Bio-inspired Visual Servoing for a Legged Robot NUS MSc Project (2019/2020)

Sun Ge*
e0452783@u.nus.edu
Mechanical Engineering
National University of Singapore

Guillaume Sartoretti
guillaume.sartoretti@nus.edu.sg
Mechanical Engineering
National University of Singapore

ABSTRACT

In this article, we present a control method that combines Central Pattern Generators (CPGs) with visual servoing for a legged robot. We implement this control strategy on two different legged robots to track a target in different scenarios. We propose two methods for onboard image processing, one based on conventional computer vision and one on deep learning. The locomotion of the robot is generated by an onboard CPG framework inspired by vertebrate animals repetitive (cyclic) movements. A vision system acts as the brain of the robot, processing the information collected by an onboard camera, and generates higher-level control commands to the CPG so that the robot can adjust gaits by perceiving changes in the external environment.

KEYWORDS

legged robot, central pattern generator, neural networks, visual servoing, hardware experiments

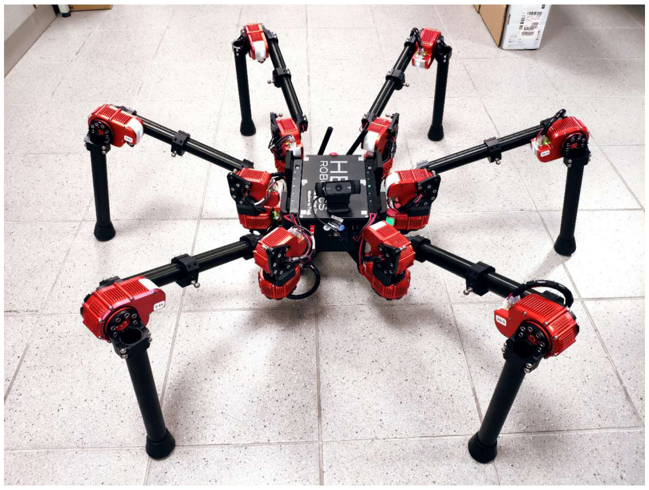


Figure 1: Hexapod robot with a camera onboard

1 INTRODUCTION

Creatures in nature possess very advanced neural control systems and body structures after billions of years of evolution. They can generate coordinated and flexible gaits with feedback from their different senses to walk on various complicated terrains. As the most important and most complex sensory modality, vision plays a vital role in perceiving external environments for creatures [1]. Inspired by creatures, there is a wide interest in developing better locomotor skills based on visual feedback from onboard cameras for legged robots, thereby improving the performance of robots in real-life deployments such as search and rescue as well as mapping and explore dangerous areas where human should not venture [2] [3].

Legged robot locomotion mechanisms, mostly inspired by vertebrate, are very successful in moving through different kinds of terrain [4]. The rhythmic locomotion of vertebrate is usually generated by biological neural circuits in their spinal cord referred to as Central Pattern Generators (CPGs) [5]. CPGs of animals can produce rhythmic motor behaviors such as walking, swimming, flying, and breathing in the absence of sensory input from higher levels in central nervous system [6]. Inspired by vertebrate, CPGs based motion control have been widely used for generating gait for legged robot [7] [8] [9] [10] [11].

Acting as the robot's spinal cord, CPGs produce fast and rhythmic commands, and various sensors perceive the surrounding environment as sensory organs of the robot. In particular, visual feedback control, usually called vision-based control, has been introduced into robotics systems to increase flexibility and intelligence of robot due to the rapid development of image processing [12]. In this work, we use an onboard camera as the sensory organ of the robot to provide visual feedback to CPG model. Our robots can detect, track, and locomote to a target of interset based on visual feedback. In image processing part, we present two different approaches which include conventional computer vision based on color segmentation and a deep-learning method based on the YOLO (You Only Look Once) network. To compare these two methods, we perform experiments on two robots respectively. Due to onboard computation limit, we demonstrate each of these methods on a different hardware.

In this article, we develop a close-loop control model with CPGs and visual feedback for a legged robot. To be more specific, the robot can track a given, potentially moving, object in a complex environment in real-time. The main implementation principle of this model is to adapt the parameters of the CPG model according to the perceived changes in the external environment through the visual feedback from an onboard camera. The hexapod robots, composed of a main body and six articulated legs, are used as experimental platforms in our work.

The structure of this article is as follows: Section 2 presents some previous works about CPG-based controllers and visual feedback control. Section 3 introduces our CPG model and describes the algorithms we used for visual servoing. In section 4, we conduct some experiments based on simulation and robots. Section 5 discusses the experimental results, analyses the performance of our model and discusses future works. Finally, section 6 summarizes our work.

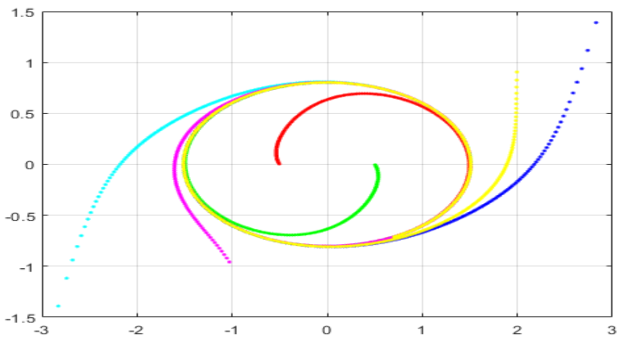


Figure 2: Central Pattern Generator (CPG) model shown to converge to the same limit cycle from different initial positions (different colors) with $a = 1.5$, $b = 0.8$, $\omega = 1$, $\gamma = 0.5$.

