Action Analysis and Control Strategy for Rat Robot Automatic Navigation

by

Minlong Lu

B.Eng., Zhejiang University, 2011

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2018

Approval

Name:	**Minlong Lu**
Degree:	**Doctor of Philosophy (Computing Science)**
Title:	**Action Analysis and Control Strategy for Rat Robot Automatic Navigation**
Examining Committee:	**Chair:** Qianping Gu Professor

Ze-Nian Li
Senior Supervisor
Professor _____

Gang Pan
Co-Supervisor
Professor
Zhejiang University _____

Yueming Wang
Supervisor
Professor
Zhejiang University _____

Mark Drew
Internal Examiner
Professor
School of Computing Science _____

Jiying Zhao
External Examiner
Professor
School of Electrical Engineering
and Computer Science
University of Ottawa _____

Date Defended: September 17, 2018

Abstract

A rat robot is an animal robot, where a rat is connected to a machine system via a brain-computer interface. Electrical stimuli can be generated by the machine system and delivered to the rat's brain to control its behavior. The sensory capacity and flexible motion ability of rat robots highlight their potential advantages over mechanical robots.

There are two challenges of rat robot automatic navigation. The first challenge is to recognize the action status of the rat robot, which is an essential feedback for determining the stimuli/instructions for it to accomplish certain movements. The second challenge is the design of the automatic instruction model that steers the rat robot to perform navigation. Due to inherent characteristics and instincts of the rats, the controlling strategy of the rat robots is different from mechanical robots.

In this thesis, we propose a new idea for analyzing the action states of the rat robot. A miniature camera is mounted on the back of the rat robot and the egocentric video captured by the camera is used to analyze its action. We propose two action analysis methods. The first method is based on an optical flow algorithm and the second method incorporates deep neural networks. We propose two automatic instruction models. The first model learns from manual control data to mimic the human controlling process, and the second model issues instructions according to human experts' knowledge. We build a rat robot and apply these models to enable it to navigate in different scenes automatically.

In order to produce more accurate optical flow estimation, we propose a row convolutional long short-term memory (RC-LSTM) network to model the spatial dependencies among image pixels. Our RC-LSTM network is integrated with Convolutional Neural Networks and achieves competitive accuracy. To analyze potentially more complex actions from the egocentric videos, we extend our deep neural networks used for rat states analysis to be a two-stream architecture. A spatial attention network is incorporated to help our model to focus on the relevant spatial regions to recognize actions. Our model is evaluated on two egocentric action recognition datasets and achieves competitive performance.

Keywords: Automatic Navigation; Action Recognition; Instruction Model; Rat Robot; Deep Neural Networks

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Research Background

An animal robot is an animal that is connected to a machine system, usually by a brain-computer interface (BCI). Electrodes are implanted into the specific brain regions, through which electrical stimuli can be delivered to the animal's brain to induce certain behavior, thereby driving the animal to take actions that are specified by humans [33]. During the past years, various animal robots have be developed based on aerial, aquatic, and terrestrial animals. Animal robots have advantages over traditional mechanical robots due to the specific motion and perceptual abilities of animals [67] and have great potential for use in rescue and search applications [23].

A rat robot is an animal robot which was first developed in [95]. The rat robot system is built based on rat. It also contains three main components: implanted electrodes, a rat-mounted pack, and a controller, as shown in Figure 1.1. The controller is a computer or a portable device that allows human to specify the instructions and send them to the rat robot through its wireless modules. The rat-mounted pack is a small set of hardware circuits assembled on the rat's back, which receives the instructions and generates corresponding electrical stimulation pulses. The electrical stimuli are delivered to the specific brain regions via the implanted electrodes, which motivates the forward and turning movements of the rat robot. The rat robot could be guided to navigate along human specified routes under manual control. During the manual control process, humans need to identify the arrangements of objects in the environment and observe the action status of the rat before giving appropriate instructions to facilitate navigation, which limits the applications of rat robots in environments that humans cannot observe. This disadvantage motivates the need of rat robots that are capable of performing automatic navigation without the need of human operations.

The research on rat robot automatic navigation is still in a preliminary stage. A few rat robots have been developed that can perform simple tasks automatically in constrained artificial scenes [91, 123]. The rat robot automatic navigation systems employ computer vision

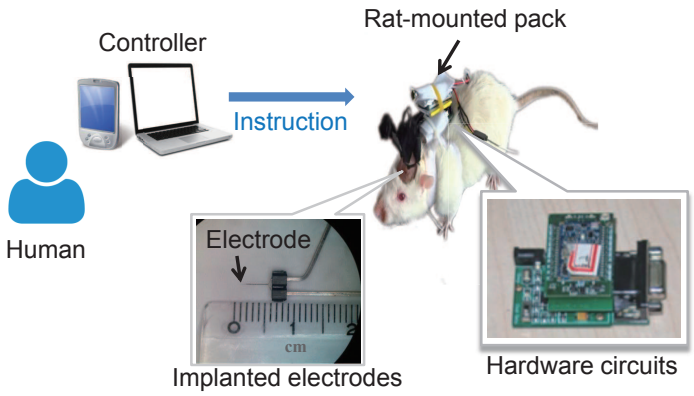


Figure 1.1: The main components of the rat robot system.

algorithms, such as object detection and tracking, to help understand the environmental layouts and analyze the rat robot action states. These visual cues are utilized by the controlling algorithms to issue instructions automatically to guide the rat robot to accomplish navigation.

1.2 Challenges

There are two challenges of rat robot automatic navigation:

The first challenge is to automatically recognize the action and motion states of the rat robot. The rat states provides essential feedback for determining the appropriate instructions to be sent for the rat robot to accomplish certain movements. For example, suppose that the rat robot reaches a junction and it is expected to take a left turn. One "left" instruction may be sufficient if its head is pointing forward in the same direction as its body, whereas two "left" instructions would be necessary if its head is currently pointing to the right. In addition, to obtain a successful motion, the "left" instruction should be followed by a "forward" instruction when the rat's head actually turns left. In order to monitor the rat states, existing automatic navigation systems equip a bird's eye camera on top of the scene, as illustrated in Figure 1.2. With the video captured by this third person view camera, computer vision algorithms are employed to segment the rat from the background and analyze its action states. However, the requirement of a top camera limits the applications of these rat robots.

The second challenge is the design of the instruction model that steers the rat robot to perform navigation. Due to the inherent characteristics and instincts of the rats, the control of the rat robots is very different with the mechanical robots. For example, the thigmotactic scanning [64, 62] is the one of the instinct behaviors of the rat to use the vibrissae to sense the edge/wall of a path while traversing. During navigation, the head of the rat robot tends to lean to one side if it's using the vibrissae of this side to scan the edge. In this case, it is easier to make the rat robot to turn to this side, while more

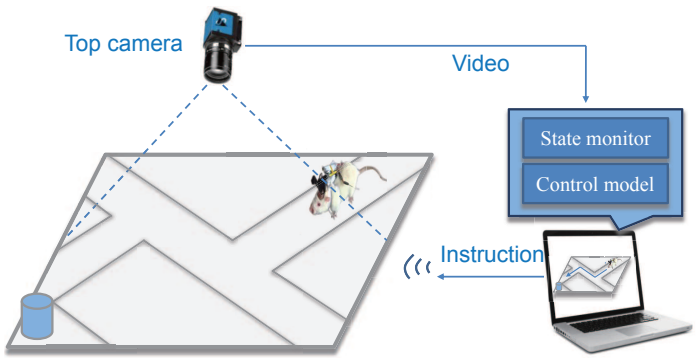


Figure 1.2: Existing automatic navigation systems with a bird's eye camera.

instructions would be needed to make the rat robot to turn to the path of the other side at a junction. Besides, the response to a stimulus varies slightly for different rats and depends on the physical condition of the rat, and it may also be affected by the action triggered by its own instincts and habits. Therefore during manual control, humans need to observe the rat's states and issue a series of instructions to enable the rat robot to perform a corresponding motion successfully. Some existing automatic control algorithms attempt to mimic the human control process by building the connections between the rat states and the instructions. However, the effectiveness of the automatic control is still not as good as that of the manual control.

1.3 Contributions

We address the aforementioned challenges and achieve effective rat robot automatic navigation. We propose a new idea for analyzing the states of the rat robot without the need of a top bird's eye camera. A miniature camera is mounted on the back of the rat robot, and the rat action states are analyzed using the video captured by the rat-mounted camera. To the best of our knowledge, our work is the first attempt to use first-person camera to facilitate rat robot automatic navigation. In this work, we propose two action analysis methods and automatic instruction models, based on which we build a new rat robot for automatic navigation. We then propose two models based on deep neural networks, which are useful for potentially more complex rat robot action analysis tasks. In the following, we highlight our main contributions in detail.

1.3.1 Egocentric Action Analysis for Rat Robot

We mount a miniature camera on the back of the rat in the way that its optical axis is arranged in the same direction as the rat head. When the rat head moves, the camera moves accordingly. The rat action states are defined to be its head orientation and head motion direction, which can be represented by the orientation and motion of the camera

and therefore be analyzed from the egocentric video captured by the camera. We propose two methods for the egocentric action analysis for the rat robot. The first method is based on a traditional optical flow algorithm, in which the rat head motion direction is estimated using the average motion of the feature points in the consecutive frames. The rat head orientation is inferred based on the position of the objects in the frame and the motion of the rat head. In the second method, we model the rat states analysis as a classification problem, where each video frame is assigned to a orientation and motion category. We incorporate Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) for orientation analysis. A devised CNN that takes the two consecutive frames as input is used to predict the motion direction. These models produce the rat action states for automatic navigation in open scenes without the need of a top camera above the scene.

1.3.2 Automatic Instruction Models for Rat Robot Control

In order to achieve successful manual control, the human operators usually observe the environmental layout as well as the rat states, based on which they give a series of stimuli to the rat robot. Inspired by this fact, we believe that automatic instruction models need to take into consideration of the following two visual cues: 1) the feedback of the rat robot states obtained from the egocentric videos, 2) the objects of interest that indicate the action expectations for the rat robot, e.g. road signs. We propose two instruction models that issue a stimulus sequence automatically according to these visual cues. The first model employs a human-like instruction model which learns to mimic the human controlling process using data collected during the manually controlled navigation. The visual cues that correspond to each instruction are extract from the video to train a three-class support vector machine (SVM). The second model incorporates a rule-based instruction model designed according to human experts' experience and knowledge. A thorough set of rules of issuing stimulus are defined for all possible scenarios during the automatic navigation. This model is able to decide whether to issue a stimulus and what stimulus to issue, based on the rat states, detected objects as well as the previous stimuli.

We build a new rat robot system, which we refer to as a *rat cyborg*, and apply the egocentric action analysis methods and the instruction models to enable it to perform automatic navigation in different scenes. The rat cyborg was built by a group of colleagues. The electronics work and rat surgical work were completed by the collaborators. The author's contributions were the analysis of the visual data and transforming the results to the neural stimuli to control the rat robot for automatic navigation. The experiments that involve rats were conducted and finished when the author was at Zhejiang University, China. All of the rats used in the experiments were cared well by the animal keepers. All of the experiments were conducted in accordance with the guidelines issued by the Ethics Committee of Zhejiang University as well as the Guide for the Care and Use of Laboratory Animals of China Ministry of Health. This study was approved by the Ethics Committee of Zhejiang

University, and the Ethics Committee of Zhejiang University performs strict and sustained ethical monitoring to guarantee that this technology will not be misused.

1.3.3 Deep Recurrent Network for Optical Flow Estimation

Optical flow encodes the motion between the consecutive frames and is used in one of our methods to analyze the action states of the rat robot. We believe that a more accurate optical flow model can result in a better rat states estimation. Therefore we explore the research of using deep neural networks for dense per-pixel optical flow estimation. It is challenging to adapt CNNs designated for high-level vision tasks to handle pixel-level predictions. This is because CNNs do not have a mechanism to explicitly model contextual dependencies among image units. Besides, the convolution and pooling operations result in reduced feature maps and hence produce coarse outputs when upsampled to the original resolution. These aspects render CNNs limited ability to delineate object details, which often result in inconsistent predictions. We propose a recurrent neural network to alleviate this issue. Specifically, a row convolutional long short-term memory (RC-LSTM) network is introduced to model contextual dependencies of local image features. This recurrent network is integrated with CNNs to form an end-to-end trainable network, which can learn context-aware features for more accurate optical flow estimation.

1.3.4 Deep Attention Networks for Egocentric Action Recognition

In order to handle potentially more complex action analysis from the egocentric videos, we extend our deep neural networks used for rat states analysis to be a two-stream architecture, which consists of an appearance-based stream and a motion-based stream to recognize actions. This follows the successful practices to decompose videos into spatial and temporal components for action recognition [88, 63]. The appearance stream learns the spatial component, which contains appearance information about scenes and objects depicted in the raw frames. The motion stream learns temporal component, which conveys the motion of the camera and objects in the scene across the frames. We incorporate a spatial attention network in each of the streams to predict an attention map. The attention maps help our model to identify and focus on the most relevant spatial region of the frames to recognize actions. A temporal network is incorporated in each stream, which contains a bi-directional LSTM to better exploit the temporal structures of the egocentric videos for action recognition. Our model is evaluated on two egocentric action recognition datasets which contain fine-grained human actions with more categories and achieves competitive performance.

1.4 Organization

This thesis is organized as follows: in Chapter 2, we review some of the related work. We present our action analysis methods for rat robots using the egocentric videos in Chapter 3,

and we describe our automatic instruction models in Chapter 4. Our rat cyborg automatic navigation system based on the action analysis and instruction models is introduced in Chapter 5. In Chapter 6, we introduce our deep recurrent network for optical flow estimation, which is intended for providing more accurate optical flow for the first rat states analysis method in Chapter 3. In Chapter 7, we present our deep attention networks for egocentric action recognition, which is an extension of the deep neural networks used for rat states analysis in Chapter 3. Chapter 8 is the conclusion.

Chapter 2

Related Work

In this chapter we review some of the previous work that are related to the problems studied in this research. We start by introducing the different types of animal robots. We then review the action recognition methods for third-person videos as well as egocentric videos. More related work can be found in some of the following chapters.

2.1 Animal Robots and Navigation

Existing animal robots can be categorized according to three classes: aerial, aquatic, and terrestrial animal robots.

The first aerial animal robot was described in 1997, where the locomotory reaction of a *Periplaneta americana* in response to various electrical stimuli was analyzed. Using two photosensors as inputs, an electronic backpack was used to drive the insect to walk along a black line [33]. The success of the *Periplaneta americana* robot greatly inspired research into animal robots based on insects. Microprobes were implanted in *Manduca sexta* during its early metamorphosis to allow direct control of its wing motions [9], thereby controlling the flight direction in *Manduca sexta*. In [81], stimuli were delivered to the brains of beetles to elicit, suppress, or modulate wing oscillations, thereby controlling flight initiation, cessation, and elevation. Turns were triggered by direct muscular stimuli. In [5], the flight initiation and cessation behaviors of honeybees were generated in a reproducible manner by using electrical pulses between two wire electrodes implanted in the honeybee's brain. The effects of different stimulus patterns on the honeybee's behavior were compared.

The second category of animal robots is based on aquatic animals. In [45], locomotion control in the horizontal plane was accomplished in goldfish by stimulating the Nflm region in the midbrain. When the stimulation site was closer to the Nflm region, a lower stimulus intensity was required to evoke movements in goldfish. In [69], it was shown that turning left, turning right, moving forward, and moving backward behaviors could be induced in adult carp via electric stimulation of the cerebellum.

Terrestrial animal robots such as *Gekko gecko* [112] and rats [95, 23, 36, 57, 101, 113] have also been investigated. A rat robot was first developed in [95], where applications of electrical stimuli to the somatosensory cortices (SI) and medial forebrain bundle (MFB) were used as cues and rewards, respectively. The rat could easily be guided through pipes and across elevated runways, and it could even be instructed to climb or jump from trees. A new method for rat robot navigation was proposed in [36], which was based on virtual punishment. Electrical stimuli were delivered to the *thalamic ventral posterolateral nucleus* and *amygdala nucleus* of the rat brain. In response to these virtual sensations, rats would change directions and escape in an active manner. In [57], the immobility behavior triggered by *dorsolateral periaqueductal gray* (dlPAG) stimulation was investigated. By stimulating the MFB or dlPAG during navigation, a rat could be switched between active and motionless states, where the motionless period was controlled to a certain extent. In [101], a new locomotion control scheme was developed for rat navigation. A few efforts have been made to develop a rat robot that is able to perform automatic navigation [91, 123]. The existing systems usually required a bird's eye camera to be equipped on the top of the scene to monitor the rat states, which limits the applications of these rat robots.

2.2 Action Recognition and Egocentric Vision

Action recognition has been one of the key problems in computer vision [1]. It is often studied in a surveillance setting, where the action of the people captured in a third person view camera is recognized. A large family of action recognition methods were based on high-dimensional encodings of local features, such as histogram of oriented gradients (HOG) [15], histogram of flow (HoF) [16] and motion boundary histograms (MBH) [103]. These features were usually extracted from Space-Time Interest Point (STIP) [50] or along dense trajectories (DT) [102], and could be encoded into the bag of word (BoW) representation for action recognition in third person videos. There have been a number of attempts to develop deep neural networks for action recognition [42, 39, 105]. A two-stream architecture was proposed in [88], which fed video frames and optical flow images into separate CNN streams and the scores of the two streams were fused for the prediction. Temporal models such as long short-term memory (LSTM) [32] were employed on top of the CNNs for modeling long-range temporal dependencies for action recognition in videos [17, 37, 106]. The convolution and pooling operations of CNNs were extended to 3D in models such as C3D [97] and I3D [13], which directly learn spatiotemporal features from the videos. The 3D operations were factorized into separate spatial and temporal components in [98] to facilitate the optimization and obtain significant results. The non-local neural network proposed in [107] captured long-range dependencies for action recognition by computing the response at a position as a weighted sum of the features at all positions.

There have been several advances in egocentric vision, such as video summarization [115, 120], video stabilization [47, 46, 2], object recognition [73] as well as action recognition [44, 71]. A pairwise deep ranking model was proposed in [120] for video highlight detection. The obtained highlight video segments were used to generate video summarization in both video time-lapse and video skimming ways. Gaze tracking information (e.g. fixation) can be used to help the summarization task and enables deriving personalized summaries [115], and the model was formulated as sub-modular function maximization with partition matroid constraints. It was shown in [35] that the egocentric video provides unique identity information. The camera motion includes the body motion information of the camera wearer and can be used to recognize his/her identity. In [122], the dynamics of social interactions between two people were analyzed by recognizing their actions and reactions using the paired egocentric videos. Action maps were generated for the egocentric videos to indicate the action that the camera wearer can perform on the specific regions, which indicate the functionality of the parts in the scene [76]. The egocentric video were also used to predict the future behavior of the wearer, such as future localization [90] and activity forecasting [77].

