# Reinforcement Learning for Maneuver Control of a Bio-Inspired Vessel with Undulating Fin Propulsion

*Gonzalo Garcia, Mohammad Uddin, Siddhartha Verma, and Oscar Curet*
Department of Ocean and Mechanical Engineering, Florida Atlantic University
Dania Beach, Florida, United States

## ABSTRACT

Undulating fins provide biological swimmers with a remarkable propulsion system. Replication with autonomous underwater robots is not simple, both on the technical side and due to the complexity of the control system. Reinforcement learning is applied to a bio-inspired vessel for maneuver control by manipulating the kinematics of its undulating membrane. The controller is initially trained with a numerical model recreating the distributed forces and moments on the fin, and then applied to an underwater vehicle. It was found that using Reinforcement Learning the vehicle was able to perform key maneuvers including control for speed, heading, and turning. The results demonstrate that reinforcement learning has the potential to overcome high levels of system complexity while delivering an optimal solution.

KEY WORDS: Reinforcement learning; DDPG; undulating fin propulsion.

## INTRODUCTION

Biological swimmers have achieved remarkable propulsion capabilities through undulatory fins. Nature has evolved to a point where the use of a single fin in combination with pectoral fins are able to control swimming in multiple directions with high levels of accuracy. The bio-inspired application of these traits to underwater vessel locomotion presents a non-trivial task. Normally, the fish has an elongated fin along its body, driven by hundreds of bones in an undulatory manner, producing the necessary forces and moments to move freely in all directions. It has been observed that these kinds of fish have developed several fin kinematics for different kinds of motion, ranging from sinusoidal traveling wave-like kinematics for longitudinal motion, to combination of more than one wave for vertical displacement among others, all combined with body bending for lateral-directional movement. The potential artificial mimicking by underwater robotic devices imposes a level of complexity in the design of control systems that quickly rules out traditional linear model-based controllers. The operation of a robotic undulatory fin composed of several rotatory rays in place of the fish fin bones, entails a multivariable process that requires a different approach. Our current research involves an autonomous device equipped with an undulatory fin actuated by 16 rays, with the capability of controlling each ray asynchronously, regulating its position and speed for arbitrary path following. The degree of intricacy of the system, while trying to keep a close resemblance to the natural swimmer, strongly suggests the need for a machine learning approach, especially technique such as reinforcement learning that is capable of learning during operation. We have developed a two-input two-output 3 DOF path-following control system for a bio-inspired robot based on a specific pattern of ray motion. The controller modulates the amplitude of a traveling wave for speed control, and coordinates the offsetting of a subset of rays for the lateral-directional dynamics, while using the closest distance to the trajectory and its current speed as inputs. The reinforcement learning controller was tested in various conditions experimentally, including an indoor flume, and indoor and outdoor open tanks to perform different maneuvers, with different levels of external disturbances.

## REINFORCEMENT LEARNING

The level of complexity in current autonomous vehicles makes classical control approaches ineffective, or harder to design, due to an increasing number of sensors, actuators, control loops, and involved dynamics. Reinforcement learning (RL) addresses these limitations in an integrated way, providing an optimal solution independently of the complexity of the system under control.

The RL agent learns by interacting with the environment, which normally not only includes the system's dynamics, but environmental perturbations as well. The agent learns actively while operating in real time. This is in contrast to other optimal methods that base their learning on simplified mathematical models and are best designed to be tuned backward-in-time (Sutton and Barton, 2018; Busoniu, Babuska, De Schutter, and Ernst, 2017).

The specific RL version chosen for this work is known as Deep Deterministic Policy Gradient (DDPG), which is an actor-critic RL agent with a model-free, online, off-policy algorithm that maximizes the expected cumulative long-term reward. This method allows for an optimal learning process carried out during operation, based on the robot's actual dynamics, and continuous-time signals. In the limit, the critic section captures the optimal action-state combination in terms of the highest long-term reward, and the actor section implements the optimal control policy.

This RL method, among others, is based on the numerical solution of the

Bellman equation, and the Bellman's principle of optimality, which states that an optimal policy has the property that for any current state and action, the decisions that follow must be themselves optimal, starting from the resulting state. This allows for recursive solutions like dynamic optimization, which splits the problem into a sequence of simpler subproblems.
Given a system described by the dynamics $x_{k+1} = f(x_k, u_k)$ and a reward function $\sigma(x_k, u_k)$, where $x_k$ is the state of the system, and $u_k = \pi(x_k)$ the control policy, a long-term reward can be defined by:

$$\sum_{k=0}^{\infty} \gamma^k \sigma(x_k, u_k) = \sum_{k=0}^{\infty} \gamma^k \sigma\big(x_k, \pi(x_k)\big) \tag{1}$$

where $0 < \gamma < 1$ is a discount factor required to penalize future rewards and to ensure convergence of the summation. This expression represents the discounted accumulated rewards starting from the current state $x_0$ and the application of the policy $\pi$.
To apply Bellman's optimality principle, the previous long-term reward expression Eq. (1) is redefined in terms of the function $Q(x_k, u_k)$, called action-value, which allows for the splitting of the reward assignment into two consecutive steps. This action-value function conveys the long-term reward, by the contribution of the immediate reward due to applying an arbitrary action $u_k$ while in the state $x_k$, and by the discounted accumulated reward continuing with the control policy $\pi$. This is, starting from $x_0$:

$$Q^{\pi}(x_0, u_0) = \sigma(x_0, u_0) + \sum_{k=1}^{\infty} \gamma^k \sigma\big(x_k, \pi(x_k)\big) = \sigma(x_0, u_0) + \gamma \sum_{k=0}^{\infty} \gamma^k \sigma\big(x_{k+1}, \pi(x_{k+1})\big) \tag{2}$$

The optimal value is obtained by maximizing the future rewards by using the optimal policy defined by $\pi^*(x_k) = \gamma \max_{\mu} Q(x_{k+1}, \mu)$. From Eq. (2) and the optimal policy $\pi^*$, a recursive equation is obtained:

$$Q^*(x_k, u_k) = \sigma(x_k, u_k) + \gamma \max_{\mu} Q^*(x_{k+1}, \mu) \tag{3}$$

This equation encapsulates the optimally principle by stating that future optimal control actions are not specified by past optimal values, but only by the current state. And the major advance in these calculations is the viability of forward-in-time learning, as opposed to standard optimal search done backward-in-time. This method also receives the name of *Q*-learning.
From Eq. (3), the following recursive equation can be devised that asymptotically converges to the fixed manifold $Q^*$ (Watkins, 1989; Watkins and Dayan, 1992):

$$Q^{i+1}(x_k, u_k) = Q^i(x_k, u_k) + \alpha\left(\sigma(x_k, u_k) + \gamma \max_{\mu} Q^i(x_{k+1}, \mu) - Q^i(x_k, u_k)\right) \tag{4}$$

