Reinforcement learning and convolutional neural network system for firefighting rescue robot

Tien Kun Yu, *Yang Ming* Chieh and *Hooman* Samani*

Electrical Engineering, National Taipei University, 23741 Sanxia New Taipei City, Taiwan

Abstract. In this paper, we combine the machine learning and neural network to build some modules for the fire rescue robot application. In our research, we build the robot legs module with Q-learning. We also finish the face detection with color sensors and infrared sensors. It is usual that image fusion is done when we want to use two kinds of sensors. Kalman filter is chosen to meet our requirement. After we finish some indispensable steps, we use sliding windows to choose our region of interest to make the system's calculation lower. The least step is convolutional neural network. We design a seven layers neural network to find the face feature and distinguish it or not.

1 Introduction

As the advancement of robot technologies, people use robot to execute the high risk tasks for human beings. Hence, it is usual for robots entering the high-risk area. Take the earthquake for example, it is unavoidable for people to enter the collapsed building or burning house to search survivor. However, the structure for the building is too weak, when aftershock strikes, it may cause huge lost. To prevent this situation, people try to build robot to keep the rescue team from danger.

For this paper, we focus on fire situation. We want to design a hexapod robot with color sensors and infrared image sensors for firemen to check how many survivors was trapped and the environment. In our opinion, four legs robot would be unstable if it losses any leg of it. However, eight legs robot's locomotion is too complex to move fast. Without a doubt, hexapod robot is the best choice.

This paper has two parts of robot design, locomotion control and human detection. For the controlling part, we used Q-Learning, a model-free method base on the reinforcement learning technique, achieve it's control motivation by interacting environment and maximize the reward. The reinforcement learning to provide maximize reward based on the action. Make the hexapod learned the gait pattern by Q-Learning, with the reward given by the speed and direction. In natural, insect with hexapod has two different gait tripod and bipod, tripod, in every moment there are three legs on the ground which is much slower than bipod every moment there are only two legs on the ground. The hexapod in this paper is used to search survivor in the fired area, so the searching must be done as quickly as it can, with the heavy load it has, the speed by tripod will be unreliable, but compare the tripod and bipod, tripod has better stability than bipod which is also necessary for the robot. With the Q-Learning, hexapod can find the best gait for speed and stability, not only limited by bipod and tripod. Besides learning the gait for robot, robot might stranded by the unknown terrain and obstacle cause by fire, to conquer unpredictable situation, when hexapod detect the position was stranded, Q-Learning will updating the movement immediately, keeping the searching go smoothly. On the other side, we try to search the trapped in the burning house by the near infrared sensors and the color sensors. We use kalman filter to combine the information to get better result than the data only be gotten by one kind of sensor. Although we train our convolutional neural network (CNN) by IIIT-Disguise Face Database [1, 2], which provides lots of faces pictures with infrared and color version, with 130*150 pixels. But our images' size is different from the form. Thus, we would choose suitable region of interested (ROI) to meet the input limitation of CNN. To find out the ROI, we use the sliding windows concept. It would cut image into a lot of pieces. But every sub image has some overlapping part. It is necessary to avoid the face part from being cut. It also can make the detection part faster. The traditional sliding windows only move one pixel or one column in a loop. However, we consider it is too similar to change the CNN's result. We set the window's moving rules that the window would move 13 columns or 15 rows. It is obvious that the calculation can be lower.

In this paper, backgrounds would be mentioned at section two. There are some different methods other researcher done. Section three's content is about our method. We design some experiment for the system and make sure it is work. At section four, we would show the result and explain the experiments. At the final chapter, there is the conclusion.