2 BACKGROUND

CPG-based controllers for legged robots have been increasingly used in robotics [13] [14] [15] [16]. In particular, *Righetti et al.* presented a CPG model using coupled oscillators for the control of quadruped locomotion [17]. Recent works by *Sartoretti et al.* presented a close-loop method to combine inertial feedback with CPG model for the control of body posture during legged locomotion on unstructured terrain [18]. Our work is based mainly on these two previous works [17] [18]. By changing the parameters of the CPG model, sensory inputs from higher-level control system can be used to adapt the gait generated in real-time.

A few works have proposed to combine CPGs with visual-based control by fuzzy logic controller [19] [20]. Some researchers have developed vision-based control systems by using camera information to track a target or perform general navigation [21] [22]. There, Hough transform or other feature extraction method are needed to identify object or scenes, and these methods are particularly limited in unknown environments. With the rapid development of deep learning, some researchers are interested in combing CPG-based visual control with neural networks [23] [24] [25].

3 VISUAL SERVOING APPROACH

3.1 CPG MODEL

There are different methodologies to create a CPG model to obtain cyclic outputs that can be used to produce rhythmic motor patterns such as swimming, breathing or walking. For example, nonlinear equation [26], artificial neural networks [27], topology [28], etc.. In this work, we rely on a dynamical-system approach to CPGs, in which the pattern generators are expressed as a set of coupled oscillators. Then, we independently control the phases of the oscillations to generate different gaits of the robot. Therefore, establishing the mathematical model of the oscillator is the basis for controlling legged robot locomotion by CPG framework. The mathematical CPG model we used is based on previous works of *Righetti et al.* and *Sartoretti et al.* [17] [18].

We build our CPG model for a hexapod robot. The output values of the oscillators directly represent the joint angles of key articulations of the robot. In hexapod joint space, $x(t) = [x_1(t), x_2(t), x_3(t), x_4(t), x_5(t), x_6(t)]$ was used to present the rotation angles of shoulder joints in the horizontal plane, and $y(t) = [y_1(t), y_2(t), y_3(t), y_4(t), y_5(t), y_6(t)]$ represent the rotation angles of shoulder joints in the vertical plane. We define the trajectory of each shoulder joint $(x(t), y(t))$ as a limit cycle. In order to make the limit cycle variable, we represent it as ellipse [18]:

$$H(x_i, y_i) = \left|\frac{x_i}{a}\right|^2 + \left|\frac{y_i}{b}\right|^2, \tag{1}$$

where a is the semi-major axis of limit ellipse and b the semi-minor axis of limit ellipse (the maximum horizontal and vertical movements of the shoulder joint are a and b respectively). The step height of the robot can be controlled by adjusting b, while the stride length can be controlled by adjusting a.

Using Eq.(1), we express the simple 2D oscillators as:

$$\begin{cases} \dot{x}_i(t) = -\omega \cdot \partial H_{y_i} \\ \dot{y}_i(t) = +\omega \cdot \partial H_{x_i} \end{cases}, \tag{2}$$

where ω is the angular speed, and $\partial H_\zeta = \frac{\partial H}{\partial \zeta}(x_i(t), y_i(t))$.

However, this simple model only exhibits that the oscillators behavior is pure rotation around the origin at a constant radius. To make a fixed-amplitude cyclic motion, we add a constant dissipation to the limit cycle:

$$\begin{cases} \dot{x}_i(t) = -\omega \cdot \partial H_{y_i} + \gamma\left(\mu^2 - H\left(x_i(t), y_i(t)\right)\right) \cdot \partial H_{x_i} \\ \dot{y}_i(t) = +\omega \cdot \partial H_{x_i} + \gamma\left(\mu^2 - H\left(x_i(t), y_i(t)\right)\right) \cdot \partial H_{y_i} \end{cases}, \tag{3}$$

where μ denotes the radius of circular limit cycle, ω denotes the speed of gait cycle, γ denotes the forcing strength. For rhythmic motion on hexapod robot, the limit cycle should always converge to a fixed-amplitude limit cycle, regardless of the initial position of the oscillator. The parameter γ controls the speed of convergence to the limit cycle, which also plays a role when switching between different gaits.

To control the locomotion of a hexapod robot, the coupling between the six legs is crucial. Based on [17], we consider the phase coupling between legs by adding a coupling term in the y-part of the oscillator model :

$$\begin{cases} \dot{x}_i(t) = -\,\omega \cdot \partial H_{y_i} + \gamma\left(\mu^2 - H\left(x_i(t), y_i(t)\right)\right) \cdot \partial H_{x_i} \\ \dot{y}_i(t) = +\,\omega \cdot \partial H_{x_i} + \gamma\left(\mu^2 - H\left(x_i(t), y_i(t)\right)\right) \cdot \partial H_{y_i} \\ \qquad + \lambda \sum_j K_{ij}\left(y_j(t) - c_{y,j}\right) \end{cases}, \tag{4}$$

where K in Eq.(4) is the coupling matrix, which defines the gait by setting the phase relationship between six legs.

