. In the case where no near or visually matched previous observation is found, a new OEM is created.

F. OEM Path-Finding and Target Assignment

The path-finding module seeks the shortest navigable, collision-free route between an OEM and its destination. A task assignment module allocates OEMs to available targets while minimizing the cumulative length of individual routes.

1) Path-Finding: Finding navigable routes for multiple OEMs can be posed as a multi-agent path-finding (MAPF) problem [13] within the 2-D free-space S_f derived from the labeled map $\mathbb{M}$ in which the binary structures represent the obstacle-space S_o. S_f is computed for each OEM, allowing for different OEM sizes, from the distance transform of the obstacle-space S_o thresholded with a value equal to half the maximum diameter occupied by the OEM and its associated light pattern. Our previous open-loop OEM control system [6] used a decentralized prioritized planner to compute the paths of all agents in a static environment, which proved computationally expensive as the number of agents and map size increased. Closed-loop system control of OEMs in a dynamic environment requires both real-time and iterative computation to accommodate unpredictable moving obstacles. Sampling-based approaches [14], while generally sub-optimal, can scale efficiently by reducing the dimension of the configuration space. This lowers the computational cost, enabling real-time execution in larger environments. To accommodate large map sizes in a pixel-precise configuration space and an unrestricted number of agents and obstacles, we chose to use the sampling-based path-finding algorithm PRM* [14], as implemented in the OMPL library [15]. While not strictly multi-agent, nor able to prevent deadlocks occurring when agents mutually obstruct their paths in space-constrained use cases, this path-finding approach, coupled with an adaptive obstacle-avoidance system for all agents (Section III-H) proved effective for closed-loop OEM control (Section IV-C) under typical experimental conditions.

2) Target Assignment: The process of harvesting multiple targets using multiple capacity-constrained OEMs can be considered as an example of the capacitated vehicle routing problem (CVRP). However, due to the dynamic state of the configuration space, OEMs, targets, and obstacles cannot be assumed to remain in fixed positions during operation, which can invalidate any pre-computed route. Incidentally, target assignment is instead reduced to an assignment problem on a bipartite graph, for which the costs of the existing edges are the computed path lengths from each OEM to each target. The problem is solved during the initialization phase using the OR-Tool library [16]. After the initialization phase, OEMs are assigned to their closest available target.

G. Stage Motion-Planning

The stage motion-planning module seeks to maximize the total navigable path length of the OEMs within the projector image field and the average speed of the agents. The navigable length n_i of an OEM trajectory T_i associated with OEM m_i is the portion of T_i located inside the projector operating space S_p, which is determined by the stage coordinates p_s, such that $n_i = \eta(m_i, T_i, p_s)$. The total length of navigable paths at stage coordinates p_s is then computed following $L(p_s) = \sum_i n_i$. The average speed of the agents, with $\mathsf{s}^+_{t,i}$ the maximum linear speed of OEM m_i, is computed following $S(p_s) = L(p_s)/\max_i(n_i/\mathsf{s}^+_{t,i})$. Finally, the length of navigable paths for a newly evaluated stage position is corrected to account for the stage motion time $\Delta_{t,s} = d(p^0_s, p^1_s)/\mathsf{s}^+_{t,s}$, with p^0_s, p^1_s and $\mathsf{s}^+_{t,s}$ the current stage position, evaluated stage position and maximum stage speed, respectively. The corrected length is given by $L_1 = L(p_s) - 2S_0\Delta_{t,s}$, where S_0 is the current average speed of all agents. Several new sampled stage positions are evaluated iteratively in order to maximize either the average agent speed or the total navigable path length over the current position, with a priority given to the former in order to favour the simultaneous control of multiple OEMs.