* Corresponding author:hooman@mail.ntpu.edu.tw

2 Background

There are so many people tried to make the dream, which we can save everyone even we don't need to go into a crisis section, comes true. Like "Control Architecture Design for a Fire Searching Robot using Task Oriented Design Methodology" [3], which is published at 2006, it focuses on three parts on its robot. The first one is Stable mobility in a hazardous and uncertain environment. The second one is providing accurate information regarding the fire and the people in danger. The last one is Safety of the operator, and convenience in operation. To meet the requirement, the robot is designed based on task oriented design (TOD). The robot will be controlled by wireless controller. The operator could watch real time environment by the images which taken by the camera. It means the robot is not an automatic or semi-automatic system. We still need a body to control it. Using a robot to search whole environment maybe is not a good solution for a rescue mission. So "Multi-Robot Exploration and Fire Searching" [4] tries to solve the efficiency issue by mult-robot. It uses improved A* algorithm to get the target way-points. It applies employs a decentralized frontier based exploration method according to these way-points. How to avoid obstacle is practiced by potential field method. The communication between all robots is by decentralized control way. These robots share information and make a complete and accurate map. By simulation, it can get a good result in an unknown environment `To make the rescue speed faster, unmanned aerial vehicles (UAV) also are good choices. "Semi-Autonomous Indoor Firefighting UAV" [5] improved it can finish the entire searching mission with some help of human beings. Moreover, it could carry a fire extinguisher to cool the trapped in order to prevent them from the hurt caused by high-temperature. With carbon fiber and G10, its structure is fireproofed, low-density and strong. To keep the inner equipment from broken, it also has a thermoelectric cooling system. And the communication system between remote controller with UAV is done by radio control. According to the experiment the UAV can work steadily when the environment temperature is less than 1300 degrees. Moreover, the link between the controller with UAV can reach 500 meters. But it still need to somebody to check the monitor.

3 Method

In this section, we show the details of all the process including the movement part, the detection part and sliding windows. At first, we would like to show that how the robot moves by Q-learning. After then, we will explain the convolutional neural network and sliding windows concept detailed.

3.1 Locomotion

The joint circled by red circle is the joint we control in this paper. For this paper, we only control one joint to learn the walking pattern, we will finish the rest later.

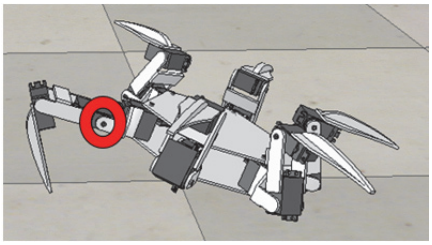


Fig. 1. Joint for Hexapod.

Control the joint roatation for the hexapod movement by the reinforcement of Q-learning. The Q-learning algorithm.

$$a_t) \leftarrow (1 - \alpha) \cdot (r_t + \gamma \cdot \max Q(s_{t+1}, a_t)) \qquad (1)$$

We define the state: θ ,by the rotate for the joint, set the horizontal from previous object to θ = 0.

Fig. 2 Define theta.

Every state have two action increase 20° or decrease 20°. a_t is the action for every state.Each state has two action, each action has it's own Q value ,which updated during the learning by algorithm and store inside the Q-table. Q-table[state,action]=[Q-value], θ+20° is countercloclwise rotation -20° is clockwise rotation.

Table. 1 Q-table for State-Action Action (Next State).

state	40°	20°	0°	-20°	-40°
20°	State20 - action2		State20 - action1		
0°		State0 - action2		State0 - action1	
-20			State-20 - action2		State-20 - action1

The Q-value is used to determine the action that which is best for the movement, the value will rapidly update during learning, when learning beacome stable, the value will converge. Like the above table, state0-action1 and state0-action2 will have different value. When the joint position was set to 0°, program will choose the higher Q-value value from one of them, and set the position to the choosen one.

The α from the algorithm is the learning rate, determine the importance for the new information, α should be low at the beginning, and increase the value along with the learning

become stable. γ the discount factor, determine the importance of future reward, if set gamma=0, the Q-learning will be short sight, may not learn the perfect movement, because it only choose the easy way to get the reward, which may cause bad result, if set γ=1, the result might be divergent. Every parameter change the learning result more or less. Adjusting the value will be neccisarily.
The most important part for Q-learning is the reward, the reward cause very different result for learning. The reward for the learning is determine by the loaction for hexapod change by every action. When the θ change, matlab will pass the θ to V-rep and Hexapod will rotate the joint order by the θ, and read the current location for Hexapod and pass back to matlab, compare the current location and previous location to set the reward. The forward directioin is set to positive x-aixs. The reward is given by below. First location is the starting location when simulation begin, if current position less than current position means the gait is still backward so give the largest negative reward.

