Autonomous Learning of Motion Parallax for Active Depth Perception

Tanapol Prucksakorn

School of Information Science Science
Japan Advanced Institute of Science and Technology
September, 2015

Master's Thesis

Autonomous Learning of Motion Parallax for Active Depth Perception

1310204 Tanapol Prucksakorn

Supervisor :	Professor Nak Young Chong
Main Examiner :	Professor Nak Young Chong
Examiners :	Associate Professor Fumihiko Asano
	Associate Professor Kazunori Kotani

School of Information Science
Japan Advanced Institute of Science and Technology

August, 2015 (submitted)

Abstract

Depth perception is a visual ability that allows humans and animals to be able to perceive the world and environments in three dimensions and distance of objects. It is one of the most fundamental task that artificial vision systems must solve. There are three different kind of depth perception which are stereo vision, motion parallax, and optic flow.

Estimating depth of an object by using vision system has been continuously researched for a long time. There are a lot of works that can estimate depth of objects by using binocular cue. However, there is little work on depth estimation by using monocular depth cue. Most of the researches focus on estimating depth from a single monocular image. Some of them require specific condition such as environment, some requires calibration. So, if there are some changes or interferences in environment or configuration of vision system, the solution seems to fail later. So, to contribute in monocular depth estimation, our goal is to propose an autonomous learning and self-calibrating motion parallax depth estimation system which is robust and able to adapt to environments.

In this thesis, we propose a model that combines knowledges of joint development of visual encoding and reinforcement learning together to find a depth of a single object with motion parallax. We use reinforcement learning to learn how to use encoded sensory data to explicitly control the movement of the eye. Perception and behavior will be developed simultaneously by minimizing the same cost function. We consider a real world experiment and a simulation. Experiment demonstrates that the framework can find estimate depths while it is an autonomous system and self-calibrating. Also, the experiment shows that an extension of active depth perception with another form of depth cue is possible.

Acknowledgements

I would like to take this opportunity to express my gratitude to those who made a difference in my research life for two years in JAIST. First and foremost, I am deeply indebted to my supervisor, Professor Nak Young Chong, for giving me an opportunity to engage researching at JAIST. He gave a very helpful supports and wonderful advices. His guidance helped me in all the time of research.

Besides my advisor, I would like to thank Assistance Professor Sungmoon Jeong for his insightful comments and advices which incented me to widen my research from various perspectives. My sincere thanks also goes to Lee Hosun, and Kshitij Tiwari for their helps, suggestions and encouragements during my research. I would also like to show my gratitude to Professor Jochen Triesch and Vikram Narayan for support form Germany.

Last but no the least, I would like to thank my parents for supporting me to study, research, and writing this thesis here in Japan.

Contents

List of Figures

List of Tables

Chapter 1

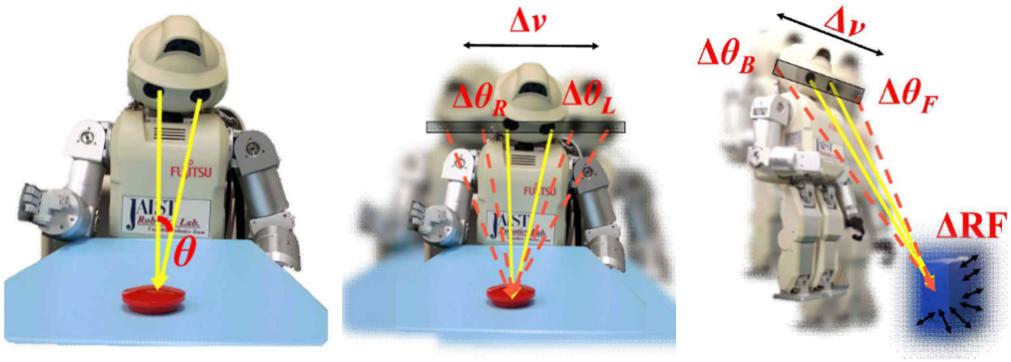
Introduction

In daily life, we use our eyes to perceive environments around us. We can grab a cup of coffee with our hands which are cooperating with our eyes. However, to robots, grabbing a cup of coffee is a very difficult task. It requires not only vision, but it also needs depth perception.

Depth perception is a visual ability to perceive the world in three dimensions and the distance of an object. Depth perception is the most fundamental artificial vision problem that must be solved. It is an active process that can involve different kinds of movement such as eyes movement, head movement, and body movement. Although currently, there are many solutions that have been developed, they seem to be unstable. In contrast, biological vision systems are robust. Because, they have abilities to learn, adapt, and self-calibrate.

The data that are collected by human or animals organs are very noisy messy data. It is not self-explanatory meaningful information [1, 2]. So, how can us make sense of these non-obvious data? In [3], they discussed that our brain did not programed to know how to use those data, but instead the brain is trained autonomously to learn how to translate those noisy unordered information into depth perception. In the same way, the robots which are not programmed with depth perception ability faced the same problem that they do not know how to utilize the data.

By adapting the biological vision systems with the current artificial vision systems, we can get rid of the dearth of robustness. In neural science, to use information from sensory system, the system should efficiently encode the sensory information by taking advantages of redundancies [4, 5, 6] So, we may use the nature of the sensory systems in humans to adapt with our artificial vision system. Neurons are the cells that are in our body. They have an ability to propagate signals rapidly over large distances. Sensory neurons fire sequences of action potentials in various temporal patterns to change their activities. To resemble the neurons in our body, sparse coding is used to represent sensory inputs [7].



(a) Vergence (b) Motion parallax (c) Optic flow

Figure 1.1: Three different depth perception

Recently, [8] proposed to apply the efficient coding principle to active vision. The proposed framework is the promising one. They use the concept of sparse coding and reinforcement learning together to create such a system that can learn and optimize the efficiency of coding and vergence eye movements [9, 10]. They proved that the concept of efficient coding in active perception has the potential to explain the simultaneous development of disparity representations and vergence behavior.

However, there is an interesting question left. The question is whether the framework could be extended to other form of active depth perception. Active depth perception is depth perception with action of agent, such as movement of the eyes, body, or manipulating the object. There are three approaches for active depth perception (Figure 1.1). The first one is depth estimation based on the vergence angle between two eyes (Stereo vision) which is used in [8] (Figure 1.1a). The following one is estimation based on motion parallax. A controlled lateral movement produces a change of the angle under which the object is perceived (Figure 1.1b). The depth can be estimated by this change. The final one is estimation based on optic flow. The pattern of optic flow or the visual size of the targeted object could be used to estimate the depth (Figure 1.1c).

Developed organisms have several method for detecting depth and integrating sensory information together. However, depth estimation strategies are not all available from birth. There is an evidence that human infants develop motion-based depth perception, motion parallax, before developing the binocular depth cue [11]. Motion parallax plays important role in development of depth perception system in our brain. So it is interesting to use motion parallax for active depth perception. This research will adapt motion parallax for the extension of the active depth perception model.

1.1 Problem Statement

To estimate the distance between a robot and an object, the robot must have depth perception mechanism in order to perceive the depth. There are many ways to estimate the depth such as, stereo vision which is widely used, and there are a lot of researches about stereo vision. It gives an depth perception ability to a robot.

However, most of monocular and binocular depth estimation researches does not only require calibrations before operating, it also requires that the configurations of the system must not be altered. So, if there is any situation or accident that interfere the configurations of the vision system a little bit, the system would begin to fail. Thus, some kind of autonomous and self calibrating mechanism would be needed in those kinds of situation.

In order to make a robot or a vision system that suitable for all environment and robust to interferences, the problems are very crucial and must be solved. A representation of vision system in developed organism, such as human, could be useful to overcome the problems, because humans vision system can adapt to many environment and can recover from interferences. A simple concept of perceiving vision or depth in our human brain is described in Figure 1.2, the action cycle. The eyes send sensory information to the brain to create vision and depth perception, while the brain learn to control the eyes movement in order to make eye perceive the environment effectively.

To create such a system that is robust and fit to all environment, a framework that is autonomous learning and self-calibrating is required. Autonomous learning will let the system learns how to improve and adapt itself to environments. Self-calibrating would make the system more resistive to changes of configuration and interference. There is one remarkable work that could create such a model that we described for binocular vision system [8]. In this thesis, we will introduce an extension of active depth perception with motion parallax of the system.

The proposed model will have two important abilities, autonomous learning and self-calibrating. The system will be able to learn how to generate an appropriate eye movement during lateral movement for estimating depth of an object. Finally, this thesis will show that extending the stereo active depth perception to another kind of active depth perception is possible. This work will be an another step to create a full representation of biological vision system for artificial vision system.

1.2 Related Work

In [12], they investigated how a humanoid robot could learn to improve binocular vision system to judge distances by using some movement. They proved that

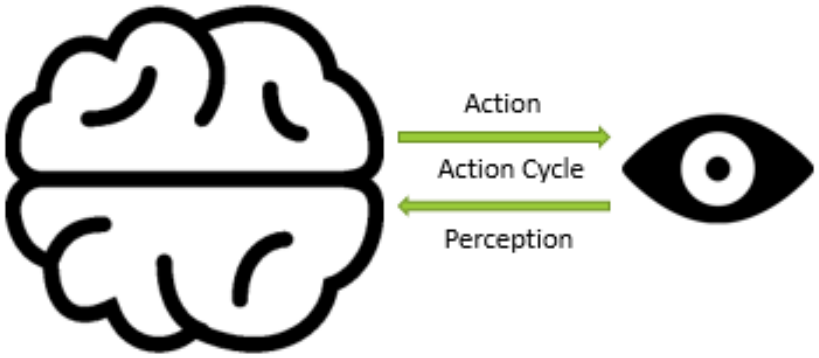


Figure 1.2: Action cycle in most of developed organism

action that does not alter the depth to the target can be a powerful method to improve depth perception ability. It is very interesting idea that some action or manipulation of an object, i.e. active depth perception, could improve depth estimation.

As discussed in [8], there is a lot of related work in artificial vision system field. Most of the works only concerns the two problem of learning perception learning behavior in isolation which is opposite to vision system in humans and animals which has the processes coupled. Some work combines the perception and behavior together, but still treated the two processes independently. Y Zhao et al. [8] has proposed a way to use reinforcement learning couple with efficient sensory coding to create a vergence eye movement for artificial vision system. Their framework integrates the joint development of perception and behavior in the context of eye movements. So, the two processes of learning and behavior are unified in this framework which creates an action cycle (Figure 1.2) like in developed organism. The framework utilized manipulation of target object to train the system to autonomously improve vergence eye movement.

However, framework in [8] has one limitation that input disparities are encoded with in a small range. In [9], they proposed a way to improve the range of input disparities. They use multi-scale images to overcome the limitation. In [13], they adapt the framework with the multi-scale improvement to create a new model that can generate smooth pursuit eye movements, i.e. tracking a moving object.

To our knowledge, there is little work on depth estimation by using monocular depth cue. Most of the researches focus on estimating depth from a single monocular image [14, 15, 16]. Nevertheless, there is some intriguing works. In [17], their approach is very interesting. They use monocular depth cue to estimate relative depth. Linear regression model is used as perception learning part. Reinforcement

learning is also used in their model as behavior learning. However, the learning and behavior learning are processed in isolation. So, behavior and perception would not be able to jointly learn together. In [18], they created a model that uses sequence of images and camera motion to estimate depth. However, it is not autonomous learning and self-calibrating system.

1.3 Research Objective

To mimic the biological vision of human, this research focuses on implementing and developing active depth perception system by using motion parallax. In [8], they prove the concept that their approach for efficient coding in active perception has potential to explain the simultaneous development of disparity representations and vergence behavior. However, they only use vergence depth perception. In this research, we will try to extend the depth perception with the other approaches which is motion parallax. For the motion parallax, the system will discover and autonomously learn to represent the latent depth information by using the lateral movements of head or upper body and the resulting retinal movement of the target object. The system will be able to obtain and learn to represent the latent depth information. The ultimate goal of this research is to create a system that is autonomous learning and self calibrating motion parallax based active depth perception system.