In egocentric action recognition, the first-person videos or egocentric videos are used to understand the camera wearer's behavior. Researchers have found that traditional spatial-temporal features do not work well due to the camera motion [20, 22]. With the help of motion compensation, large improvement were achieved using these features [54]. Object-centric features [71] were used to capture the appearance changes of objects, and egocentric cues based on head movement, hand pose and gaze were proposed for better characterizing egocentric actions [54]. Many attempts have been made to employ deep CNNs to tackle egocentric action recognition problem [63, 89, 126]. To directly incorporate egocentric cues, an Ego ConvNet [89] was proposed to train on stacked input of hand mask, homography image, and saliency maps. The ego stream was then fused with other two streams for the final prediction. In [63], networks were trained to segment hand and localize object, and then the object of manipulation was cropped as the input to the appearance stream. The appearance and motion streams were fused by a fully-connected layer to recognize the objects and action jointly.

Chapter 3

Egocentric Action Analysis for Rat Robot

The rat states provide essential feedback for determining the appropriate instructions to be sent for the rat robot to accomplish certain movements. Instead of relying on a bird's eye camera equipped on top of the scene to monitor the rat action, we mount a miniature camera on the back of the rat robot with its optical axis arranged in the same direction as the rat head. The rat action states are defined to be its head orientation and head motion direction, which can be represented by the orientation and motion of the camera and therefore be analyzed from the egocentric video captured by the rat-mounted camera. In this chapter, we introduce our methods which use a traditional optical flow algorithm and deep neural networks to analyze the rat action states.

3.1 Rat States Analysis based on Optical Flow

We define the rat action state as $S = (\theta, V)$, where V is the rat head motion direction and θ is the rat head orientation. The rat head motion direction is estimated based on the average motion of the feature points in two consecutive video frames. It should be noted that when the rat head moves in one direction, the feature points in the video move in the opposite direction. The main steps used to estimate the motion direction of the rat's head comprise feature detection, feature tracking, and direction computation.

- Feature detection: In this step, we initialize a set of feature points for tracking in consecutive frames. We use the Harris corner detection method [29] to extract corner features from the frame $I(x, y, t)$. The autocorrelation matrix M is computed from the image derivatives as follows:

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_x^2 \end{bmatrix}, \tag{3.1}$$

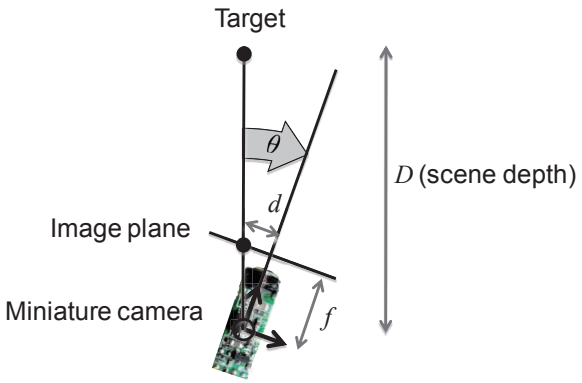


Figure 3.1: Rat head orientation estimation.

where $w(x, y)$ is a window function and I_x denotes the partial derivative of the pixel value with respect to the x direction. The corner response is defined as $R = \det(M) - k \times \text{trace}(M)^2$, where k is a constant, and $\det(\cdot)$ and $\text{trace}(\cdot)$ are the determinant and the trace of a matrix, respectively. The Harris detector finds the points where the corner response function R is greater than a threshold, and it takes the points with the local maxima of R as the corner feature points.

- Feature tracking: The Lucas-Kanade method [61] is applied to compute the optical flow between consecutive frames, $I(x, y, t)$ and $I(x, y, t + 1)$. We assume that u and v are the x and y components of the velocity of the corner feature (x, y). Thus, we have $I_x u + I_y v + I_t = 0$. This equation is computed over a 5-by-5 window around the pixel (x, y), thereby yielding the following overconstrained system:

$$A \begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{b}, \text{ where } A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix}, \mathbf{b} = \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}. \tag{3.2}$$

  The solution is $[u\ v]^T = (A^T A)^{-1} A^T \mathbf{b}$. The corresponding pixel in $I(x, y, t+1)$ of the corner (x, y) is then found using u and v.

- Direction computation: The corner feature point (x_1, y_1) in image $I(x, y, t)$ and its corresponding pixel (x'_1, y'_1) in the next frame $I(x, y, t+1)$ form a vector $\mathbf{v}_1 = (a_1, b_1)$, which indicates the motion of the feature point between the two frames. The rat's head motion direction V is calculated as the opposite direction of the average feature point motion, i.e., $V = -\sum_i^n \mathbf{v}_i / n$, where n denotes the number of feature points.

For the rat head orientation, the line between the rat cyborg and the current target is considered to be the reference direction. The rat's head orientation is defined as the angle θ between the camera's optical axis and the reference direction, as shown in Fig. 3.1. Assume

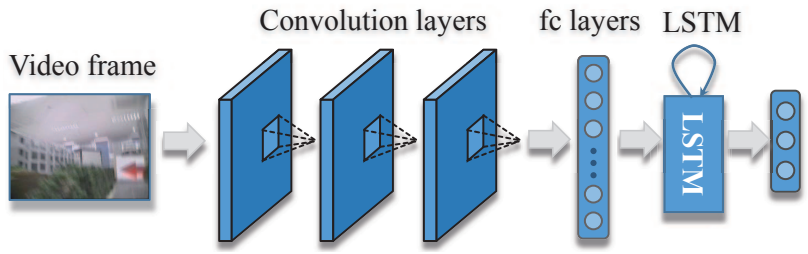


Figure 3.2: Our orientation network for rat head orientation analysis.

that the position of the target in the frame is d pixels from the center in the x direction. The rat's head orientation θ is computed as $\theta = \arctan(d/f)$, where f is the focal length. If the rat cyborg deviates from the reference direction by a large distance, the target will move outside the video frame. In this case, the distance d is estimated by the last target offset distance d_{old} and the rat's head motion, $d = d_{old} - V_x$, where V_x is the x component of V.

3.2 Rat State Analysis based on Deep Neural Networks

The rat head orientation in the first method is inferred based on the position of the detected object in each frame. Therefore the rat state estimation is vulnerable to the detection errors (e.g. false positive) and can be unreliable when the object is out of sight. We intend that this method is able to analyze the rat states based on the raw video frames, which does not rely on preprocessing steps such as object detection and feature tracking. Therefore we model the rat state analysis as a classification problem, where each video frame is assigned to a category corresponding to the head orientation and motion. An orientation network and a motion network are proposed for classifying rat head orientation and motion direction, which are described in this section.

3.2.1 Rat Head Orientation

The video captured by the rat-mounted camera contains rich set of information to indicate the camera orientation, such as road layouts and object arrangements. We consider 3 possible orientations in our model: left, middle, and right. Therefore the rat head orientation analysis is a 3-class image classification based on the video frames captured by the rat-mounted camera. Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) are incorporated for orientation analysis, as illustrated in Figure 3.2.

We employ a generic CNN in this model, which has been immensely successful in high-level vision tasks, such as image classification [48, 93], due to its powerful feature learning ability based on large-scale datasets. The CNN operates on a single video frame and aims to capture appearance features related to the rat head orientation. A small dataset is collected to train the CNN for orientation classification. We manually control the rat cyborg to

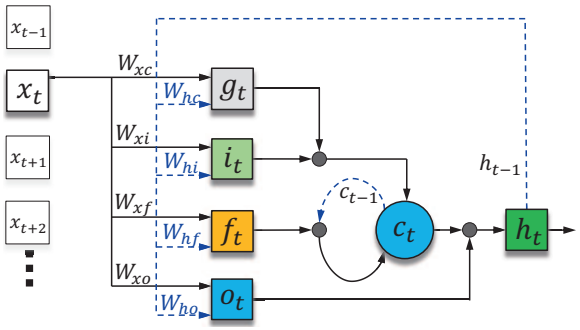


Figure 3.3: A diagram of LSTM.

navigate in an artificial maze to record the videos of the rat-mounted camera. The videos are labeled frame by frame by a person who examines the frames and decides the ground truth rat head orientation. It is very time consuming to label sufficient amount of data to train the CNN from scratch. Instead, we first pre-train the CNN on ImageNet [80] dataset, and then finetune the network on our dataset for rat head orientation classification. After the training process, the CNN is able to determine the rat head orientation based on the current video frame.

We believe that the information in the previous frames is useful for determining the current rat head orientation. This is based on the insight that the consecutive frames often have the same orientation label, due to the continuous motion of the rat head. Besides, in most cases the labels of the neighboring frames can only change either between "left and middle" or between "middle and right". In order to model the temporal dependencies of the consecutive frames, we incorporate LSTM in our orientation network.

Long short-term memory (LSTM) [32] is stable and effective for modeling long-range temporal information. Its innovation is the introduction of the "memory cell" c_t to accumulate the state information. The cell is accessed, written and cleared by several controlling gates, which enables LSTM to selectively forget its previous memory and learn long-term dynamics without the vanishing gradient problem of simple RNNs. Figure 3.3 illustrates a diagram of the LSTM model.

Assume that x_t is the input of LSTM at time t. The hidden state representation h_t is the modulated memory cell c_t by the output gate o_t:

$$h_t = o_t \odot \phi(c_t)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

where $\odot$ denotes the element-wise multiplication, ϕ stands for the tanh function, σ stands for the sigmoid function, and W_{xo}, W_{ho} are the weight matrices for the input and the previous hidden state respectively, and b_o is the bias term.

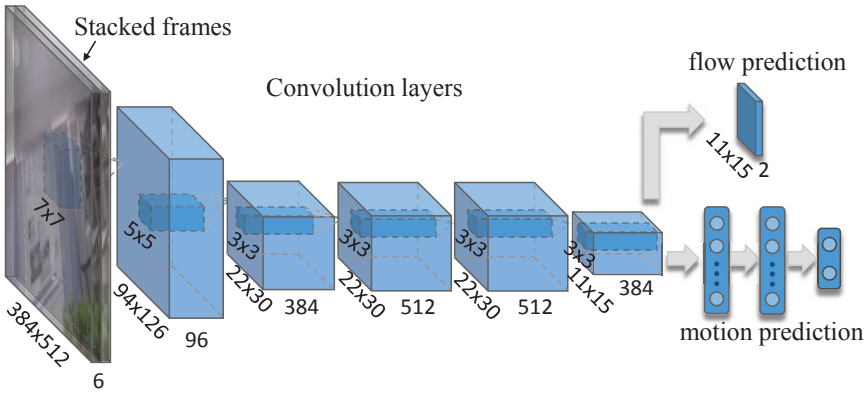


Figure 3.4: An illustration of the architecture of our motion network for analyzing the rat head motion direction.

The memory cell c_t is computed as a weighted sum of the previous memory cell c_{t-1} and the candidate new memory content g_t:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

where i_t, f_t are input and forget gates. The input gate i_t controls what information in g_t to be accumulated into the cell c_t. While the forget gate f_t helps the c_t to maintain and selectively forget information in previous cell memory c_{t-1}. The input and forget gates are computed by

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

and g_t is computed by the input modulation gate as:

$$g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c).$$

The overall structure of the memory cell and the regulating gates make LSTM suitable for modeling temporal relationships. The LSTM is placed after the first fully connected layer (fc layer) of the CNN and its output representation is used to predict the rat head orientation (Fig. 3.2).

3.2.2 Rat Head Motion Direction

We intend that our model is capable of analyzing the horizontal motion using the consecutive frames. This is because only the horizontal component of the rat head motion is of interest during our navigation control. We devise a generic CNN to take the stacked two consecutive frames with their original resolution as input. Its output indicates the two possible motion directions, i.e. left or right. An illustration of the architecture and the feature map resolutions of the CNN is shown in Figure 3.4.

In order to obtain the training data with ground truth horizontal motion labels, we employ a large dataset used for optical flow estimation and generate the left/right labels based on its ground truth flow vectors. The Flying Chairs dataset [18] is a synthetic dataset which contains 22872 image pairs with resolution 512×384. The ground truth for each image pair contains the optical flow vectors for every pixel in the first image. To generate the ground truth horizontal motion label, we sum the x component of the flow vectors of all pixels as:

$$sum_u = \sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij}, \tag{3.3}$$

where n and m are the image width and height, and u_{ij} is the x component of the ground truth flow vector (u_{ij}, v_{ij}) of the pixel at position (i, j). If sum_u is larger than 0, the overall motion of the image pixels is toward right. Therefore the camera is moving left and the ground truth motion label is set to be *left*. Otherwise, the label is set to be *right*.

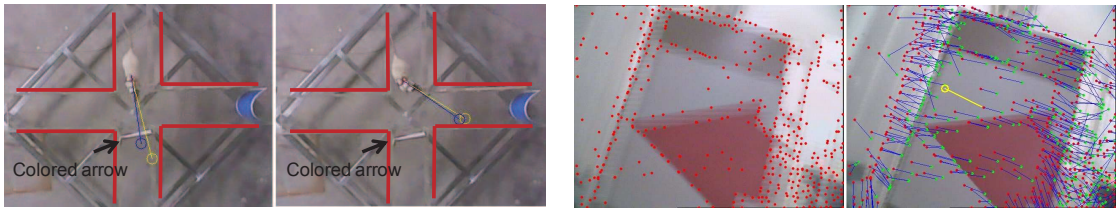
One naive way is to use the image pairs and the generated motion labels to train our motion network from scratch, which does not work in our experiments. We believe this is because the supervision of the left/right label is not explicit enough for the CNN to learn the motion related features from the image pairs.

Inspired by the successful applications of CNNs for optical flow estimation [18] and multi-task learning [72, 55], we propose a two stage multi-task training process for our motion network (see Figure 3.4). In the first training stage, we discard the fully-connected layers and use 3×3 convolution filters after the convolution layers to predict a 2-channel optical flow map. By minimizing the error between the predicted flow map and the resized ground truth flow map, the convolution layers of the CNN learn the motion related features. The standard error measure for optical flow estimation called the *endpoint error* (EPE) is used as the training loss, which is the average Euclidean distance between the predicted flow vectors and the ground truth for all pixels:

$$EPE = \frac{1}{N} \sum \sqrt{(u_i - u_{GTi})^2 + (v_i - v_{GTi})^2} \tag{3.4}$$

where N is the total number of pixels in the predicted flow map, (u_i, v_i) and (u_{GTi}, v_{GTi}) are the predicted flow vector and the ground truth flow vector for pixel i, respectively.

In the second training stage, we remove the flow prediction layer and add the fully connected layer after the convolution layers. In this stage, the kernel weights of the convolution layers are kept fixed. The fully connected layers are trained to predict the left/right motion label based on the convolution features that are related to motion. After the two stage training, our model is able to predict the motion direction from the stacked frames.



(a) (b)

Figure 3.5: (a) Estimation of the rat's head orientation (viewed from the top by a bird's eye camera). Left: Head orientation estimated using the rat-mounted video (blue arrow) and the top-mounted video (yellow arrow) when the sign was visible. Right: Estimated orientations when the sign was not visible. (b) Estimation of the rat's head motion direction (viewed from the rat-mounted camera). Left: Corner features (red dots) detected in the first frame. Right: Original feature location (red dots), the registered features (green dots) in the next frame, and the estimated rat's head motion direction (yellow arrow).

3.3 Experiments

3.3.1 Evaluation of the Optical Flow-Based Method

In this section, we present an assessment of the accuracy of the first rat state extraction method which is based on optical flow. The experiment is conducted using a four-armed maze (see Figure 3.5(a)). We design eight routes for rat state estimation. For each arm of the maze, the rat cyborg is initially placed at the end and a colored arrow is placed at a junction in the maze, where the rat cyborg is required to move from the starting point to the end of the adjacent arm indicated by the direction of the arrow, as shown in Figure 3.5(a). There are two possible directions for the arrow, so each arm had two possible routes. Thus, there are eight routes in total for the four arms. As the rat cyborg traverse the routes, we continuously estimate the rat's head motion direction V and the rat's head orientation θ from the videos recorded by the mounted camera (see Figure 3.5(a) and Figure 3.5(b)). We test each route four times, thus there are four trials and 32 routes.

To obtain the "ground truth" for V and θ, we use a bird's eye camera which is mounted above the scene, to record videos while the rat cyborg performs the tests. In these stable videos, we label the rat's head in the first frame and use the Lucas-Kanade method [61] to track the head and to compute the rat's head orientations θ. These results are compared with the results estimated from the rat-mounted camera and we compute the average differences and standard deviations, as shown in Table 3.1. For the rat's motion direction V, a similar method is used to obtain the rat head motion directions from the videos captured by the bird's eye camera. However, it should be noted that the motion direction is in a vector space and the results obtained from the bird's eye videos use different scales compared with those computed from the videos recorded by the rat-mounted camera. Thus, we perform

Table 3.1: Average difference and standard deviations between the rat's head orientations estimated from the rat-mounted camera and from the top-mounted camera, and the accuracy of the estimated rat's head motion direction.

		Route #1	Route #2	Route #3	Route #4	Route #5	Route #6	Route #7	Route #8
Trial #1	AD*(degree)	8.92	13.10	12.68	7.71	10.21	8.41	9.10	9.74
	SD*(degree)	6.60	6.09	6.90	5.11	6.69	6.34	6.17	6.67
	Accuracy (%)	88.37	84.67	82.49	93.33	91.21	88.89	89.30	92.31
Trial #2	AD (degree)	8.68	6.86	8.44	7.74	8.19	8.58	11.54	11.38
	SD (degree)	6.55	5.97	6.81	5.33	5.52	5.95	6.18	7.14
	Accuracy (%)	91.03	89.77	90.39	91.65	93.59	84.17	85.21	90.67
Trial #3	AD (degree)	10.55	9.67	11.04	10.07	8.44	11.93	10.87	11.54
	SD (degree)	7.27	6.87	7.27	7.02	6.81	6.43	6.59	6.99
	Accuracy (%)	94.19	89.84	89.96	91.58	93.26	85.16	87.24	92.13
Trial #4	AD (degree)	8.27	13.57	8.58	8.48	11.02	8.41	9.94	8.13
	SD (degree)	7.18	7.21	5.90	6.66	6.66	5.90	6.61	6.36
	Accuracy (%)	86.09	85.48	88.16	94.17	91.71	89.58	91.67	89.38

* AD denotes the average difference, SD denotes the standard deviation.

a qualitative comparison to determine whether the two motion estimates are in the same direction: "left" or "right."

Table 3.1 shows the average difference and standard deviations between the rat's head orientations estimated from the rat-mounted camera and those from the bird's eye camera, as well as the accuracy of the estimates of the rat's head motion direction. In all trials, the average differences are usually about 8 degrees. These differences generally have trivial effects in determining whether the rat's head is currently located left or right of its body. On average, approximately 90% of the rat's motion directions are estimated correctly. In our navigation experiments, this performance can satisfy the requirement of the automatic instruction model.

3.3.2 Evaluation of the Deep Neural Networks-Based Method

In this section, we evaluate the second rat state analysis method which is based on deep neural networks, and we compare it with our first optical flow-based state analysis method. In the previous section, the first method is evaluated by comparing with the reference rat states extracted using a top camera, which can be incorrect in some cases. In order to conduct a more reliable comparison, we use new datasets and design new experiments. We assess the performance of our orientation network on four videos with human labeled ground truth orientation, which are recorded by the rat-mounted camera during manual control. The rat head motion direction accuracy is then evaluated using four datasets with reliable motion labels, which are generated based on public available optical flow datasets.