In order to allow the model to be applied to different hexapod robot platforms, offsets were added on each joint by modifying Eq.(1) as:

$$H_c(x, y) = \left|\frac{x - c_x}{a}\right|^2 + \left|\frac{y - c_y}{b}\right|^2, \tag{5}$$

where c_x is the offset in horizontal plane, c_y is the offset on vertical plane. By changing c_x and c_y We can adjust the plane where the limit cycle lies, so that it can adapt the hexapod morphology. The final CPG equation used in this work reads :

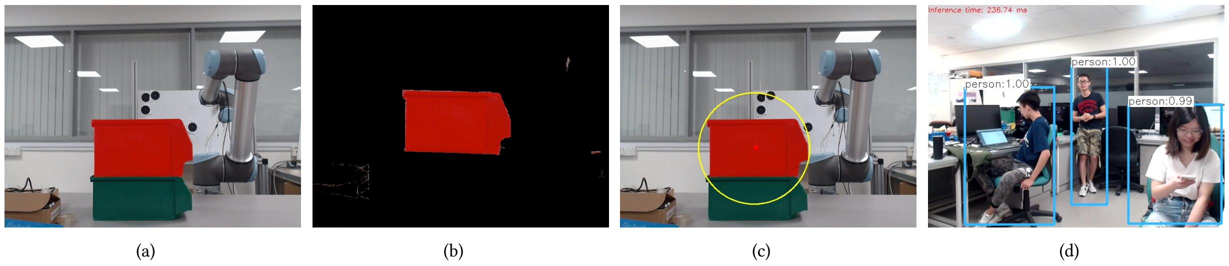


Figure 3: Object detection with two methods. (a) Original Frame: A red box and a green box are placed on the laboratory desk. (b) Red color segmentation based on HSV. (c) Visualized output of the color detection algorithm. (d) Visualized output of the deep-learning algorithm.

$$\begin{cases} \dot{x}_i(t) = -\omega \cdot \partial H_{y_i} + \gamma \left(1 - H_{c_i}\left(x_i(t), y_i(t)\right)\right) \cdot \partial H_{x_i} \\ \dot{y}_i(t) = +\omega \cdot \partial H_{x_i} + \gamma \left(1 - H_{c_i}\left(x_i(t), y_i(t)\right)\right) \cdot \partial H_{y_i} \\ \qquad + \lambda \sum_j K_{ij}\left(y_j(t) - c_{y,j}\right) \end{cases} . \tag{6}$$

In this project, the step size and steering direction of the robot were controlled by changing the parameters a and b in Eq.(5). That is, we allow a higher-level controller to send commands to the CPG based on information obtained from visual feedback from the onboard camera. In detail, a is used to adjust the stride length of each step and b is used to change the height of each step.

3.2 Image Processing

This section introduces the method to extract information from the environment by processing images taken from the onboard camera, and giving a high-level command to the CPG model presented in the previous section. To meet the needs of intelligent control, we present two different approaches for image processing. The first approach is based on conventional computer vision, while the second one is based on recent deep-learning advances.

3.2.1 Object Detection Algorithm. To achieve real-time and high accuracy control, our object detection algorithm should satisfy two conditions: high accuracy of the result, and real-time detection speed. The speed is essential for a rapid response of a low-computing onboard processor on a robot. According to different application scenarios, we present two algorithms used in this project.

Conventional Computer Vision. For relatively simple scenarios, the color of the target is significantly different from the background. Segmentation based on color gives good performance in identifying objects in an image. Additionally, compared with deep-learning method, color-based segmentation is computationally inexpensive.

The basic principle of the algorithm is to rely on thresholding to only keep regions of an image that have a high amount of a color of interest. This way, the background as well as other objects are filtered out and the frame with only the target is obtained as shown in Fig.3 (a) (b) (c).

The frame from the onboard camera is RGB (Red-Green-Blue) image. We first experimented with thresholding based on RGB channel. The result of segmentation does not work like we expected since the RGB values are highly sensitive to illumination. Therefore, we transformed the color space of the images into HSV (Hue, Saturation and Value) [29]. There are three values in HSV color space:

- `Hue`: Encodes color information.
- `Saturation`: Encodes the intensity/purity of color.
- `Value`: Encodes the brightness of color.

RGB is defined based on primary colors. In contrast to RGB, the principle that HSV defined is consistent with the way human perceives color. That is, if the color of the target is known, we should pay more attention to the Hue component. Because the Saturation and Value represent the shades and intensities of the particular color [30]. HSV-based object detection method can adapt to the color change of objects caused by different illumination. After segmenting the object, we extract the largest pixel block in the connected area to prevent noise interference in the picture. We treat the position of the centroid of the pixel block as the coordinates of the object. We calculate the centroid of this pixel block by raw moments:

$$M_{pq} = \int_{a_1}^{a_2} \int_{b_1}^{b_2} x^p y^q f(x,y)\,dx\,dy, \tag{7}$$

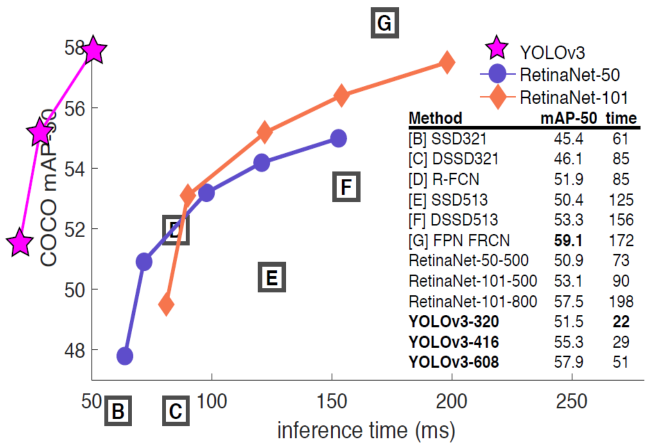
where $f(x,y)$ denotes continuous function of the pixel block,a_1, a_2, b_1, b_2 denotes the Range of the pixel block.