1.4 Structure of this Thesis

- Chapter 1 is about the research background, problem statement, research objective, and some related works.
- Chapter 2 is a study of autonomous learning of active binocular vision framework.
- Chapter 3 explains the method of extending the framework's active depth perception with motion parallax with experimental and simulation results.
- Chapter 4 contains concluding remarks, contributions and future direction.

Chapter 2

Autonomous Learning of Active Binocular Vision Framework

In this chapter, we will discuss about the research, and the fundamental of each module in the framework proposed in [8] and developed of Zhao et al.'s model proposed in [9]. First, to understand and extend the framework, we have studied the framework and created the framework simulation. The setup is that there is one object that we want to find the distance between the camera and the object placed in an environment. Then as shown in Figure 2.1, two images are input to the sensory coding model which is the perception learning part from camera A and camera B. After that, the sensory coding model sends out sparse coefficient to the natural actor critic reinforcement learning algorithm which is the behavior learning part. Finally, vergence command (command that tells how much camera need to adjust) is sent to the cameras. The framework will repeat this procedure until it can achieve the vergence angle that gives zero disparity of the object. In the next section, we will discuss about the fundamental of each step in this framework.

2.1 Vergence Eye Movement

Vergence eye movement is simply the function that control both eyes to point their fovea on an interested object. Left and right eye rotates in opposite direction to obtain or maintain single binocular vision (Figure 2.2). When we looks at an object, the eyes must rotate around the a vertical axis so that the projection of the image is in the center of the retina of both eyes. In this framework, we try to make both of the cameras point to the same object.

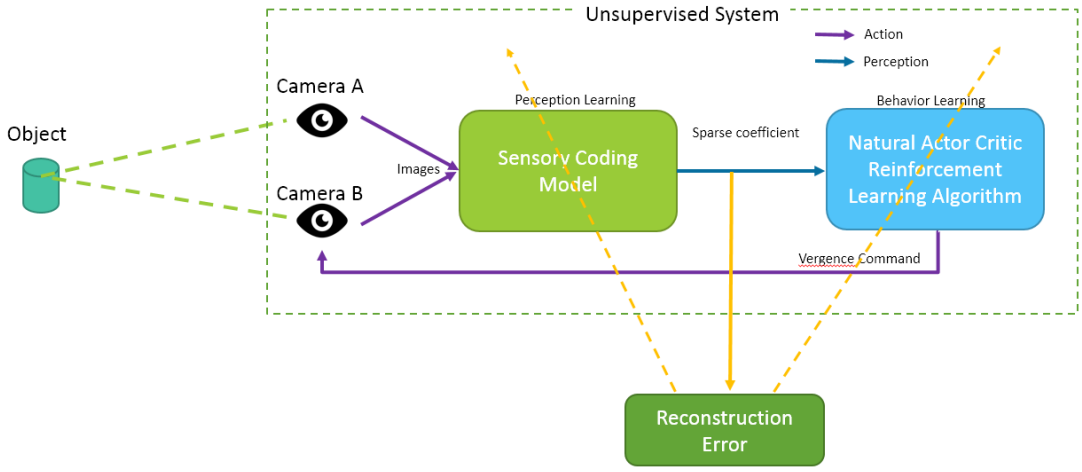


Figure 2.1: Zhao et al.'s framework

2.2 Sensory Coding Model

As we have discussed in Chapter 1, sensory systems should encode sensory information in an efficient manner by exploiting redundancies in their inputs. In this framework, to encode the sensory information, we use sparse coding to learn an efficient representation of the sensory input (or the images from camera A and B). The key idea of this efficient encoding is that the reinforcement learner receives the reward signal on how well the sensory model can represent the input. The flow of the sensory coding model is shown in Figure 2.3 The inputs to the model are stereo images from the cameras. The horizontal shift of image of the object that is measured from left to right is referred as input disparity. At different depth, the binocular images provide different input disparity. For the experiment, we artificially generated stereo image pairs from 6 images (Figure 2.4). The left camera is represented by the original image, while the right camera sees the horizontal shifted version of the original image.

Stereo image sequences are created by randomly choosing the an image from the database. For each 10 iterations, one image will be randomly selected again. To virtually represent images taken from cameras, for each pair of stereo images, the images are cropped with windows size of 128 by 128 pixel in the center of the images. The window location of left eye is fixed, while the window location of right eye can be shifted horizontally to virtually represent the vergence of the eyes. The actual retinal disparity is the difference between input disparity and the vergence.

So, the goal of this framework is to generate vergence equal to the input disparity which will create zero retinal disparity. After we cropped the images, we converted

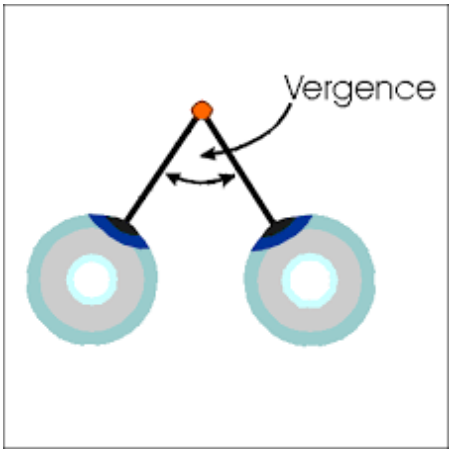


Figure 2.2: Vergence eye movement

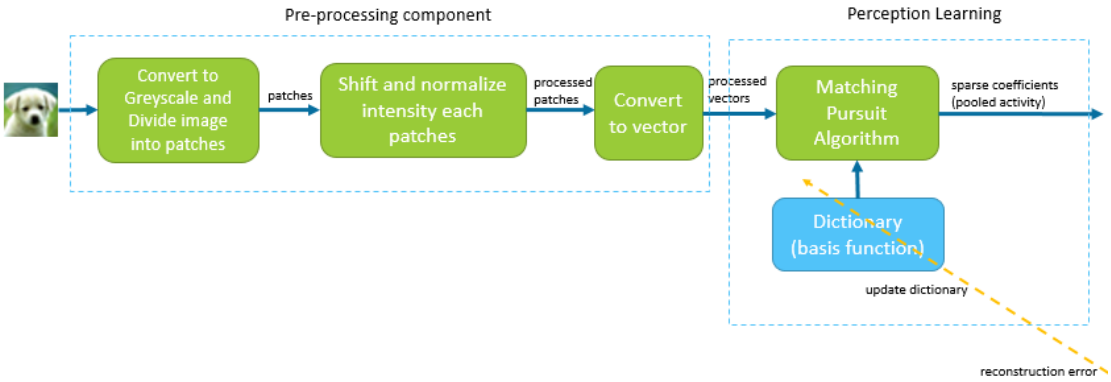


Figure 2.3: Inside of sensory coding model

the images to gray scale first. Then, we extract the gray scales images into 8 by 8 pixel patches whose locations are generated by 1 pixel shifts horizontally and vertically. The patches is sub-sampled by using Gaussian pyramid algorithm by a factor of 8. Then the patches are shifted and normalized to have zero mean and unit norm. The processed patch is then converted to a vector. Corresponding vector $x_i(t)$, where i is indexes of the patches, from the left and right images are then combined into a single vector, $x(t)$. The first 64 elements of the vector are from left eye and the remaining are from the right eye. This results in vector $x(t)$ has $P = 128$ elements

To represent the neural vision system in developed organism, the idea of sparse coding is used. The concept of sparse coding is that the neuron encode the images in linear fashion. We use combination of basis function drawn from an over-complete dictionary $\phi(t) = \{\phi_n(t)\}_{n=1}^{N}$[7]. The number of basis function used in

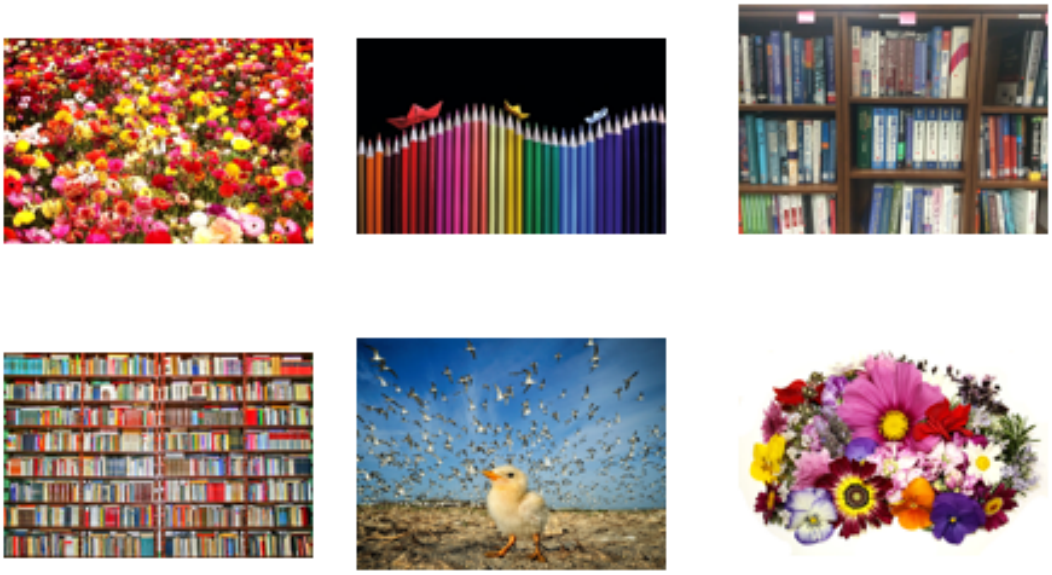


Figure 2.4: Images that are used in binocular vision framework simulation

this experiment is $N = 288$. The dictionary is initially randomly generated and normalized. We use matching pursuit algorithm provided in MATLAB tools to estimate and find the sparse representation of the input vector by the weighted sum

$$x_i(t) \approx \sum_{n=1}^{N} a_{i,n}(t)\phi_n(t) \tag{2.1}$$

We use the matching pursuit algorithm to estimate $x_i(t)$ because it can limit the number of coefficients used which reduce complexity. We set the maximum number of non-zero scalar coefficients $a_{i,n}(t)$ to be 10 elements. This is to create sparseness for efficient encoding. The coefficients generated by the matching pursuit algorithm will then be used in reinforcement learning, behavior learning part, later as pooled activity. Pooled activity represent the activity of each neuron cell, the coefficients $a_{i,n}(t)$, from every patches. The model that we use for pooled activity is

$$f_n(t) = \sum_{i=1}^{P} a_{i,n}(t)^2 \tag{2.2}$$

By estimating vector $x(t)$, there will be some errors. To reduce the errors that will generate by matching pursuit algorithm in the next iteration, we use gradient

descent method to update the dictionary by using reconstruction error as a cost function. The reconstruction error is defined as below

$$e(t) = \frac{1}{P} \sum_{i=1}^{P} \frac{\|x_i(t) - \sum_{n=1}^{N} a_{i,n}(t)\phi_n(t)\|^2}{\|x_i(t)^2\|} \tag{2.3}$$

After each update, the dictionary is normalized again. The update of dictionary part is the perception learning part which improves cognitive of the system over time.

2.3 Reinforcement Learning

In machine learning, the environment is often treated as Markov decision process (MDP) [1] which can get very complicate for an environment that has many variables and dimensions. In some cases, MDP can be solved analytically, and in many cases can be solved by dynamic programming. However, reinforcement learning does not require any prior knowledge about the Markov decision process model. Also, it is not supervised system.