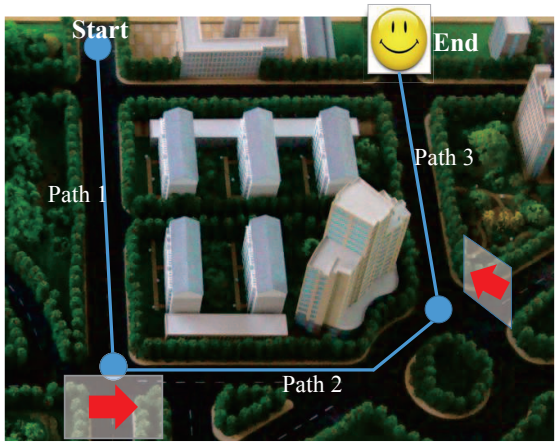


Figure 3.6: The U-shaped route on the urban planning model.

Table 3.2: The number of frames in the videos captured during each trial for evaluating our orientation network.

	Path 1	Path 2	Path 3	Total
Trial #1	239	282	88	609
Trial #2	371	581	160	1112
Trial #3	198	570	211	979
Trial #4	491	409	239	1136

Rat Head Orientation

We manually control the rat cyborg to navigate on an urban planning model to record videos to evaluate the rat head orientation analysis performance of our orientation network. We design a U-shaped route on the maze which consists of 3 pathes (see Figure 3.6). The objects at the end of path 1 and 3 provide cues for orientation estimation, which makes the estimation on these two pathes relatively simple. The path 2 is more challenging due to the limited visual cues in sight.

We perform the manual navigation 4 times and one video is recorded during each trial. The frame number of each video are listed in Table 3.2. We label each video frame with a ground truth rat head orientation label. We test our orientation network on each of the videos. When one video is considered as testing set, the rest three videos are used as training data. Our model is pre-trained on ImageNet dataset and fine-tuned on the training videos. In order to analyze how the LSTM in our model contributes to the accuracy, we conduct an ablation study by training only the generic CNN of our orientation network and testing its performance.

Table 3.3 shows the rat head orientation estimation accuracy of our orientation network, the generic CNN, as well as our first method, which is denoted as M1. We report the orientation classification accuracy on each of the 3 pathes and the overall accuracy on the

Table 3.3: The orientation estimation accuracy of our orientation network, the generic CNN, and our optical flow-based method, which is denoted as M1. The numbers are in percentage.

	Path 1			Path 2			Path 3			Overall		
Method	M1	CNN	Ours	M1	CNN	Ours	M1	CNN	Ours	M1	CNN	Ours
Trial #1	92.89	89.12	**94.56**	53.90	77.66	**87.23**	92.04	94.32	**97.73**	74.71	84.56	**91.62**
Trial #2	90.84	95.69	**97.57**	57.14	74.01	**82.62**	84.38	91.25	**93.75**	72.30	83.72	**89.21**
Trial #3	92.42	93.43	**98.48**	57.19	85.61	**88.77**	84.83	93.84	**96.21**	70.27	88.97	**92.34**
Trial #4	84.32	93.89	**96.33**	63.57	86.55	**91.69**	92.05	91.63	**97.49**	78.49	90.78	**94.91**

Figure 3.7: Sample video frames that our orientation network classifies correctly. The top frames contain object of interest while the bottom frames do not. The ground truth orientation label for each frame from left to right: left, middle, and right.

whole route. It can be found that our model consistently outperforms the generic CNN. This demonstrates that by incorporating the LSTM, our orientation network is able to capture the temporal information and obtain better orientation estimation.

Our first method is able to produce a continuous number θ which is the angle between the head orientation and the reference direction. In order to compare it in the 3-class classification task, we use human defined threshold to discretize the estimated rat head orientation. The first method uses the location of the object of interest as reference to infer the rat head orientation. When the object is out of the video frames due to the rat head motion, it uses feature tracking algorithm to estimate the object location. This method achieves good accuracy on path 1 and 3, where the object is often available in the video. However, the estimated object location can be unreliable when the object is unavailable for long time. This results in the significant performance drop of this method on the path 2. Our orientation network is able to outperform the first method, especially in the challenging scenario of path 2. Sample frames that are classified correctly by our orientation network are shown in Figure 3.7.

Table 3.4: Comparison of the motion estimation accuracy of our motion network with our first method based on optical flow. The numbers are in percentage.

	Our First Method	Our Motion Network
Flying Chairs	85.78	**90.47**
Sintel Clean	83.77	**85.30**
Sintel Final	79.44	**80.69**
KITTI	70.10	**74.74**

Rat Head Motion Direction

It is difficult to obtain accurate left/right motion label for the rat video frames, because the videos are usually blurry and shaky due to the unpredictable motions of the rat. In order to quantitatively evaluate our motion network, we incorporate 4 publicly available optical flow datasets and generate the motion labels by averaging the ground truth flow vectors as described in section 3.2.2. The datasets used in our experiments are briefly introduced as follows:

- **Flying Chairs dataset** [18] is a large synthetic dataset which is built to provide sufficient data for training CNNs for optical flow estimation. It contains 22232 training image pairs and 640 testing image pairs with dense per-pixel ground truth. The images have resolution 512×386 and are generated by rendering 3D chair models on background images from Flickr.

- **KITTI dataset** [25] contains 194 image pairs. An average of 50% pixels in the images have ground truth optical flow. The images are captured using cameras mounted on an autonomous car in real world scenes, and the image resolution is about 1240×376. This dataset contains strong projective transformations and special types of motions.

- **Sintel dataset** [12] contains computer rendered artificial scenes of a 3D movie with dense per-pixel ground truth. It includes large displacements and pays special attention to achieve realistic image properties. The dataset provides "Clean" and "Final" versions. The Final version includes atmospheric effects (e.g. fog), reflections, and motion blur, while the Clean version does not include these effects. Each version contains 1041 image pairs and the image resolution is 1024×436.

We train our motion network on the training set of the Flying Chairs dataset, and test it on the testing set of the Flying Chairs dataset, as well as on the Sintel Clean, Sintel Final, and KITTI datasets. We compare our model with the motion estimation method based on optical flow, and the results are shown in Table 3.4. Our motion network is able to produce more accurate motion direction estimation than the first method.

We also test our motion network on the videos captured by the rat-mounted camera and the qualitative results are shown in Figure 3.8. We stack the previous frame and the

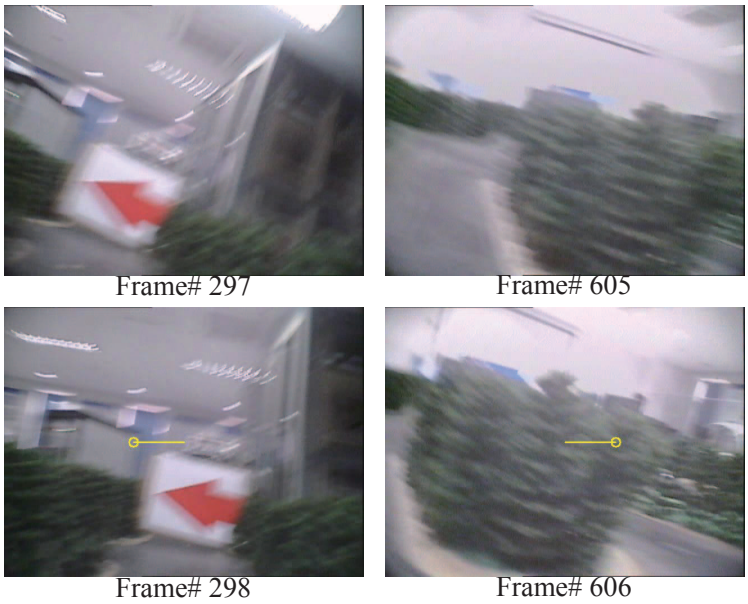


Figure 3.8: The estimated rat head motion direction from consecutive video frames, which is indicated by the yellow arrow drawn on the second frame.

current frame as the input of our model, which outputs the motion direction between these two consecutive frames. A yellow arrow is drawn on the current frame to indicate the rat head motion direction (see Figure 3.8).

Implementation Details and Time Efficiency

We implement our model using Caffe [40]. The generic CNN in our orientation network have the same architecture as the CNN used in [17]. It is a minor variant of the network proposed by Zeiler and Fergus [124], which contains 5 convolution layers and 3 fully-connected layers. We devise the input of this architecture to take stacked image pairs with resolution of 512×386 and use it as our motion network. The LSTM implementation in [17] is adopted in our orientation network. We use 8 timesteps and set the hidden vector dimension to be 1024 for the LSTM. Our orientation network is able to run at a frame rate of 35 fps and the motion network runs at 55 fps, which is sufficient for real-time applications. The running speeds are measured using an NVIDIA TITAN-X GPU on a desktop with an Intel i7-6850K CPU.

3.4 Summary

In this chapter, we present two methods which are based on optical flow and deep neural networks for analyzing the rat action states using the egocentric video captured by the rat-mounted camera. Our methods provide rat head orientation and motion direction for

the automatic control without the need of a bird's eye camera on top of the scene. This enables the rat robot to navigate in a wider range of scenes.

Chapter 4

Automatic Instruction Models for Rat Robot Control

In this chapter, we present our instruction models for rat robot automatic navigation. Our automatic instruction models are able to consider the following two visual cues: 1) the feedback of the rat robot states obtained from the egocentric videos, 2) the objects of interest that indicate the action expectations for the rat robot, e.g. road signs. This is inspired by the manual control process, where the human operators usually observe the rat states and the environmental layouts, and then give a series of stimuli/instructions to the rat robot. We propose two automatic instruction models. The first employs a human-like instruction model which learns to mimic the human controlling process. The second is a rule-based control model designed according to human experts' experience and knowledge.

4.1 Human-like Instruction Model

The human-like instruction model issues instructions based on rat states and the object of interest. This method attempts to learn from the human control data and operate in a similar manner to human when they encounter similar situations during the manual control.

4.1.1 Learning from Human Control Process

In the manual control process, after the instruction C_{i-1} is issued to the rat robot, its posture changes from the state S_{i-1} to the next state S_i. Next, humans observe the change in the rat state and determine the current instruction C_i to adjust the incorrect action or to reward the correct action. Thus, the state change is considered an important factor for deciding the current instruction in this model. In addition, the current object affects the selection of the instruction. Different objects are related to different motion expectations for the rat cyborg. Thus, different instructions will be sent even if the same state change is observed for different objects. Therefore, the current motion expectation is treated as another factor when deciding the current instruction.

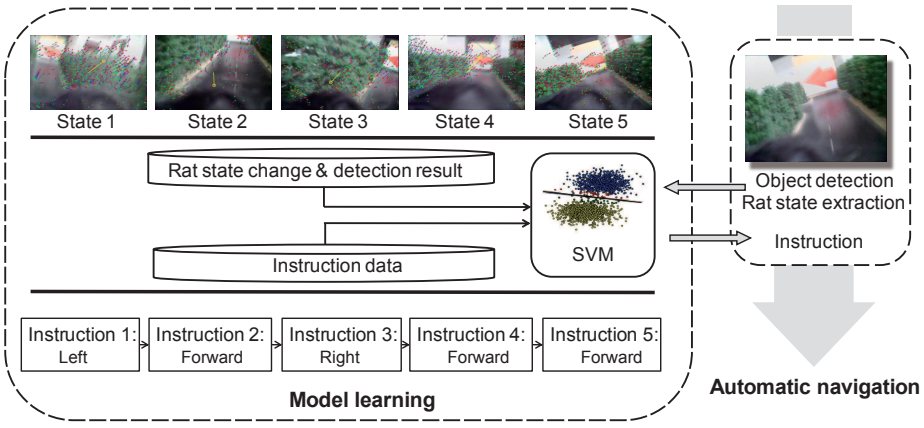


Figure 4.1: Human-like instruction model.

We assume that the state change extracted by our method is $\Delta S_i = \{\theta_i - \theta_{i-1}, V_i - V_{i-1}\}$, the current motion expectation E_i indicated by the object detection result is one of Forward (0), Left Turn (-1), and Right Turn (1), and the current instruction issued by humans is C_i. We use $\mathbf{X}_i = (\Delta S_i, E_i)$ as input features and C_i as output labels to train a support vector machine (SVM) [14] classifier to construct the human-like instruction model. Given a set of l training data points $(\mathbf{X}_i, C_i)$, $i = 1, 2, ..., l$, the SVM aims to learn a classifier for the input data $\mathbf{X}$ with the form:

$$C = \text{sign}(\mathbf{w}^T \mathbf{X} + b), \tag{4.1}$$

where $\mathbf{w}^T$ and b are parameters obtained by solving the following optimization problem:

$$\begin{aligned} &\min \frac{1}{2}||\mathbf{w}||^2 + \alpha \sum_{i=1}^{l} \xi_i, \\ &s.t. \quad C_i(\mathbf{w}^T \mathbf{X}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0, \end{aligned} \tag{4.2}$$

where ξ is a slack variable and the parameter α balances the weights between the two terms. Because there are three possible values for an instruction: C (Left (-1), Right (1), or Forward (0)), we build a three-class classifier as the human-like instruction model using a one-against-all scheme.

The training data are collected during the manually controlled navigation of the rat robot. Both the control instructions and the videos from the rat-mounted camera are recorded. The rat motion state S_i that corresponds to each instruction C_i is extracted from the videos by the method described in section 3.1 and used to compute the rat state change ΔS_i. The action expectation E_i is obtained based on the object detection results. In the testing stage, we compute the real-time state change ΔS and motion expectation E of the rat robot. The instruction for the rat robot is then obtained based on the classification results produced using our model. Figure 4.1 shows the training and testing framework employed by our human-like instruction model.

4.1.2 Detecting Objects of Interest with Soft-cascade and Color Model

In the automatic navigation, we utilize two common objects: colored objects and human faces to indicate the action expectations. The colored turning signs indicate the direction that the rat robot should turn and the human face acts as the destination. We develop object detection algorithms for these two types of objects.

In colored object detection, a specific color is treated as a random variable $\mathbf{c}$, which conforms to a single Gaussian distribution, $\mathbf{c} \sim N(\boldsymbol{\mu}, \Sigma)$, where $\mathbf{c} = (R, G, B)^T$ is the color vector, and $\boldsymbol{\mu}$ and Σ are the mean vector and the covariance matrix of the distribution respectively. The parameters for a specific color are estimated from a group of natural training images by:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{c}_j, \quad \Sigma = \frac{1}{n-1}\sum_{j=1}^{n}(\mathbf{c}_j - \boldsymbol{\mu})(\mathbf{c}_j - \boldsymbol{\mu})^T, \tag{4.3}$$

where n is the total number of color training samples $\mathbf{c}_j$. The probability of a pixel with color vector $\mathbf{x}$ belonging to the specific color can be computed as:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{1/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{4.4}$$

During detection, if $p(\mathbf{x})$ is greater than a threshold T_c, the pixel is considered to be this color. When the area of the bounding box of connected pixels in a frame exceeds a threshold T_a, the object is considered to be detected. In our experiments, T_c is set to 0.9 and T_a is set to 25×25 pixels, which obtains satisfactory performance.

For face detection, we develop a modified version of the fast face detection method called soft cascade [8], which employs real AdaBoost as the learning algorithm. The classifier used by soft cascade is,

$$H_T(x) = \sum_{i=1}^{T} h_i(x), \tag{4.5}$$

where x is a test sample and $h_i(x)$ denotes a weak classifier. Given a set of rejection thresholds $\{\gamma_1, \gamma_2, ..., \gamma_T\}$, x will be accepted as a face if and only if every partial sum $H_t(x) > \gamma_t$. This cascade structure makes the detector fast. The Haar feature is used in the detector and a stump method is used to train the weak classifiers [8]. The detector is trained on a face image set containing more than 20,000 face images and 100,000 non-face images, which measures 10×10 pixels, and they are collected from the Internet.

4.2 Rule-based Instruction Model

The human-like instruction model in previous section issues instructions based on the SVM classification result, which lacks a mechanism to determine whether it is appropriate to issue an instruction. One of the three instructions (left, right, or forward) is issued at a fixed

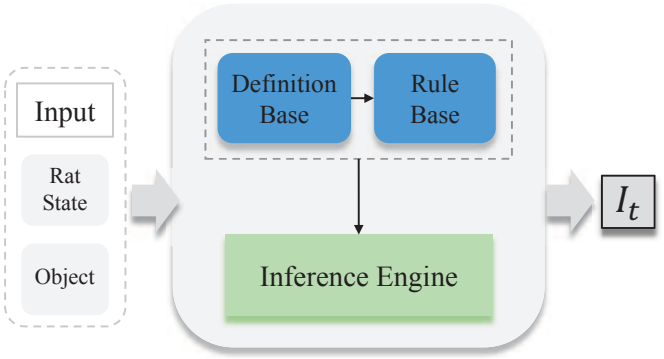


Figure 4.2: An illustration of the rule-based instruction model.

time interval even when it is unnecessary or inappropriate. However, the human operator sometimes choose not to give an instruction during the manual control, rather than keeping issuing one of the three instructions. This "no instruction" operation is nontrivial to be learned by the classifier, because it is difficult to record the corresponding training data during the manual control.

We propose a rule-based instruction model which is able to decide whether to issue an instruction and what instruction to issue. This model is constructed according to human experts' experience and knowledge, in which a thorough set of rules of issuing instructions are defined for all possible scenarios during the automatic navigation. We explicitly define rules to specify the scenarios where no instructions will be given, which better mimics the manual control strategy. Figure 4.2 is an illustration of our rule-based instruction model. The input of our model is the visual information, including the rat states and the objects that indicate the specified path. The rule-based model consists of a definition base, a rule base and an inference engine, which are described in this section.

4.2.1 Definition Base

The definition base provides the definitions and terminologies, which are used to define the control rules and describe the input visual information. The terms/variables used in our model are listed as follows:

1. $\mathbf{E_t} \in \{l, f, r\}$: the current motion expectation of the rat cyborg. Its possible values are l, f, r, which represent left, forward, right and take numerical values of 0, 1, 2, respectively. For example, suppose that the rat is on a straight path and there is a left arrow placed at the intersection. The rat is expected to first walk forward towards the arrow and then turn to the left path following the arrow. The motion expectation is f at first, then it switches to l when the rat reaches the intersection and gets close to the arrow. Finally E_t becomes f after the rat successfully turns into the left path.

2. $\mathbf{O_{bj}} \in \{face, arrow_l, arrow_r, null\}$: the detected object. We place left/right arrows and human face in the scene to indicate the motion expectation of the rat cyborg, which is expected to follow the arrows and reach the human face. We use the faster R-CNN [72] detector in our experiments. $O_{bj} = null$ means object of interest is not detected in the current frame.

3. $\mathbf{B_{sz}} \in N_0^*$: the bounding box size of the detected object, which indicates the distance between the object and the rat cyborg. If B_{sz} is larger than a threshold θ_1, the rat is considered to be close to the object.