Table. 2 Define reward.

Situation	Reward
Current location > previous location	Reward = 20
The action makes hexapod move forward.	
Current location < previous location	Reward = -5
Also compare to first location, if current location < first location reward = -100.	
Current location = previous location	Reward =0
If the action is converge to regular, but the location didn't increase the reward will change negative. Reward = -10.	

To find the fastest gait, there is also a reward is given by the velocity. Each action will have current location and previous location, substract them, it can seem to be the velocity for the action, velocity=distance/time, time for every action can approximate to the same. If current location larger than previous location, the reward for the velocity will add into the reward for the action.

To read the Q-table and each value easier, we use matlab to compile the moving module, use V-rep to run the simulation, the controlling value given by matlab will pass to V-rep and return the data back to matlab, and pass to program to calculate the data and update the Q-value.
When the joint increase or decrease 20°, matlab will take the location for hexapod from V-rep, and determine the location and update the Q-value, as the program started, first rotate the joint from 0° to 120° and decrease to -120° and back to 0°, and record the movement and update Q-value, it will have a initial Q-table, these help the learning much faster than only by random action, after the schedule rotation, the program start to have random action, the action will compare the random action with every action's Q-value and choose the largest, beside the current action, it also compare the next action for next position, which can get the highest reward, and make it under consideration for current action.

3.2 Vision

3.2.1 Sensor fusion

Because we use two kinds of sensors, we need to combine the color images with infrared images. We use extended kalman filter as our solution. Kalman filter is a kind of efficient recursive filter. It always is used on denoising or predicting. There are some functions for kalman filter to finish its mission. Kalman filter has two parts. The first one is predicting, the second one is updating. For the predicting part, we use follow two formulas [6]

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \quad (2)$$

$$P_k^- = AP_{k-1}A^T + Q \quad (3)$$

There are some definitions as bellowed. $\hat{x}_k^-$ means the uncorrected state matrix at time k. u_{k-1} means the control matrix at time k-1. The formula (2) can get the state from the previous moment. A is the state transfer function. B is the control gain. Q is the noise covariance P_k^- is the uncorrected estimated covariance matrix. By formula (1) and (2), we can predict the following state by previous state, but we hope we can get a more precise estimation. There are three functions to correct the variable X^- and P^-

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}, \quad (4)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(Z_k - H\hat{x}_k^-), \quad (5)$$

$$P_k = (1 - K_k H)P_k^-. \quad (6)$$

According to formula (4), we calculate the variable kalman gain (K_k). It is an important value for system to correct the final output. If the kalman gain is low, it means the output is closer to predicting measurement than the real situation. Otherwise, the output is closer to the actual measurement. After updating the measurement, updating the estimated covariance would be done. Although kalman filter is a time-domain filter, we transfer the pixels from a 2D image to an array to make our images fit the form.

3.2.2 Convolutional Neural Network

After sensor fusion, we get some image information for convolutional neural network. We train the convolutional neural network by the face pictures which provided by IIIT-Disguise Face Database [1, 2]. But its images' size is 130*150. So, we use sliding windows to found out the face

shown in any place in the pictures. Why the CNN structure is chosen is because CNN has the feature that it can simplify the calculation.

The first layer is input layer. It allows us to get the test data and training data. The second layer is a convolutional layer. It has twenty kernels. Every kernel is a 5*5 filter to find out the features in the input layer. The formula (7) would be calculated by the weights are showed at bellowed:

$$F_2(x) = \sum_{i=1}^{5}\sum_{j=1}^{5} x_{ij} \times \omega_{ij}. \quad (7)$$

ω_{11}	ω_{12}	ω_{13}	ω_{14}	ω_{15}
ω_{21}	ω_{22}	ω_{23}	ω_{24}	ω_{25}
ω_{31}	ω_{32}	ω_{33}	ω_{34}	ω_{35}
ω_{41}	ω_{42}	ω_{43}	ω_{44}	ω_{45}
ω_{51}	ω_{52}	ω_{53}	ω_{54}	ω_{55}

Fig. 3. Filter weight position.