H. Kinematics and Steering Behaviors

The kinematics module computes and updates the poses of the light patterns which control the OEMs and the poses of the virtual objects, both in real and simulated conditions, using specific dynamic behavioral modes. Several methods for path-following and trajectory tracking have been proposed in the literature, *e.g.*, [17], [18]. For computational simplicity, the motion of each simulated object is computed using a point particle approximation and differential kinematic equations integrated in real-time with the Euler method. This solution is inspired by the framework originally proposed by Reynolds [19], adapted to mass-less objects since inertia in microscopic environments is negligible [20]. The effect of viscous drag is addressed through parametrized linear and angular acceleration values to ensure motion smoothness. Formally, a movable object o is represented by ten elements: its centroid position on the map $p_o \in \mathbb{M}$, its orientation θ_o, its current acceleration and velocity vectors $\mathbf{a}_{t,o}$ and $\mathbf{s}_{t,o}$, its current angular acceleration and velocity $\mathsf{a}_{\theta,o}$ and $\mathsf{s}_{\theta,o}$, its maximal linear and angular accelerations $\mathsf{a}^+_{t,o}$, $\mathsf{a}^+_{\theta,o}$ and velocities $\mathsf{s}^+_{t,o}$, $\mathsf{s}^+_{\theta,o}$, and its last time of update t_o. For linear motions, the new position of an object is computed from an input acceleration vector $\mathbf{a}_{t,d}$ following:

$$p^{t_1}_o = p^{t_0}_o + \left[\mathbf{s}^{t_0}_{t,o} + \left(\hat{\mathbf{s}}^{t_1}_{t,o} \cdot \max\left(\|\mathbf{s}^{t_1}_{t,o}\|, \mathsf{s}^+_{t,o}\right)\right)\right] \cdot \Delta_t \tag{1}$$

with $\mathbf{s}^{t_1}_{t,o} = \frac{\mathbf{s}^{t_0}_{t,o} + (\hat{\mathbf{a}}_{t,d} \cdot \max(\|\mathbf{a}_{t,d}\|, \mathsf{a}^+_{t,o}) \cdot \Delta_t)}{2}$ and $\Delta_t = t_1 - t_0 = t_{current} - t_o$ the elapsed time. Similarly, angular motions are computed from the angular difference Δ_θ between the normalized current orientation θ_o and a desired orientation θ_d following:

$$\theta^{t_1}_o = \theta^{t_0}_o + \left[\mathsf{s}^{t_0}_{\theta,o} + \max\left(\mathsf{s}^{t_1}_{\theta,o}, \mathsf{s}^+_{\theta,o}\right)\right] \Delta_t \tag{2}$$

with $\mathsf{s}^{t_1}_{\theta,o} = \frac{\mathsf{s}^{t_0}_{\theta,o} + (\mathrm{sgn}(\mathsf{a}_{\theta,d}) \cdot \max(|\mathsf{a}_{\theta,d}|, \mathsf{a}^+_{\theta,o}) \cdot \Delta_t)}{2}$ and $\mathsf{a}_{\theta,d} = \mathsf{a}^+_{\theta,o}[\mathrm{sgn}(\Delta_\theta) \cdot \sqrt{2 \cdot \mathsf{a}^+_{\theta,o} \cdot |\Delta_\theta|} - \mathsf{s}^{t_0}_{\theta,o}]$. When an object is at rest $\mathsf{s}_{l,o} = 0$, rotation to a desired orientation $\theta_d = \mathrm{atan2}(\mathsf{a}_{t,d})$ is performed before linear motion, with π added to the desired orientation value in the case of reverse motion. Within this kinematic framework, a set of discrete steering behavioral modes is defined to allow OEMs and simulated objects to perform a variety of operations.

1) Seek, Arrival and Stop: Seek moves an object at maximum acceleration to a destination p_d in a straight line without stopping, with an acceleration vector $\mathbf{a}_{t,d} = (\mathbf{s}^{t_1}_{t,o} - \mathbf{s}^{t_0}_{t,o}) \cdot \mathsf{a}^+_{t,o}$ and desired velocity vector $\mathbf{s}^{t_1}_{t,o} = p_d - p_o$. *Arrival* is similar to *Seek*, but gradually decelerates the object to a stop at the destination, by modifying the acceleration magnitude according to the remaining distance to create an acceleration vector $\mathbf{a}_{t,d} = (\mathbf{s}^{t_1}_{t,o} - \mathbf{s}^{t_0}_{t,o}) \cdot \mathsf{a}^+_{t,o}$, with $\mathbf{s}^{t_1}_{t,o} = \frac{p_d - p_o}{\|p_d - p_o\|} \cdot \sqrt{2 \cdot \mathsf{a}^+_{t,o} \cdot d(p_o, p_d)}$ the desired velocity vector. Finally, *Stop* applies the maximum deceleration to bring the object to a halt with an acceleration vector $\mathbf{a}_{t,d} = -\mathbf{s}^{t_0}_{t,o} \cdot \mathsf{a}^+_{t,o}$.