4. $\mathbf{O_t} \in \{l, f, r\}$: the rat head orientation of time t. The orientation change of the rat head can be represented by $\hat{\mathbf{O}}_\mathbf{t} = O_t - O_{t-1}$. For example if $\hat{O}_t < 0$, the rat head orientation has changed towards left.

5. $\mathbf{M_t} \in \{l, r\}$: the rat head motion direction of time t.

6. $\mathbf{I_t} \in \{l, f, r, n\}$: the instruction of time t. $I_t = n$ means no instruction at this time.

7. $\mathbf{A} \in \{T, F\}$: a boolean variable indicating if it is appropriate to issue a turning instruction. During the manual control, experienced human operators usually avoid giving too many consecutive turning instructions. In out model, A is set to be False if there has been 2 previous turning instructions.

8. $\mathbf{D} \in N_0^*$: the duration since the last instruction was issued, which is used to ensure that the separation between two instructions are larger than a threshold θ_2, (e.g. 800ms).

4.2.2 Rule Base and Inference

The rule base contains a set of control rules which are specially designed according to the experience and control strategy of the human experts, as listed in Table 4.1. With the explicitly defined rules with $I_t = n$, this model is able to decide when there is no instruction to issue. The core of the rule base are the rules in part 2 and part 3 of Table 4.1. In part 2, the detected object and the distance of the object are used to derive the current motion expectation of the rat cyborg. When the rat cyborg gets close enough to the human face, we consider the navigation task to be successfully completed (see Rule #6). In part 3, the instruction for the rat cyborg is decided based on the motion expectation, the rat states, and previous instruction information.

We utilize a simplified forward chaining algorithm [30] in our inference engine, which starts with the observed data and reasons towards the answer by repeated application of modus ponens. Our model accumulates the duration D from the last instruction, and after it is larger then a threshold, our model initializes the variables and starts to infer the instruction to be sent. After the inference engine receives data O_{bj} and B_{sz}, it triggers one of the rules in part 2 and derives E_t. Once E_t is ready, the rule with its condition satisfied

Table 4.1: The Rule Base.

Part 1: Variable initialization.
1. if $D < \theta_2$, then no operation.
2. if $D \geq \theta_2$, then $I_t = f, A = T$.
3. if $I_{t-2} = l, I_{t-1} = l$, then $A = F$.
4. if $I_{t-2} = r, I_{t-1} = r$, then $A = F$.
Part 2: Motion expectation derivation.
5. if $O_{bj} = face, B_{sz} < \theta_1$, then $E_t = f$.
6. if $O_{bj} = face, B_{sz} \geq \theta_1$, then $Success$.
7. if $O_{bj} = arrow_l, B_{sz} < \theta_1$, then $E_t = f$.
8. if $O_{bj} = arrow_l, B_{sz} \geq \theta_1$, then $E_t = l$.
9. if $O_{bj} = arrow_r, B_{sz} < \theta_1$, then $E_t = f$.
10. if $O_{bj} = arrow_r, B_{sz} \geq \theta_1$, then $E_t = r$.
11. if $O_{bj} = null$, then $E_t = E_{t-1}$
Part 3: Determining the instruction.
12. if $E_t = f, O_t = f$, then $I_t = f$.
13. if $E_t = f, O_t = r, M_t = r, A = T$, then $I_t = l$.
14. if $E_t = f, O_t = r, M_t = r, A = F$, then $I_t = n$.
15. if $E_t = f, O_t = r, M_t = l$, then $I_t = n$.
16. if $E_t = f, O_t = l, M_t = l, A = T$, then $I_t = r$.
17. if $E_t = f, O_t = l, M_t = l, A = F$, then $I_t = n$.
18. if $E_t = f, O_t = l, M_t = r$, then $I_t = n$.
19. if $E_t = l, O_t = l$, then $I_t = f$.
20. if $E_t = l, O_t = f, I_{t-1} = l, \hat{O}_t < 0$, then $I_t = f$.
21. if $E_t = l, O_t = f, I_{t-1} = l, \hat{O}_t \geq 0, A = T$, then $I_t = l$.
22. if $E_t = l, O_t = f, I_{t-1} = l, \hat{O}_t \geq 0, A = F$, then $I_t = n$.
23. if $E_t = l, O_t = f, I_{t-1} = f$, then $I_t = l$.
24. if $E_t = l, O_t = r, A = T$, then $I_t = l$.
25. if $E_t = l, O_t = r, A = F$, then $I_t = n$.
26. if $E_t = r, O_t = r$, then $I_t = f$.
27. if $E_t = r, O_t = f, I_{t-1} = r, \hat{O}_t > 0$, then $I_t = f$.
28. if $E_t = r, O_t = f, I_{t-1} = r, \hat{O}_t \leq 0, A = T$, then $I_t = r$.
29. if $E_t = r, O_t = f, I_{t-1} = r, \hat{O}_t \leq 0, A = F$, then $I_t = n$.
30. if $E_t = r, O_t = f, I_{t-1} = f$, then $I_t = r$.
31. if $E_t = r, O_t = l, A = T$, then $I_t = r$.
32. if $E_t = r, O_t = l, A = F$, then $I_t = n$.
Part 4: Resetting variables.
33. if $I_t \neq n$, then $D = 0$.

in the part 3 will be triggered, which results in the selected instruction for the rat cyborg. If an instruction is issued, our model will reset D.

Table 4.2: Average confusion matrix for the off-line instruction classifications obtained using our human-like instruction model.

%	Forward	Left	Right
Forward	**93.83**	2.95	3.22
Left	11.81	**88.19**	0
Right	12.37	0	**87.63**

4.3 Experiments

4.3.1 Evaluation of the Human-like Instruction Model

In order to evaluate the human-like instruction model, we first conduct an off-line instruction classification test for this model. We manually controlled the rat cyborg to collect data for training and testing the human-like instruction model. Next, we design several turning tasks to test whether this model can automatically direct the rat cyborg to perform navigation.

To train the human-like instruction model and to verify its performance in an off-line test, we perform 60 trials in the four-armed maze (Figure 3.5(a)) and the urban planning model (Figure 3.6). In these trials, we place the pictures of colored arrows and human faces in different positions. The rat cyborg is manually controlled to walk toward the arrow pictures, to turn in the directions indicated by the arrows, and it finally reached the human face target. During navigation, we collect the instructions issued by humans and record the videos from the rat-mounted camera. The objects in the videos are detected using the object detection methods and the rat's states are extracted. For each manually issued instruction, we determine the synchronous rat state changes and the objects detected to form a dataset. There are 1171 instructions in 60 trials, thus the dataset contained 1171 samples. We select 574 random samples as the training data to learn the human-like instruction model and use the remaining 597 samples as testing data. The off-line test results are shown in Table 4.2. There are three types of instruction: "Forward," "Left," and "Right." The average confusion matrix shows that the accuracies of classification for the three instructions are 93.83%, 88.19%, and 87.63%, respectively.

To verify whether the human-like instruction model can automatically direct the rat cyborg to perform a specified motion, we design four simple routes: a single left/right turn, and a left/right turn followed by a right/left turn. The pictures of arrows are placed at intersections to indicate the turning directions, and a human face picture is placed at the end of the route to represent the destination. We use two rat cyborgs and test the first two routes 50 times and the other two routes 20 times for each rat cyborg. If the rat cyborg follows the arrow and reaches the destination, we consider it as a successful trial. We also record the time costs to compute the speed during each trial to evaluate the efficiency of our model. Moreover, the same experiments are conducted using manual control for the purposes of comparison.

Table 4.3: Comparison of the success rates obtained by our human-like instruction model and manual control in four turning tasks.

		Ours	Ours	Manual	Manual
		Success/Total	Average speed (m/min)	Success/Total	Average speed (m/min)
Rat Cyborg No.1	Left Turn	42/50	2.88	47/50	2.84
	Right Turn	44/50	2.60	48/50	2.63
	Left→Right	15/20	2.83	17/20	2.89
	Right→Left	17/20	2.33	19/20	2.38
Rat Cyborg No.2	Left Turn	45/50	3.78	49/50	3.85
	Right Turn	44/50	3.42	43/50	4.10
	Left→Right	16/20	3.26	18/20	3.74
	Right→Left	15/20	3.34	18/20	3.34

Table 4.3 compares the success rates for achieving the specified motions using our automatic instruction method and manual control in the four turning tasks. In all cases, the success rates of our model are close to those of manual control. The speed of motion completion is also similar for the two methods. This indicates that the human-like instruction model can automatically control the rat cyborg to perform specified motions.

4.3.2 Evaluation of the Rule-based Instruction Model

To evaluate the rule-based instruction model, we use the same turning tasks for the rat cyborg to perform automatic navigation as in the previous section, i.e., a single left/right turn and a left/right turn followed by a right/left turn. We use two rat cyborgs and test the first two routes 50 times and the other two routes 20 times for each rat cyborg, and we record the success rates for each task. We also record the number of instructions during each trial, this is because we believe that fewer number of instructions usually indicates more fluent and efficient control process. In order to facilitate a direct comparison, we conduct the same experiments using manual control and the human-like instruction model using these two rat cyborgs.

Table 4.4 shows the success rates and the average instruction numbers of the human-like instruction model, the rule-based instruction model, and manual control for these turning tasks. In most cases, our rule-based instruction model is able to achieve higher success rate than the human-like instruction model. It also requires fewer number of instructions to complete these tasks. This indicates that our rule-based model can provide more effective and fluent control. It can be found that the success rates of our model are even comparable to those of the manual control.

Finally, we perform some relatively complex automatic navigation missions on a small urban planning model, using both the rule-based model and the human-like model. We place pictures of colored arrows and human face in the scene to specify the routes for the rat cyborg, which mimics the search-and-rescue application. The experiments show that both of our instruction models can steer the rat cyborg to complete the automatic navigation

Table 4.4: Comparison of the success rates and the average instruction numbers of the human-like instruction model, the rule-based instruction model, and manual control based on four turning tasks.

		Human-like Model		Rule-based Model		Manual Control	
		Success /Total	Average Instr. #	Success /Total	Average Instr. #	Success /Total	Average Instr. #
Rat Cyborg No.3	Left Turn	44/50	42.3	**47/50**	**39.7**	48/50	26.2
	Right Turn	49/50	27.8	49/50	**27.4**	50/50	16.4
	Left→Right	18/20	40.8	**20/20**	47.5	20/20	35.8
	Right→Left	15/20	42.4	**17/20**	**41.1**	18/20	30.6
Rat Cyborg No.4	Left Turn	45/50	31.0	**47/50**	31.6	48/50	30.9
	Right Turn	47/50	29.8	**48/50**	**24.1**	48/50	12.7
	Left→Right	17/20	59.5	17/20	**39.5**	18/20	40.2
	Right→Left	16/20	72.9	**19/20**	**53.5**	18/20	55.1

tasks successfully by following the path indicated by the arrows and reaching the target human face, more details are in Section 5.5.

4.4 Summary

In this chapter, we propose two instruction models that issue a stimulus sequence automatically according to the states of the rat robot as well as the objects of interest. The first model learns to mimic the human controlling process by employing a human-like instruction model. The second model uses a rule-based control model designed according to human experts' knowledge. Our models are able to provide effective automatic control for the rat robot navigation.

Chapter 5

Rat Robot Automatic Navigation System

We build our rat robot system and apply the egocentric action analysis methods and control models to enable it to perform automatic navigation. In this chapter, we introduce our rat robot system, which we refer to as a *rat cyborg*. We first give an overview of the components of our rat cyborg. Then we introduce the hardware modules in our rat cyborg system. We also describe the principle about the electrical stimuli and the induced rat behavior. Then we introduce the training procedure that is required to establish and reinforce the correspondence between the stimuli and the rat behavior. Finally we give an automatic navigation example of our rat cyborg.

5.1 Overview

Figure 5.1 shows the three main components of our rat cyborg, i.e., implanted electrodes, a rat-mounted pack, and a computational component.

The electrodes are implanted in specific regions of the rat's brain and electrical stimuli can be delivered to the rat's brain via the implanted electrodes. The stimuli induce the corresponding rat behaviors of turning left, turning right, or moving forward.

The rat-mounted pack consists of the following components:

- A stimulator that generates electrical stimuli to be delivered to the rat's brain via the electrodes implanted in the brain.
- A miniature camera that captures real-time video of the scene in front of the rat. The miniature camera measures 20 mm×8 mm×1 mm and its optical axis is in the same direction as the rat's head.
- A wireless module that receives stimulus instructions from a PC and sends videos from the miniature camera to the computational component on the PC for action analysis. Thus, it includes an instruction receiver and a video transmitter.

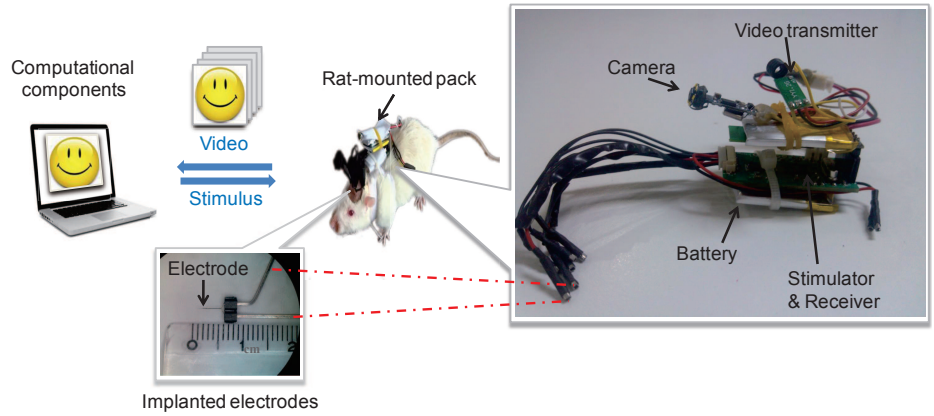


Figure 5.1: Three main components of our rat cyborg system. The electrode picture is taken under a microscope. The rat-mounted pack includes a miniature camera, a wireless module, and a stimulator.

The computational component comprises the egocentric action analysis methods and instruction models that have been described in previous chapters. The action analysis method extracts the rat states in the video data transferred from the rat-mounted pack. Based on the results, the instruction model determines the stimulus sequence delivered to the rat-mounted pack to stimulate the rat to take the correct actions.

5.2 Hardware Modules

In this section, we present the details of the stimulator in the rat-mounted pack.

The stimulator circuit generates stimulation pulses. The size of the circuit is minimized by using surface-mounted devices. As shown in Fig. 5.2, the main processor in the stimulator is a Mixed-Signal ISP FLASH MCU (C8051F020), which is characterized by its high speed, small size, and low power consumption. These features make it suitable for use in the small rat-mounted pack. This processor has two 12-bit Digital to Analog Converters (DACs), which produce outputs for jitter-free waveform generation.

The electrical stimulation pulses exported from the two DACs of the C8051F020 MCU are used to control a constant voltage driver circuit and a constant current driver circuit, thereby producing a monopolar pulse. The pulses of the constant voltage/current pass through three analog switches and are then delivered to the implanted electrodes.

The first analog switch is used to select between the input pulses, where the output is a monopolar pulse with either a constant voltage or a constant current. The second analog switch converts the monopolar input into a bipolar output. It reverses each half of the positive pulse into a negative pulse, and the resulting bipolar pulse has the same duration and amplitude in the positive and negative phase. The third analog switch acts as a selector to choose among the four output channels, which are connected to the implanted electrodes. Using the three analog switch circuits, the stimulator operates as a pulse generator, which

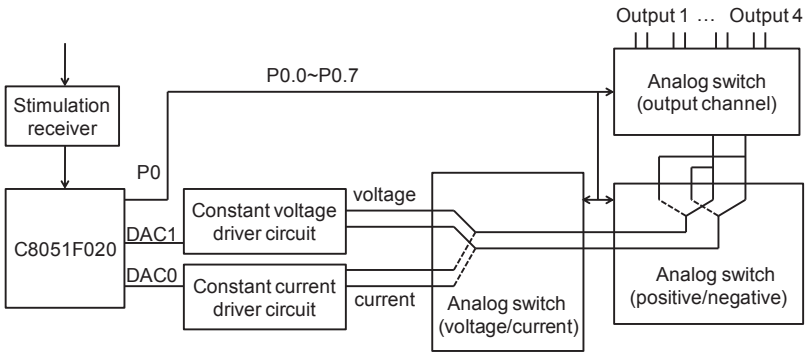


Figure 5.2: Circuit schematic diagram of the stimulator. The stimulator obtains inputs from the instruction receiver and sends outputs to the implanted electrodes. The system comprises a C8051F020 MCU as the main processor, constant voltage/current drive circuits, and analog switch circuits.

outputs a voltage or current pulse. The amplitudes of the output pulses are variable. Thus, the stimulator can produce signals with various waveforms to satisfy different requirements.

5.3 Stimulation-action Principles

In this section, we describe the principles about the stimuli and the induced rat behavior. Electrical stimuli can be delivered to specific brain regions as rewards [75, 31] and as steering cues [79] to control rat behavior. The MFB in the rat's brain is known as a pleasure center, thus the application of electrical stimuli to the MFB can be used as rewards [31]. Applying a stimulus to the MFB will increase the level of dopamine (a neurotransmitter with an important role in reward-motivated behavior) in the rat's brain [82, 31], thereby motivating its motion and reinforcing its behavior [95]. The application of stimulation to the SI can be used as a steering cue [108]. Rats use their vibrissae (whiskers) to sense object surfaces while exploring the environment. The whisker barrel fields in the SI receive projections from the contralateral facial vibrissae. A stimulus on one side of the SI is represented as a virtual touch on the contralateral vibrissae, which makes the rat perform a turn [95]. Thus, three pairs of electrodes are implanted in the rat's brain. One of the electrodes is placed in the MFB and the other two are implanted symmetrically in the whisker barrel field of the left and right SIs.

5.4 Rat Preparation and Training

Surgery is performed on rats to implant electrodes in their brains. Adult Sprague-Dawley rats are used in our experiment. During surgery, the rats are anesthetized with chloral hydrate and placed on a stereotaxic apparatus. One millimeter holes are drilled in the skull in order to insert three electrodes in the brain. One of the electrodes is placed in the MFB

(AP -3.8, ML +1.6, DV +8.2) [31]. The other two electrodes are implanted symmetrically in the whisker barrel field of the left and right SIs (AP -1.8, ML ±5.0, DV +2.8) [68]. Dental acrylic is used to fix the electrodes to the skull. After surgery, each rat is allowed at least 7 days of post-operative recovery.

Before the rat robot can perform navigation tasks, a training procedure is required to establish and reinforce the correspondence between the stimuli and the behavior of the rat. We employ the rat behavior training method described in [23], which consists of the MFB reward training and SI steering training methods.

In MFB reward training, the rat is first trained to press a bar to obtain the MFB stimulus reward, until it presses the bar continuously to obtain the MFB stimuli after it has been placed before the bar. Next, the rat is placed on a narrow runway to train the continuous movement behavior by delivering MFB stimuli continuously. After training is complete, if an MFB reward is sent to the rat, it will move forward. In SI steering training, the rat is trained to make correct turns in an eight-armed maze. SI stimulations are delivered to drive the rat so that it performs turns. After each correct turn, an MFB reward is given immediately to reinforce the correct behavior.