We use reinforcement learning algorithms to train the behavior of the system which is defined by policy. Policy maps the state of the actor in its environment to a specific action. The agent, or the system, is trained by giving a reward with respect to its action as described in Figure 2.5. There are many reinforcement learning algorithms. But the basic of reinforcement learning model is very simple as described in 2.5. It has a set of environment states S, a set of actions A, and policy. The basic flow of all reinforcement learning algorithm would be

1. Observe state, s_t
2. Decide on an action, a_t
3. Perform action
4. Observe new state, s_{t+1}
5. Observe reward, r_{t+1}
6. Learn from experience
7. Repeat step 1

[1] Markov Decision Process is a discrete time stochastic control process. At each time step, the process is in some state s, and the decision maker may choose any action a that is available in state s. The process responds at the next time step by randomly moving into a new state s', and giving the decision maker a corresponding reward. - wikipedia

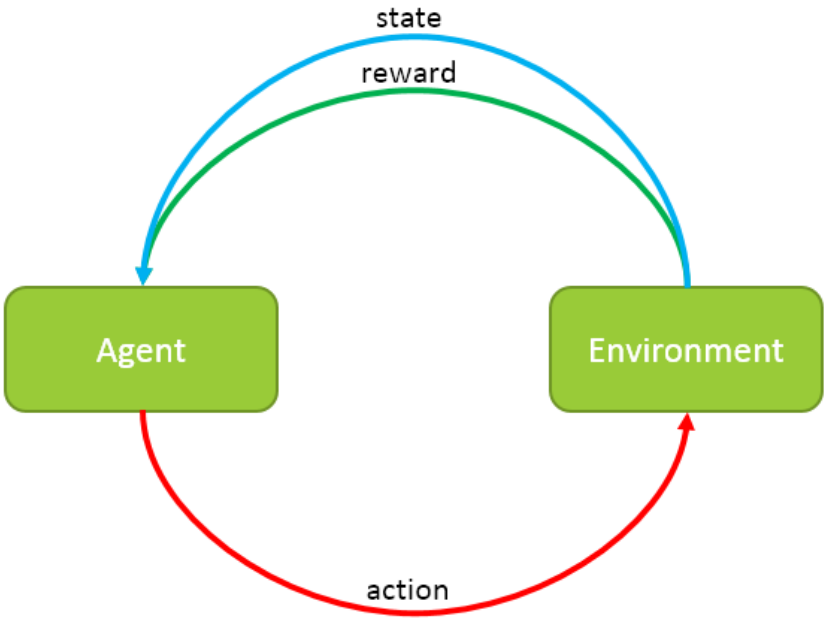


Figure 2.5: Basic diagram of reinforcement learning

The goal of the agent is to find a policy that would maximize the observed rewards over the lifetime of the agent. There are two important value function in reinforcement learning. First one is value function. It tells how much is the best reward the agent could get from that state.

$$V^{\pi}(s_t) = R(s_t, \pi(s), s_{t+1}) + V^{\pi}(s_{t+1}) \tag{2.4}$$

Where, $R(s_t, \pi(s_t), s_{t+1})$ is a reward that the agent would get, if the agent perform action at state s_t with respect to the policy π to the state s_{t+1}. The another value function is state-action value function $Q(s_t, a_t)$. It is a little bit different from value function $V^{\pi}(s_t)$. It shows the best reward that the agent could get if take the action a_t from state s_t.

$$Q(s_t, a_t) = R(s_t, a_t, s_{t+1}) + \max_{a'} Q(s_{t+1}, a') \tag{2.5}$$

Before explaining the Natural Actor-Critic Reinforcement Learning, a simple reinforcement learning algorithm will be explained first in the following subsection.

2.3.1 Q-Learning

Q-learning is one of reinforcement learning algorithm. It keeps track the state, action, and state-action value $Q(s, a)$ in a Q-table (Figure 2.6). The flow of the

Q-learning can be described in Figure 2.7. An action that will be performed is selected form the actions list. The selection is based on the policy that is used. The main idea is that it will most likely choose an action that can get good reward. For updating Q-Value, we can use

$$Q(s_t, a_t) = (1 - l)Q(s_t, a_t) + l(R(s_t, a_t, s_{t+1}) + \gamma \max_{a'} Q(s_{t+1}, a')) \tag{2.6}$$

Where, l is learning rate, which tells how much information the agent would use from the new experiences. γ is discounted factor. It determines the importance of the future rewards for the agent. If γ approaches to 1, the agent will make agent strive for long term high reward. But if γ is close to 0, the agent will consider only most of current reward and will decide to quit quickly. Generally, we use $0 < l < 1, 0 < \gamma < 1$. To select an action, there are many method to select. In

State (s)	Action (a)	Q(s,a)
s_1	a_1	Q(1,1)
s_1	a_2	Q(1,2)
s_2	a_1	Q(2,1)
	. . .	
s_n	a_k	Q(n,k)

Figure 2.6: Q-Table

general, we use Gibbs distribution for selecting the action.

$$\pi(s_t, a_t) = \frac{e^{Q(s_t, a_t)}}{\sum_{a' \in A} e^{Q(s_t, a')}} \tag{2.7}$$

It gives probability of an action being selected for each action based on the value in Q-table, $Q(s_t, a_t)$.

2.3.2 Actor-Critic

Actor-Critic is a reinforcement learning algorithm that is similar to Q-learning. But the difference is there is separate actor and critic. The actor corresponds

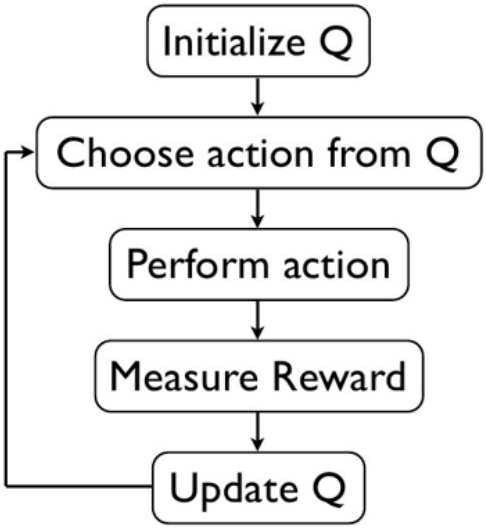


Figure 2.7: Q-Learning flow chart

to an action-selection policy, mapping states to actions based on probabilistic. The critic corresponds to a conventional state-value function, mapping states to expected cumulative future reward. So, the critic handle a problem of prediction, while the actor is concerned with control of the action. The information that is shared by actor and critic is TD (Temporal Difference) error as shown in Figure 2.8. TD error is used to estimate the average reward for a state-action pair. TD error, δ_t, is defined by

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.8}$$

The TD error can be used to evaluate the selected action. If the TD error is positive, it means that the selected action, a_t, generate better reward. So, the action a_t should be strengthened in the future. While, for the negative TD error, it suggests the action a_t should be weakened. This part is considered to be critic part. For the actor part, we used the information from critic to update the policy parameter of the actor, $\theta(s_t, a_t)$. The update is performed by

$$\theta(s_t, a_t) = \theta(s_t, a_t) + \beta\delta_t \tag{2.9}$$

To select an action, we may use Gibbs distribution, equation 2.7. The only difference is changing $Q(s_t, a_t)$ to $\theta(s_t, a_t)$

2.3.3 Natural Actor Critic

Unfortunately, both Q-learning and Actor-critic algorithm is only for discrete state space system. So, this algorithm is used for the reinforcement learning part. The

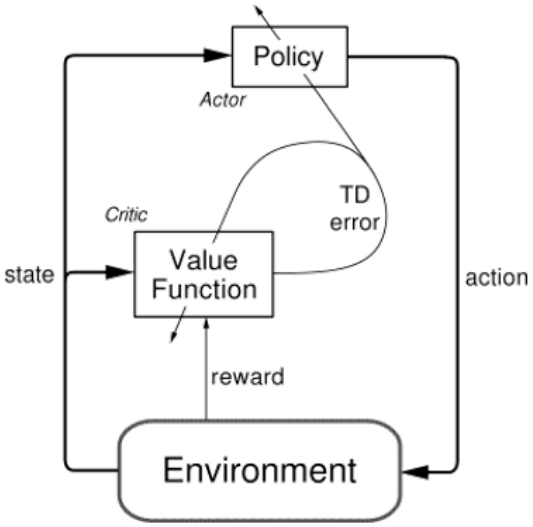


Figure 2.8: Actor-Critic model

algorithm has been proposed in [19]. They presented four reinforcement learning algorithm based on actor-critic, and natural gradient ideas. Two linear neural networks are used to implement actor and critic (Figure 2.9).

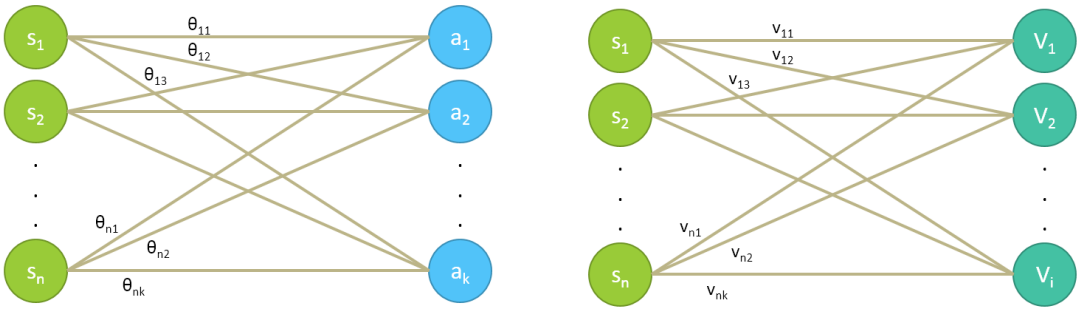


Figure 2.9: Two neural network implementing actor and critic

We use the actor-critic algorithm number 3 proposed in [19]. The algorithm is explained in Table 2.1 below.

- t is the iteration number.
- $\hat{J}$ is average reward.
- f_{s_t} is a feature vector for state s_t.
- v is neural network weights for feature vector f_{s_t}.

Table 2.1: Natural Actor Critic Algorithm 3 in [19]

1:	**Input**:	
	• Randomized parameterized policy π	
	• Value function feature vector f_s	
2:	**Initialization**:	
	• Policy parameters $\theta = \theta_0$	
	• Value function weight vector $v = v_0$	
	• Step sizes $\alpha = \alpha_0, \beta = \beta_0, \xi = c\alpha_0$	
	• Initial state s_0	
3:	**for** $t = 0, 1, 2, ...$ **do**	
4:	**Execution:**	
	• Draw action $a_t \sim \pi(s_t, a_t)$	
5:	**Average Reward Update:**	$\hat{J}_{t+1} = (1 - \xi_t)\hat{J}_t + \xi_t r_{t+1}$
6:	**TD Error:**	$\delta_t = r_{t+1} - \hat{J}_{t+1} + v^\intercal f_{s_{t+1}} - v_t^\intercal f_{s_t}$
7:	**Critic Update:**	$v_{t+1} = v_t + \alpha_t \delta_t f_{s_t}$
		$w_{t+1} = [I - \alpha_t \psi_{s_t a_t} \psi_{s_t a_t}^\intercal] w_t + \alpha_t \delta_t \psi_{s_t a_t}$
8:	**Actor Update:**	$\theta_{t+1} = \theta_t + \beta_t w_{t+1}$
9:	**endfor**	
10:	**return** Policy and value function parameters θ, v	

- w is neural network weights for policy parameter vector θ.
- θ is policy parameter vector.
- α, β, ξ are step sizes for updating weight vector w, θ, and average reward $\hat{J}$ respectively.
- $\phi_{s_t a_t}$ is a feature vector for state-action pair.

for softmax activation policy, Gibbs distribution, which we use in this project

$$\pi(s_t, a_t) = \frac{e^{\theta^\intercal \phi_{s_t a_t}}}{\sum_{a' \in A} e^{\theta^\intercal \phi_{s_t a'}}} \tag{2.10}$$

$$\psi_{s_t a_t} = \phi_{s_t a_t} - \sum_{a'_t \in A} \pi(s_t, a'_t) \phi_{s_t a'_t} \tag{2.11}$$

2.4 Multi-Scale Extension

Binocular cells tuned to different disparity ranges in visual cortex areas. These cells adjust and adapt the controlling mechanism to generate fast or slow vergence

response depending to the range of disparity [20]. As discussed in [9], there is one limitation of the framework which is the maximum input disparities is quite low. In order to increase the range of detectable disparities, they has extend the model to multi-scale approach. The developed framework uses two scale of images which are coarse and fine scale, in order to have both characteristic of the two scales of image. The model is very similar to the previous model. The changed is only that the model now consider two scale of images, as shown in Figure 2.10. The only difference is only in the sensory coding model part. For the reinforcement learning part, the only difference is input state and reward. In the following subsection, we will explain the change of the sensory coding model part.