The third layer is a Rectified Linear Units (ReLU) layer. Our data will not be changed if the data is positive. However, the negative data is replaced by zero just like function (8) because the negative data means the feature is too different to use. We delete the information to lower system loading and speed up the calculation:

$$F_3(x) = \begin{cases} x, if\ x > 0 \\ 0, if\ x < 0 \end{cases} \quad (8)$$

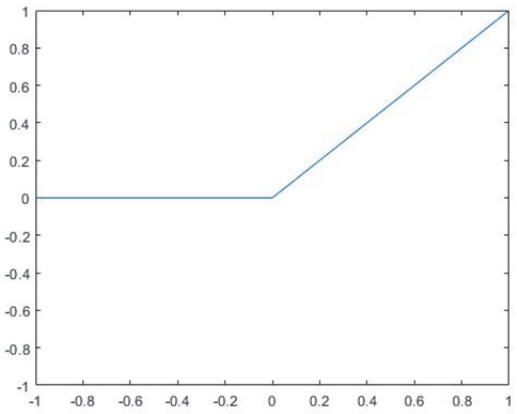


Fig. 4. ReLU function.

After then, the next layer is pooling layer. This pooling layer is max pooling. It is another method to relieve the system loading. We use 2*2 maximum filters to choose the maximum value as the layer's output. The image only gives quarter information to fifth layer.

$$F_4(x_{i,j}) = \max(x_{i,j}, x_{i+1,j}, x_{i,j+1}, x_{i+1,j+1}). \quad (9)$$

At fifth layer, it is a fully connected layer. It turns the image into an array and connects by the perception structure. The last two layers are softmax layer and classification layer. Its missions are calculating weights and finish detection separately. Classification layer is a standard to detect the image is face image or not. If the value is higher than the standard, it is a face image. If not, the image doesn't have face. The weight is from softmax layer. Softmax layer uses softmax function to calculate the final value. Softmax function is showed at bellowed. The range of softmax function is from zero to one. The function (10) is softmax function.

$$F_6 = \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{Z_k}}, j = 1 \sim K. \quad (10)$$

Because our image is an array now, we show the all data as Z. K means how many pixels in an image. We are able to get the weight for every pixel by the function. The weight means how similar between the pixel and face. So the final layer is trying to detect any face by the result the previous six layers calculated.

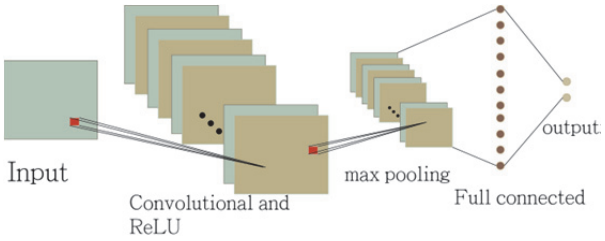


Fig. 5. Convolutional neural network structure.

3.2.3 Sliding windows

Because the input image's size is different from the first layer, we use sliding window to choose region of interested (ROI). Sliding window is a common concept for image processing and signal processing. It sets a 130*150 rectangle at the upper left corner of images. It is the first ROI for CNN input. Our system will detect human being by the selected ROI with the convolutional neural network we trained. Different from the traditional sliding windows, the rectangle doesn't move a column by a column because neural network is a time-cost technology. We hope our system can make a good balance between time-cost and accuracy. We choose move 13 columns (10% of rectangle's length). If the ROI is at the rightest part, the ROI would move 15 rows downward and start from the leftest side. Just like the Figure 6 showed.

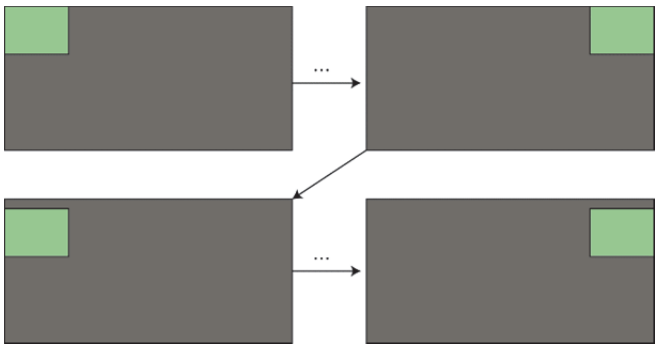


Fig. 6. Sliding windows filter moving steps.