After training, the basic rat robot can perform approximate turning left, turning right, or moving forward actions given the corresponding stimulus. Thus, humans can control a rat to navigate in an environment by observing its states and sending suitable stimulus instructions.

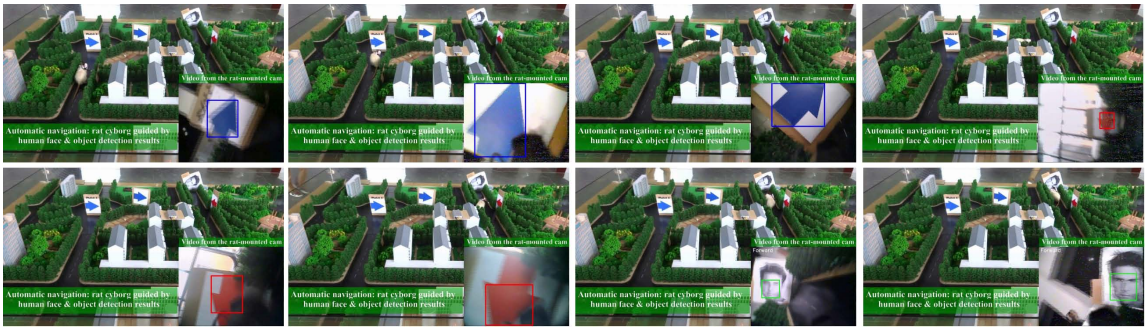
5.5 Automatic Cue-guided Navigation

We design two automatic navigation tasks for our rat cyborg. In the first task, the rat cyborg is expected to walk towards a human. When it reaches the human, the human moves to the next position for the rat cyborg to search. In the second task, we place colored pictures of arrows with left/right directions and a human face picture in a small urban planning model. Figure 5.3 shows the urban planning model, which contains buildings, trees, roads, and junctions. The colored arrows indicate the next route that the rat cyborg should take at junctions and the face picture is the target. The positions of the objects are changed to specify the different routes/missions. The rat cyborg is expected to follow the signs and reach the target human face. These tasks mimic the search-and-rescue applications, such as during earthquake remedy. Our experiments show that our rat cyborg successfully completes the cue-guided navigation tasks (Fig. 5.3).

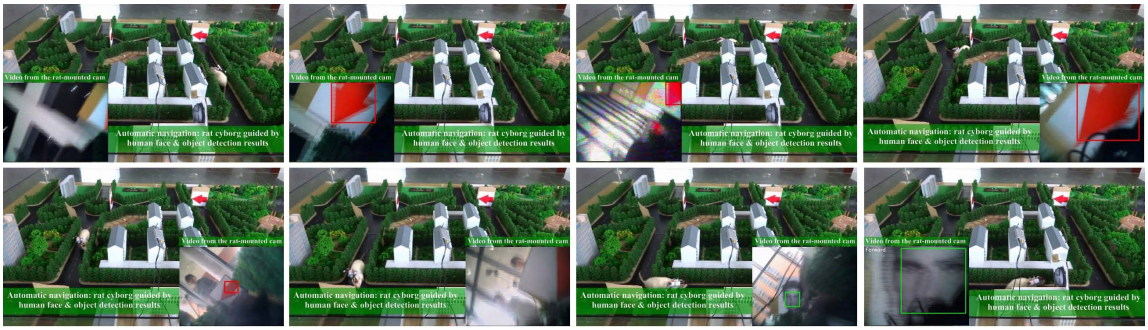
5.6 Summary

In this chapter, we present an overview of our rat cyborg and introduce the components in the system. We also describe the principle and the training procedure that allows the

(a)



(b)



(c)

Figure 5.3: Automatic navigation examples. (a) Task 1: the rat cyborg walks towards a human. (b)(c) Task 2: the rat cyborg follows the signs to reach the target human face picture in the urban planning model.

rat cyborg to be controlled to take specified actions. Two examples of visual-cue guided automatic navigation performed by our rat cyborg are also presented.

Chapter 6

Deep Recurrent Network for Optical Flow Estimation

Optical flow encodes the motion between the consecutive frames and is used in one of our methods to analyze the rat action states, as described in Chapter 3. The rat head motion direction is directly computed by the average optical flow direction, and the object offset used for rat head orientation is also inferred based on optical flow estimation. We believe that a more accurate optical flow model can result in a better rat states estimation. Therefore, we explore the research of using deep neural networks for dense per-pixel optical flow estimation. We propose a row convolutional long short-term memory (RC-LSTM) network to model contextual dependencies of local image features. This recurrent network can be integrated with CNNs to learn context-aware features for more accurate optical flow estimation.

6.1 Introduction

Convolutional Neural Networks (CNNs) [51] have brought a revolution in computer vision community with its powerful feature learning capability based on large-scale datasets. They have been immensely successful in high-level computer vision tasks, such as image classification [48, 93] and object detection [26, 65]. CNNs are good at extracting abstract image features by using convolution and pooling layers to progressively shrink the feature maps, which produces translation invariant local features and allows the aggregation of information over large areas of the input images.

Recently, researchers have been attempting to employ CNNs to tackle pixel-level prediction tasks, such as semantic segmentation [60, 125] and optical flow estimation [18, 96]. These tasks differ from the previous high-level tasks in that they not only require precise single pixel prediction, but also require semantically meaningful and contextually consistent predictions among a set of pixels within objects. Optical flow estimation is even more diffi-

cult because it requires finding the x-y flow field between a pair of images, which involves a very large continuous labeling space.

There are significant challenges in adapting CNNs to handle dense per-pixel optical flow estimation. First, CNNs do not have a mechanism to explicitly model contextual dependencies among image pixels. Although the local features learned with CNNs play an important role in classifying individual pixels, it is similarly important to consider factors such as appearance and spatial consistency while assigning labels in order to obtain precise and consistent results. Besides, the convolution and pooling operations result in reduced feature maps, and hence produce coarse outputs when upsampled to the original resolution to produce pixel-level labels. These two aspects render CNNs limited ability to delineate object details, and can result in blob-like shapes, non-sharp borders and inconsistent labeling within objects.

In this method, Recurrent Neural Networks (RNNs) are incorporated to alleviate this problem. RNNs have achieved great success in modeling temporal dependencies for sequential data, and have been widely used in natural language processing [27], image captioning [41], etc. Long short-term memory (LSTM) [32] is a special RNN structure that is stable and powerful for modeling long-range dependencies without suffering from the vanishing gradient problem of vanilla RNN models [66]. We propose a row convolutional LSTM (RC-LSTM), which has convolution operators in both the input-to-state and state-to-state transitions to handle structure inputs. We treat an image as a sequence of rows and use our RC-LSTMs to explicitly model the spatial dependencies among the rows of pixels, which encodes the neighborhood contexture into local image representation.

The proposed RC-LSTM structure can be integrated with CNNs to enhance the learned feature representations and produce context-aware features. In our experiments, we integrate our RC-LSTM with FlowNet [18], the first successful CNN-based model for optical flow estimation, to form an end-to-end trainable network. We test the integrated network on several datasets, the experimental results demonstrate that our RC-LSTM structure can enhance the CNN features and produce more accurate and consistent optical flow maps.

6.2 Related Work

6.2.1 Optical Flow Estimation

Optical flow estimation has been one of the key problems in computer vision. Starting from the original approaches of Horn and Schunck [34] as well as Lucas and Kanade [61], many improvements have been introduced to deal with the shortcomings of previous models. Most of those methods model the optical flow problem as an energy minimization framework, and are usually carried out in a coarse-to-fine scheme [10, 109, 92]. Due to the complexity of the energy minimization, such methods have the problem of local minima and may not be able to estimate large displacements accurately. The methods in [11, 110] integrate descriptor

matching into a variational approach to deal with large displacements problem. The method in [74] emphasizes on sparse matching and uses edge-preserving interpolation to obtain dense flow fields, which achieves significant flow estimation performance.

6.2.2 CNNs for Pixel-level Prediction

CNNs require fixed-size inputs, and the fully connected layers transform the feature maps into vector representations which are difficult to reconstruct for 2D predictions. Therefore, it is not straightforward in adapting CNNs to tackle the tasks of pixel-level predictions. Fully Convolutional Networks (FCN) is proposed in [60] which can take input of arbitrary size and produce output in the same size. The key insight is to transform the fully connected layers into convolution layers which produce coarse predictions. Then deconvolution layers are incorporated to iteratively refine/upsample the prediction to the original size. A similar scheme is utilized in FlowNet [18] to predict an optical flow field with convolution and deconvolution layers. The difference is that not only the coarse predictions, but the whole coarse feature maps are "de-convolved", allowing the transfer of more information to the final prediction.

The central issue in the methodology of [60, 18] is that the convolution and pooling operations result in reduced feature maps and hence produce coarse outputs when upsampled to the original resolution. Besides, CNNs lack smoothness constraints to model label consistency between pixels. To solve this problem, Zheng et al. [125] combines CNNs with a Conditional Random Fields (CRFs)-based probabilistic graphical model. Mean-field approximate inference for the CRFs is formulated as Recurrent Neural Networks, which enables training CRFs end-to-end together with CNNs. This CRF-RNN network is integrated with FCN [60] and achieves more precise segmentation results. However, the inference of mean-field approximation in [125] is for discrete labeling, making it not applicable to refine optical flow maps which is a continuous labeling task with very large label space.

6.2.3 RNNs for Structural Modeling

Recurrent Neural Networks (RNNs) have been extended to model spatial and contextual dependencies among image pixels [56, 99, 100]. The key idea is to define different connection structures among pixels within an image and build spatial sequences of pixels. Multi-dimensional RNNs are proposed in [28] and are applied to handwriting recognition. 2D-RNNs [85], tree-structured RNNs [94], and directed acyclic graph RNNs [86] are proposed to model different connections between image pixels for different tasks. Applying RNNs to specifically defined graph structures on images are different with the idea of CRF-RNN [125]. Instead of implicitly encoding neighborhood information with a pairwise term in an energy minimization framework, the RNNs enables explicit information propagation via the connections.

Our approach intergrates ideas from these methods. We propose to model the contextual dependencies among each row of pixels with a row convolutional LSTM (RC-LSTM). Similar as [84], our RC-LSTM utilize convolution operators in both the input-to-state and state-to-state transitions instead of the full matrix multiplication in LSTM to handle structure inputs. The feature vectors of the pixels in each row form one input to the RC-LSTM, and the rows are processed in sequence. The RC-LSTM enables the information propagation/message passing among the rows of pixels, which encodes the contextual dependencies into the local feature representations. This RC-LSTM structure is integrated with convolution and deconvolution layers, giving rise to an end-to-end trainable network. To the best of our knowledge, our work is the first attempt to integrate CNNs with RNNs for optical flow estimation.

6.3 The Proposed Approach

To predict dense pixel-level optical flow from a pair of images, the images are processed by three types of network components: convolution layers, deconvolution layers, and our RC-LSTM network. Functionally, the convolution layers transform raw image pixels to compact and discriminative representations. The deconvolution layers then upsample the feature maps to the desired output resolution. Based on them, the proposed RC-LSTM models the contextual dependencies of local features and produce context-aware representations, which are used to predict the final optical flow map.

In this section, we will introduce our RC-LSTM for modeling contextual dependencies among image rows. After that, we will explain how our RC-LSTM model can be integrated with CNNs to form an end-to-end trainable network.

6.3.1 RC-LSTM for Modeling Spatial Dependencies

Long short-term memory (LSTM) [32] has been introduced in section 3.2, which is designed to model temporal dependency for a data sequence. To applying LSTMs to one image, a spatial order of the image pixels needs to be defined. The simplest way is to consider each pixel as an individual input data, and the image is reshaped to a sequence of pixels to feed into the LSTM model. However, the interactions among pixels are beyond this chain-structured sequence. This simple way will loss the spatial relationships, because adjacent pixels may not be neighbors in this sequence.

In order to model the spatial relationships among pixels, we consider each row of the image as one input data, and consider the image as a sequence of rows. This results in structure inputs which LSTM has difficulty to handle. We propose a row convolutional long short-term memory (RC-LSTM) to cope with the structure inputs and learn the contextual dependencies among the rows of pixels. Our RC-LSTM is illustrated in Figure 6.1. The feature vectors of the pixels in the rth row form one input matrix $X_r \in \mathcal{R}^{m \times n}$ to the RC-

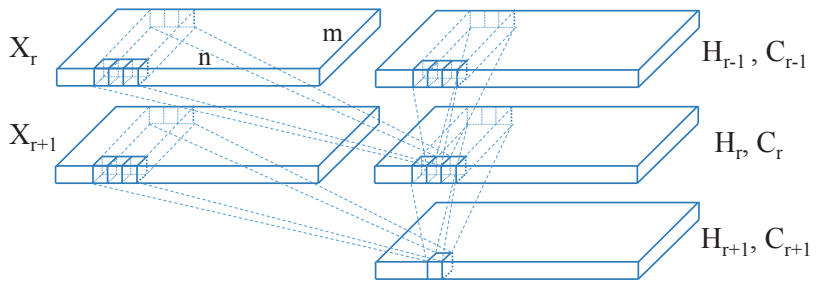


Figure 6.1: The structure of the RC-LSTM.

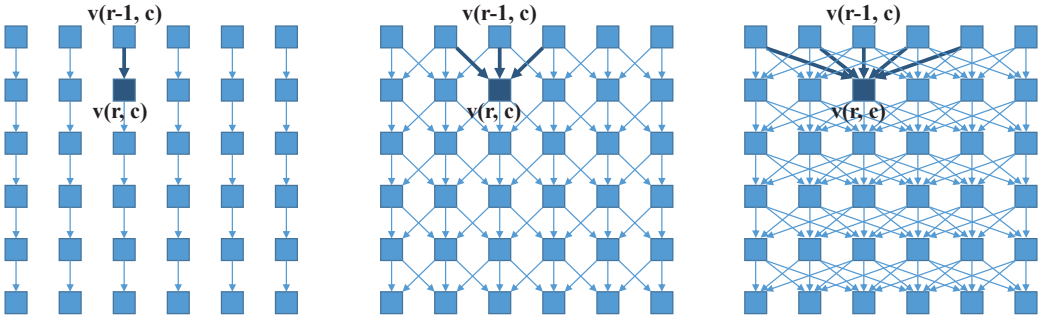


Figure 6.2: Illustration of top-to-bottom message passing among the image pixels with k equals 1, 3, and 5.

LSTM, where n is the number of pixels in each row and m is the feature dimension. The RC-LSTM determines the hidden state H_r by the input X_r and the states H_{r-1}, C_{r-1} of the previous row. Convolution operations are used in both the input-to-state and state-to-state transitions, and the RC-LSTM can be formulated as:

$$
\begin{aligned}
i_r &= \sigma(w_{xi} \otimes X_r + w_{hi} \otimes H_{r-1} + b_i) \\
f_r &= \sigma(w_{xf} \otimes X_r + w_{hf} \otimes H_{r-1} + b_f) \\
o_r &= \sigma(w_{xo} \otimes X_r + w_{ho} \otimes H_{r-1} + b_o) \\
g_r &= \phi(w_{xc} \otimes X_r + w_{hc} \otimes H_{r-1} + b_c) \\
C_r &= f_r \odot C_{r-1} + i_r \odot g_r \\
H_r &= o_r \odot \phi(C_r)
\end{aligned}
\tag{6.1}
$$

where the w-s are convolution kernels with size $1 \times k$ and $\otimes$ denotes convolution. The distinguishing feature of our RC-LSTM is that the inputs X_r, cell states C_r, hidden states H_r, and the gates i_r, f_r, g_r, o_r are all matrices. In this sense, our RC-LSTM can be considered as a generalization of the traditional vector-based LSTM to handle structure input.

The RC-LSTM model enables the message passing in one direction: from the top to the bottom of the image. The pixel $v_{(r,c)}$ at location (r, c) will get the information propagated from its ancestors in the previous row, which are a small neighborhood of k pixels near

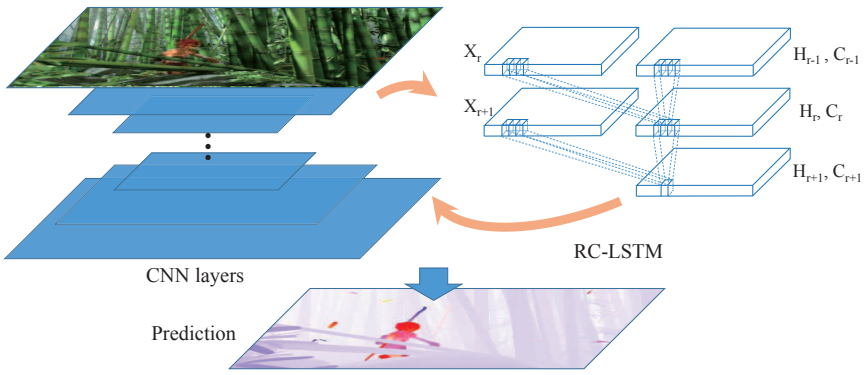


Figure 6.3: The overall framework of an end-to-end trainable network. RC-LSTMs can be used in any point to refine the feature maps, by modeling the spatial dependencies of local features and produce context-aware representations.

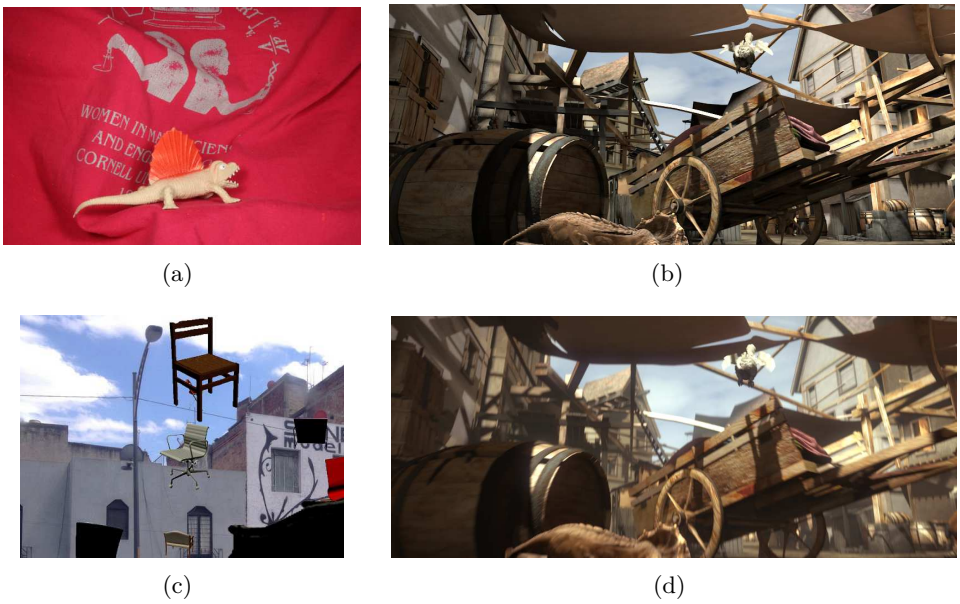
the pixel $v_{(r-1,c)}$. Figure 6.2 illustrated the message passing among the pixels. Our RC-LSTM explicitly models the contextual dependencies among the pixels using this 1-direction message passing.