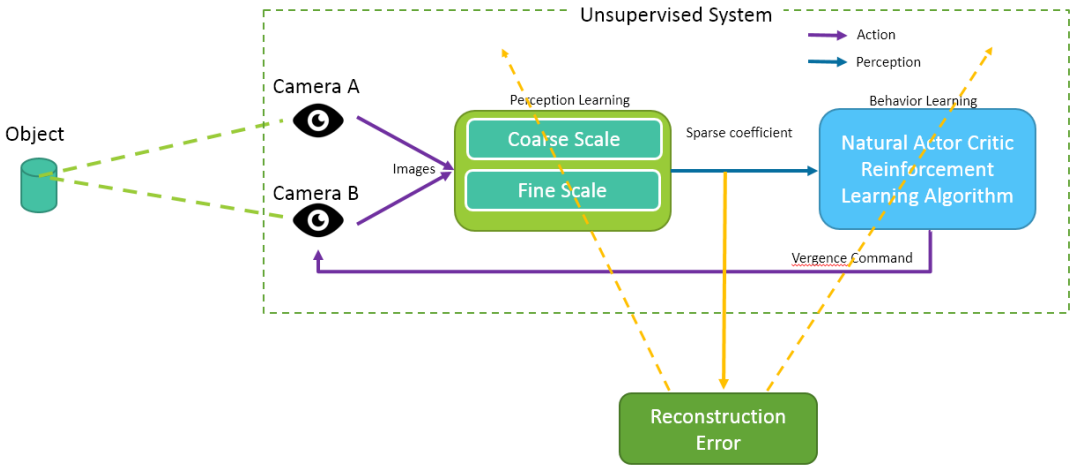


Figure 2.10: Multi-scale binocular vision model

2.4.1 Sensory Coding Model for Multi-Scale Extension

Input images and method for virtual left and right camera are still the same. For this extension, we cut more one 80 by 80 pixels image for left and right camera. The addition cropped image represent a fine scale image, while the original one represent coarse scale image. This is to mimic the human vision system, i.e. we can get more detail in the center of our vision. The images will be processed in the same way as explained in section 2.2. The settings for coarse scale image are the same. But for the fine scale image, the image is sub-sampled by factor of 2. We shift the image by 4 pixels horizontally and vertically to generate image patches.

For the coding part, we use two over-completed dictionary, each for coarse scale and fine scale. The process for encoding is the same, except for that we encode coarse scale and fine scale image separately.

2.5 Simulation

2.5.1 Simulation Setup

Original Framework

Policy which maps the state of the system to an action define behavior. In this framework, the state is represented by feature vector, the pooled activity

$$f_{s_t} = f(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_P(t) \end{bmatrix} \tag{2.12}$$

Actions are vergence commands, which tells the cameras how much they have to rotate. Negative of the reconstruction error from the sensory coding model is used as a reward for the reinforcement learning.

$$r_t = -e(t) \tag{2.13}$$

The goal of the reinforcement learning is to select actions that maximize the discounted cumulative future reward, namely minimizing the total reconstruction error.

We chose a set of 5 actions as follow, $A = \{-2, -1, 0, 1, 2\}$. The actions are the number of pixels that the image for right eye will be shifted. The probability of choosing an action is computed according to a softmax operation (Gibbs distribution). The step sizes for reinforcement learning algorithm is set as below.

- $\alpha = 0.1$
- $\beta = 0.01$
- $\xi = 0.01$

The neural network weights v, w, and policy parameter θ are initially randomized.

Original Framework With Multi-Scale Extension

The input state for reinforcement learning part is combined pooled activity of coarse and fine scale image. For the state and reward given to the reinforcement learning algorithm, we use

$$f_{s_t} = f(t) = \begin{bmatrix} f_1^C(t) \\ f_2^C(t) \\ \vdots \\ f_P^C(t) \end{bmatrix} + \begin{bmatrix} f_1^F(t) \\ f_2^F(t) \\ \vdots \\ f_P^F(t) \end{bmatrix} \tag{2.14}$$

$$r_t = -(e^C(t) + e^F(t)) \tag{2.15}$$

Where, superscript F states for fine scale, while superscript C means coarse scale. We use the set of actions, step sizes, and softmax operation as the same in original framework simulation setup.

2.5.2 Simulation Results

To evaluate the performance of the model, we use absolute mean error (AME) in vergence over 100 trials from [9]. This method will track vergence errors only in the iteration before changing the image, i.e. every 9^{th} iteration. The AME is defined as

$$AME(t) = \frac{1}{100}\sum_{k=0}^{99} |\alpha(t+9+10k) - \alpha^*(t+9+10k)| \tag{2.16}$$

where α^* is the target vergence at the current iteration. Figure 2.11 shows AME of a one run through simulation.

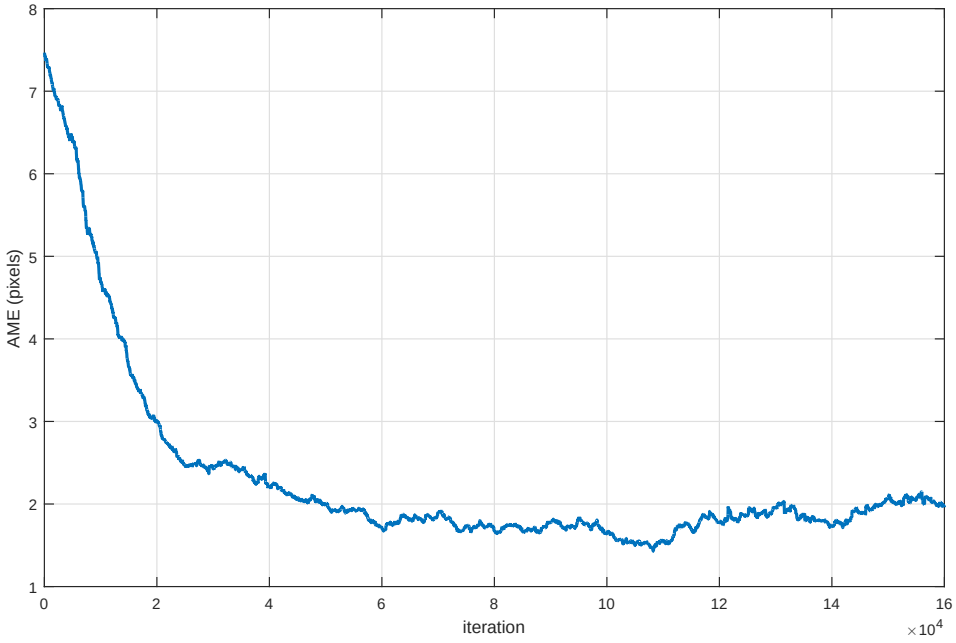


Figure 2.11: AME of the simulation

We can see that the error is reduced over time and stays around 2 pixels. This tells that the framework has nothing to learn anymore from the input images.

After the training is satisfied, i.e. the framework can generate overlapped images from left and right camera (Figure 2.12), we test the framework by changing the input disparity. The result is shown in Figure 2.13. The vergence error is shown in Figure 2.14.

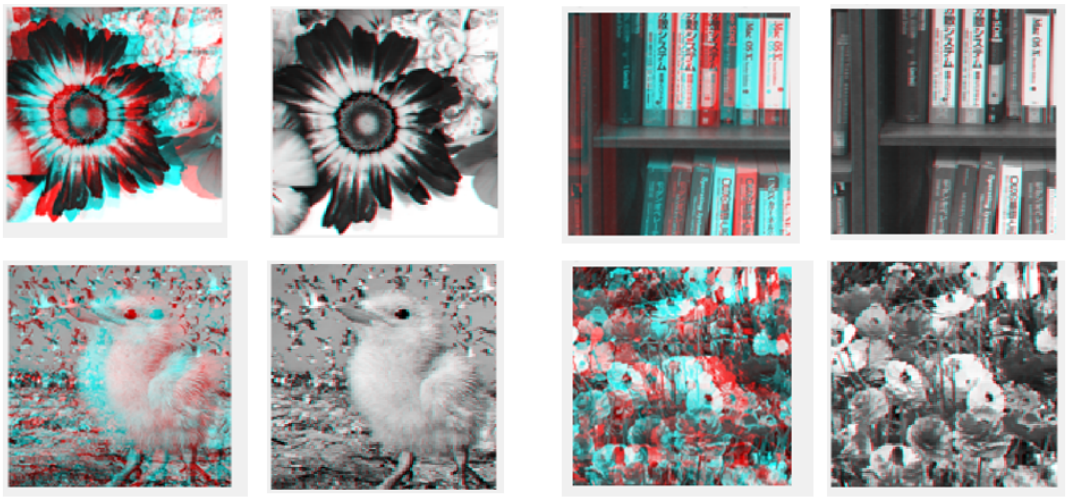


Figure 2.12: Example of some of results of the simulation

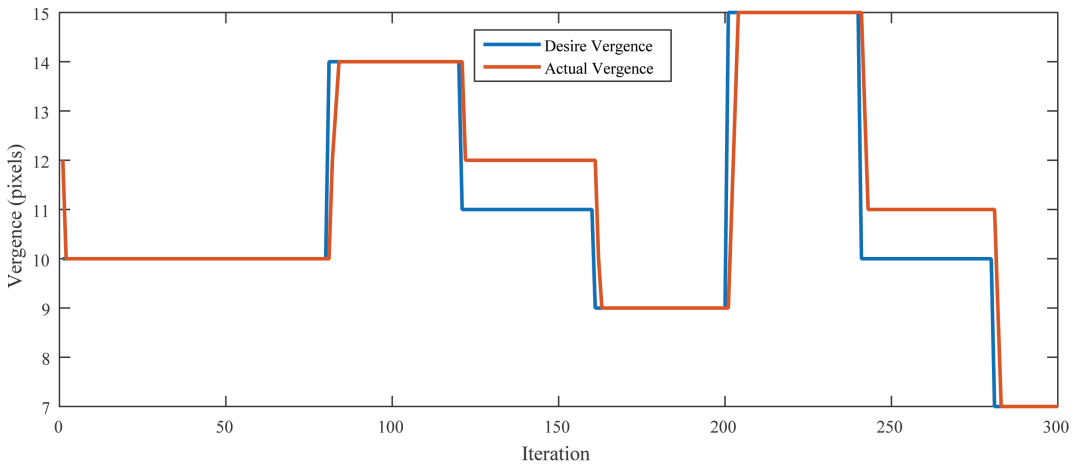


Figure 2.13: Vergence tracking after training is finished

We can see that the system can response to the input disparity changes quickly. Vergence stays stable after reaches zero retinal disparity. By neglecting the error between transition to a new disparity, the maximum error we get is only 1 pixels.