The term $\sigma(x_k, u_k) + \gamma \max_{\mu} Q^i(x_{k+1}, \mu) - Q^i(x_k, u_k)$ is typically labeled temporal difference $TD^i(x_k, u_k)$, or error between the target value $\sigma(x_k, u_k) + \gamma \max_{\mu} Q^i(x_{k+1}, \mu)$ and the current value $Q^i(x_k, u_k)$, with $0 < \alpha < 1$ a learning rate. The expression in Eq. (4) resembles a gradient descend numerical search. Another interpretation of (4) is the structure of a low pass filter, by rearranging it as $Q^{i+1}(x_k, u_k) = \alpha TD^i(x_k, u_k) + (1-\alpha) Q^i(x_k, u_k)$.
The learning rate, or numerical search step size, establishes the effect of new information overriding previous information. A small value will reduce the rate of learning, while a larger value will rely more heavily on new data despite what was previously learned. This factor does not need to be kept constant and can be modified during the learning process. *Q*-learning can be directly applied as a look-up table, where states and actions are discretized as indices. But this approach becomes intractable for large numbers of states and actions and requires some sort of discretization. An alternative method is to use function approximators to parametrize both the action-value function and the control policy (see Lillicrap, Hunt, Pritzel, Heess, Erez, Tassa, Silver, and Wiestra, 2015). The approach used here, known as DDPG actor-critic, is based on feedforward artificial neural networks (ANN), given their capacity as effective universal approximators, allowing the application of the Q-learning concept to continuous-time system with many signals.
In the limit, a feedforward ANN with a single hidden layer with a finite number of neuron cells, with sigmoid-type activation functions, and a linear-type activation function at the output cells, inherently materializes the universal approximation theorem. It has the ability of approximating any multivariable continuous function on a compact subset of $\mathbb{R}^q$ to any given degree of accuracy (Hornik K, Stinchcombe M, White H, (1989). This states that for any given continuous function $f_j(\boldsymbol{y}) \in [0,1] \in \mathbb{R}$, with $\boldsymbol{y} = \left[y_1, y_2, \cdots, y_q\right]^T \in [0,1]^q \in \mathbb{R}^q$, and for $\varepsilon > 0$, there exists an integer $M$ and a set of parameters $\boldsymbol{w}_i = \left[w_{1i}, w_{2i}, \cdots, w_{qi}\right] \in \mathbb{R}^q, a_{ji} \in \mathbb{R}$ (weights) and $b_i, c_j$ (biases), with $i = 1 \cdots M$, and $j = 1 \cdots p$, such that

$$F_j(\boldsymbol{y}) = \sum_{i=1}^{M} a_{ji} \phi\big(\left[w_{1i}, w_{2i}, \cdots, w_{qi}\right]\boldsymbol{y} + b_i\big) + c_j \tag{5}$$

is an approximation of the function $f_j(\boldsymbol{x})$, with $\left|F_j(\boldsymbol{y}) - f_j(\boldsymbol{y})\right| < \varepsilon$, with $\phi$ a bounded and monotonically increasing activation function.
This ability makes an artificial neural network an extremely useful means for mapping signals. With a proper mechanism for adjusting its parameters, for example, back propagation and gradient descent numerical search, the ANN is capable of approximating any static relation between inputs and outputs. If the training is kept active during operation, then an ANN can also track slow changing causal relations.
Two ANNs are used here, one for the action-value function $Q\big(x_k, u_k | \theta_Q\big)$ and one for the control policy $\pi(x_k | \theta_\pi)$, with $\theta_Q$ and $\theta_\pi$, the respective parameters to be tuned during the learning process.
The update of $\theta_Q$, based on a gradient descent method, is based on the minimization of:

$$L = \frac{1}{N} \sum_{j=1}^{N} TD^i\big(x_{i-j}, u_{i-j}\big)^2 \tag{6}$$

this is the square of the temporal difference over a moving window of data of length $N$.
Instability during the training stage may arise due to the stochastic dependency between the target and the current value. Another solution presented by Lillicrap, 2015 was the application of a low pass filter to the target values, effectively reducing the variability of the estimated parameters $\theta_Q$.
On the other hand, the update of $\theta_\pi$ is done by calculating the direction of the maximum variation of the critic with respect to the actions, i.e., computing the gradient, what constitutes a continuous-time version of the maximization in the Q-learning discrete logic.
For this work, MATLAB toolbox Reinforcement Learning was used.

## BIO-INSPIRED VESSEL

Inspired from the black ghost knifefish, the robot uses a lengthwise fin that produces net forces and torques effecting the longitudinal and vertical motions. This membrane is driven by parallel rays extended from the body, producing a field of distributed forces and moments. The body is an elongated ellipsoidal cylinder carrying all the electronics and motors inside, from which each ray is subtended and independently actuated. A photo and the internal configuration are shown in Fig. 1. More details of the original design and performances can be found in (Liu, 2017; Liu and Curet, 2018).

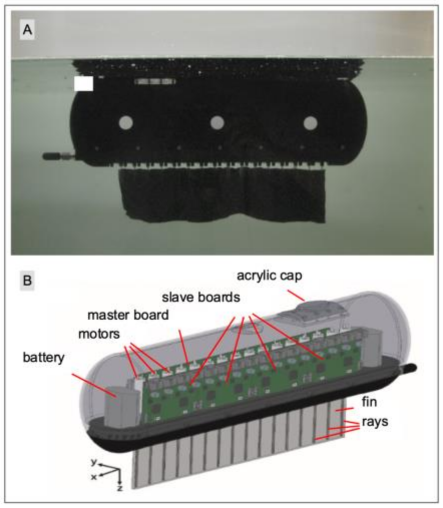


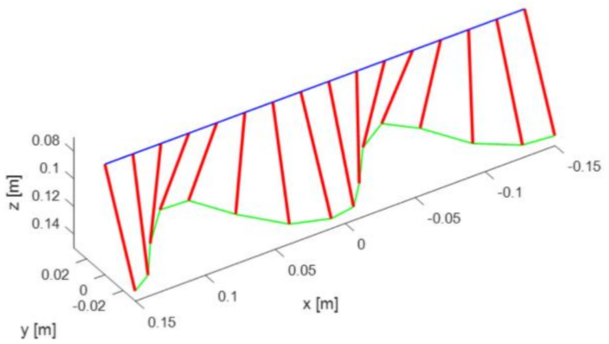
Fig.1. A. Bio-inspired vessel. B. Interior configuration, electronics, and undulating fin in the lower part.