$$\bar{x} = \frac{M_{10}}{M_{00}}, \quad \bar{y} = \frac{M_{01}}{M_{00}}, \tag{8}$$

where $(\bar{x}, \bar{y})$ is the coordinate of object's centroid. Based on the coordinates of the object in frame, the robot can determine the relative position of the target in real world and decide its next action.

Deep Learning for Computer Vision. Object detection based on color is extremely fast. However, for complex scenarios (multiple colors in the background) or for target without color characteristics, color detection does not work. In this subsection, for more intelligent visual servoing, we rely on a neural-based approach to object detection. In particular, we use YOLOv3 as the main object detection algorithm [31].

YOLO is a neural network-based approach for real-time object detection. The object detection task contains two aspects: Firstly, determine the positions of the detected objects by bounding boxes on the image. Secondly, classifying objects in each bounding box.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 4: Adapted from [31], common object detection models speed/accuracy on the mAP (mean Average Precision) at .5 IoU (Intersection over Union) metric. YOLOv3, the model we used in our work, is an extremely fast and accurate model as shown above.

For the previous object detection methods, such as R-CNN and its variants, completed object detection task by a pipeline with through several steps [32] [33] [34]. In contrast, YOLO regards object detection as a single regression problem and straightly detects from image pixels to bounding box coordinates and class confidence. This makes YOLO much faster than other models (as shown in Fig.4). Specifically, the neural network gets an image as input, and the output is a tensor contains bounding boxes and class predictions.

The network structure of YOLO v3 is based on DarkNet-53, which has 53 convolutional layers network trained on an image database called Imagenet for feature extraction [35]. There are another 53 layers stack onto it for the task of detection. 106 layers are fully convolutional underlined architecture for YOLO v3. And the loss function is mainly divided into three parts [31]:

$$Loss = \lambda_1 L_{conf} + \lambda_2 L_{cla} + \lambda_3 L_{loc}, \tag{9}$$

where L_{conf} denotes the object confidence loss, L_{cla} denotes class loss, L_{loc} denotes location loss. λ is the weight of different parts of loss function. Exactly like the original YOLO, the model we used with these losses is trained in a supervised manner.

The input image is divided into a discrete $S \times S$ grid of cells. For each object on the image, the grid cell where the center of the object lies on will be responsible for predicting it. Each grid cell predicts B bounding boxes (usually set as 3 or 5), and each bounding box contains C class probabilities, where C is the number of class in the training dataset. The bounding box prediction contains 5 components: (x, y, w, h, c). x, y denotes the coordinates of the center of the bounding box. w,h denotes the width and height of the bounding box. x, y, w, h are normalized to $[0, 1]$. c denotes the confidence which is defined as:

$$c = Pr(Object) * IOU(pred, truth). \tag{10}$$

If an object exists in the cell, $Pr(Object)$ should be 1, otherwise, it should be equal to 0. $IOU(pred, truth)$ represents the intersection over union between the ground truth and the predicted box. In all, the output of the neural net is a $S \times S \times (B \times 5 + C)$ tensor. We trained it only with one class (person). So, the outputs in our neural network are three $S \times S \times 6$ tensors, where S is equal to 13, 26, and 52 respectively. However, one object may be detected multiple times so there are more than one bounding box for one object. Non-Maximum Suppression (NMS) is used to fix this problem. And the confidence threshold and NMS threshold values are set to select the best bounding box. Then we get the coordinates and class of the object in the image as shown in Fig.3 (d).

3.2.2 *Object Tracking Algorithm.* Object tracking is the process of locating the same objects of interest over multiple frames in videos. To this end, we can assume that the coordinates of the object were obtained via object detection (e.g., like described in the previous section). However, there are some problems for a robot to track a specified target. What will happen if there are more than one object in the frame? In such cases, we cannot always match a specified target in the current frame to the previous frame. Essentially, during object detection, we worked on one image at a time and we had no idea about the motion and past movement of the object, so we cannot uniquely track a specified target by camera in real-time. To solve this problem, we propose to compare and match objects in successive frames based on their relative positions. By calculating the Euclidean distance (Equation.11) between all objects detected in this frame and the target in the previous frame, we can get the result that the object corresponding to the minimum Euclidean distance is regarded as the target of this frame.

$$\text{dist}((x, y), (a, b)) = \sqrt{(x-a)^2 + (y-b)^2}. \tag{11}$$

The object to be tracked is determined by the initialization in the first frame. With that, the robot can continuously track one target even though there exist multiple similar objects in the frame. However, note that this method is still not comprehensive enough because the object tracked can switch if another object passes in front of it. We will improve this algorithm in our future works.