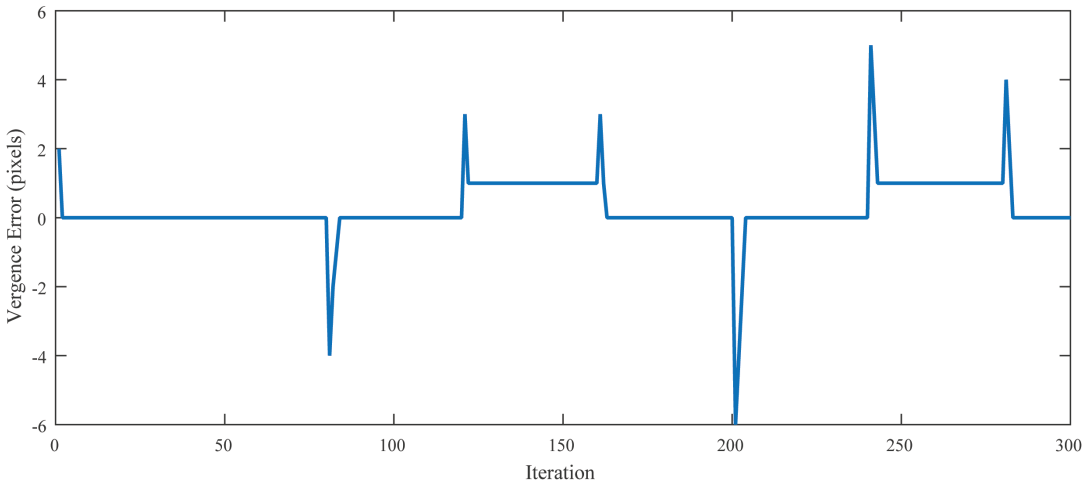


Figure 2.14: Vergence error

From the simulation results, we can say that the framework can generate vergence eye movement.

2.6 Chapter Conclusion

In this chapter, a novel framework proposed by Zhao et al. and multi-scale extension of the framework have been explained and discussed. We showed that the system can autonomously learn how to control left and right camera to generate vergence eye movement.

Chapter 3

The Extended Framework

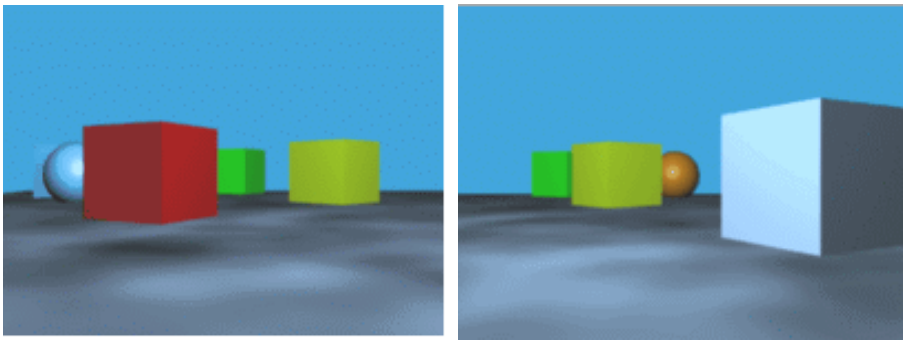
After we have studied and understand the previous framework in chapter 2, in this chapter we will explain the framework that active depth perception is extended with motion parallax.

3.1 Motion Parallax

Parallax is a difference in apparent position of an object viewed along two different viewpoint. The term is derived from Greek word "parallaxis" meaning alteration. Parallax has many application such as for astronomers, they use the concept of parallax to measure distances to the closer stars.

Thus, motion parallax is a method that gives the parallax effect when the subject or object is moved. Motion parallax is one of the depth perception that we use everyday in our daily life. It is a monocular depth cue which means that it uses only one vision organ. The fundamental of the motion parallax is that as we move the object that are closer to us move faster than the object that is farther as shown in Figure 3.1. In Figure 3.1a, as we start moving from the initial position we can see red box and yellow box. But after we moved to the right, as in Figure 3.1b, we still see the yellow box, but now we can not see the red box. So, from these two images we can immediately conclude that the yellow box is farther than the red box.

To see the depth of the object from motion parallax, the subject maintains fixation on the object. In order to fixate the vision of the object, the subject must counter-rotate the eye during translation (Figure 3.2). If the subject move to the left, the eye must rotate to the right, and vice versa.



(a) Start position (b) End position

Figure 3.1: Images created by lateral movement from left to right

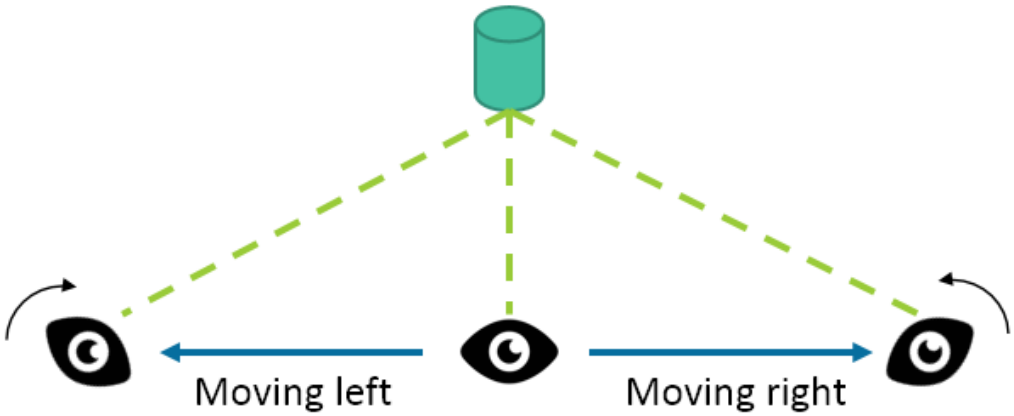


Figure 3.2: Depth perception from motion parallax

3.2 The Extended Framework

To extend the framework with motion parallax, we use different method for image input and camera control as shown in Figure 3.3. In this case, we use only one camera for image input. We generate lateral movement for the camera. The camera then capture images from different viewpoints. Two successive images is input to the framework. The output of the framework is movement of the eye (rotating left or right). The framework will try to fixate the object at the same position for two successive image, i.e. make the object in the images overlapped. A simple two layer neural network is used to estimate the depth by using motion parallax information, the movement of the eye. The neural network will be trained by using the provided ground truth depth information at first. After finished the

training the framework will be able to estimate the depth of an object by moving only once. For sensory coding model and reinforcement learning part, we use the same setting as the previous framework with multi-scale extension.

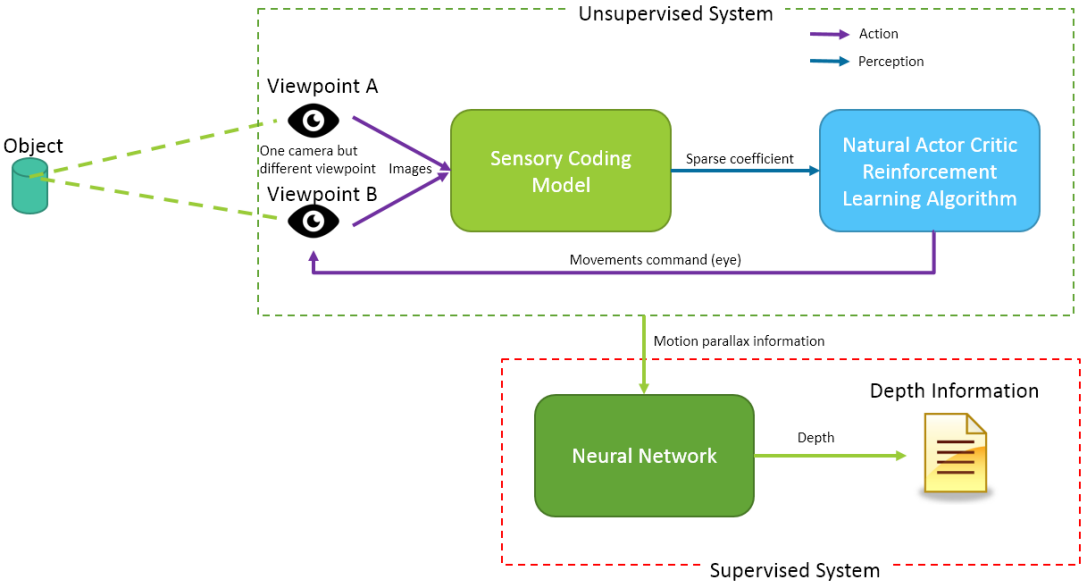


Figure 3.3: Motion parallax framework

3.3 Simulation

To test the extended framework, we test it on a simulation first. For motion parallax simulation, we use virtual experiment platform called V-REP to generate input images for motion parallax framework in MATLAB. The simulation environment is shown in Figure 3.4. The scene composes of a HOAP3 robot, a bookshelf, and background.

3.3.1 Simulation Setup

In this simulation, the lateral movement of the robot is generated by simply changing the position of the robot in the environment. The initial distance between the bookshelf and the robot is 1 meter. The robot moves from left to right by 50 centimeters for 5 steps. Each step, the robot moves for 10 centimeters. Thus we get 5 images for one lateral movement from left to right. We use only left eye of the robot to capture the images for motion parallax. The example of the images are shown in Figure 3.5. As mentioned above, two successive images will be input

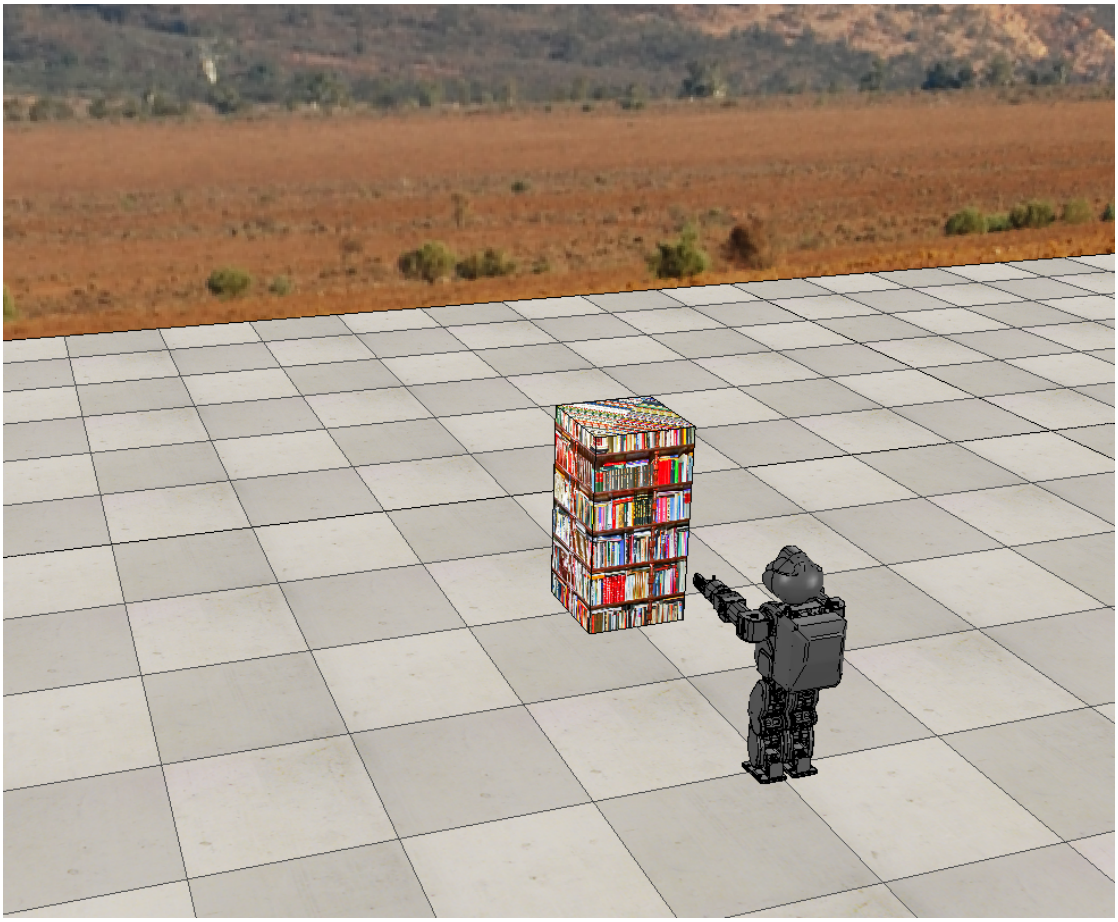


Figure 3.4: Motion parallax framework simulation by using V-REP

to the sensory coding model. The sensory coding model randomly selects two successive images for the input. After the two successive images are processed, we get the movement command for the eye rotation. However, to reduce the movement required for the robot, we use image shifting to virtually rotate the eye of the robot. Because, if we use the real rotation of the eye, the robot needs to perform lateral movement every time when eye rotates.