One of the basic motions of the rays is their joint actuation with a given phase difference and maximum amplitude to generate travelling waveforms to interact with the surrounding environment and produce net forces for longitudinal and vertical motion. The physics involved in these maneuvers have been studied and described in the related literature (Shirgaonkar, Curet, Patankar, Lauder, and MacIver, 2008; Curet, Patankar, Lauder, and MacIver; Neveln, Bale, Bhalla, Curet, Patankar, MacIver, 2014).In the present work, the propulsion is obtained by a backward travelling sinusoidal, whose amplitude is modulated to control the speed. The robot is operated in the surface, so its dynamics are confined to lateral-directional. To produce directional forces and torques for yaw maneuvering during the forward motion, this harmonic pattern is modified based on the principle tested in (Uddin and Curet, 2018; Uddin, Garcia, and Curet, 2019), by the superposition of a gradual offsetting of a subset of the rays, located at the end.

## Robot Dynamics

The propulsive fin, as depicted in Fig. 2, consists of a flexible fabric attached to rotating rays. To obtain a dynamic model for initial RL training, this arrangement is mathematically modeled.
The membrane between consecutive rays is linearly discretized, and subsequently each quadrilateral (a skewed rectangle), is subdivided into two triangles from where the forces and moments are ultimately computed. This characterization was inspired in the works done in (Nguyen Phan, Pham, Kim, Nguyen, 2018; Sfakiotakis, Fasoulas, and Gliva, 2015).

Although each ray within the fin can be independently actuated, they are moved harmonically, propagating backward a single sinusoidal wave producing a net thrust. One of the constraints in an arbitrary rotation pattern of the rays is the maximum stretch of the membrane between consecutive rays. This limitation is not important for the case of a travelling sinusoidal wave, as the phase difference between consecutive rays is small, independently of the chosen amplitude, frequency, and wavelength. The location of the rays is considered an input to the model and is prescribed by a dedicated close loop controller connected to each ray motor.



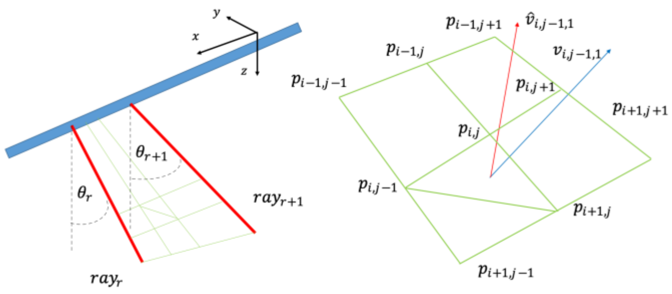


Fig. 2. Geometrical and dynamical modeling of the undulating fin during backward sinusoidal wave progression.

For both triangles within all quadrilaterals, two forces are computed, one based on the normal velocity, $v_n$ associated to drag phenomena $\overrightarrow{f_d}(r,i,j,m,k) = -\frac{1}{2}\rho S_m(r,i,j,m,k)C_d\overrightarrow{v_n}(r,i,j,m,k)|\overrightarrow{v_n}(r,i,j,m,k)|$ characterized by the coefficient $C_d$, and another one using the normal acceleration, linked to added mass effects $\overrightarrow{f_i}(r,i,j,m,k) = -\frac{1}{2}\rho S_m(r,i,j,m,k)C_i\overrightarrow{a_n}(r,i,j,m,k)|\overrightarrow{a_n}(r,i,j,m,k)|$ proportional to coefficient $C_i$, shown in Fig. 2. These two forces together constitute the thrust $\overrightarrow{f_t}$ model delivered by each element, with $\rho$ the density. This is $\overrightarrow{f_t}(r,i,j,m,k) = \overrightarrow{f_d}(r,i,j,m,k) + \overrightarrow{f_i}(r,i,j,m,k)$, shown in Fig. 2. The total force and torque exerted by the fin is computed by the accumulation of all individual forces and individual torques and given by $\overrightarrow{F_F}(k) = \sum_r\sum_i\sum_j\sum_m\overrightarrow{f_t}(r,i,j,m,k)$ and $\overrightarrow{T_F}(k) = \sum_r\sum_i\sum_j\sum_m p_c(r,i,j,m,k) \times \overrightarrow{f_t}(r,i,j,m,k)$.
The description of the aquatic robot is based in the Euler's laws of motion for a rigid body, allowing the device to display a six degree of freedom (6DOF) behavior observing six linear and angular accelerations. For this purpose, two main coordinate systems are set in place, a body frame $\{x,\ y,\ z\}$ fixed attached to the vehicle at the buoyancy center, and an inertial local frame $\{N,\ E,\ D\}$, shown in Fig. 2. The translational and rotational dynamics, with linear velocity components $U$, $V$, and $W$, and angular velocity components $P$, $Q$, and $R$, are described by $\begin{bmatrix} dU/dt \\ dV/dt \\ dW/dt \end{bmatrix} = -\begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}\begin{bmatrix} U \\ V \\ W \end{bmatrix} + \begin{bmatrix} 1/M_x & 0 & 0 \\ 0 & 1/M_y & 0 \\ 0 & 0 & 1/M_z \end{bmatrix}\begin{bmatrix} F_x - D_x \\ F_y - D_y \\ F_z - D_z \end{bmatrix}$ and

$$\begin{bmatrix} dP/dt \\ dQ/dt \\ dR/dt \end{bmatrix} = -\begin{bmatrix} 1/I_{xx} & 0 & 0 \\ 0 & 1/I_{yy} & 0 \\ 0 & 0 & 1/I_{zz} \end{bmatrix} \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix} \begin{bmatrix} PI_{xx} \\ QI_{yy} \\ RI_{zz} \end{bmatrix} + \begin{bmatrix} 1/I_{xx} & 0 & 0 \\ 0 & 1/I_{yy} & 0 \\ 0 & 0 & 1/I_{zz} \end{bmatrix} \begin{bmatrix} T_x - B_x \\ T_y - B_y \\ T_z - B_z \end{bmatrix}$$

with and total mass array defined by $[M_x\ M_y\ M_z]^T = M[1+k_{11}\ \ 1+k_{22}\ \ 1+k_{33}]^T$, for hull mass $M$, and $I_{xx}$, $I_{yy}$, and $I_{zz}$ the principal moments of inertia of the hull, functions of coefficients $k_{44}$, $k_5$, and $k_{55}$, respectively, and the geometry of the hull approximating an ellipsoid, assuming negligible cross moments. Hydrostatic forces $\overrightarrow{F_H}$ and moments $\overrightarrow{T_H}$, including weight and buoyancy, together with the fin make up the total force $\{F_x,\ F_y,\ F_z\}$ and torque $\{T_x,\ T_y,\ T_z\}$ driving the robot. The hydrostatic force is described by $\overrightarrow{F_H} = M(g-b)[-\sin(\theta), \cos(\theta)\sin(\phi), \cos(\theta)\cos(\phi)]^T$, while the torque components are $\overrightarrow{T_H} = [-z_g Mg\cos(\theta)\sin(\phi), -z_g Mg\sin(\theta) - x_g Mg\cos(\theta)\cos(\phi), x_g Mg\cos(\theta)\sin(\phi)]^T$ with $g$ and $b$ accounting for the gravity and buoyancy accelerations, and $\{\theta,\ \phi,\ \psi\}$ being the Euler rotation angles whose dynamics are derived from the angular rates by the equation $\begin{bmatrix} d\phi/dt \\ d\theta/dt \\ d\psi/dt \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}^{-1} \begin{bmatrix} P \\ Q \\ R \end{bmatrix}$.
Hull drag forces $\{D_x,\ D_y,\ D_z\}$ and moments $\{B_x,\ B_y,\ B_z\}$ are defined as proportional to the square of body translational and rotational velocities.