4 Results

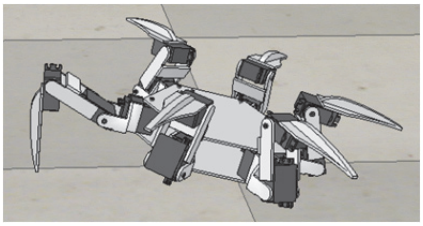
In order to prove our system is work, some experiments are designed. Experiment results about locomotion part will be showed as followed. After that, face detection function's experiment result will be displayed.

4.1 Locomotion

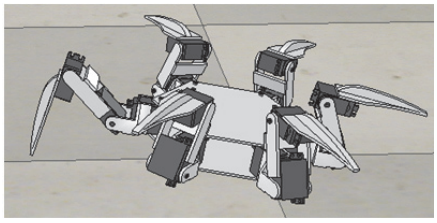
For the value form the table below, column represent the state (θ) and row is the action (next θ), set column=7 when rotation=0, column=6 when rotation=-20 and so on. For the result, When θ=-20 and column=6, row 5 and row 7 has two different value, each represent the Q-value of action decrease to rotation=-40° and increase to rotation=0°, row 5 is 20 that is much higher than row 7, so the action will be row 5, decrease to rotation=-40°, and θ change to column 5, and compare the value between the two action, makes the θ back to rotation=-20°. So the gait of the rotation will alternate between -20° and -40°.

Table. 3 Q-table for the result.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	-150	0	0	0	0	0	0	0	0	0	0	0
2	-100	0	-100	0	0	0	0	0	0	0	0	0	0
3	0	-100	0	-100	0	0	0	0	0	0	0	0	0
4	0	0	-100.0000	0	-100.0000	0	0	0	0	0	0	0	0
5	0	0	0	-99.9878	0	30	0	0	0	0	0	0	0
6	0	0	0	0	20	0	-99.3750	0	0	0	0	0	0
7	0	0	0	0	0	-37.8125	-3	-98.5156	0	0	0	0	0
8	0	0	0	0	0	0	-49.5313	0	-95	0	0	0	0
9	0	0	0	0	0	0	0	-76.1500	0	-76.0938	0	0	0
10	0	0	0	0	0	0	0	0	-76.2375	0	-62.5000	0	0
11	0	0	0	0	0	0	0	0	0	-57.1375	0	-75	0
12	0	0	0	0	0	0	0	0	0	0	-88.0250	0	-87.5000
13	0	0	0	0	0	0	0	0	0	0	0	-119.5375	0



θ=-20°



θ=-40°

Fig. 7. Rotation for the one joint hexapod gait.

The discount factor γ is very important at this learning, when θ larger than 0, joint rotate counterclockwise, hexapod will go forward so reward will be positive, and rotate clockwise the hexapod will get negative reward, on the contrary counterclockwise makes backward movement, clockwise makes forward movement. It makes the action stock at θ=100° and 120° or -100° and -120°, because the action for -120° or 120° only have one option is increase or decrease to θ=100° and -100°, and θ=100° and -100° will select the highest Q-value that will makes them go back to θ=-120° and 120°. So the discount factor need to set very high to makes the algorithm know there is a higher reward.

Table. 4 Wrong result from wrong define γ.