2) Obstacle-Avoidance: This behavior is somewhat different from the other modes, as it checks for obstacles along a desired acceleration vector $\mathbf{a}_{t,d}$. *Obstacle-avoidance* detects and avoids obstacles around the object within a radius $r_{\text{o-a}} = \mathbf{s}^{t_0}_{t,o} \cdot \frac{\mathbf{s}^{t_0}_{t,o}}{2 \cdot \mathsf{a}^+_{t,o}}$ computed from the current object speed, and works similarly to a 2-D LIDAR range-scanner [21]. Object surroundings are scanned on the labeled map $\mathbb{M}$ following lines in every direction at one-degree angle intervals, and the position of the first collision in each direction is converted into polar coordinates. Similarly, the 2-D rectangle representing the minimal space for navigation along the object's forward axis, with a width equal to the object's maximum dimension, is also converted to polar coordinates. All directions for which the scanned range is less than the minimal navigable range are discarded. When there is no obstacle in the direction of the acceleration vector $\mathbf{a}_{t,d}$, *Obstacle-avoidance* returns the same vector. If an obstacle is detected, the previous vector is cleared and the object either stops or chooses the closest available direction θ^{t_1} by jointly calling *Seek*, with destination $p_d = p_o + [r_{\text{o-a}} \cdot \cos(\theta^{t_1}), r_{\text{o-a}} \cdot sin(\theta^{t_1})]^T$, and *Stop* to avoid a collision, until the path is clear.

3) Wander: Wander randomly changes the magnitude and orientation of the object's current acceleration vector $\mathbf{a}^{t_0}_{t,o}$ to produce a new acceleration vector. A wander rate factor w_r changes the orientation of the new vector $\theta^{t_1} = \text{atan2}(\mathbf{a}^{t_0}_{t,o}) + \text{rand}(-w_r, w_r)$ and the magnitude of the vector $m^{t_1} = \|\mathbf{a}^{t_0}_{t,o}\| + \text{rand}(-w_r, w_r)$. If the current acceleration vector is null, a wander strength value w_s is used to compute the magnitude of a new acceleration vector $m^{t_1} = w_s + \text{rand}(-w_r, w_r)$, while the orientation is chosen randomly $\theta^{t_1} = \text{rand}(-\frac{\pi}{2}, \frac{\pi}{2})$. The resulting acceleration vector is then computed from the magnitude and orientation parameters following $\mathbf{a}^{t_1}_{t,o} = [m^{t_1} \cdot \cos(\theta^{t_1}), m^{t_1} \cdot sin(\theta^{t_1})]^T$.

4) Path-Following: Finally, *Path-following* follows the object's trajectory T_o within a radius r_t. An object trajectory is composed of several nodes n, with the current node n_0 being the next closest node to the object on the trajectory. This behavioral mode estimates the distance between the object's expected position $p^p_o = p_o + (\mathbf{s}^{t_0}_{t,o} \cdot \Delta_t)$ at its current velocity to the remaining nodes on the trajectory $d^{n_i}_o = d(p^p_o, n_i)$. The trajectory node n_i for which the distance $d^{n_i}_o > r_t$ is then used as a destination point p_d by *Arrival*.

5) Combining Steering Behaviors: All the described behaviors can be combined using a steering accumulator, *i.e.* by simple addition of the computed acceleration vectors, to create more complex behaviors. *Path-following* can for instance be combined with *Obstacle-avoidance* $p^{t_1}_o = \text{obstacle-avoidance}(\text{path-following}(o))$, which is the behavior used by the light patterns for OEM manipulation. Virtual obstacles use a combination of *Wander* and *Obstacle-avoidance*. An example of OEM and obstacle motions in simulation are shown Fig. 3.