## Robot Actuation

Two actuations are considered to control heading $\psi$ and forward speed $U$ for trajectory tracking. Both control variables are obtained from the manipulation of the fin kinematics.
Although the motors of the fin can be actuated independently to produce an arbitrary shape of the fin, the present work has chosen to drive them conjointly. A backward traveling sinusoidal wave of a fixed frequency $f$, and wavelength $\lambda$, is applied to the fin with a variable amplitude $A$ as a way to modify the surge force and speed, and a variable offset $\alpha$ from the vertical position of the last motors, to regulate the yawing torque and heading of the robot. This last actuation is implemented by the incremental offset deflection added of the last four rays to generate a net yawing torque different from zero.
This idea of the offset deflection in the last rays of the fin, and the change in amplitude applied to all rays, is shown in Fig. 3. It shows a vertical view of the rays and their maximum deflection. The first twelve have no offset and their maximum deflection is vertically symmetric, while the last four have an increasing offset from the vertical breaking the symmetry and thus producing a net torque different from zero. from It was shown in Uddin and Curet, 2018; Uddin, Garcia, and Curet, 2019, that this strategy is a viable means to vary the heading of the vessel during the undulation of the fin. The fin has a total of sixteen rays.

Both actuations are described by the following equation:

$$\theta^r(t) = A(t)\sin\left(2\pi ft - \frac{2\pi(r-1)}{\lambda}\right) + k_\alpha(r)\alpha(t) \tag{7}$$

for $r = 1 \cdots 16$, and $\lambda = (16-1)/2$, and with $k_\alpha$ defined as $k_\alpha(r) = 0$ for $r \le 12$, and $k_\alpha(r) = r - 12$ for $r > 12$. In this way the offset deflection only affects the rays $r = 13 \cdots 16$, with a linear increment, as discuss previously. Before implementation, this equation is discretized.

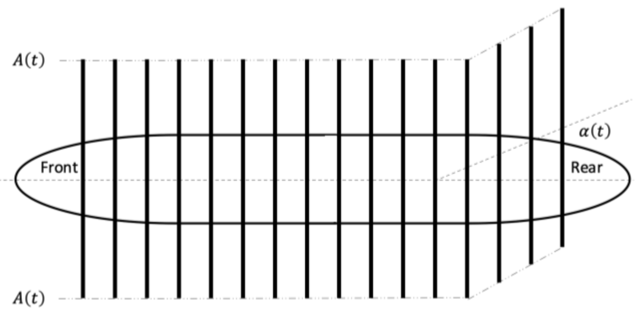


Fig. 3. Fin actuations, including offset deflection for heading control, and amplitude change for speed control.

Figs. 4 shows the relationship between amplitude $A$ of the fin and surge forces $F_x$ and speed $U$. Fig. 5 on the other hand, illustrates the effect of different offset values $\alpha$ in yawing torque $\tau_\psi$ and heading rate $\psi$. These results are obtained in simulation using the dynamic model.

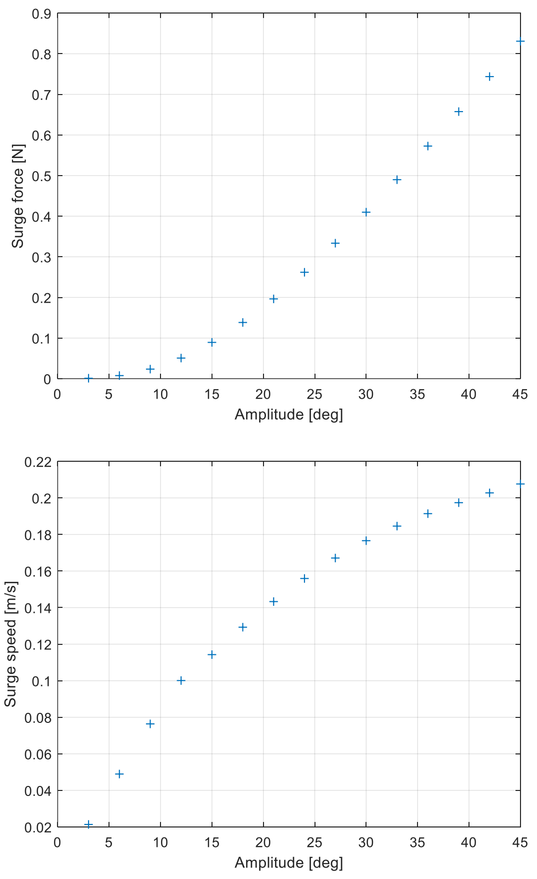


Fig. 4. Surge force and speed as a function of fin amplitude ($f = 2\ Hz$ and $\lambda = 2$).

These two figures show the capability of the two actuators to effectively control the vessel.

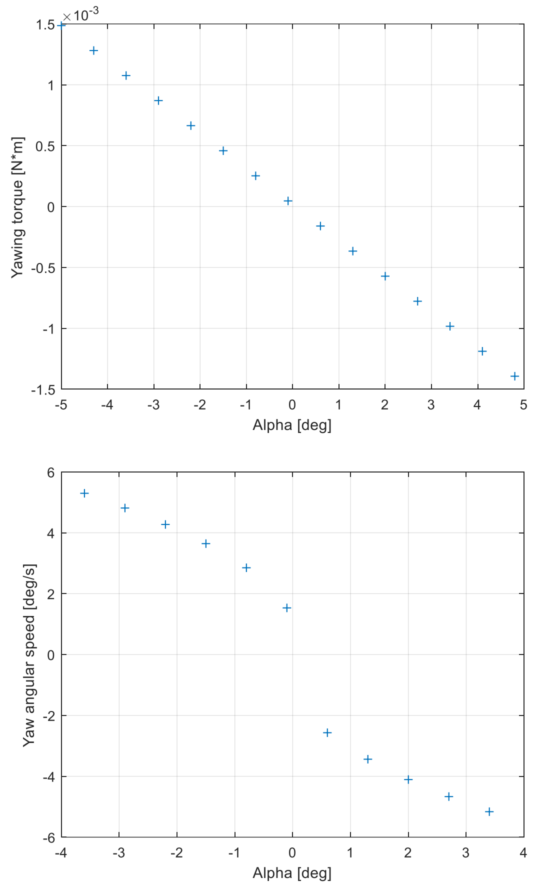


Fig. 5. Yawing torque and yawing angular speed as a function of alpha ($f = 2\ Hz$ and $\lambda = 2$).