Theta=-120 and -100

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	-0.6000	0	0	0	0	0	0	0	0	0	0	0
2	52.4692	0	-0.6000	0	0	0	0	0	0	0	0	0	0
3	0	46.8109	0	-0.6000	0	0	0	0	0	0	0	0	0
4	0	0	35.0227	0	-0.6000	0	0	0	0	0	0	0	0
5	0	0	0	10.4640	0	16.8109	0	0	0	0	0	0	0
6	0	0	0	0	16.8000	0	16.8000	0	0	0	0	0	0
7	0	0	0	0	0	1.1520	-0.6000	14.7658	0	0	0	0	0
8	0	0	0	0	0	0	-0.8400	0	18.4416	0	0	0	0
9	0	0	0	0	0	0	0	-0.8400	0	12.3034	0	0	0
10	0	0	0	0	0	0	0	0	-0.8400	0	9.4330	0	0
11	0	0	0	0	0	0	0	0	0	-0.8400	0	9.7776	0
12	0	0	0	0	0	0	0	0	0	0	-0.8400	0	5.9705e-...
13	0	0	0	0	0	0	0	0	0	0	0	-1	0

Theta=100 and 120

When I first started the simulation I expect the gait will be the joint rotate 120° to 120° regularly, but no matter how I rewrite the program the result can't be what I expect, than I discovered the reason is the velocity for the backward and forward, for example, joint regularly rotate between θ=0° and -120°, when joint rotate from -120° to 0°, the distance will increase the most, but when rotate 0 to -120, the location will also decrease the most. Compare the movement than the gait rotate between -40° and -20°, the velocity for the gait between -40° and -20° is higher than gait between 0° to -120°.

4.2 Face detection

To determine our kalman filter is work or not, we use thermal images and RGB images which are provided by IIIT- Disguise Face Database. There are some images showed bellowed.

Fig. 8. Infrared and RGB Image from IIIT-Disguise Face Database.

We use kalman filter which mentioned at chapter three to combine two kinds of image to get the result.

Fig. 9. Composite Image by kalman filter.

According to the image, we can keep the important information, like nose and contour. The most important thing is that the image includes the temperature information.

We detect human not only with shape but temperature. The image also helps us to understand our method is work.

Though image fusion is work, we still need to make sure the CNN is useful. It is very important that CNN has a good performance with high noise. For this reason, we do the experiment with different salt and pepper noise. We use matlab to make some noise.

Fig. 10. Face image with 30% salt and pepper noise.

We use some images which are like the above image and some noise images to check the accuracy of convolutional neural network. Because the difference between non-face images with face images is too huge, our convolutional neural network's performance is very well. Actually, it reaches 100%. However, it is not for real situation. We will continue to change and correct our experiment [7].

5 Conclusion

We use Q-learning to learn the gait for robot locomotion. By the simulation we, it works at one-leg situation. On the other hand, we use kalman filter and CNN to finish face detection as well. It provides a trusted result for the system. In other words, we made robot move efficiently and detect human accurately with infrared and RGB sensor. Which is necessary for fire rescue robot to work at fire ground in short time. Although we only get one-leg situation and face detection with normal temperature environment, we will continue to improve it. Our final system would be tested by hexapod situation and real fire ground situation. Even we can't fulfil the dream now, we still provide a new rescue concept that hexapod can go into fire ground and search people. We also finish the first step experiment to prove it is useful.

The authors gratefully acknowledge the financial support of the Ministry of Science and Technology of Taiwan through grant MOST 106-2221-E-305 -013.

References

1. T.I. Dhamecha, R. Singh, M. Vatsa, A. Kumar, PLoS ONE, **9** (7), e99212 (2014)
2. T.I. Dhamecha, A. Nigam, R. Singh, M. Proceedings of International Conference on Biometrics, 1–8 (2013)
3. P.H. Chang et al, SICE-ICASE International Joint Conference, Bexco, Busan, Korea 3126–3131 (2006)
4. A. Marjovi, J.G. Nunes, L. Marques, A. de Almeida, IEEE/RSJ International Conference on Intelligent Robots and Systems, 1929–1934 (2009)
5. A. Imdoukh, A. Shaker, A. Al-Toukhy, D. Kablaoui, M. El-Abd, 2017 18th International Conference on Advanced Robotics (ICAR), 310–315, (2017)
6. G. Welch, G. Bishop. *An introduction to the kalman filter* (Department of Computer Science, University of North Carolina //ed: Chapel Hill, NC, unpublished manuscript, 2006)
7. Samani H., SPIIRAS Proceedings. **56**, 56-75. (2018)