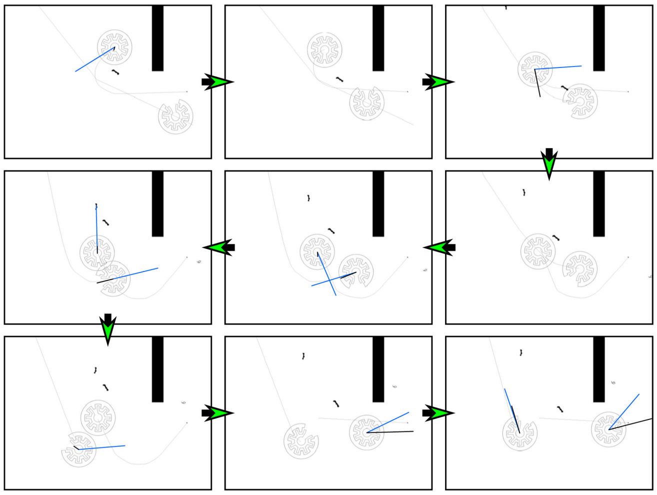
Fig. 3. Cropped frames from a time-lapse sequence showing adaptive motion and trajectories of two OEMs in simulation. OEMs move using both *Path-following* and *Obstacle-avoidance* behaviors, while dynamic obstacles (small black objects) move using both *Wander* and *Obstacle-avoidance*. The black vertical bar in the upper right is a static obstacle. The trajectory, instantaneous linear acceleration and velocity of each OEM are represented by gray, blue and black lines starting from their centroid, respectively. Note the updated trajectories between each frame to avoid collisions. In the 6th frame, the light patterns of the two OEMs overlap, which is by design, as overlapping light patterns do not interfere in real-world conditions.

I. Implementation in Concurrent Framework

The SLAM, image-processing, path-finding, stage motion-planning, and kinematics modules are integrated into a concurrent C++ framework to enable simultaneous, real-time operation following an off-line initialization stage.

1) Initialization: After a complete system calibration (Section III-C), the device map is created and labeled by capturing and tiling camera images spanning the usable area of the OET electrodes. All OEMs are simultaneously detected (virtual ones are created during simulations) and virtual targets and obstacles are created in the free-space region of the finalized map. Each OEM is then assigned to a target (Section III-F2) and its trajectory is computed.

2) Asynchronous Concurrent Modules With Multi-Threading: Following initialization, the system launches all modules concurrently in separate asynchronous CPU threads, with protected access to shared data. This allows all the modules to run at different cycle speeds. Critical components, such as the kinematics of the projected light patterns, which are essential for smooth OEM control, can run at high cycle speeds, whilst less sensitive tasks such as path-finding or target assignment can run more slowly. The image grabber module acquires and undistorts images from the camera using the pre-computed calibration parameters. These images are used by the SLAM and image processing modules to synchronously update the labeled map and the current poses of OEMs within the camera image, while virtual objects are updated by the kinematics module. The updated map and objects are used by separate modules to compute the paths for each OEM, assign them to a destination, and position the stage to maximize the total navigable path length or average agent speed. Finally, the kinematics module updates the position of the light patterns for projection onto the lower OET electrode to move OEMs to their destinations. During stage motion, all other modules are disabled and the external electric field is switched off to minimize OEM drift. After the stage reaches its target position, all modules resume their tasks.

IV. Experiments and Results

The OEM control system was evaluated using a simple harvesting task in which OEMs were asked to collect and transport a virtual target object, release it in a predefined area and dock in a separate parking area after task completion. In the case where there were fewer OEMs than targets, each OEM was allowed to collect several targets before the release operation, provided it had sufficient cargo capacity. Experiments were conducted both in simulation (Fig. 3) and on a real device (Fig. 4) to investigate the system's ability to complete the task, using virtual static targets and virtual moving obstacles. Both types of experiments