## Robot Sensors

To perform trajectory tracking, position and yaw information is required. For normal underwater operation acoustic positioning systems or in some specific cases visual systems can give the required information. In this application the robot is operated at the surface, and the yaw angle is determined by two spatially separated markers whose positions are obtained by an external digital camera, capable of covering the whole trajectory. The trajectory is fully contained within the camera field of view.

The digital camera delivers a sequence of frames to a computer from where the positions are extracted. Each frame consists of the current view field digitized into pixels with three colors associated to each one. These matrices are then used to extract the position, by using thresholds deciding what background is and what the robot is. To help the process of visual tracking markers are installed on the upper part of the robot, which after visual processing are used to provide relative position and heading. The visual tracking is supported by a set of Kalman filters to smooth out possible variability in the raw data.

The sensor information is sent to the robot in real time for its application to the fin. The software code running on a computer sends every cycle a message to the robot encoding the yaw information, consisting in the mismatch between a desired yaw and the current yaw. This is done using a radio link and requires the vehicle to stay on the surface. The message is decoded upon reception, and then applied to an inner controller for the final actuation of the fin.

Frames delivered to the computer consist of two-dimensional arrays $Im$ with a three-valued RGB vector per entry, each one ranging from $0$ to $255$. Fig. 6 shows a sample frame depicting the robot in a small pool with two markers, located over the upper part of the body, signaled by the arrows and centered by rectangles. These rectangles also indicate the area where the filtering algorithm will search for the markers. For each video frame a three-valued RGB two-dimensional array is obtained (for a predefined pixel resolution), then a grayscale array is calculated using a weighted average of the level of the three colors, and finally a binary array is obtained by comparing it to a threshold. Then an image processing algorithm is used to extract the markers and discard clutter.

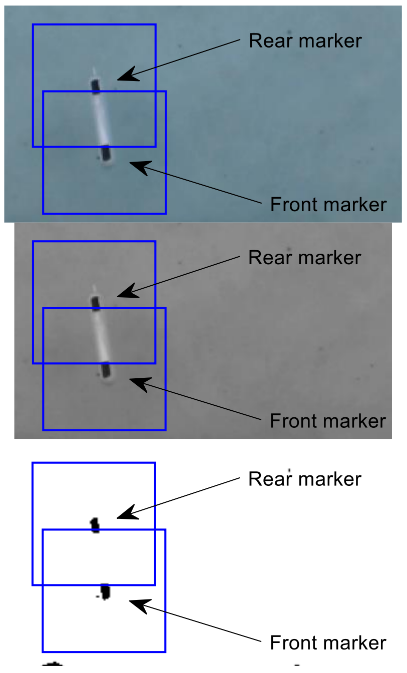


Fig. 6. Sample video frame. Position extraction of markers.

Before the digital filtering and tracking are carried out, visual processing subroutine is used to produce useful numerical data from the frames. Each of these arrays is first converted to a grayscale version with the usual weighted average $0.299R + 0.587G + 0.114B$ (coefficients must add to one). Next step is to then compare it the to a predefined level $\gamma$ (this parameter is different from the one used in the RL algorithm), a design parameter, to obtain a binary matrix of ones and zeros. The value of $\gamma$ has to be obtained empirically as a function of the current conditions of luminosity present on the place. The idea of the comparison is to leave the markers as distinctive closed shapes able to be identified by a common set of pixels from which a centroid can be obtained.

From the binary matrices, the exterior boundary of connected sets of non-zero values are extracted and labeled as distinctive objects, and finally the centroid of each of these objects is computed. The result is a list of 2D points each one representing the center of the existing objects. It is important to notice that due to the unpredictability of the binary matrix, the number of objects can vary from frame to frame. This information is then fed into the digital filtering stage for a final calculation of the position of the robot, using Kalman filters. The whole process is detailed in pseudocode detailed in Algorithms I, II, and III in the Appendix.

Given the stochastic nature of the visual processing stage, the time elapsed between each cycle $\Delta_k$ becomes variable, should not be assumed fixed, and must be measured. The accuracy of the elapsed time for each of the algorithm cycle is vital in the precision of the Kalman filters. In the algorithms $i$ and $j$ are the indices of the number of markers and the

number of centroids, which are not necessarily the same value. The distance units used in the algorithms are pixels, so can be converted to meters, although for yaw computation, is not absolutely necessary. The usual nomenclature for the Kalman filters is used where $\hat{v}_x$ represents an estimated value of the $x - axis$ velocity, and for example $k/k-1$ refers to an estimated quantity at time $k$ given information used up to time $k-1$. The parameters $\alpha$ and $\beta$ are in general adjusted experimentally, they are related to the figure of noise of the system. Larger values produce faster responses amplifying with more abrupt transient changes.
The markers are placed at known longitudinal distances from the geometrical center of the robot. This allows the calculation of the position of the vehicle, from the filtered positions of markers. A simple logic computes the robot position independently of the number of markers and the current reliability and accuracy of the estimated position of each one of them. Although more available position estimates from more markers at any given time implies a better estimation of the robot's position, in the worst case just a single marker's estimated position should be enough to compute the position of the robot. The setting up of the experiment defined the relation between pixels and distance in meters. From at least two reliable marker positions, the heading of the vessel is directly obtained from the relation $tan\,\psi_k = (\hat{p}_{x_{k/k}}^{\,i} - \hat{p}_{x_{k/k}}^{\,i+1})/(\hat{p}_{y_{k/k}}^{\,i} - \hat{p}_{y_{k/k}}^{\,i+1})$.
The yaw information calculated outside the vehicle, is then transferred via RF link. This is not the preferred method for communicating the sensor information as it requires the vessel to be on the surface, severely limiting its operation, where in normal conditions the final sensing should rely on acoustic devices. As the purpose of the present stage of this research is the proof of concept that the undulating fin is able to autonomously control the heading, this temporary solution has been adopted.
The remote sensing algorithm running on the external computer establishes a serial communication with the Master board onboard the robot, by modulating a RF link. Strings of characters are sent serially encoding the yaw information, which in this particular application has been chosen to be the yaw error, this is the difference between the desired yaw and the current one. This election is arbitrary, and this part of the guidance logic could have been written in the robot's embedded code, but by keeping the desired heading coded on the exterior computer, its modification for different heading maneuvers for testing purposes, is greatly simplified.
Each yaw error value is encoded into an ASCII message including its integer part and two decimals. This digital word modulates a 2.4 Giga Hertz carrier and is transmitted. The RF receiver, part of the Master board, demodulates the signal and it is delivered to the microcontroller via an interruption subroutine. After an immediate verification for inconsistency within the message, it is communicated to the Slave boards via an internal common channel, where it is converted into actual offset deflection. These two communications are independent of each other, so their operation is asynchronous. By design, the rate at which the rays are actuated by each slave boards, depends on the chosen operation frequency, but are in this case is 16 time per second. On the other hand, the resultant rate at which the messages are sent to from the laptop to the robot oscillates around 20. The main restriction here is that this last rate should not be lower than the actuation rate, to avoid losing data.