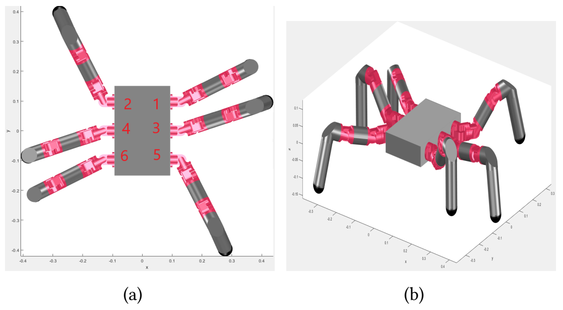


Figure 5: MATLAB Simulation. (a) top view of the hexapod robot at initial position showing the leg numbering convention. (b) tripod gait: three legs always remain on the ground.

3.3 Visual-based Control Method

The output of two image processing methods includes the coordinates (x, y) of centroid of the target in the image. Therefore, we consider the target as a point where its centroid lies on. To find out the relative position of the target respect to the robot, we divide the input image into three areas as shown in Fig.6.

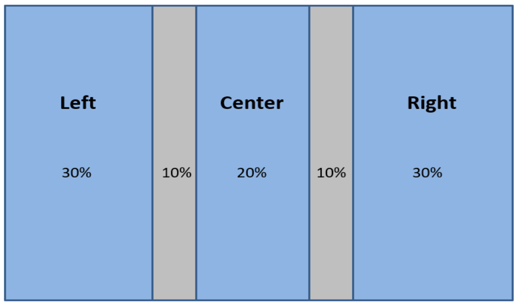


Figure 6: We divide the frame into three areas. Each of blue area corresponds to different relevant parameter in our CPG model. The grey areas are "buffer areas". If center point of the target lies on the grey area, the relevant parameter remain the same with the previous step.

Based on limit cycle equation, we can steer the robot by changing the parameters a (stride length) in the CPG model. If the point (x, y) lies on the right area in the frame, the a of the limit cycle corresponded to the three legs on the right side will be decreased by 20%, and the remaining a for the left side will be increased by 20%, so that the robot can turn to the right. If the point (x, y) lies on the center area of the frame, the values of a on both sides will keep equal to each other, so that the robot could go forward. We set buffer area to improve the fault tolerance of the detection algorithm. If the point (x, y) lies on the buffer area (gray area in Fig.6), a remain the same as the last detection and the gait of the robot will not be changed. The degree of turning can be set by changing the difference of a in right side and left side.

4 EXPERIMENTS

In this section, we conduct experiments on a simulated model and two types of robots respectively. But first, we present the initial parameters of CPG in the experiment and how to combine the output of CPG model with joint angles of hexapod robot to generate locomotion.

4.1 Matlab Simulation

Before going for robot experiments, we use MATLAB simulations to verify the feasibility of the CPG model and tune the relevant parameters. For simulations and experiments, each of the six robot legs has three modular joints. From robot body, the first joint aligns with the yaw axis, second and last joints align with the roll axis. The two most proximal joints act as the shoulder of the robot and correspond to the x and y of the CPG model respectively.

We select the tripod gait for robot in simulation, which means the robot maintains at least three legs on the ground at any given time [36]. The tripod gait has the best performance both in speed and stability of locomotion for hexapod robot [37]. We applied tripod gait to the robot through the K matrix shown as below:

$$K = \begin{bmatrix} 0 & -1 & -1 & 1 & 1 & -1 \\ -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & 1 & 0 & -1 & -1 & 1 \\ 1 & -1 & -1 & 0 & 1 & -1 \\ 1 & -1 & -1 & 1 & 0 & -1 \\ -1 & 1 & 1 & -1 & -1 & 0 \end{bmatrix}.$$

Based on our choice of gait, as well as the robot morphology, we defined initial position x_0,y_0 and constant offsets c_x,c_y as:

$$\begin{aligned} x_0 &= \begin{bmatrix} \frac{\pi}{8} & -\frac{\pi}{8} & -\frac{\pi}{8} & \frac{\pi}{8} & \frac{\pi}{8} & -\frac{\pi}{8} \end{bmatrix}, \\ y_0 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\ c_{x_0} &= \begin{bmatrix} \frac{\pi}{4} & \frac{\pi}{4} & 0 & 0 & -\frac{\pi}{4} & -\frac{\pi}{4} \end{bmatrix}, \\ c_{y_0} &= \begin{bmatrix} \frac{\pi}{16} & \frac{\pi}{16} & \frac{\pi}{16} & \frac{\pi}{16} & \frac{\pi}{16} & \frac{\pi}{16} \end{bmatrix}, \end{aligned} \tag{12}$$

where c_x, c_y determines the plane in which the limit cycle lies on. In fact, we could start the oscillators in nearly any position and they would converge to the right limit cycle we set, but the stability of the robot cannot be guaranteed. Therefore, the initial position of six legs are set on the limit cycle in order for robot to start moving steadily. The initial position of the hexapod robot in simulation is shown in Fig.5(a).