After the processes of the two successive images are finished for 15 iteration, the sensory coding model will randomly choose new two successive images from the same set of 5 images. After finish processing 5 sets of two successive images, the bookshelf in the V-REP simulation will be moved farther by 10 centimeters. The process is repeated. When the depth between the robot and the bookshelf reaches to 2 meters, the depth is reset to 1 meter. This is to ensure that every depth between 1 meter to 2 meters is trained. Every 14 iterations, we record the number of shifting pixel q (how much the eye rotates) correspond to the depth d

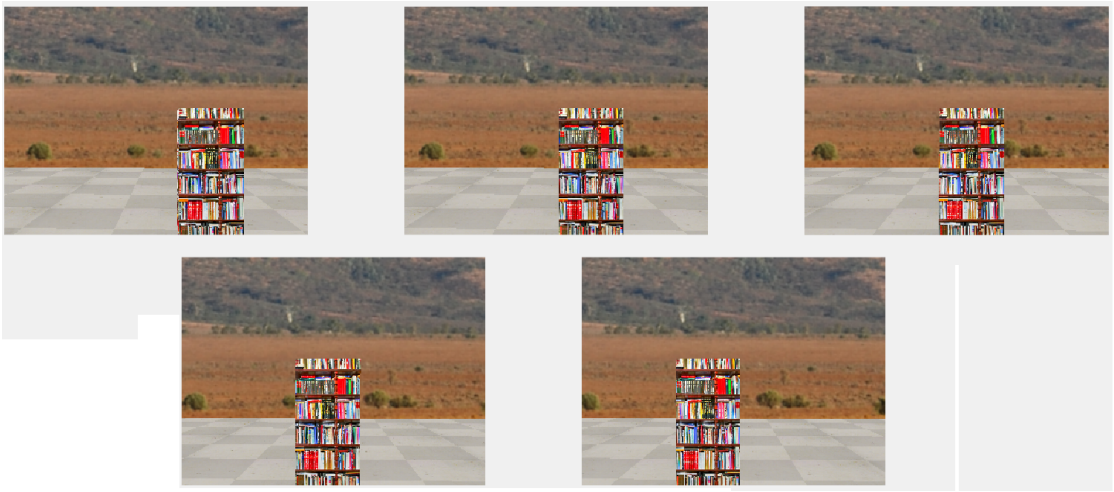


Figure 3.5: Example of motion parallax images from simulation (left to right)

at that iteration in a depth data matrix D.

$$D = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots \\ d_1 & d_2 & d_3 & \cdots \end{bmatrix} \tag{3.1}$$

When the training of the framework is satisfied [1], we continue to train the depth data. For the depth training part, we use a neural network toolbox provided in MATLAB to train the depth data D. We use a simple two layer feed-forward neural network with a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer (Figure 3.6). The number of neurons in hidden layer is 10. For training algorithm, we use Levenberg-Marquardt method. Because it is capable of solving most of the problems, and the depth data is very simple. In the first row of the depth data matrix D, we use it for the input of neural network. We use the second row of the matrix to be the target. 70-percent of the data is reserved for training. 15-percent is for validating. And another 15-percent is for testing.

3.3.2 Simulation Results

From Figure 3.7, we can see that we can fixate object in the same position for two successive images. However, there are still some errors as shown as AME (equation 2.16) in Figure 3.8. We can see that AME converges around 1 to 2 pixels. Although we could not achieve zero pixel AME, but we can still use the

[1] the framework can generate an overlapped of two successive images, or AME starts to saturate (see section 2.5.2)

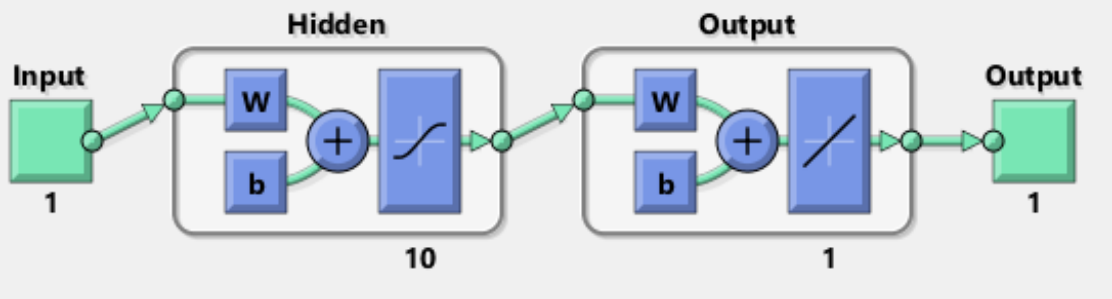


Figure 3.6: The neural network used in this simulation

eye movement information (amount of shifting in pixels) for finding depth of the object. Figure 3.9 shows error histogram of the trained neural network. It shows

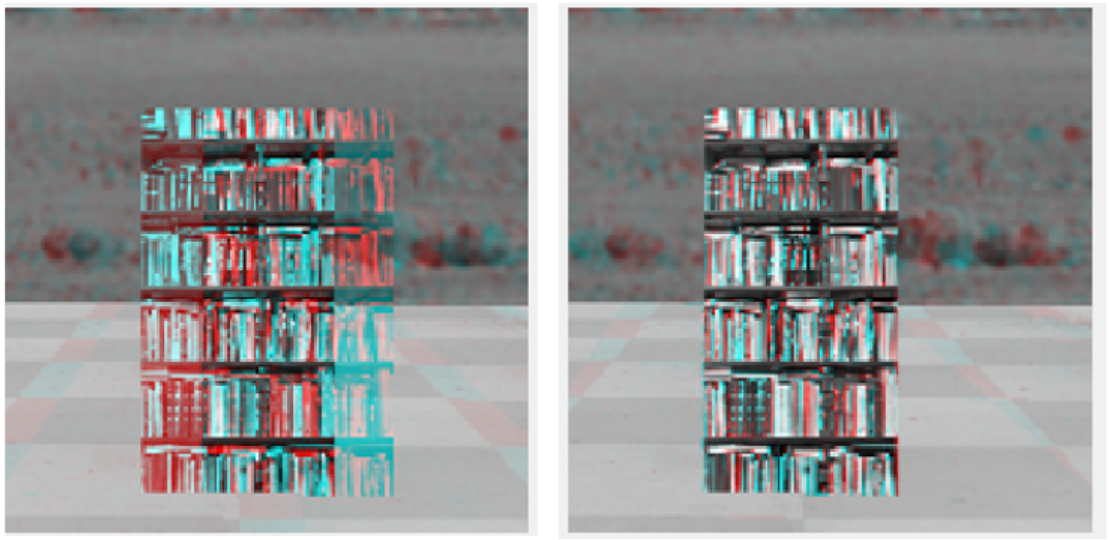


Figure 3.7: Example of object fixating in simulation

depth estimation error, the difference between real depth and output depth. Each bin contain instances that have error in that range. We can see that the most of the error is closed to zero. This tells that the neural network we used can handle the data very well.

Then, we test the framework by using the image at the same depth as in training. The depth of the object is varied from 1 meter to 2 meters increasing by 10 centimeters in the same way as in training period. The results and errors are shown in Table 3.1. Then we input some images at different depth other than the ones that are used in training. The result is shown in Table 3.2.

From the results, we can see that the framework can estimate the depth of the

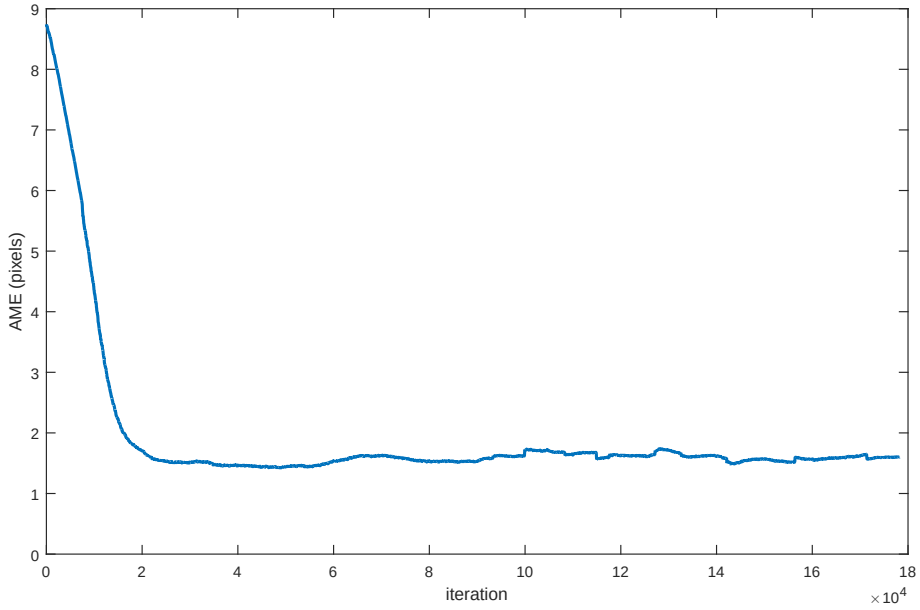


Figure 3.8: AME of HOAP3 simulation

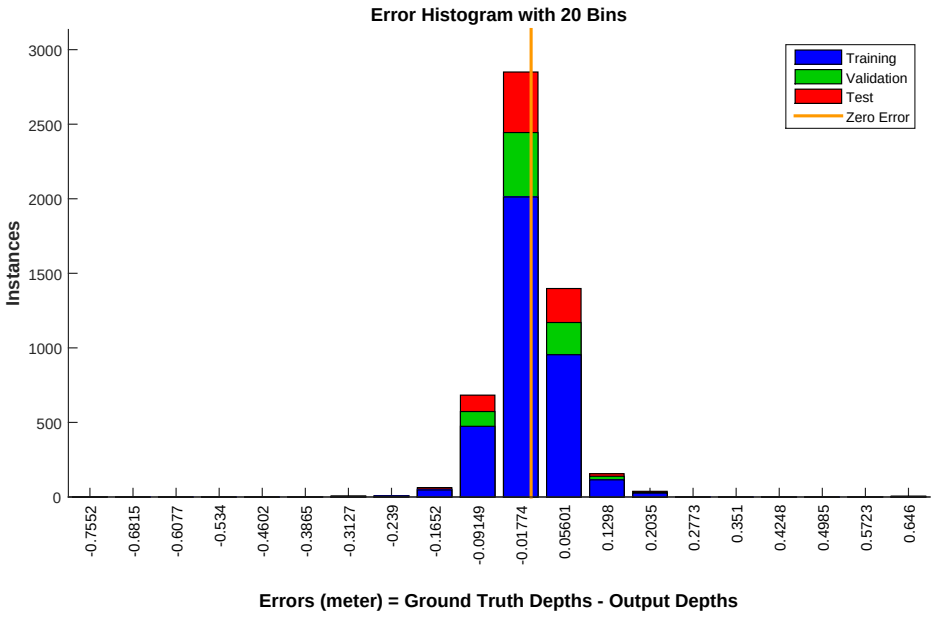


Figure 3.9: Neural network error histogram

Table 3.1: HOAP3 simulation result (training depths)

Input Depth (meter)	Output Depth (meter)	Error (centimeter)
1.00	1.02	2
1.10	1.10	0
1.20	1.20	0
1.30	1.27	3
1.40	1.47	7
1.50	1.47	3
1.60	1.60	0
1.70	1.81	11
1.80	1.86	6
1.90	1.91	1
2.00	1.99	1

Table 3.2: HOAP3 simulation result (random depths)

Input Depth (meter)	Output Depth (meter)	Error (centimeter)
1.25	1.29	4
1.53	1.60	7
1.77	1.86	9
1.92	1.90	2

object with some small errors. Although, for some depths, the framework outputs the same result. This is because there is still a little offset error of pixel shifting. Also, there is not enough space for eye movement (pixel shifting) to represent a certain depth. So, for the depths that are close together, there is a chance that the results are the same. So, in this case, we could only estimate depths in step of 10 centimeters. This problem could be solved by increasing size of image patch and resolution of input images to increase resolution of eye movement. Thus, there are more eye movement pixels to represent depths. However, increasing the patch and input image size will increase computation time.