## RL RESULTS

The RL was learned using the mathematical model described before, and then exported to the robot's control system for initial tests. These tests were carried out indoor in a small testing pool of 12 feet diameter.

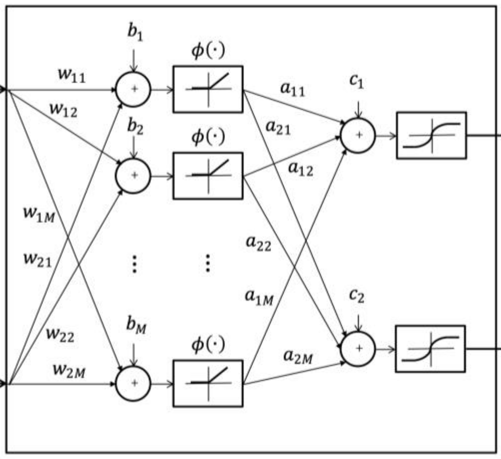
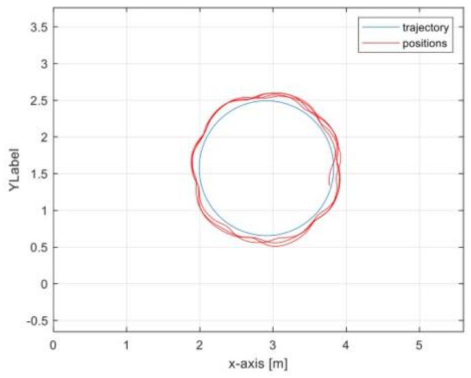
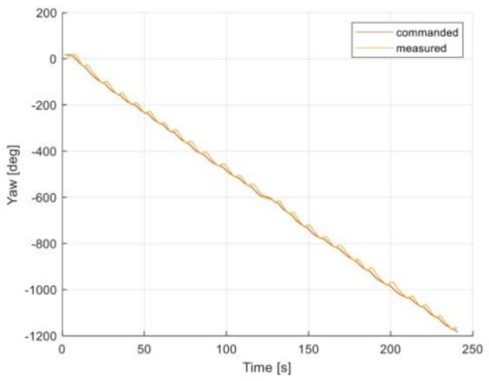
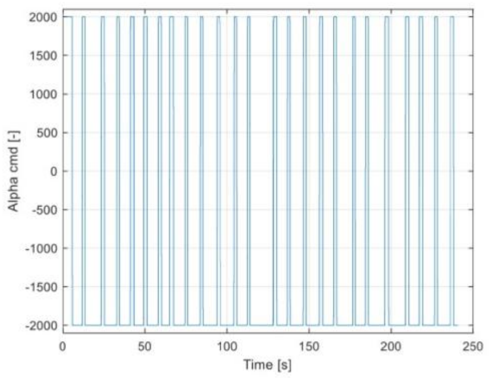


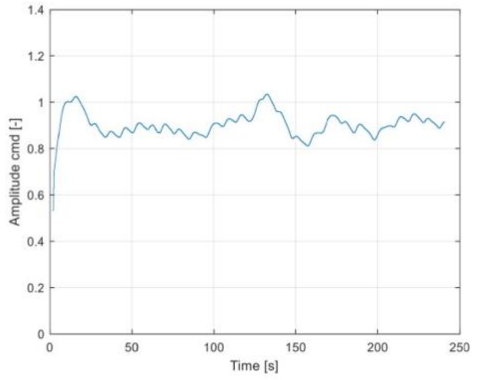
Fig. 7. ANN RL Control Policy $\pi(x_k|\theta_\pi)$. Inputs and outputs are normalized appropriately (not shown).

The ANN control policy $\pi(x_k|\theta_\pi)$, shown in Fig. 7, was designed with a single hidden layer of $M = 32$ neurons, 2 inputs ($q = 2$) and 2 outputs ($p = 2$), a rectified linear unit as the activation function $\phi(z) = max\{0, z\}$. This function has become the most popular activation function, allowing a better training in deeper networks compared to the sigmoid-based functions. It was also argued in (Hahnloser, H., Sarpeshkar, R., Mahowald, M., Douglas, R., and Seung, H., 2000) that it has a strong mathematical and biological motivation. The ANN's parameters to be tuned during training process, are then $\theta_\pi = \{\boldsymbol{w}_1, \boldsymbol{w}_2, a_{11}, a_{12}, \cdots, a_{1M}, a_{2M}, b_1, \cdots, b_M, c_1, c_2\}$. The reward function was designed as a negative quadratic cost of trajectory tracking errors, so the smaller the errors, the less negative, the reward. This choice of reward function is common in control design and serves the purpose of reward equally well as standard positive reward functions.
For path-following, the closest distance from the vessel to the trajectory is obtained every control cycle, from where the two inputs to RL are determined. The closest distance including its sign is converted into a desired yaw angle, $\psi_{cmd}$, and the closest point in the trajectory is associated to a speed $U_{cmd}$. These quantities are compared to actual vehicle states of yaw $\psi$ and speed $U$, to get the error quantities $y_1 = e_U = U_{cmd} - U$, and $y_2 = e_\psi = \psi_{cmd} - \psi$. These values are the two inputs to the RL agent. The RL algorithm works to reduce these two errors by outputting the two actuations $F_1 = A$ and $F_2 = \alpha$. These inputs are normalized before going into the ANN, as well as the outputs, after leaving the ANN (refer to Eq. 5).
The first test consisted of a circular trajectory and a fixed speed. Figure 8 shows the result.









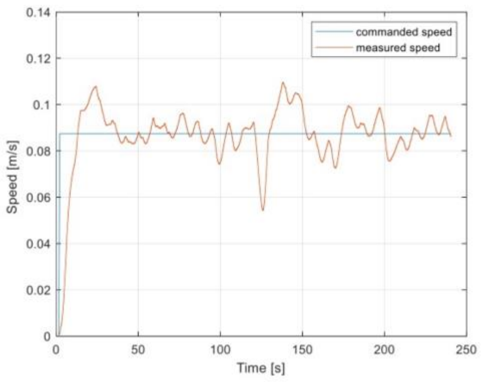


Fig. 8. Trajectory tracking using RL. First test results.

This test was designed to impose a lesser complexity to the controller, consisting of a circular trajectory and a constant speed command. Three complete loops were carried out. The amplitude actuation $A(t)$ is seen to be fairly constant between $0.8$ and $1$, which translates into a range of $24$ and $30$ degrees of maximum deflection, mainly responsible for the speed regulation, in this case close to $9$ $[cm/s]$. On the other hand, it can be seen that both the lateral trajectory tracking, and heading, the intermediate control variable, are also well managed. In this case, and due to the inherent dynamics of the vessel, the deflection of the last rays, $\alpha(t)$, have an abrupt behavior, switching between the extreme values. This remains an important challenge for future designs to reduce this oscillation and improve the overall tracking.