Spatial dependencies of a pixel come from surrounding pixels in all directions within an image, and the 1-direction message passing might not be enough to model all these dependencies. This can be addressed by using the RC-LSTM 4 times: row by row from top to bottom, from bottom to top, column by column from left to right, and from right to left. We call this method 4-direction message passing RC-LSTM, which is able to model more complete contextual dependencies.

6.3.2 Integration with CNNs

Our RC-LSTMs can be integrated with any CNNs structures, other networks (e.g. auto-encoder [78]), and even hand-crafted features, to model the spatial dependencies of the local features and produce context-aware feature representations. Figure 6.3 shows a generic framework that integrates the RC-LSTM model with a general CNN. The RC-LSTM model gets input from the CNN feature maps, refine the local features with message passing, and output context-aware feature maps, which can be further processed by CNN layers or directly used to predict output flows.

In our experiment, we integrate our RC-LSTM with FlowNetS [18], which has a generic network structure and is shown to have better generalization abilities. The image pair is stacked as the input of the FlowNetS, and the network has 10 convolution layers and 4 deconvolution layers. Detailed structure of the integrated network can be found in Section 6.4.1.



(a) (b) (c) (d)

Figure 6.4: Sample images from (a) Middlebury, (b) Sintel Clean, (c) Flying Chairs, and (d) Sintel Final dataset. The Final version of Sintel (d) includes motion blur and atmospheric effects to the Clean version (b).

6.4 Experiments

6.4.1 Datesets and Experiment Setup

Optical flow evaluation requires dense per-pixel ground truth, which is very difficult to obtain from real world images. Therefore, the amount of real images with dense ground truth optical flow field is very small. In Middlebury dataset [4], optical flow ground truth of 6 image pairs are generated by tracking the hidden fluorescent paint applied to the scene surfaces, with computer-controlled lighting and motion stages for camera and scene. The KITTI dataset [25] is larger (389 image pairs) and the semi-dense optical flow ground truth is obtained by the use of 4 cameras and a laser scanner.

In order to facilitate the evaluation of optical flow algorithms, some larger synthetic datasets with a variant of motion types are generated [12, 18]. In our experiments, we attempt to use both real world images and synthetic images to evaluate our method. The datasets used in our experiments are introduces below, and example images from some of these datasets are shown in Figure 6.4.

- **Middlebury dataset** [4] contains 8 training image pairs and 8 testing image pairs with dense optical flow ground truth. Six image pairs are real world images and others are synthetic ones. The image resolution ranges from 316×252 to 640×480, and the displacements are small, usually less than 10 pixels. The ground truth flows of the training images are publicly available, while the ground truth of the testing images is

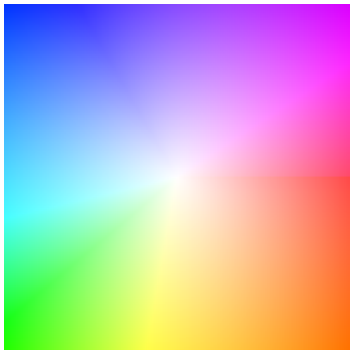


Figure 6.5: Optical flow color coding scheme: the vector from the center to a pixel is encoded using the color of that pixel.

hidden from the public, researchers can upload their testing results to an evaluation server.

- **KITTI dataset** [25] contains 194 training image pairs and 195 testing image pairs. An average of 50% pixels in the images have ground truth optical flow. The images are captured using cameras mounted on an autonomous car in real world scenes, and the image resolution is about 1240×376. This dataset contains strong projective transformations and special types of motions. Similar with Middlebury dataset, the ground truth flows of the training images are publicly available, while the testing results are evaluated on a server.

- **Sintel dataset** [12] contains computer rendered artificial scenes of a 3D movie with dense per-pixel ground truth. It includes large displacements, and pays special attention to achieve realistic image properties. The dataset provides "Clean" and "Final" versions. The Final version includes atmospheric effects (e.g. fog), reflections, and motion blur, while the Clean version does not includes these effects. Each version contains 1041 training image pairs, and 552 testing image pairs. The image resolution is 1024×436. The ground truth flows of the training images are publicly available, while the testing results are evaluated on a server.

- **Flying Chairs dataset** [18] is a large synthetic dataset which is built to provide sufficient data for training CNNs for optical flow estimation. It contains 22232 training image pairs and 640 testing image pairs. The images have resolution 512×386 and are generated by rendering 3D chair models on background images from Flickr. The ground truth flows of the whole dataset are publicly available.

We train our network on the training set of Flying Chairs dataset, and test the network on the Sintel, KITTI, Middlebury, and the testing set of Flying Chairs datasets. Note that we only train our model on the training set of Flying Chairs dataset, and we do not train or fine-tune our model on the other datasets. Therefore, both the "train" and "test" data of

Table 6.1: A summary of the datasets used to TEST our method.

Dataset	image pairs #	Evaluation Method
Sintel Clean Train	1041	Compare results with GT
Sintel Clean Test	552	Upload to evaluation server
Sintel Final Train	1041	Compare results with GT
Sintel Final Test	552	Upload to evaluation server
KITTI Train	194	Compare results with GT
Middlebury Train	8	Compare results with GT
Flying Chairs Test	640	Compare results with GT

other datasets can serve as testing data in our experiments, the same scheme is used in [18]. The datasets used to test our method are summarized in Table 6.1.

Two architectures of our RC-LSTM as described in Section 6.3.1, i.e. 1-direction and 4-direction message passing, are integrated with FlowNetS to build an end-to-end trainable network. We do not include the variational refinement because it is essentially using CNN results to initialize a traditional flow estimation [11] and is not end-to-end trainable. Specifically, our RC-LSTMs are plugged into FlowNetS after the last deonvolution layer and before the final prediction layer. The detailed network structure can be found in Table 6.2 and our implementation is based on Caffe [40]. The kernel size is set to be 1×3. For the sake of convenience, we call the integrated networks Proposed-1dir and Proposed-4dir. The networks are evaluated both qualitatively and quantitatively described as follows.

The endpoint error (EPE) is used to evaluate the performance of different methods, which is define as:

$$EPE = \frac{1}{N} \sum \sqrt{(u_i - u_{GTi})^2 + (v_i - v_{GTi})^2} \tag{6.2}$$

where N is the total number of image pixels, (u_i, v_i) and (u_{GTi}, v_{GTi}) are the predicted flow vector and the ground truth flow vector for pixel i, respectively.

The optical flow field of an image pair is also visualized as an image by color-coding the flow field as in [4, 12]. Flow direction is encoded with color and flow magnitude is encoded with color intensity. Figure 6.5 shows the color coding scheme: the vector from the center to a pixel is encoded using the color of that pixel. We use the open source tool provided in [4] to generate color-coded flow maps. The estimated flow maps on Flying Chairs test, Sintel train, and Middlebury train are visualized and compared with the visualized ground truth.

6.4.2 Results

Figure 6.6 compares the Proposed-1dir and Proposed-4dir networks to the FlowNetS method on the Flying Chairs and Middlebury datasets. The results produced by FlowNetS are

Table 6.2: The detailed structures of the Proposed-1/4dir networks with RC-LSTM + FlowNetS. The illustrated input/output resolutions are based on the input image with size 512×384. pr stands for prediction, and pr' stands for the upsampled pr.

Layer name	Kernel sz	Str	I/O Ch#	InputRes	OutputRes	Input
conv1	7×7	2	6/64	512×384	256×192	Stacked image pair
conv2	5×5	2	64/128	256×192	128×96	conv1
conv3	5×5	2	128/256	128×96	64×48	conv2
conv3_1	3×3	1	256/256	64×48	64×48	conv3
conv4	3×3	2	256/512	64×48	32×24	conv3_1
conv4_1	3×3	1	512/512	32×24	32×24	conv4
conv5	3×3	2	512/512	32×24	16×12	conv4_1
conv5_1	3×3	1	512/512	16×12	16×12	conv5
conv6	3×3	2	512/1024	16×12	8×6	conv5_1
conv6_1	3×3	1	1024/1024	8×6	8×6	conv6
pr6+loss6	3×3	1	1024/2	8×6	8×6	conv6_1
deconv5	4×4	2	1024/512	8×6	16×12	conv6_1
pr5+loss5	3×3	1	1026/2	16×12	16×12	conv5_1+deconv5+pr6'
deconv4	4×4	2	1026/256	16×12	32×24	conv5_1+deconv5+pr6'
pr4+loss4	3×3	1	770/2	32×24	32×24	conv4_1+deconv4+pr5'
deconv3	4×4	2	770/128	32×24	64×48	conv4_1+deconv4+pr5'
pr3+loss3	3×3	1	386/2	64×48	64×48	conv3_1+deconv3+pr4'
deconv2	4×4	2	386/64	64×48	128×96	conv3_1+deconv3+pr4'
RC-LSTM	1×3	1	194/200	128×96	128×96	conv2+deconv2+pr3'
pr2+loss2	3×3	1	200/2	128×96	128×96	RC-LSTM
upsample	-	-	2/2	128×96	512×384	pr2

often blurry and inconsistent, which lose some of the object details (e.g. chair legs). Both our proposed networks are able to produce better visualized flow maps, which are more accurate and contains more consistent objects with finer details. This demonstrates that our RC-LSTM is able to enhance the CNN features of FlowNetS, by explicitly modeling the spatial dependencies among pixels and producing context-aware features for optical flow estimation. It can be found the Proposed-4dir network produces lower endpoint errors (EPE) than the Proposed-1dir network.

In Figure 6.7, we qualitatively compare our Proposed-1dir to FlowNetS as well as the state-of-the-art method, EpicFlow [74], on more examples from Sintel Final dataset. The images of EpicFlow and FlowNetS are from the results in [18]. It can be seen from the figure that in most cases (Row 1-8), the Proposed-1dir method produces visually better results and lower endpoint errors (EPE) than the FlowNetS. Although the EPEs of the proposed

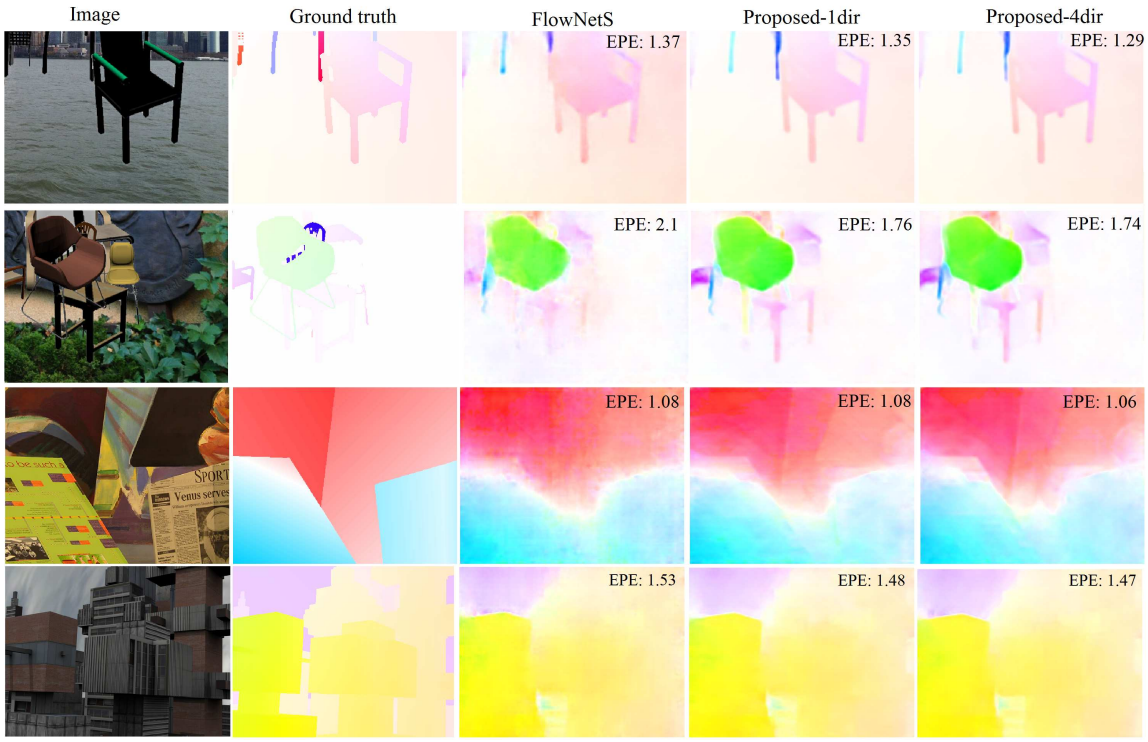


Figure 6.6: The estimated optical flow maps from the Flying Chairs dataset(top two) and the Middlebury dataset(bottom two).

method are somewhat worse than that of EpicFlow, our model often produces better object details, see Row 3-5, 7-8, and 10 in Figure 6.7.

Table 6.3 shows the quantitative comparison between our Proposed-1dir and Proposed-4dir methods to many well-performing methods on Sintel Clean, Sintel Final, KITTI, Middlebury, and Flying Chairs datasets. It is shown that both our Proposed-1dir and Proposed-4dir methods consistently outperform the FlowNetS method on all these datasets. This demonstrates that our RC-LSTM is able to produce more powerful context-aware features for optical flow estimation. The running time of FlowNetS and our models is measured using an *NVIDIA TITAN* GPU on Middlebury dataset, and the time of other methods are from [18].

On Sintel and KITTI train datasets, our models outperform the LDOF [11] method. And our models are comparable to the real-time method EPPM [6], while our Proposed-1dir is two times faster. Although the proposed methods performs not as well as the state-of-the-art EpicFlow [74], it has been shown in Figure 6.7 that our model often produce more consistent results. It is more interesting to see the quantitative results on the Flying Chairs test dataset. Since our models are trained on the training set of Flying Chairs, they are expected to perform better when testing on this dataset than on others. Table 6.3 shows that both our models outperform all the state-of-the-art methods.

Figure 6.7: Predicted optical flows on the Sintel Final dataset. In each row from left to right: overlaid image pair, ground truth flow, the predicted results of EpicFlow [74], FlowNetS [18], and Proposed-1dir RC-LSTM. Endpoint error (EPE) is shown at the top-right corner of each prediction. In most cases (Row 1-8), the proposed method produces visually better results with lower EPE than the FlowNetS. Although the EPEs of the proposed method are somewhat worse than that of EpicFlow, our model often produces better object details, see Row 3-5, 7-8, 10.

Table 6.3: Average endpoint errors (in pixels) of our networks compared to several well-performing methods on several datasets. Since we trained our network on Flying Chairs dataset, we can test our model on both the train and test images on other datasets.

Method	Sintel Clean		Sintel Final		KITTI	Middle	Chairs	Time(s)	
	train	test	train	test	train	train	test	CPU	GPU
EpicFlow [74]	2.40	4.12	3.70	6.29	3.47	0.31	2.94	16	-
DeepFlow [110]	3.31	5.38	4.56	7.21	4.58	0.21	3.53	17	-
EPPM [6]	-	6.49	-	8.38	-	-	-	-	0.2
LDOF [11]	4.29	7.56	6.42	9.12	13.73	0.45	3.47	65	2.5
FlowNetS [18]	4.50	7.42	5.45	8.43	8.26	1.09	2.71	-	0.05
Proposed-1dir	3.77	6.72	4.93	7.94	7.55	1.02	2.64	-	0.08
Proposed-4dir	3.77	6.69	4.90	7.91	7.54	1.01	2.56	-	0.2

The overall experimental results show that our models achieve best performance on the Flying Chairs dataset, and generalize well to other existing datasets. Note the fact that training data is essential to the performance of the CNN models, while the Flying Chairs dataset contains unrealistic images with 3D chair models rendered on background images. These results indicate that our method is very promising and may perform even better, if sufficiently large datasets with more realistic images are available.

6.5 Summary

In this chapter, we introduce a row convolutional long short-term memory (RC-LSTM) network for modeling contextual dependencies of image pixels. The RC-LSTM is integrated with FlowNetS to enhance its learned feature representations and produce context-aware features for optical flow estimation. Our model can produce more accurate and consistent optical flows than the comparing CNN-based models, and achieves competitive accuracy on several datasets.

Optical flow is used in one of our methods to analyze the rat action states (Chapter 3). The rat head motion direction is directly computed as the average optical flow direction, and the object offset used for rat head orientation is also inferred based on optical flow estimation. Our model introduced in this chapter can achieve more accurate optical flow estimation, therefore it should be able to provide better rat states estimation for automatic navigation tasks.

Chapter 7

Deep Attention Networks for Egocentric Action Recognition

The recognition of horizontal rat head orientation and motion is sufficient for our current rat robot automatic navigation tasks. However, more complex navigation tasks may require the recognition of more sophisticated actions. Therefore, we extend our deep neural networks used for rat states analysis in Chapter 3 to be a two-stream architecture, which consists of an appearance-based stream and a motion-based stream to handle more complex action analysis for egocentric videos. We incorporate a spatial attention network in each of the streams to predict an attention map. The attention maps help our model to identify and focus on the most relevant spatial regions of the frames to recognize actions. A temporal network is incorporated in each stream to better exploit the temporal structure of the egocentric videos for action recognition. Our model is evaluated on two publicly available egocentric datasets which contain fine-grained human actions with more categories.

7.1 Introduction

Understanding human behavior from videos has been a highly active research topic in computer vision. With the availability of various wearable cameras, there is a growing interest in using first-person videos to understand the camera wearer's behavior. The wearable camera is usually mounted on a person's head and its optical axis is aligned with the person's field of view. Recognizing actions using first-person videos, or egocentric videos, is different from that using third-person videos. This is because the camera wearer's poses are mostly unavailable in these videos. And unlike third person videos where the camera is either static or moving smoothly, strong motions are commonly present in egocentric videos due to the head motion of the camera wearer. These aspects make egocentric action recognition very challenging.

Researchers have explored a rich set of visual features, including object-centric features and egocentric cues for action recognition in first-person videos. The object features aim at

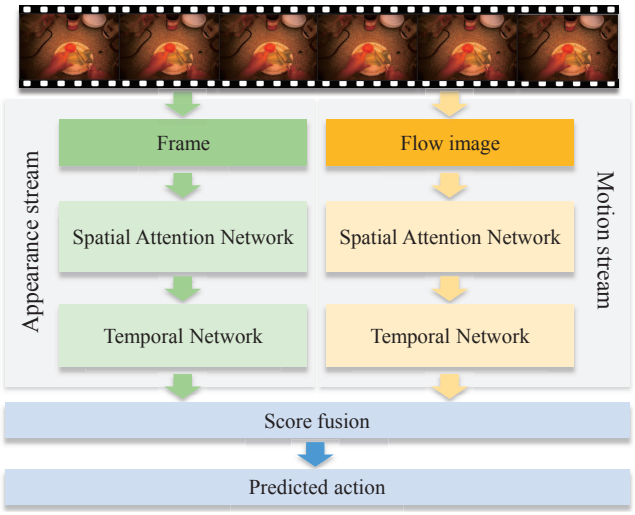


Figure 7.1: The overview of our approach. The spatial attention networks predict attention maps to select relevant regions to focus on. The temporal networks model the forward and backward information for action recognition.

capturing the appearance changes of objects and are shown to be effective in characterizing egocentric actions [20, 71]. The egocentric cues include the first person's head/hand motion and hand pose [21, 54], which can reveal the underlying actions of the camera wearer and are shown to be complementary to object-centric representations. Recent works have attempted to employ the feature learning capability of convolutional neural networks (CNNs) [51] for egocentric action recognition and have achieved good performance [63, 89]. In order to directly incorporate object and egocentric cues, these models use preprocessed inputs such as hand mask, homography [89], and localized objects [63].