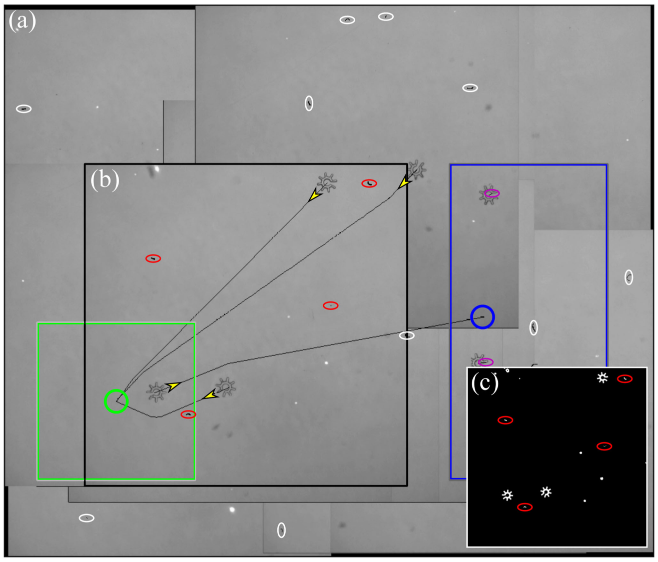

Fig. 4. Screen capture from an OEM collect, transport, and release experiment in real-world conditions, as described in Sections IV and IV-B (shown in full in supplementary video). (a) Brightfield map of the device, containing 6 OEMs (yellow arrows indicate the direction of their motion), 2 virtual targets (not visible, the purple ellipses indicate their position), OEM trajectories to their destination (black lines), virtual obstacles (red and white ellipses), drop-off area (green box) and parking area (blue box). (b) Camera field-of-view inside the map with virtual moving obstacles (red ellipses) and 3 OEMs. (c) Crop of the labeled map $\mathbb{M}$ corresponding to the camera field-of-view shown in (b). In this example, two OEMs (top-right, bottom-right, above (c)) are in the process of collecting their assigned virtual targets (purple ellipses), three OEMs are moving to the drop-off area (green circle at the center of the green box) to release their cargo and one is going to the parking zone (blue circle inside the blue box) after releasing its target.

were repeated 8 times each in the same conditions using an AMD Ryzen$^{\text{TM}}$ Threadripper$^{\text{TM}}$ 3060X with 64 GB of RAM. The cycle speeds of the path-finding and stage motion-planning modules were fixed at 2 Hz and 0.2 Hz, respectively, to avoid incomplete or varying trajectories and prevent the stage from moving too frequently. Examples of simulated and on-device experiments are shown in the supplementary video.

A. Simulation

A virtual environment was created as a binary map of 27.66 mm $\times$ 15.13 mm containing static obstacles, 10 OEMs, 10 targets, and 100 moving obstacles randomly placed in the free-space region of the map. These targets and moving obstacles were generated with morphologies and dimensions corresponding to cells and debris using the simulated dissociated tissue labels proposed in our previous work [6]. The virtual OEMs used a combination of *Path-following* and *Obstacle-avoidance* steering behavioral modes, with maximum acceleration and velocity vectors of 0.4 mms^{-2} and 0.4 mms^{-1} and maximum angular acceleration and velocity values of 4 rads^{-2} and 4 rads^{-1}. The moving virtual obstacles used a combination of the *Wander* and *Obstacle-avoidance* modes, with maximum acceleration and velocity vectors of 0.1 mms^{-2} and 0.1 mms^{-1}, maximum angular acceleration and velocity values of 0.1 rads^{-2} and 0.1 rads^{-1} and wander rate and strength values $w_r = 0.02$ and $w_s = 0.1$.

TABLE I
FRAMEWORK PERFORMANCE IN SIMULATED AND REAL CONDITIONS

Avg. per experiment ($\times$8)	Simulation, 10 OEMs	On-device, 6 OEMs
Task success rate (%)	95.00 $\pm$ 7.07	72.92 $\pm$ 21.95
Task failure rate (%)	5.00	27.08
OEM out of FOV (%)	-	14.58
Unreachable destination (%)	1.25	8.33
Non-observable OEM (%)	-	2.08
FOV blocked by OEM (%)	2.50	2.08
Unstable path-finding (%)	1.25	-
OEM tracking (%)	-	92.39 $\pm$ 4.64
OEM path-finding (%)	93.00 $\pm$ 6.28	78.84 $\pm$ 11.08
OEM displacement (mm)	283.96 $\pm$ 32.63	46.32 $\pm$ 10.49
Time (s)	1700.88 $\pm$ 498.54	770.57 $\pm$ 200.75
Speed of OEMs (mms^{-1})	1.74e^{-1} $\pm$ 2.88e^{-2}	6.15e^{-2} $\pm$ 9.31e^{-3}
Speed per OEM (mms^{-1})	1.74e^{-2} $\pm$ 2.88e^{-3}	9.29e^{-3} $\pm$ 1.68e^{-3}

The stage maximum acceleration and velocity vectors were set at 5 mms^{-2} and 5 mms^{-1}.