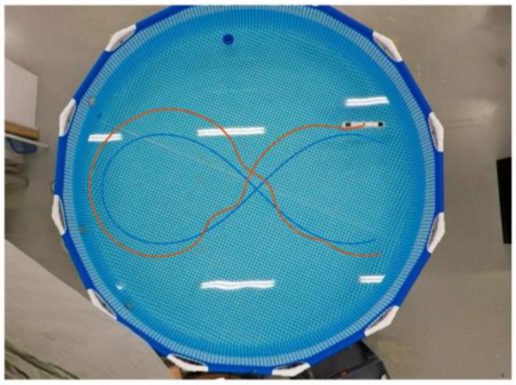
The second trajectory was designed as a lemniscate with varying speed. Results are shown in Fig. 9.

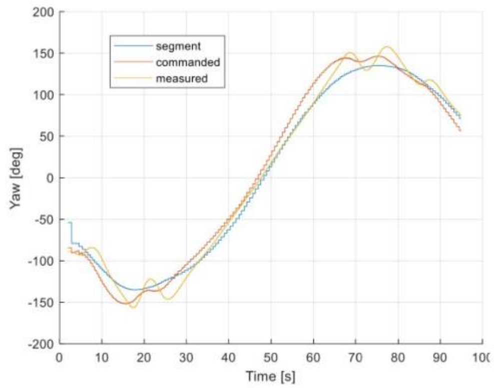
This test was designed to impose a higher level of complexity, by varying both commands together, i.e., the speed command's magnitude, and the lateral distance to the trajectory, in this case switching sign and magnitude.

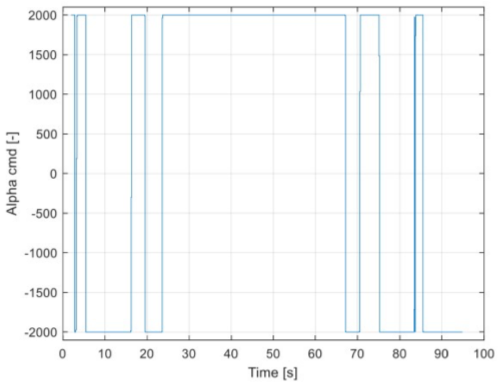
The overall behavior is considered acceptable, where the heading control has the best response. The same effect in the second actuation is present, where the values are changing abruptly between the saturating values. Nonetheless, the control manages to keep the robot fairly close to the trajectory, and desired speed.

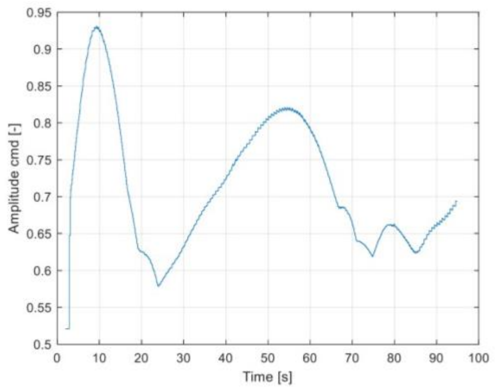
## CONCLUSIONS

A two-input, two-output RL has been successfully designed and tested experimentally in a bio-inspired robot. The training was mainly done virtually using a comprehensive mathematical model including a numerically discretized propulsive fin intended to capture the spatiotemporal and distributed behavior of the different forces and moments. The tuned artificial neural network implementing the optimized control policy was exported to the vessel's control system and tested in two different path following cases, with increasing levels of complexity. RL shows its potential applicability to control maneuverable robots in different scenarios, with increasing levels of difficulty and larger number of sensors and actuations.









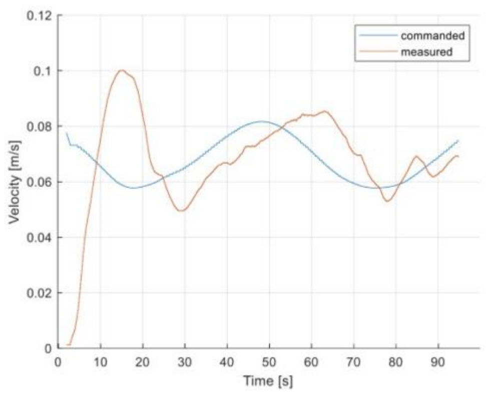


Fig. 9. Trajectory tracking using RL. Second test results.

## ACKNOWLEDGEMENTS

This research is based upon work supported by the National Science Foundation under Grant No. 1751548 awarded to O.C.

## REFERENCES

Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., (2017), *Reinforcement Learning and Dynamic Programming Using Function Approximators*, CRC-Press, Boca Raton, FL, USA.

Curet, O., Patankar, N., Lauder, G., MacIver, M., (2011), “Aquatic maneuvering with counter-propagating waves: a novel locomotive strategy”, *Journal of The Royal Society*, 8, 041-1050.

Hahnloser, H., Sarpeshkar, R., Mahowald, M., Douglas, R., and Seung, H., (2000), “Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit”, *Nature*, 405 (6789), 947–951.

Hornik K, Stinchcombe M, White H, (1989), “Multilayer Feedforward Networks are Universal Approximators”, *Neural Networks*, 2, 359-366.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., (2015), “Continuous control with deep reinforcement learning”, ICLR.

Liu, H., (2017), *Propulsive Performance and Maneuver Control of Undulatory Ribbon Fin Propulsion using Bio-Inspired Robotic Systems*, PhD Dissertation, FAU, USA.

Liu, H., and Curet, (2018), “Swimming Performance of a Bio-Inspired Robotic Vessel with Undulating Fin Propulsion”, *Bioinspir. Biomim*. 13(5):056006.

Neveln, I., Bale, R., Bhalla, A., Curet, O., Patankar, N., and MacIver, M., (2014), “Undulating fins produce off-axis thrust and flow structures”, *The Journal of Experimental Biology*, 217, 201-213.

Nguyen, V., Phan, D., Pham, C., Kim, D., and Nguyen, T., (2018), “Study on Determining the Number of Fin-rays of a Gymnotiform Undulating Fin Robot”, DOI: 10.1007/978-3-319-69814-4_72.

Sfakiotakis, M., Fasoulas, J., and Gliva, R., (2015), “Dynamic Modeling and Experimental Analysis of a Two-Ray Undulatory Fin Robot”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Congress*, Hamburg, Germany.

Shirgaonkar, A., Curet, O., Patankar, N., and MacIver, M., (2008), "The hydrodynamics of ribbon-fin propulsion during impulsive motion", *The Journal of Experimental Biology*, 211, 3490-3503.