Using eye-tracking devices, the gaze or eye fixation of the person can be recorded while interacting with the physical world, which is also utilized to facilitate egocentric action recognition during object manipulation tasks [21, 52]. The eye movements reflect a person's thinking process and represent human attention [121], which can be divided into two categories: bottom-up attention and top-down attention [7, 43]. The bottom-up attention includes the human attention when they are performing free-viewing on a scene or an image, in which the objects or regions that "stand out" relative to the neighboring parts (saliency) attract human attention. In comparison, the human attention when they are performing certain tasks (e.g. object manipulation) belongs to the category of top-down attention, which is task-driven. The human eye movements and fixation during performing these tasks are very different from during free-viewing [121]. It is demonstrated in [49] that a substantial percentage of human eye fixations falls on the task-relevant regions during object manipulation tasks, which require hand-eye coordination. For example, when pouring water into a bottle, instead of paying attention to the salient objects, the person has to fixate on the opening of the bottle and also monitor the water level in the bottle. In these tasks, the point

of eye fixation may not be at the location which is the most visually salient (bottom-up attention), but rather will correspond to the most relevant location depending on the task demands and human action (top-down attention).

Based on these insights, we propose an attention-based deep network that exploit the spatial and temporal structure of the egocentric videos for action recognition. The proposed model has a two-stream architecture which consists of an appearance stream and a motion-based stream, as shown in Figure 7.1. A spatial attention network is incorporated in each stream to learn human attention using gaze as ground truth. It shares the convolutional layers in the stream and has a separate branch to predict a human attention map. This attention map helps our model to focus on the most relevant spatial region of the inputs to predict actions. The temporal network incorporates bi-directional long short-term memory (LSTM) to model the long-range dependencies of the video frames.

Our contribution can be summarized as follows: (1) We propose a spatial attention network and a temporal network, which are incorporated in a two-stream architecture for egocentric action recognition. Our model achieves state-of-the-art performance on GTEA Gaze dataset. (2) We provide detailed ablation analysis to demonstrate how the proposed spatial attention network and temporal network contribute to the overall performance. (3) To the best of our knowledge, our method is the first successful deep network-based method that models human gaze behavior and top-down attention. By comparing to a prediction-oriented attention model, we demonstrate both quantitatively and qualitatively that our spatial attention network with gaze supervision is capable of learning better attention mechanism for egocentric action recognition.

7.2 Related Work on Attention Model

Attention models have been proved successful in variance vision tasks, such as object recognition [3], image and video captioning [119, 117], action recognition [83, 19, 59], and visual question answering [114, 118]. The visual attention mechanism aims to identify interesting regions in the visual data and focus on these regions to extract relevant information, which mimics the human perception and thinking process during accomplishing certain tasks.

In the attention models, a probability distribution over a grid of features is first predicted to indicate the level of attention on each region. The soft attention models use the attention distribution to re-weight the features, while the hard attention models select the feature with the highest probability to represent the data. Both soft and hard attention models are explored in [117] for generating image captions. The soft attention model is trained using back-propagation and the hard attention model is trained using reinforce algorithm. Attention mechanism is extended to temporal domain in [119, 70], where the models learn to select more relevant video segments for video description and action recognition. Based on the insight that image question answering requires multiple steps of reasoning, stacked

attention networks are proposed in [119] to progressively focus on different regions of the image to infer the answer. The spatial transformer networks [38] introduce affine transformations to the CNN feature maps, which allows the model to attend to arbitrary regions of the data.

These attention models learn to select the most relevant part of the data for the task automatically. With the availability of human gaze information in egocentric videos, we are able to use this real human attention to train our spatial attention model in a supervised way. To the best of our knowledge, our work is the first attempt to use deep spatial attention models in egocentric action recognition, which models gaze behavior and the task-dependent top-down human attention..

7.3 The Proposed Approach

In this work, we employ a two-stream architecture composed of an appearance stream and a motion stream to recognize egocentric actions. The spatial attention network is incorporated in each stream to predict human attention distribution using gaze information as ground truth. The attention distribution helps to selectively focus on the most relevant part of the data to predict actions. The temporal network incorporates bi-directional LSTM to model the long range temporal structure of the videos for recognizing actions. In this section, we will describe our spatial and temporal networks, and provide detailed framework of our two-stream architecture.

7.3.1 Spatial Attention Network for Predicting Spatial Relevant Regions

Our spatial attention network takes the feature map of the last convolutional layer in the generic CNNs as input. We denote the feature map as $X \in \mathbf{R}^{K \times K \times D}$, where K is the spatial resolution of the feature map and D is the number of the feature channels. We feed the feature map to a convolutional layer to predict an attention distribution $A \in \mathbf{R}^{K \times K}$ over the grid of features as:

$$A = f(X \otimes w + b) \tag{7.1}$$

where w is the convolution kernel and b is the bias term.

The previous attention models do not have direct supervision on this predicted attention distribution. Instead, they use the distribution to either weighted average the features or select the features with highest attention, then the model is trained in a prediction oriented manner by attempting to minimize the prediction error. Therefore, these attention models implicitly learns an attention mechanism to focus on certain regions of the input that in favor of the final prediction.

In this work, we utilize the gaze information of the human as ground-truth to enforce the training our spatial attention network, see Fig 7.2. Therefore our model is able to predict top-down task dependent human attention to select the relevant regions for better action

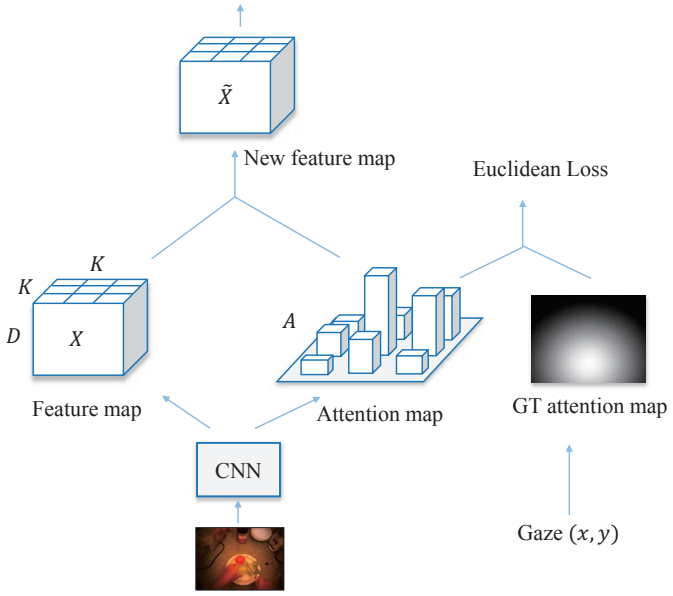


Figure 7.2: The spatial attention network, which is trained with ground truth (GT) attention map generated using human gaze location.

recognition. The eye movement can be tracked using wearable tracking system such as Tobii, and synchronized with the egocentric videos. The eye fixation position in the scene is then transformed to the image coordinate and represented as the gaze location (x, y) in each frame. We compute the ground-truth human attention distribution by applying a Gaussian bump on the gaze location:

$$A_{ij}^{gt} = \exp(-\frac{(i - x')^2 + (j - y')^2}{2\sigma^2}) \tag{7.2}$$

where $A^{gt} \in \mathbf{R}^{K \times K}$, $i, j \in [1, K]$ are the spatial index in the attention map, and x',y' are the scaled gaze location in the interval of $(0, K)$. Euclidean loss is used to measure the prediction error of our attention distribution as:

$$L = \frac{1}{2K^2} \sum_{i=1}^{K} \sum_{j=1}^{K} (A_{ij} - A_{ij}^{gt})^2. \tag{7.3}$$

Based on the attention distribution, we re-weight the feature map X by:

$$\tilde{X} = A \odot X \tag{7.4}$$

where $\odot$ represent element-wise multiplication. Therefore the feature vector of position (i, j), denoted as $\tilde{X}_{ij}$ is the product of the scalar A_{ij} and the corresponding feature vector $X_{ij} \in \mathbf{R}^D$. This attention mechanism constructs a more informative feature map $\tilde{X}$, since higher weight can be assigned to visual regions that are more relevant to the current action.

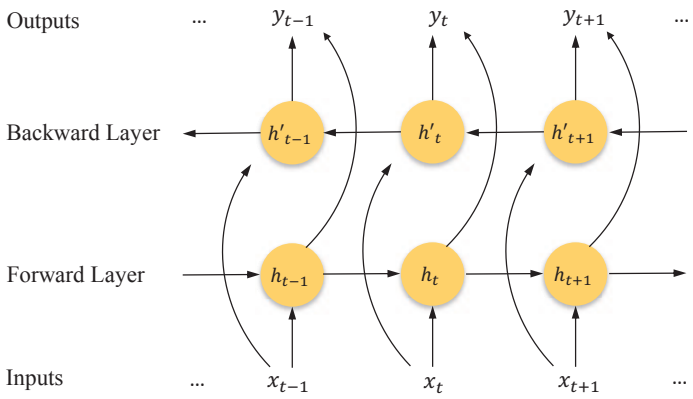


Figure 7.3: The temporal network: bi-direction LSTM.

The feature map $\tilde{X}$ is then processed by the consecutive fully connected layers to produce a feature vector x.

7.3.2 Temporal Network for Modeling Temporal Structure

In the previous deep models, stacked optical flow is used to model the temporal dynamics of the videos. In order to better exploit the temporal structure of egocentric videos for action recognition, we incorporated a bi-directional LSTM in our temporal network.

Long short-term memory (LSTM) [32] is stable and powerful for modeling long-range temporal dependencies without the vanishing gradient problem of the simple RNNs. Its innovation is the introduction of the "memory cell" c_t to accumulate the state information. The cell is accessed, written and cleared by several controlling gates, which enables LSTM to selectively forget its previous memory states and learn long-term dynamics. Given x_t as the input of an LSTM cell at time t, the cell activation can be formulated as:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ g_t &= \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \phi(c_t) \end{aligned} \tag{7.5}$$

where σ stands for the sigmoid function, ϕ stands for the tanh function, and $\odot$ denotes the element-wise multiplication. In addition to the hidden state h_t and memory cell c_t, LSTM has four controlling gates: i_t, f_t, o_t, and g_t, which are the input, forget, output, and input modulation gate respectively. The input gate i_t controls what information in g_t to be accumulated into the cell c_t. While the forget gate f_t helps the c_t to maintain and selectively forget information in previous state c_{t-1}. Whether the updated cell state c_t will

be propagated to the final hidden state h_t representation is controlled by the output gate o_t.

The overall structure of the memory cell and the regulating gates make LSTM suitable for modeling complex temporal relationships that may span a long range. However, one shortcoming of the conventional LSTM is that it is only able to make use of previous context. We incorporate the bi-directional LSTM to process the data in both directions with two separate hidden layers, see Fig 7.3. In addition to the forward LSTM layer that produces a sequence of hidden states h_t, we have a backward LSTM layer that produces a sequence of hidden states h'_t based on the information of the next time step $t+1$:

$$\begin{aligned} i'_t &= \sigma(W'_{xi}x_t + W'_{hi}h'_{t+1} + b'_i) \\ f'_t &= \sigma(W'_{xf}x_t + W'_{hf}h'_{t+1} + b'_f) \\ o'_t &= \sigma(W'_{xo}x_t + W'_{ho}h'_{t+1} + b'_o) \\ g'_t &= \phi(W'_{xc}x_t + W'_{hc}h'_{t+1} + b'_c) \\ c'_t &= f'_t \odot c'_{t+1} + i'_t \odot g'_t \\ h'_t &= o'_t \odot \phi(c'_t) \end{aligned} \tag{7.6}$$

The hidden states of the two LSTM layers are combined to produce the output by:

$$y_t = W_{hy}h_t + W'_{hy}h'_t + b_y \tag{7.7}$$

where W_{hy} and W'_{hy} are the weight matrices and b_y is the bias term.

7.3.3 Two-stream Architecture

It has been proved successful to decompose videos into spatial and temporal components for action recognition [88, 63]. The spatial component, in the form of raw frame appearance, contains information about scenes and objects depicted in the video. The temporal component, in the form of motion across the frames, conveys the motion of the camera and objects in the scene. The appearance stream and motion stream in our model cope with the spatial and temporal video components respectively.

Each stream of our model contains a generic CNN to extract feature maps from the spatial and temporal components. The CNN in the appearance stream operates on raw video frames. It captures appearance features such as hand-object configuration and object attributes. The CNN in the motion stream takes optical flow images as input and learns complementary features. We adopt the flow image encoding method in [17]. The first two channels of the flow image are computed by centering x and y flow values around 128 and scaling the values to fall between 0 and 255. The third channel of the flow image is created by computing the flow magnitude. This flow image models local temporal structure and conveys short term information about the camera, hand or object motion.

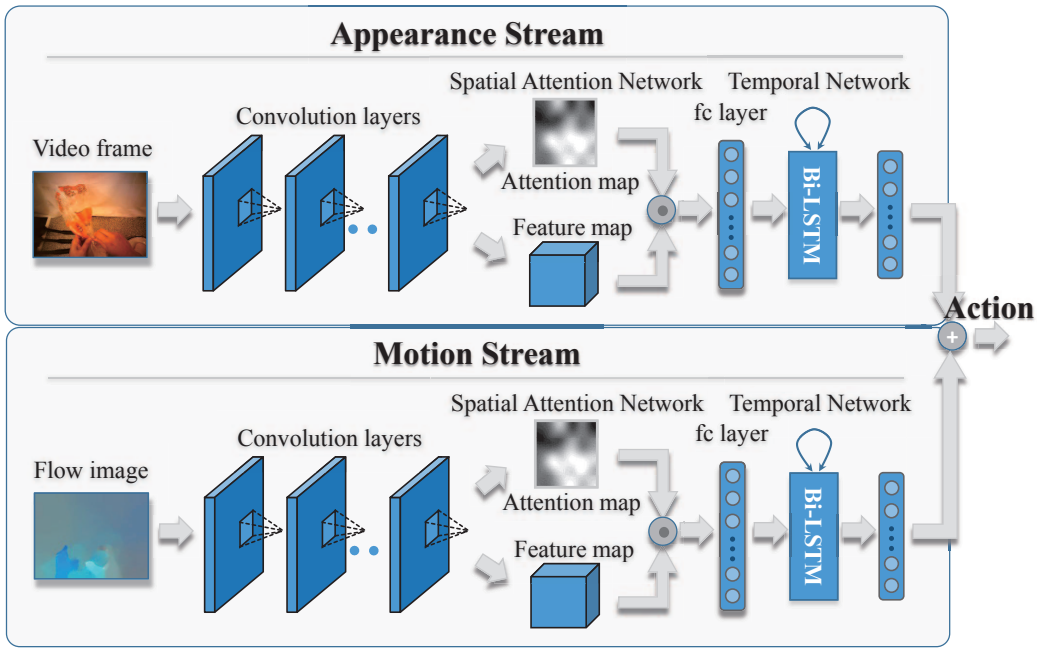


Figure 7.4: Framework details of our two-stream model.

The advantage of decoupling the appearance and motion streams is that it allows us to utilize the large amount of annotated image data (e.g. ImageNet) for pre-training each of the CNN. The CNNs extract discriminative feature maps, which are then used by our spatial attention and temporal networks to recognize egocentric actions, as illustrated in Fig 7.4. The softmax scores of the two streams are fused to predict the action labels.

7.4 Experiments

7.4.1 Datasets and Experimental Setup

We evaluate our proposed method on two public datasets: GTEA Gaze (Gaze) and GTEA Gaze+ (Gaze+). These datasets are collected using a head-mounted camera and the activities performed by the camera wearer involve hand-object interactions. These datasets include action annotations as well as gaze information. Each action is represented by a verb and a set of nouns, for example "put lettuce (on) plate". The gaze information is represented as a coordinate in the video frame, indicating the location where the person is looking at. The details of the datasets are introduced below.

- **Gaze dataset** [21] contains 17 video sequences performed by 14 different subjects. The total duration of these videos is one hour, and the frame resolution is 640×480. Previous works usually report results on fix splits on this dataset, where 13 sequences are used as training data and 4 sequences are used as testing data. There are 40 action categories and a total of 331 action instances.

Table 7.1: Comparison of the action recognition accuracy of our method with previous methods.

Methods	Gaze (40)	Gaze+ (44)
Wang *et al.*: DT [103]	34.10	42.40
Wang *et al.*: IDT [104]	27.70	49.60
Li *et al.*: O+E+H [54]	35.10	57.40
Li *et al.*: O+M+E+H [54]	35.70	60.50
Li *et al.*: O+M+E+G [54]	39.60	60.30
Ma *et al.*: object-cnn [63]	35.56	46.38
Ma *et al.*: motion+object-svm [63]	16.00	34.75
Ma *et al.*: motion+object-joint [63]	43.42	66.40
Ours: appearance stream	42.86	57.63
Ours: motion stream	37.36	57.42
Ours: two-stream fusion	48.35	64.74

- **Gaze+ dataset** [21] contains 37 video sequences performed by 6 different subjects. The total duration of these videos is 9 hours, and the frame resolution is 1280×960. Previous works usually report leave-one-subject-out cross validation accuracy on this dataset. There are 44 action categories and a total of 1958 action instances.

Each action instance in these datasets is a segment of the video with tens to hundreds of consecutive frames, during which the camera wearer completes one action. Therefore each instance and all the frames it contains have a single action label. We use the instance level accuracy to evaluate the performance of our method, which is the percentage of the instances that are classified correctly in the testing set.

Our model takes a set of consecutive frames as input (16 in our experiments), and use the bi-directional LSTM to learn the forward and backward dependencies among the frames. During each training iteration, we randomly select a start frame for an instance and use the consecutive 16 frames to train the model. At test time, we extract 16 frame clips with a stride of 8 frames from each test instance and average the score across the instance to predict its action label.

7.4.2 Comparison with Previous Methods

We compare our method with well-performing methods using traditional features such as DT [103] and improved DT (IDT) [104], as well as the best-known methods proposed in [54, 63]. Li *et al.* [54] provide a systematic evaluation of different combinations of features for egocentric action recognition, and the best performing ones are included in Table 7.1. **O** stands for object features obtained by concatenating the Fisher Vectors (FVs) from HoG, LAB color histogram and LBP; **M** stands for motion features, which are a concatenation of the FVs from trajectory features, MBHx, MBHy and HoF; **E** stands for egocentric features

which are computed by concatenating FVs from head motion and manipulation point; **G** and **H** represents the selected local descriptors near the gaze point and the manipulation point estimated by hand shape. Ma *et al.* [63] propose a deep architecture that contains an object CNN and a motion CNN to recognize actions.