The remaining parameters in CPG model are set with: $\gamma = 1$, $\lambda = 0.3$. The relationship between the output of CPG model and the angle of each joint is as follow:

$$\theta_{1,i} = x_i, \quad \theta_{2,i} = \max\left(y_i, c_{y,i}\right), \quad \theta_{3,i} = f\left(\theta_{1,i}, \theta_{2,i}\right), \tag{13}$$

where $\theta_{1,i}$,$\theta_{1,i}$,$\theta_{3,i}$ are the angles of first joint, second joint and last joint respectively. We define the $y_1 < c_{y,i}$ as the leg is on the ground. In contrast, $y_1 > c_{y,i}$ means the set of legs is in the air. $f(\theta_{1,i}, \theta_{2,i})$ make the trajectory of the end of each leg a straight line, shown in Fig.7, so that the robot able to walk forward.

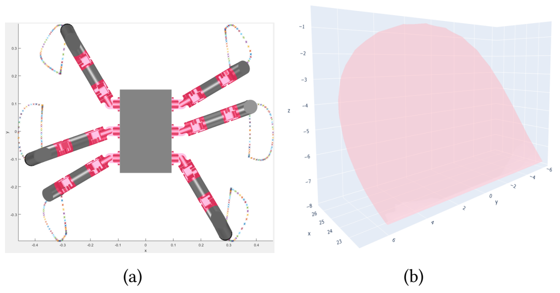


Figure 7: Trajectory of the end of each leg is a straight line. (a) Top view of the trajectory of the end of six legs. (b) Trajectory of the end of leg in three-dimensional space

4.2 Hardware Experiments

In this section, we use two different hexapod Robots to validate our CPG model. We apply the visual servoing based on color detection and YOLOv3 to two hexapod robots respectively. During these experiments, we introduce feedback from visual information into the CPG model to make it a closed-loop system.

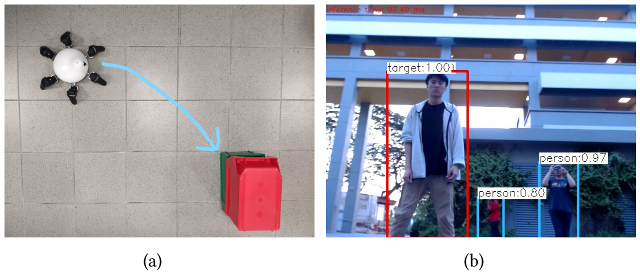


Figure 8: (a) Hexa robot walk towards the red box on the left front with its trajectory (blue line). (b) Daisy perspective when the robot is tracking people (the bounding box in red is target, the bounding box in blue is other objects in frame).

4.2.1 HEXA (Vincross). HEXA is a 6-legged (18 motors), highly maneuverable, compact robot that comes complete with an onboard camera (shown as Fig.8 (a)). We applied our CPG model to it by MIND, a robotics OS and SDK, in Golang programming language. According to the leg structure of the robot, the constant offsets c_x,c_y were defined as:

$$\begin{aligned} c_{x_0} &= \begin{bmatrix} -\frac{\pi}{4} & -\frac{\pi}{4} & 0 & 0 & \frac{\pi}{4} & \frac{\pi}{4} \end{bmatrix}, \\ c_{y_0} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{14}$$

Other CPG parameters are the same as those in the MATLAB simulations. We test our CPG framework based on color detection on HEXA (Fig.8 (a)). The target is placed at 1m, 2m, 5m from the robot respectively. At different distances, the robot can accurately track the position of the target and follow it as the target moves. However, if there is some interference in the background that is similar to the color of the target, the accuracy will decrease as the distance between target and the robot increases. This is the weak point of color-based object tracking. Next, we tested the network-based method on another robot.

4.2.2 Daisy (HEBI ROBOTICS). The robot we used (Fig.1) is a hexapod robotics kit designed by HEBI ROBOTICS. HEBI Robotics' X-Series actuators are used for Modular configuration. Force sensing and position control are available at each joint. We use the HEBI Python API to implement control of each actuator's motion and connected an external Logitech camera (C930c) as an onboard vision sensor. The c_x and c_y were defined as 0 based on the structure of the joints. Other CPG parameters are the same as those in MATLAB simulation.

We trained our own version of the YOLOv3 neural network on the COCO dataset, which has 80 classes. We only used one class (the 'person' class) in our experiment, and the object tracking algorithm was used to handle situations where multiple targets appear in the frame at the same time (Fig.8 (b)). We tested neural network-based visual control models in complex environments. We change the distance between robot and target from 0.5m to 5m. The result shows that the robot can continuously track the target even in complex environments and is not affected by another similar object in the frame. A video of the experiments can be found at https://youtu.be/hmV5AVvZPf0.

5 DISCUSSION

We tested two detection models with two different Robots. For the model based on conventional computer visionit takes 45ms and 33ms for HEXA and Daisy to process one frame respectively. The model based on deep-learning method takes 100ms to process one frame when it is running on Daisy. The neural network-based model was slower than color-based model when it works on Daisy. However, the performance of robots with two models were almost same in experiments. Because the speed of the target is limited in normal scenarios. High-speed moving targets are not considered in this work.

Table 1: Frame rate of two object detection models

Model	Processor	frames per second
HSV color	ARM Cortex-A9 (HEXA)	22fps
HSV color	Intel i3-8109U (Daisy)	30fps
YOLOv3	Intel i3-8109U (Daisy)	10fps

Two methods of image processing are suitable for different scenarios. The conventional computer vision method can be used in scenarios where the background is relatively simple, and the color characteristics of the tracked objects are distinguishable from the environment, thus achieving faster speed with less computing power. The deep-learning method can be used in complex scenarios for hard missions. However, the deep-learning method is computationally expensive. Therefore, our next step is to simplify the network and increase the speed of the network. We will try to implement some shallow networks both on HEXA and Daisy, so that the robot could track the high-speed moving target.