Sutton, R. S., and Barto, A. G., (2018), *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA.

Uddin, M., and Curet, O., (2018), "Modeling and Control of a Bio-Inspired Underwater Vessel with Undulating-Fin Propulsion", DOI: 10.1109/OCEANS.2018.860454.

Uddin, M., Garcia, G., and Curet, O., (2019), "Yaw Turning Experiments of a Bio-Inspired Vessel with Undulating Fin Propulsion", 72nd Annual Meeting of the APS Division of Fluid Dynamics, Seattle, WA.

Watkins, C., (1989), *Learning From Delayed Rewards*, King's College, Cambridge, UK.

Watkins, C., and Dayan, P., (1992), "Q-learning", *Mach. Learn.*, 8(3-4), 279–292.

APPENDIXES

## Algorithm I: Position extraction algorithm.

```
1:    Program robot positioning:
2:        Input the number of markers $num_{markers}$ and number of sample times $num_{samples}$
3:        Input Kalman filter parameters $\alpha$ and $\beta$
4:        Input visual processing threshold $\gamma$ and mask length $\delta$
5:        Get snapshot, generate matrix $Im_0$, determine $(\hat{p}_{x_{0/0}}^{\,i}, \hat{p}_{y_{0/0}}^{\,i})$, set $\left(\hat{v}_{x_{0/0}}^{\,i}, \hat{v}_{y_{0/0}}^{\,i}\right) = 0$, set $\Delta_1 = 0$
6:        From $k$ 1 to $num_{samples}$:
7:            Call program visual processing:
8:                Pass $num_{markers}$, $\gamma$, $\delta$, $(\hat{p}_{x_{k-1/k-1}}^{\,i}, \hat{p}_{y_{k-1/k-1}}^{\,i})$
9:                Receive markers measured positions $\left(p_{x_k}^{\,i}, p_{y_k}^{\,i}\right)$
10:           Call program Kalman filter:
11:               Pass $\Delta_k$, $\alpha$, $\beta$, $(\hat{p}_{x_{k-1/k-1}}^{\,i}, \hat{p}_{y_{k-1/k-1}}^{\,i})$, $(\hat{v}_{x_{k-1/k-1}}^{\,i}, \hat{v}_{y_{k-1/k-1}}^{\,i})$, $\left(p_{x_k}^{\,i}, p_{y_k}^{\,i}\right)$
12:               Receive $(\hat{p}_{x_{k/k}}^{\,i}, \hat{p}_{y_{k/k}}^{\,i})$, $(\hat{v}_{x_{k/k}}^{\,i}, \hat{v}_{y_{k/k}}^{\,i})$
13:           Calculate lapsed time $\Delta_k$
14:       end
```

## Algorithm II: Visual data extraction algorithm.

```
1:    Program visual processing
2:        Receive $num_{markers}$, $\gamma$, $mask$, $(\hat{p}_{x_{k-1/k-1}}^{\,i}, \hat{p}_{y_{k-1/k-1}}^{\,i})$
3:        Get snapshot, generate matrix $Im_k$
4:        Generate grayscale matrix $Gr_k = 0.299 Im_k(R) + 0.587 Im_k(G) + 0.114 Im_k(B)$
5:        Generate binary matrix $Bw_k = Gr_k < 255\,\gamma$
6:        Centered in $(\hat{p}_{x_{k-1/k-1}}^{\,i}, \hat{p}_{y_{k-1/k-1}}^{\,i})$ and delimited by $\delta$ mask outside values of $Bw_k$, generate $Bw_k^{\delta}$
7:        Detect connected sets of ones in $Bw_k^{\delta}$, define their centroids $C_k^j = (c_{x_k}^{\,j}, c_{y_k}^{\,j})$
8:        Determine $\left(p_{x_k}^{\,i}, p_{y_k}^{\,i}\right) = \underbrace{min}_{\left(p_{x_k}^{i}, p_{y_k}^{i}\right)} \sqrt{(\hat{p}_{x_{k-1/k-1}}^{\,i} - c_{x_k}^{\,j})^2 + (\hat{p}_{y_{k-1/k-1}}^{\,i} - c_{y_k}^{\,j})^2}$
9:        Return $\left(p_{x_k}^{\,i}, p_{y_k}^{\,i}\right)$
```

## Algorithm III: Kalman filter algorithm.

```
1:    Program: Kalman filtering:
2:        Receive $\Delta_k$, $\alpha$, $\beta$, $(\hat{p}_{x_{k-1/k-1}}^{\,i}, \hat{p}_{y_{k-1/k-1}}^{\,i})$, $(\hat{v}_{x_{k-1/k-1}}^{\,i}, \hat{v}_{y_{k-1/k-1}}^{\,i})$, $\left(p_{x_k}^{\,i}, p_{y_k}^{\,i}\right)$
3         Do:
4             $\left(\hat{p}_{x_{k/k-1}}^{\,i}, \hat{p}_{y_{k/k-1}}^{\,i}\right) = \left(\hat{p}_{x_{k-1/k-1}}^{\,i}, \hat{p}_{y_{k-1/k-1}}^{\,i}\right) + \Delta_k(\hat{v}_{x_{k-1/k-1}}^{\,i}, \hat{v}_{y_{k-1/k-1}}^{\,i})$
5             $\left(\hat{v}_{x_{k/k-1}}^{\,i}, \hat{v}_{y_{k/k-1}}^{\,i}\right) = (\hat{v}_{x_{k-1/k-1}}^{\,i}, \hat{v}_{y_{k-1/k-1}}^{\,i})$
6             $\left(\hat{p}_{x_{k/k}}^{\,i}, \hat{p}_{y_{k/k}}^{\,i}\right) = \left(\hat{p}_{x_{k/k-1}}^{\,i}, \hat{p}_{y_{k/k-1}}^{\,i}\right) + \alpha\left\{\left(p_{x_k}^{\,i}, p_{y_k}^{\,i}\right) - \left(\hat{p}_{x_{k/k-1}}^{\,i}, \hat{p}_{y_{k/k-1}}^{\,i}\right)\right\}$
7             $\left(\hat{v}_{x_{k/k}}^{\,i}, \hat{v}_{y_{k/k}}^{\,i}\right) = \left(\hat{v}_{x_{k/k-1}}^{\,i}, \hat{v}_{y_{k/k-1}}^{\,i}\right) + \beta\left\{\left(p_{x_k}^{\,i}, p_{y_k}^{\,i}\right) - \left(\hat{p}_{x_{k/k-1}}^{\,i}, \hat{p}_{y_{k/k-1}}^{\,i}\right)\right\}$
8:        Return $(\hat{p}_{x_{k/k}}^{\,i}, \hat{p}_{y_{k/k}}^{\,i})$, $(\hat{v}_{x_{k/k}}^{\,i}, \hat{v}_{y_{k/k}}^{\,i})$
```