3.4 Hardware

After we finished testing on the simulation, we test the framework in a real world. For the moving part, we did not use the HOAP3 robot, because time is limited and generating lateral motion for the robot takes time and difficult. Using the real HOAP3 robot will be considered in the future work.

3.4.1 Hardware Setup

The setup that we use in this experiment is shown in Figure 3.10. We use MATLAB to run the framework. A micro-controller, Arduino, is used for receiving command from MATLAB and controlling an XY-table. A camera is attached to the XY-table.

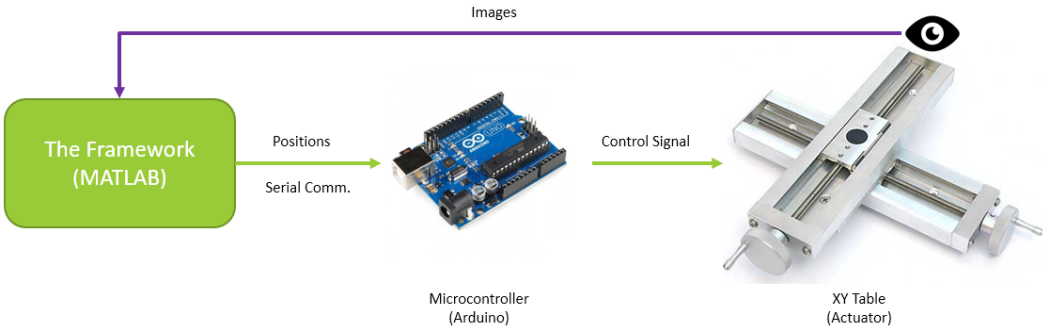


Figure 3.10: Setup for real world experiment

However, motion parallax requires only lateral movement, so we use only one axis of the XY-table. The flow of the system is the same as in the simulation, except that the camera and camera controlling part are in the real world.

In this experiment, we have the XY-table and object on a floor. As shown in Figure 3.11, the camera (blue eye symbol) can move laterally to generate the motion parallax images. The depth between the camera and the object (black cube with red stripe) will be varied by hand manually. In this case, the camera will move laterally for 12 centimeters. Each step move for 3 centimeters, thus we have 4 images per lateral movement. The view from the camera is shown in Figure 3.12.

In order to make training easier, we gather all of the data required to train before run the training. We capture all images generated from lateral movement in various depth, from 40 centimeters to 1 meter (each step increased by 10 centimeters). Then we use the set of images that we have gathered to train the framework.

3.4.2 Experimental Results

Figure 3.13 shows an example of tracking of object from two successive frames. Figure 3.14 shows AME of pixel shifting.

Figure 3.15 shows error histogram of the trained neural network. We test the framework in the same way as in HOAP3 simulation. The results are shown in Table 3.3, and Table 3.4.

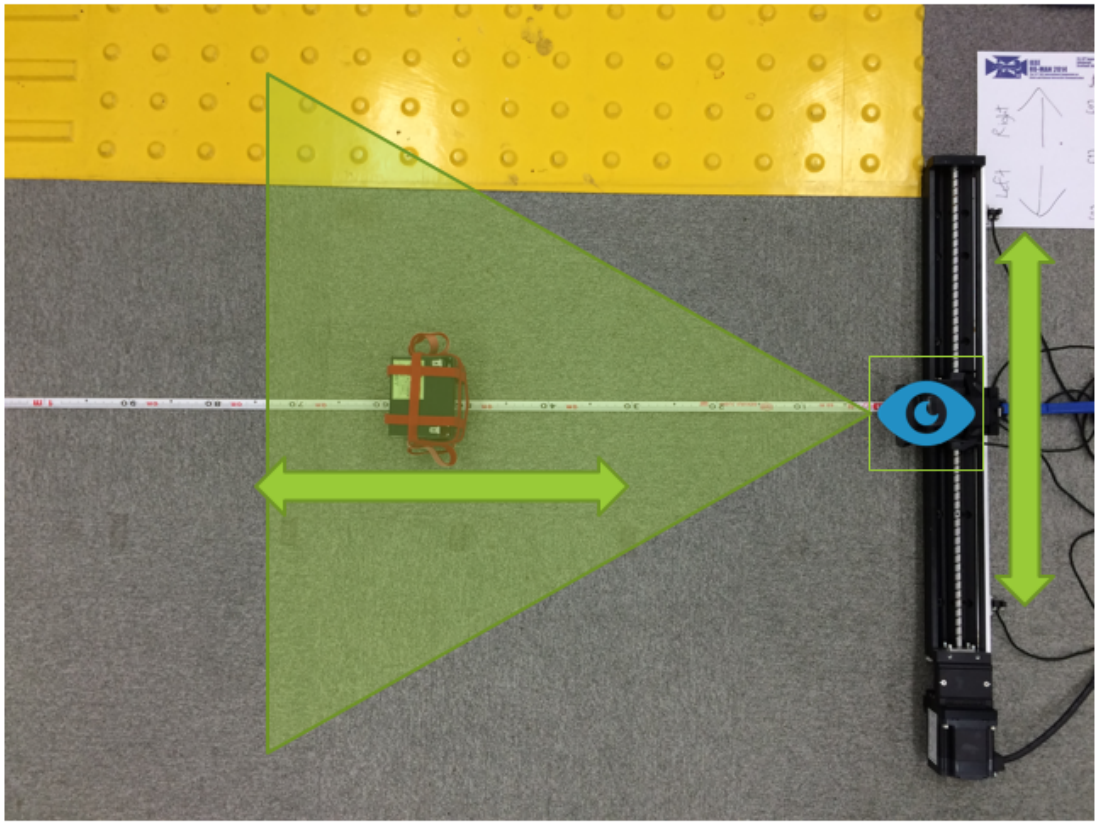


Figure 3.11: XY-table and the object

Table 3.3: Experimental result (training depths)

Input Depth (centimeter)	Output Depth (centimeter)	Error (centimeter)
40	35.34	4.66
50	49.40	0.60
60	54.33	5.67
70	72.18	2.18
80	83.45	3.45
90	88.76	1.24
100	91.98	8.02

The experimental results are similar to the simulation results in the HOAP3 simulation. As discussed in the simulation section 3.3.2, we can increase the resolution of the input images and patch size to increase perceivable depth resolution. But, it comes with costs of computation time.