B. On-Device

Real-world experiments were conducted on a smaller device area measured at 3.30 mm $\times$ 2.30 mm, containing debris, 6 OEMs, 6 virtual targets, and 20 virtual moving obstacles, again placed randomly in the free-space region of the labeled map (Fig. 4). OEMs were trapped and positioned using light patterns with maximum acceleration and velocity vectors of 0.1 mms^{-2} and 0.05 mms^{-1} and maximum angular acceleration and velocity values of 0.2 rads^{-2} and 0.5 rads^{-1}. The characteristics and placement of the virtual objects were the same as in the simulation. The stage maximum acceleration and velocity vectors were set at 200 mms^{-2} and 2 mms^{-1}.

C. Performance Assessment

The performance measures shown in Table I quantify the effectiveness of our approach, with the task completion success rates (the fraction of OEMs successfully executing the harvesting, release, and parking operations) measured at 93.75% in simulation and 72.92% in real conditions. The cycle speed of the joint SLAM and image processing modules was measured at ≈ 8 Hz when operating with full resolution images (2048×2048 pixels). This allowed both the labeled map (downscaled at 1/4 resolution) to be updated and for reliable tracking of OEMs, with 92.39% accuracy, over several non-adjacent camera fields-of-view. The path-finding success rate (the ratio of successful OEM planning queries) was evaluated at 93.00% and 78.84% for simulations and on-device experiments, respectively. The cycle speed of the kinematics module was measured at ≈ 500 Hz, which allowed for smooth update of the light patterns on the PC substrate and well-controlled OEM positioning. While not an issue in simulation, where the OEMs were directly positioned by the kinematics module, the achievable linear and angular speeds of OEMs (measured on average at $9.29\ \mu$s^{-1} per OEM) proved dependent on the cycle speed of the OEM detector in real experiments. This was mainly due to maintain close proximity between each light pattern and its corresponding OEM to ensure a sufficient DEP force, and avoid loss of control when the two are significantly offset. A further limitation encountered during on-device experiments was due to untrapped OEMs drifting outside the camera or projector image field, which accounted for 14.58% of task execution failures. Another occasional cause of failure both in simulation (1.25%) and on-device (8.33%) experiments originated from the *Obstacle-avoidance* steering behavior and path-finding module blocking the progress of OEMs trying to avoid obstacles very close to their destination (unreachable destination). Finally, we observed three other minor causes of task failures. Firstly, OEMs could become undetectable when passing close to an imperfection or scratch on the device substrate which scattered excess light towards the camera, overexposing the sensor (2.08%). Secondly, unreachable destinations could prevent the stage moving to continue shorter trajectories in other locations of the map (2.50% and 2.08%). Thirdly, unstable path-finding computations in complex areas of the virtual device produced an endlessly oscillating OEM trajectory (1.25%). Although these failure cases can affect the ability of an individual OEM to complete a particular operation, having control of multiple OEMs ensures the system has sufficient redundancy to complete the overall task, albeit in a longer execution time.

V. Conclusion

This letter describes the first adaptive autonomous system for multi-agent micromanipulation in dynamic environments using optoelectronic microrobots. The physical constraints imposed by the system hardware and changing environmental conditions are addressed using SLAM, visual-servoing in closed-loop, dynamic path-finding, and a custom kinematics solution operating in a concurrent asynchronous framework. The experimental results in simulated and real-world environments demonstrate the ability of this framework to simultaneously control multiple OEMs over relatively large distances in challenging real-world conditions, by successfully avoiding moving obstacles and replanning their routes according to the updated configuration space of the environment in real-time. While presenting certain limitations in regard to multi-agent path-finding optimality, this control framework is a robust foundation that can be further developed and improved, particularly for multi-agent navigation and cooperation. Notably, the ability of the system to operate in simulation offers the possibility to apply deep-learning approaches such as reinforcement learning for multi-agent path-finding, stage motion-planning, kinematics, and task allocation. The new OEM control framework overcomes many of the limitations of manual [2] and open-loop approaches [6], increasing the potential application of OEMs to various manipulation tasks. In particular, many biological micromanipulation operations are time critical due to the short period within which cells are viable within a microfluidic environment. Harnessing multiple OEMs to perform operations simultaneously greatly reduces overall task execution time. With further modification to the OET system hardware (including the use of a lower magnification objective lens and a corresponding increase in light intensity from the digital projector), our simulations indicate that the framework can directly scale to allow simultaneous control of tens of OEMs. Finally, we note that the framework can be easily adapted to accommodate new developments in OEM design, with small modifications to the OEM detector, and support more complex micromanipulation tasks via the addition of new steering behaviors. In summary, our work removes many of the barriers preventing to the wider adoption of OEM technology, providing a powerful new tool for micromanipulation.