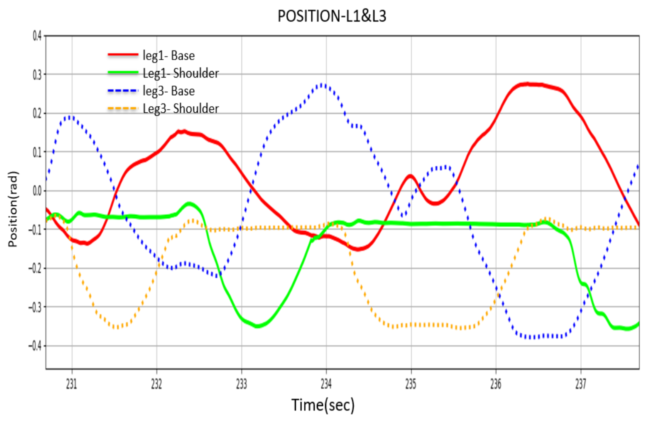


Figure 9: The positions of Leg1 and Leg3 when locomotion changed from forward to turning (joint1 and joint2).

To further improve the performance and simplicity of the visual-based control architecture for the robot, we believe that the next step is an end-to-end neural network based on the existing network model. That is, the input of the neural network would be the frame and the output would directly be the parameters of the CPG model. Then the visual feedback can be further combined with inertial feedback for more intelligent control.

6 CONCLUSION

In this paper, we built a CPG model for hexapod robot and presented two approaches for visual servoing of a hexapod robot using a CPG framework. We showed how the gait of a robot can be changed by adjusting the relevant parameters of the CPG model. Inspired by vertebrate animals, vision sensor with neural networks acts as brain of the robot to send a high-level command to the CPG model which acts as the spinal cord of the robot to generate locomotion. Specifically, this work presents two approaches to visual-based control. For simple scenarios, we can use the method based on conventional computer vision that requires less computing power. Second, our deep-learning-based method can handle more general tasks in complex environments but at the cost of more computing power. We implement our approaches on two types of hexapod robot respectively. Experimental results showed that the robot can change gait smoothly based on visual information in real-time and exhibited good performance even in a complex scenario.

We believe that visual control can further improve the performance of robots from two main aspects. Firstly, this approach will improve the environmental adaptability of the robot. The robot can get information about the environment through an onboard camera, such as terrain, obstacle and scenarios. According to the information from vision, the robot can adjust the gait to avoid obstacles or change gait to suit the complex terrain. Based on the convergence characteristics of the limit cycle, the robot can switch gait smoothly. Fig.9 shows that joints position feedback of the first and third legs joints when the limit cycle of the robot is changed. The limit cycle's parameters is changed at t=235, then the amplitude of oscillation for the base joint is increased smoothly, showing smooth convergence.

Secondly, combining vision sensor with an advanced neural network, the robot can complete different complex tasks intelligently, for example, the identification, detection, and tracking in different scenarios. It also can be used in search-and-rescue scenarios, such as in buildings damaged by earthquake or fire.