We showed that the framework can be trained by using only one cost function

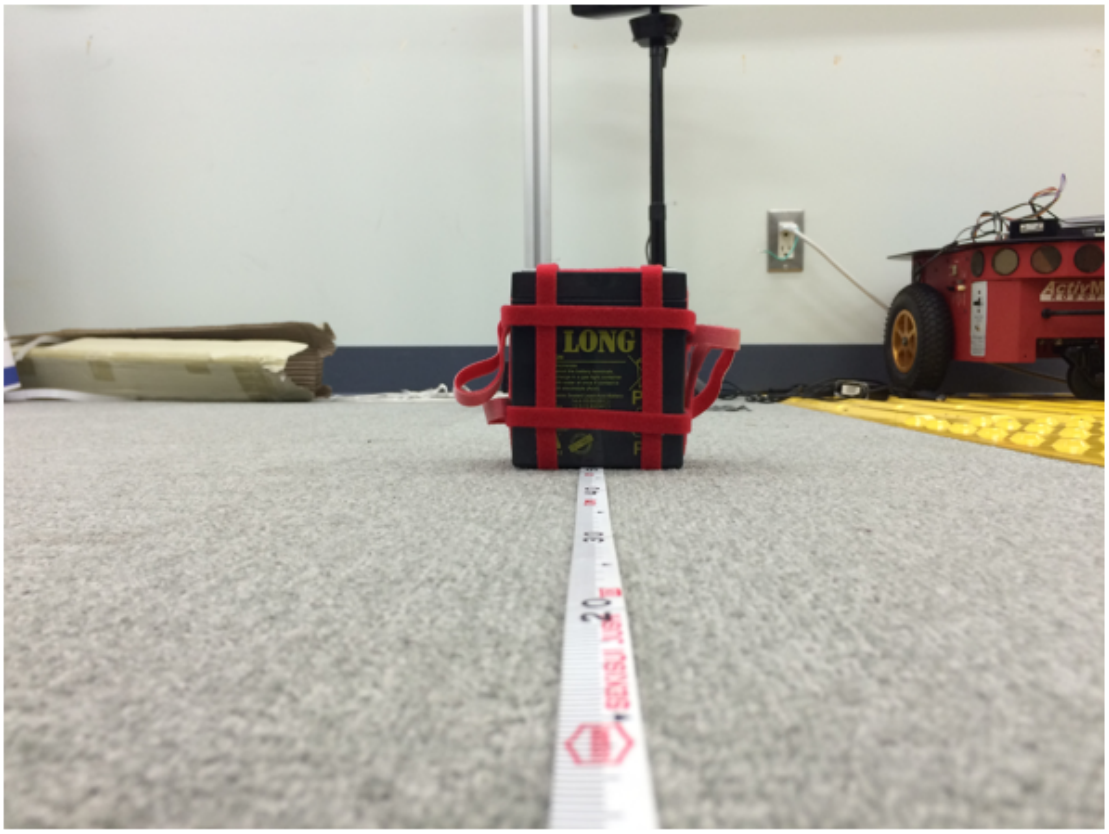


Figure 3.12: View from camera

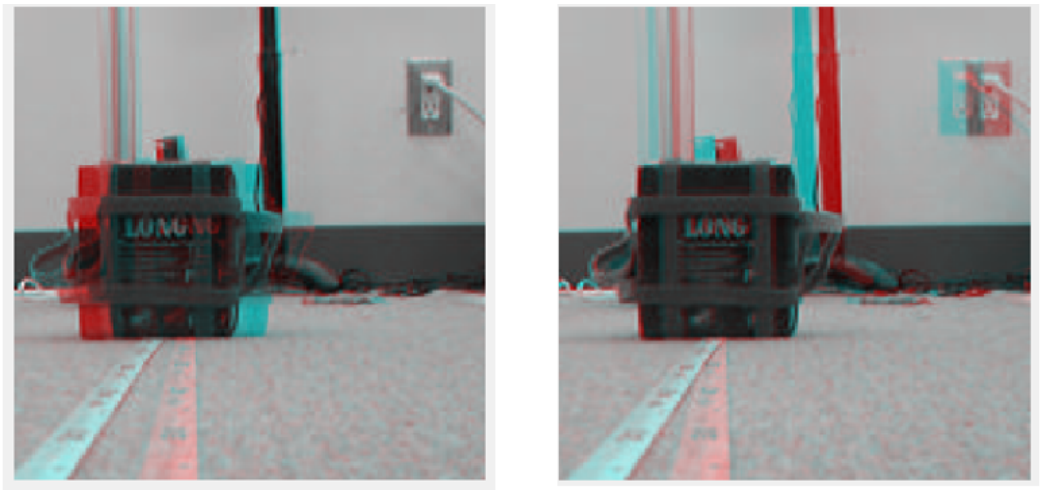


Figure 3.13: Example of object fixating image from real world

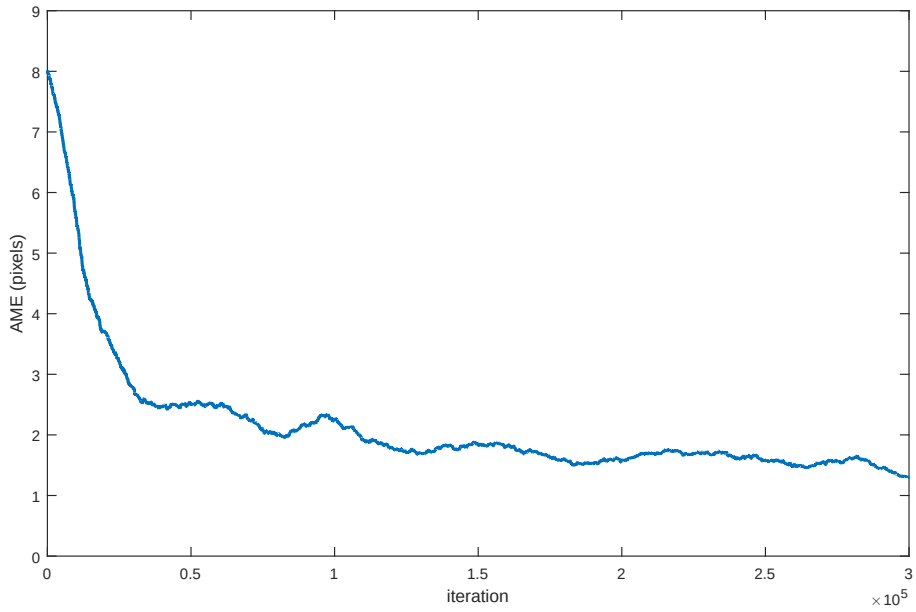


Figure 3.14: AME of real world experiment

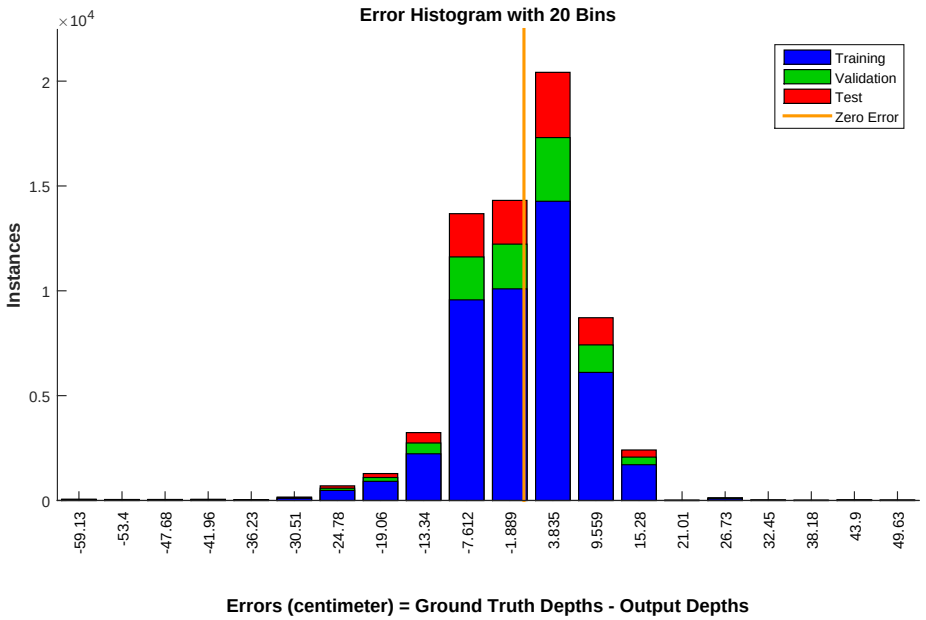


Figure 3.15: Neural network error histogram

Table 3.4: Experimental result (random depths)

Input Depth (centimeter)	Output Depth (centimeter)	Error (centimeter)
45	47.90	2.90
65	60.35	4.65
85	88.76	3.76

which is reconstruction error. The framework autonomously learn perception and behavior simultaneously, i.e. joint development learning. They could improve each other. Only two successive images from lateral movement are used to estimate depth. The experiment and simulation setup shows that the system does not require calibration. Finally, we can use a simple linear regression model, neural network, to utilize eye movement to estimate the depths. So, the framework is autonomous learning and self-calibrating.

3.5 Chapter Conclusion

We have proposed a method to extend the active depth perception of original framework proposed in [8]. In addition, we proposed a method to use the information from motion parallax to estimate the depth between camera and the object.

Chapter 4

Conclusions

This last chapter will conclude our work done in this thesis and the contribution made to network throughput. It will also give some points which can further studied and improved in this thesis.

4.1 Concluding Remarks

In order to perceive world in three dimension, depth perception is required. Depth perception is the visual ability to perceive the world in three dimensions and the distance of an object. Depth perception is the most fundamental artificial vision problem that must be solved. There are three kinds of depth perception which are stereo vision, motion parallax, and optic flow.

In Chapter 1, we first give a brief explanation of depth perception. We pointed out the lack of robustness in conventional artificial vision system. To deal with this problem, we proposed a method that adapted biological vision systems with artificial vision system by using motion parallax active depth perception. At first, we studied a framework proposed in [8].

In Chapter 2, we have explained the step and procedure for each module in the framework. We also considered to use the multi-scale image extension proposed in [9]. We showed that the system is can generate vergence eye movements. In addition, we made the simulation to test our understanding of the framework and to be extended later with motion parallax active depth perception.

In Chapter 3, we proposed a method to extend te active depth perception with motion parallax. We use the concept of motion parallax movement which when we are moving laterally, our eyes try to fixate the object. We utilized eye rotation information to generate depth information by using two layers neural network. We tested the framework in both simulation and real world experiment. The result of both simulation and real world experiment are acceptable, although there are

some depth errors. The results could be improved, if we used larger resolution of input images and patch size. However, it will increases computation time.

4.2 Contributions

The main contributions of this thesis would be

- Motion parallax extension

  We have proved that an extension to active depth perception part of the framework is possible. We can use the framework to fixate the object while moving laterally. This tells us that we could extend and develop this framework farther.

- Depth estimation

  The framework could only fixate the object between two cameras or two successive images, but it still lacked of depth estimation. We have developed a way to utilize movement information to extract depth information.

- Smooth pursuit of lateral movement

  As we have discussed in Chapter 1, in [13], they developed a model that can pursuit a moving object by using stereo vision while maintaining zero disparity. However, in our case, the framework could fixate the object while the camera is moving at a unit speed with one camera.

4.3 Open Questions

In this thesis, there are still some open questions remaining for further development.

4.3.1 Optic Flow Extension

As we have mentioned before that there are three types of depth perception which are stereo vision, motion parallax, and optic flow. We have studied the stereo vision framework. We have proposed a way to extended the active depth perception with motion parallax. However, there is one remaining type of depth perception that have not yet been used yet which is optic flow. To perceive depth by optic flow, we have to move forward and backward. When we are moving forward and backward, we can sense that the closer object have the size increased more than the object that is far away. So, the question is left that can we utilize those information to extend the framework.

4.3.2 Depth Perception Integration

The prospective of this thesis is to mimic the depth perception system in developed organisms. So, it is interesting that whether it is possible to integrate depth perception into a single framework or not. For example, we may integrate motion parallax and stereo vision together. The framework is able to decide to choose which depth perception is more reliable in a specific situation and environment.

4.3.3 Depth of Multiple Objects

Even though, we can extended the active depth perception part with motion parallax and estimate the depth, we can only find depth of a single object in the scene. So, there is an interesting question that whether the framework can estimate depths for multiple objects or not. We may implement an algorithm that divide the image for each object and input into the framework.

4.4 Future Directions

In the future, this project will be farther developed in my Doctoral research project. The remaining active depth perception, optic flow, will be used to extended the depth perception of the framework. Finally, a way to complete integrated of three active depth perception will be proposed.

Bibliography

[1] B. Kuipers, P. Beeson, J. Modayil, and J. Provost, "Bootstrap learning of foundational representations," *Connection Science*, vol. 18, no. 2, pp. 145–158, June 2006.

[2] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *Autonomous Mental Development, IEEE Transactions on*, vol. 4, no. 1, pp. 70–86, March 2012.

[3] J. Weng and M. Luciw, "Brain-like emergent spatial processing," *Autonomous Mental Development, IEEE Transactions on*, vol. 4, no. 2, pp. 161–185, June 2012.

[4] F. Attneave, "Some informational aspects of visual perception," *Psychol. Rev*, pp. 183–193, 1954.

[5] H. B. Barlow, *Possible principles underlying the transformation of sensory messages*. Cambridge, MA: MIT Press, 1961.

[6] D. J. Field, "What is the goal of sensory coding?" *Neural Comput.*, vol. 6, no. 4, pp. 559–601, Jul. 1994.

[7] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311 – 3325, 1997.

[8] Y. Zhao, C. Rothkopf, J. Triesch, and B. Shi, "A unified model of the joint development of disparity selectivity and vergence control," in *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, Nov 2012, pp. 1–6.

[9] L. Lonini, Y. Zhao, P. Chandrashekhariah, B. Shi, and J. Triesch, "Autonomous learning of active multi-scale binocular vision," in *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, Aug 2013, pp. 1–6.

[10] L. Lonini, S. Forestier, C. Teulire, Y. Zhao, B. E. Shi, and J. Triesch, "Robust active binocular vision through intrinsically motivated learning," *Frontiers in Neurorobotics*, vol. 7, no. 20, 2013.

[11] E. Nawrot, S. Mayo, and M. Nawrot, "The development of depth perception from motion parallax in infancy," *Perception and Psychophysics*, vol. 71, no. 1, pp. 194–199, 2009.

[12] T. Mann, Y. Park, S. Jeong, M. Lee, and Y. Choe, "Autonomous and interactive improvement of binocular visual depth estimation through sensorimotor interaction," *Autonomous Mental Development, IEEE Transactions on*, vol. 5, no. 1, pp. 74–84, March 2013.

[13] C. Teulire, S. Forestier, L. Lonini, C. Zhang, Y. Zhao, B. Shi, and J. Triesch, "Self-calibrating smooth pursuit through active efficient coding," *Robotics and Autonomous Systems*, vol. 71, pp. 3 – 12, 2015, emerging Spatial Competences: From Machine Perception to Sensorimotor Intelligence.

[14] T.-Y. Kuo and Y.-C. Lo, "Depth estimation from a monocular view of the outdoors," *Consumer Electronics, IEEE Transactions on*, vol. 57, no. 2, pp. 817–822, 2011.

[15] A. Saxena, J. Schulte, and A. Y. Ng, "Depth estimation using monocular and stereo cues." in *IJCAI*, vol. 7, 2007.

[16] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," *International journal of computer vision*, vol. 76, no. 1, pp. 53–69, 2008.

[17] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 593–600.

[18] H. Zhuang, R. Sudhakar, and J.-y. Shieh, "Depth estimation from a sequence of monocular images with known camera motion," *Robotics and autonomous systems*, vol. 13, no. 2, pp. 87–95, 1994.

[19] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[20] J. Semmlow, G. Hung, and K. Ciuffreda, "Quantitative assessment of disparity vergence components," *Investigative ophthalmology and visual science*, vol. 27, no. 4, p. 558564, April 1986.