References

[1] B. Ahmad, M. Gauthier, G. J. Laurent, and A. Bolopion, “Mobile microrobots for in vitro biomedical applications: A survey,” *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 646–663, Feb. 2022.

[2] S. Zhang et al., “The optoelectronic microrobot: A versatile toolbox for micromanipulation,” *Proc. Nat. Acad. Sci.*, vol. 116, no. 30, pp. 14823–14828, 2019.

[3] J. R. Moffitt, Y. R. Chemla, S. B. Smith, and C. Bustamante, “Recent advances in optical tweezers,” *Annu. Rev. Biochem.*, vol. 77, no. 1, pp. 205–228, 2008.

[4] P. Y. Chiou, A. T. Ohta, and M. C. Wu, “Massively parallel manipulation of single cells and microparticles using optical images,” *Nature*, vol. 436, no. 7049, pp. 370–372, 2005.

[5] S. Zhang et al., “Reconfigurable multi-component micromachines driven by optoelectronic tweezers,” *Nature Commun.*, vol. 12, no. 1, pp. 1–9, 2021.

[6] C. Bendkowski et al., “Autonomous object harvesting using synchronized optoelectronic microrobots,” in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2021, pp. 7498–7504.

[7] R. Tsai, “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses,” *IEEE J. Robot. Autom.*, vol. 3, no. 4, pp. 323–344, Aug. 1987.

[8] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[9] Y. Zhou and B. J. Nelson, “Calibration of a parametric model of an optical microscope,” *Opt. Eng.*, vol. 38, no. 12, pp. 1989–1995, 1999.

[10] M. Ammi, V. Frémont, and A. Ferreira, “Automatic camera-based microscope calibration for a telemicromanipulation system using a virtual pattern,” *IEEE Trans. Robot.*, vol. 25, no. 1, pp. 184–191, Feb. 2009.

[11] G. Bradski, “The opencv library,” *Dr Dobb’s J.: Softw. Tools Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[12] J. Lai, L. Lei, K. Deng, R. Yan, Y. Ruan, and Z. Jinyun, “Fast and robust template matching with majority neighbour similarity and annulus projection transformation,” *Pattern Recognit.*, vol. 98, 2020, Art. no. 107029.

[13] R. Stern, “Multi-agent path finding—an overview,” in *Artif. Intell.*, pp. 96–115, 2019.

[14] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[15] I. A. Şucan, M. Moll, and L. E. Kavraki, “The open motion planning library,” *IEEE Robot. Automat. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012, [Online]. Available: https://ompl.kavrakilab.org

[16] L. Perron and V. Furnon, “Or-tools,” Google, Mar. 15, 2022. [Online]. Available: https://developers.google.com/optimization/

[17] M. Breivik and T. I. Fossen, “Guidance laws for planar motion control,” in *Proc. IEEE 47th Conf. Decis. Control*, 2008, pp. 570–577.

[18] Q.-C. Pham, “A general, fast, and robust implementation of the time-optimal path parameterization algorithm,” *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1533–1540, Dec. 2014.

[19] C. W. Reynolds et al., “Steering behaviors for autonomous characters,” in *Proc. Game Developers Conf.*, 1999, pp. 763–782.

[20] G. I. Taylor, “Analysis of the swimming of microscopic organisms,” *Proc. Roy. Soc. London Ser. A. Math. Phys. Sci.*, vol. 209, no. 1099, pp. 447–461, 1951.

[21] T. Weerakoon and K. Ishii, “2D obstacle avoidance algorithm for mobile robots,” in *Proc. JSME Conf. Robot. Mechatron.*, 2012, pp. 1A2-F06_1–1A2-F06_4, doi: 10.1299/jsmermd.2012._1A2-F06_1.