REFERENCES

[1] Rebecca J Hirst, Lucy Cragg, and Harriet A Allen. Vision dominates audition in adults but not children: A meta-analysis of the colavita effect. *Neuroscience & Biobehavioral Reviews*, 94:286–301, 2018.
[2] Aaron M Hoover, Samuel Burden, Xiao-Yu Fu, S Shankar Sastry, and Ronald S Fearing. Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot. In *2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 869–876. IEEE, 2010.
[3] Jeffrey Ackerman and Justin Seipel. Energetics of bio-inspired legged robot locomotion with elastically-suspended loads. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 203–208. IEEE, 2011.
[4] Horacio Rostro-Gonzalez, Pedro Alberto Cerna-Garcia, Gerardo Trejo-Caballero, Carlos H Garcia-Capulin, Mario Alberto Ibarra-Manzano, Juan Gabriel Avina-Cervantes, and César Torres-Huitzil. A cpg system based on spiking neurons for hexapod robot locomotion. *Neurocomputing*, 170:47–54, 2015.
[5] Sten Grillner and Peter Wallen. Central pattern generators for locomotion, with special reference to vertebrates. *Annual review of neuroscience*, 8(1):233–261, 1985.
[6] Eve Marder and Dirk Bucher. Central pattern generators and the control of rhythmic movements. *Current biology*, 11(23):R986–R996, 2001.
[7] Alexander Sproewitz, Lorenz Kuechler, Alexandre Tuleu, Mostafa Ajallooeian, Michiel D'Haene, Rico Moeckel, and Auke Ijspeert. Oncilla robot: a light-weight bio-inspired quadruped robot for fast locomotion in rough terrain. In *5th International Symposium on Adaptive Motion of Animals and Machines*, number CONF, 2011.
[8] Paolo Arena, Luigi Fortuna, Mattia Frasca, and Giovanni Sicurella. An adaptive, self-organizing dynamical system for hierarchical control of bio-inspired locomotion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(4):1823–1837, 2004.
[9] Shinkichi Inagaki, Hideo Yuasa, and Tamio Arai. Cpg model for autonomous decentralized multi-legged robot system—generation and transition of oscillation patterns and dynamics of oscillators. *Robotics and Autonomous Systems*, 44(3-4):171–179, 2003.
[10] Shinkichi Inagaki, Hideo Yuasa, Takanori Suzuki, and Tamio Arai. Wave cpg model for autonomous decentralized multi-legged robot: Gait generation and walking speed control. *Robotics and Autonomous Systems*, 54(2):118–126, 2006.
[11] Bernhard Klaassen, Ralf Linnemann, Dirk Spenneberg, and Frank Kirchner. Biomimetic walking robot scorpion: Control and modeling. *Robotics and autonomous systems*, 41(2-3):69–76, 2002.
[12] Yongchun Fang, Xi Liu, and Xuebo Zhang. Adaptive active visual servoing of nonholonomic mobile robots. *IEEE Transactions on Industrial Electronics*, 59(1):486–497, 2011.
[13] Alessandro Crespi and Auke Jan Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *IEEE Transactions on Robotics*, 24(1):75–87, 2008.
[14] Jörg Conradt and Paulina Varshavskaya. Distributed central pattern generator control for a serpentine robot. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, pages 338–341, 2003.
[15] Daisy Lachat, Alessandro Crespi, and Auke Jan Ijspeert. Boxybot: a swimming and crawling fish robot controlled by a central pattern generator. In *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, pages 643–648. IEEE, 2006.
[16] Jun Morimoto, Gen Endo, Jun Nakanishi, S Hyon, Gordon Cheng, Darrin Bentivegna, and Christopher G Atkeson. Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1579–1584. IEEE, 2006.
[17] Ludovic Righetti and Auke Jan Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *2008 IEEE International Conference on Robotics and Automation*, pages 819–824. IEEE, 2008.
[18] Guillaume Sartoretti, Samuel Shaw, Katie Lam, Naixin Fan, Matthew Travers, and Howie Choset. Central pattern generator with inertial feedback for stable locomotion and climbing in unstructured terrain. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–5. IEEE, 2018.
[19] Jose Hugo Barron-Zambrano, Cesar Torres-Huitzil, and Jose Juan Garcia-Hernandez. Fpga-based cpg robot locomotion modulation using a fuzzy scheme and visual information. In *2011 International Conference on Reconfigurable Computing and FPGAs*, pages 291–296. IEEE, 2011.
[20] Feihu Sun, Junzhi Yu, De Xu, and Ming Wang. Target tracking of robotic fish based on embedded vision and cpg model. In *2013 IEEE International Conference on Information and Automation (ICIA)*, pages 1206–1211. IEEE, 2013.
[21] Paulo Borges, Robert Zlot, Michael Bosse, Stephen Nuske, and Ashley Tews. Vision-based localization using an edge map extracted from 3d laser range data. In *2010 IEEE International Conference on Robotics and Automation*, pages 4902–4909. IEEE, 2010.
[22] Arati Gopalakrishnan, Sheldon Greene, and Ali Sekmen. Vision-based mobile robot learning and navigation. In *ROMAN 2005. IEEE International Workshop on*

Robot and Human Interactive Communication, 2005., pages 48–53. IEEE, 2005.

[23] Guillaume Sartoretti, William Paivine, Yunfei Shi, Yue Wu, and Howie Choset. Distributed learning of decentralized control policies for articulated mobile robots. *IEEE Transactions on Robotics*, 35(5):1109–1122, 2019.

[24] Takeshi Mori, Yutaka Nakamura, Masa-Aki Sato, and Shin Ishii. Reinforcement learning for cpg-driven biped robot. In *AAAI*, volume 4, pages 623–630, 2004.

[25] Yutaka Nakamura, Takeshi Mori, Masa-aki Sato, and Shin Ishii. Reinforcement learning for a biped robot based on a cpg-actor-critic method. *Neural networks*, 20(6):723–735, 2007.

[26] Yasuhiro Fukuoka, Hiroshi Kimura, and Avis H Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research*, 22(3-4):187–202, 2003.

[27] Paolo Arena and Luigi Fortuna. Collective behaviour in cellular neural networks to model the central pattern generator. *International Journal of Systems Science*, 31(7):827–841, 2000.

[28] MG Felipe, F Yang, and Zhijun Yang. Building artificial cpgs with asymmetric hopfield networks. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 4, pages 290–295. IEEE, 2000.

[29] JR Smith. Color for image retrieval. image databases: Search and retrieval of digital imagery. v. castelli and r. baeza-yates, 2002.

[30] Shamik Sural, Gang Qian, and Sakti Pramanik. Segmentation and histogram generation using the hsv color space for image retrieval. In *Proceedings. International Conference on Image Processing*, volume 2, pages II–II. IEEE, 2002.

[31] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[32] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[33] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[35] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[36] Pavan Ramdya, Robin Thandiackal, Raphael Cherney, Thibault Asselborn, Richard Benton, Auke Jan Ijspeert, and Dario Floreano. Climbing favours the tripod gait over alternative faster insect gaits. *Nature communications*, 8(1):1–11, 2017.

[37] Dariusz Grzelczyk, Bartosz Stanczyk, and Jan Awrejcewicz. Kinematics, dynamics and power consumption analysis of the hexapod robot during walking with tripod gait. *International Journal of Structural Stability and Dynamics*, 17(05):1740010, 2017.