The action recognition accuracies of our appearance stream, motion stream, and the overall two-stream model are shown in Table 7.1. The score fusion can result in a significant improvement over both the appearance stream and the motion stream, which demonstrates that these streams learn complementary information for action recognition. Our method consistently outperforms the DT [103], IDT [104] methods, as well as the top 3 feature combinations proposed in [54] on both the Gaze and Gaze+ datasets. The confusion matrices of our method are shown in Fig 7.5. The action categories are sorted with the decreasing number of instances. Our method can get most of the categories correct on Gaze+ dataset. The deviation of the last few categories in Gaze dataset is due to the imbalanced data distribution in this dataset, which is preliminary and incomplete [54]. There are limited training and testing instances (1-2) for those categories and misclassifying one instance can result in a large penalty in the confusion matrix, similar problem is also discussed in [54].

The method in [63] performs action recognition with the help of an explicitly designed object recognition network, which requires additional annotations of hand masks, object locations and object labels. It first utilizes a fully convolutional network to segment hand masks, based on which it fine-tunes an object localization network to detect the objects being manipulated in egocentric videos. The cropped object images are used as input to the object recognition network, which is combined with the motion network for joint action recognition. Since an action is defined as "verb+object", this object network provides additional information for recognizing the action. Therefore the method in [63] is not directly comparable to our model, and we would have also benefited if we were to utilize the object recognition results. However, our model is able to achieve comparable performance on the two datasets. Besides, our model has a simple end-to-end trainable architecture, while the method in [63] has a complex pipeline with several stages.

7.4.3 Ablation Study

To analyze the performance of the spatial attention network and the temporal network in our two-stream architecture, we conduct a detailed ablation study by testing each component in each stream and their combinations to see how they contribute to the action recognition accuracy. The ablation study is conducted on the Gaze+ dataset. Each stream contains a generic CNN, a spatial attention network and a temporal network. We add the components one by one for each stream, which results in the models listed below:

1. **RGB-o**: this model contains only the generic CNN of the appearance stream.

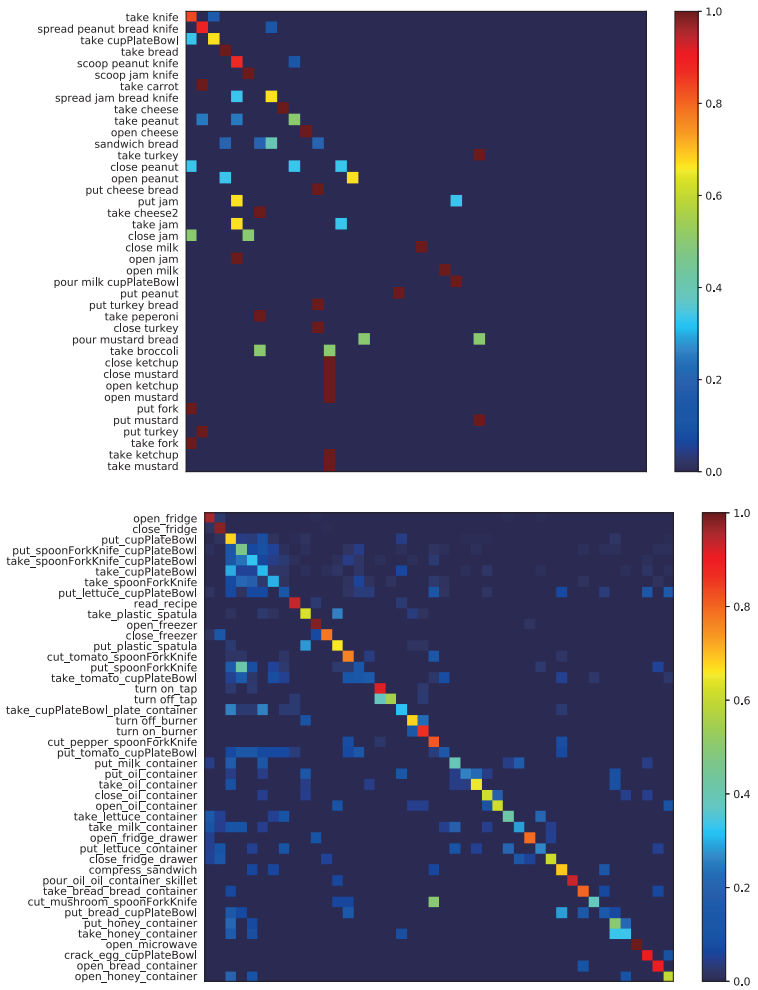


Figure 7.5: Confusion matrices of our method on Gaze (left) and Gaze+ (right) datasets. Action categories are sorted based on decreasing number of instances as in [54].

2. **RGB-s**: this model contains the generic CNN and the spatial attention network of the appearance stream.

3. **RGB-st**: this is the appearance stream of our method, which contains the generic CNN, the spatial attention network and the temporal network.

4. **Flow-o**: this model contains only the generic CNN of the motion stream.

5. **Flow-s**: this model contains the generic CNN and the spatial attention network of the motion stream.

6. **Flow-st**: this is the motion stream of our method, which contains the generic CNN, the spatial attention network and the temporal network.

7. **Fuse-o**: the score fusion from RGB-o and Flow-o.

Table 7.2: Detailed ablation study of our method on Gaze+ dataset. There are 6 subjects in this dataset and we produce action recognition accuracies on each of the subjects and compute the average.

Methods	RGB-o	RGB-s	RGB-st	Flow-o	Flow-s	Flow-st	Fuse-o	Fuse-s	Fuse-st
Subject #1	49.85	51.65	51.65	50.83	52.27	53.10	54.55	57.02	58.26
Subject #2	72.26	72.63	71.17	67.15	67.88	66.42	75.18	75.18	76.28
Subject #3	51.22	51.00	52.77	51.00	53.44	52.11	55.65	56.98	58.31
Subject #4	62.00	62.29	61.43	66.29	64.57	65.14	68.86	71.42	72.57
Subject #5	57.64	61.81	59.03	52.78	50.00	52.08	57.64	62.50	61.81
Subject #6	52.74	55.27	55.70	56.96	56.96	57.81	63.71	64.97	67.09
Average	56.39	57.42	57.63	56.86	57.37	57.42	61.65	63.56	64.74

8. **Fuse-s**: the score fusion from RGB-s and Flow-s.

9. **Fuse-st**: the score fusion from RGB-st and Flow-st, which is our two-stream model.

The detailed results of these models are listed in Table 7.2. We can see a general increasing trend from the accuracy of the '-o','-s', and '-st' models. By including each component, the performance of our model increases consistently. This demonstrates that our spatial attention network and temporal network are able to better model the video structures for action recognition. The results of different fusion combinations of the corresponding RGB and Flow models are shown in columns 8-10 in Table 7.2. The score fusion can result in the improvement of accuracy over each of the individual stream. The best performance is achieved by the Fuse-st model, which fuses the score of our appearance stream and motion stream.

Figure 7.6 shows three examples that the RGB-o model misclassifies but the RGB-s model recognizes correctly. The first row is the video frames and we draw a blue dot in each frame to represent the ground truth (GT) gaze location. The second row is the visualized attention maps of the RGB-s model, which illustrate where the model actually attend to. The values in each attention map are scaled so that the maximum value becomes 255 and the minimum becomes 0. The attention maps are then shown as black-white images and resized to the frame resolution. In these examples, the RGB-o model recognizes the objects that exist in the frames but are not being manipulated, and it considers them as the 'noun' part of the action. The attention map helps the RGB-s model to identify the object being manipulated and predict the correct action label. For example in the second column, the RGB-o model predicts the action to be 'take oil container' due to the appearance of the object in the upper part of the frame. The attention map enables the RGB-s model to focus on the lower part of the frame and obtain the correct label 'take knife'.

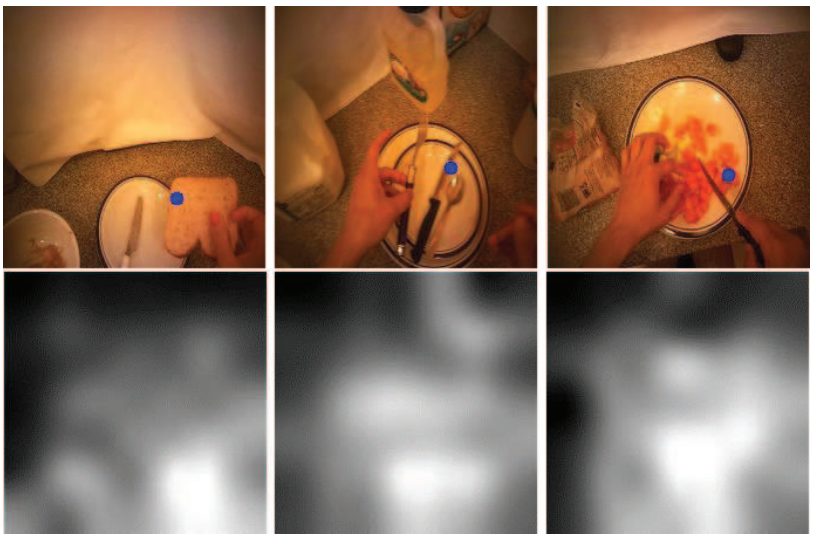


Figure 7.6: First row: the video frames with GT gaze locations drawn in blue dots. The RGB-o model misclassifies the frames as 'put knife','take oil container', and 'cut mushroom'. The GT labels are 'put bread', 'take knife', and 'cut tomato'. With the help of the predicted attention map (second row), our RGB-s model is able to recognize the actions correctly.

Table 7.3: Comparison of our spatial attention network (RGB-s) with the variance (RGB-v) that does not use gaze to learn the attention mechanism.

Methods	RGB-o	RGB-v	RGB-s
Subject #1	49.85	50.41	51.65
Subject #2	72.26	70.80	72.63
Subject #3	51.22	50.78	51.00
Subject #4	62.00	60.29	62.29
Subject #5	57.64	61.11	61.81
Subject #6	52.74	56.12	55.27
Average	56.39	56.65	57.42

7.4.4 Analysis of the Spatial Attention Network

Previous attention models do not have direct supervision on the predicted attention distribution. The success of these attention models indicates that the models can implicitly learn an attention mechanism to focus on certain regions of the input to facilitate final prediction. Therefore, there are two natural questions. *Are these attention models helpful in egocentric action recognition tasks?* and *Does our spatial attention network learn better attention mechanism than these models with the help of human gaze?*

To answer these two questions, we design an attention network variant and test its performance on Gaze+ dataset. This model uses the same architecture as our spatial attention network to predict the attention distribution. However, we does not use the Euclidean loss

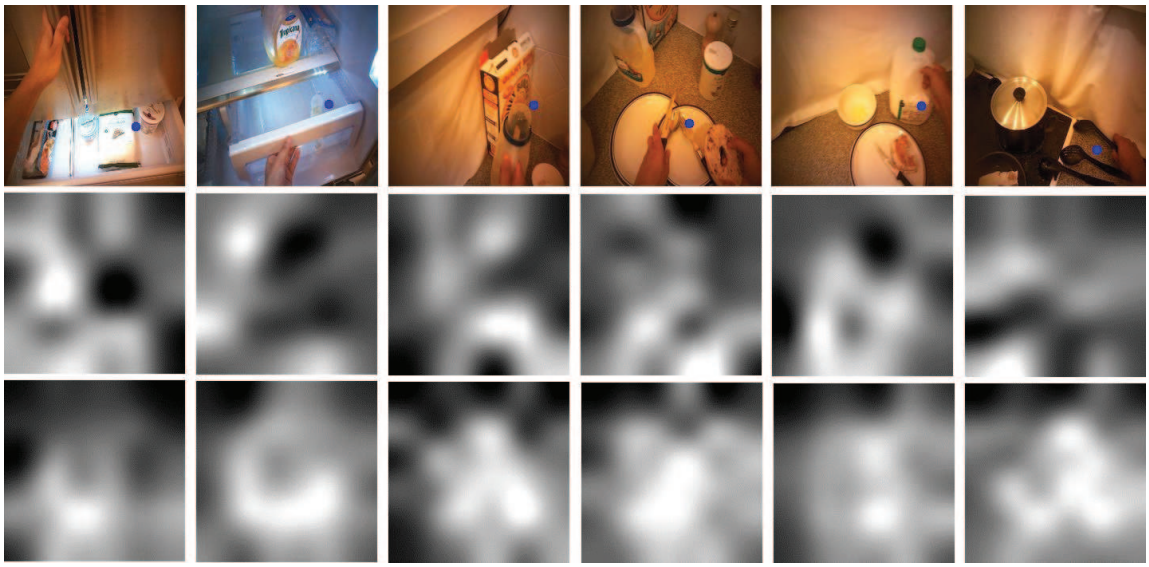


Figure 7.7: The video frames with ground truth gaze locations drawn in blue dots (top), the visualized attention maps of RGB-v (middle), and the visualized attention maps of our spatial attention network RGB-s (bottom). The ground truth action label for each frame from left to right: open freezer, open fridge drawer, take oil container, take plate, take milk container, and take plastic spatula. Our model is able to attend to the image regions that are more relevant to the actions and is more consistent with the human attention (gaze).

with the ground truth attention map to train the layer. Instead, we let the network learn the attention prediction in the prediction-oriented manner by minimizing the final action recognition loss. We combine this attention prediction network with the generic CNN of the appearance stream, and we call this model RGB-v. Table 7.3 shows the action recognition performance of RGB-o, RGB-v and RGB-s methods on the Gaze+ dataset. The RGB-v method outperforms the RGB-o model, which indicates that learning attention mechanism can implicitly facilitate egocentric action recognition to a certain extent. Our spatial attention network RGB-s is able to achieve higher accuracy than RGB-v. This demonstrates that using the human gaze to explicitly train the network can result in a better attention mechanism for egocentric action recognition.

We visualize the attention maps produced by the RGB-v and RGB-s methods, which are shown in Fig 7.7. The first row are the raw video frames with the ground truth human gaze locations drawn as blue dots. The second and third rows are the visualized attention maps of RGB-v and RGB-s methods. It can be found that our RGB-s model is able to attend to the image regions that are more relevant to the action and is more consistent with the human gaze. For example, the first column represent the action of “open freezer”. The RGB-v method tends to focus more on the upper region of the images, specially the hand regions. While our RGB-s method is able assign most of the high weights to the bottom region which represents the freezer.

7.4.5 Implementation Details

We implement our model using Caffe [40], and we adopt the LSTM implementation in [17]. The VGG net [87] 16 layer architecture is adopted as the generic CNN in our appearance and motion streams. We utilize the network weights pre-trained on the ImageNet [80] dataset to initialize our CNNs and then train the networks on the Gaze and Gaze+ datasets.

Leave-one-subject-out cross-validation is performed by the comparing methods on Gaze+ dataset. However, this is very computationally intensive for deep learning frameworks and makes hyper-parameter tuning challenging. In our experiment, we choose our parameters using the first subject as validation set on Gaze+ dataset, and fix parameters for the rest of the dataset as well as for the Gaze dataset. We used 16 timesteps and batch size 8 for the bi-directional LSTM layers, and the LSTM has hidden vector with a dimension of 1024.

Similar to the training procedure used in [17], we first train the generic CNNs with a base learning rate of 0.001 and decrease it by a factor of 0.1 for every 3000 iterations until the maximum iteration 30000. Then we add the spatial attention network and train the model by fixing the CNN convolution kernels. The base learning rate is set to be 10^{-6} and the same learning rate decreasing criteria is used. Finally, the temporal network is added and the whole stream is trained for 10000 iterations with the weights of the previous components fixed. The base learning rate of this step is 0.01 and the learning rate decreases in the same way. The advantage of this training procedure is that we do not need to worry about the learning speed and loss magnitude differences of each component, and it also makes the ablation study easier.

7.5 Summary

In this chapter, we propose a two-stream deep architecture each of which contains a spatial attention network and a temporal network for egocentric action recognition. The temporal network utilizes bi-directional LSTM to model the long-range dependencies among the video frames. The spatial attention network learns to predict attention maps by using gaze information as ground truth. The produced attention maps help our model to identify the most relevant parts in the frames and predict actions more accurately. By visualizing the attention maps of our spatial attention network and a comparing model, we demonstrate that our model is able to predict attention maps that are more consistent with human attention while performing the actions. The overall two-stream model also achieves competitive action recognition performance with the state-of-the-art methods on GTEA Gaze and GTEA Gaze+ datasets.

The two-stream network proposed in this chapter laid the foundation for some fundamental issues in terms of ego-centric action recognition research. It is an extension of our deep neural networks used for rat state analysis. The deep networks described in Chapter 3 are sufficient in current navigation tasks to extract rat states, which are defined to be the

horizontal head orientation and motion direction. In the future, more complex action analysis from the egocentric videos would be needed in order to facilitate navigation in more complex scenes such as real world environment. This two-stream network will be beneficial in such navigation tasks. Currently, this model is not applicable to our rat cyborg automatic navigation, this is because it is computationally expensive and unable to meet the real-time navigation requirement.

Chapter 8

Conclusion

Rat robot has advantages over traditional mechanical robots due to its special motion and perceptual abilities and has great potential for use in rescue and search applications. In this thesis, we address the two major challenges in the rat robot automatic navigation, which are the recognition of the rat action states and the design of the automatic control strategies. We propose a new idea for analyzing the states of the rat robot. We propose two action analysis methods and automatic instruction models, based on which we build a new rat robot for effective automatic navigation. We also propose two models based on deep neural networks, which are useful for potentially more complex action analysis tasks. More specifically, the contributions of this thesis can be summarized as follows:

1. **Egocentric Action Analysis for Rat Robot**

   We mount a miniature camera on the back of the rat and the optical axis is placed in the same direction as the rat head. We use the egocentric video captured by the rat-mounted camera to analyze the rat action states, which are defined to be its head orientation and head motion direction. We propose two egocentric action analysis methods for the rat videos. The first method is based on a traditional optical flow algorithm and the second method utilizes deep neural networks to analyze rat action states. These methods produce the rat action states for automatic navigation in open scenes without the need of a top camera above the scene.

2. **Automatic Instruction Models for Rat Robot Control**

   Inspired by the human control process, we propose two instruction models that issue an instruction sequence automatically according to the following visual cues obtained from the egocentric videos: 1) the feedback of the rat robot states, 2) the objects of interest that indicate the action expectations for the rat robot, e.g. road signs. The first model learns to mimic the human controlling process using data collected during the manually controlled navigation. The second model issues instructions according to a set of control rules defined by human experts' experience and knowledge.

We build a rat robot system, which we refer to as a rat cyborg, and apply the egocentric action analysis and automatic instruction models to enable it to perform automatic navigation in different scenes.

3. **Deep Recurrent Network for Optical Flow Estimation**

   Optical flow encodes the motion between the consecutive frames and is used in one of our method to infer the rat states. In order to produce more accurate optical flow estimation, we propose a row convolutional long short-term memory (RC-LSTM) network to model the spatial dependencies among image pixels and produce context-aware features. Our RC-LSTM network is integrated with Convolutional Neural Networks (CNNs) to form an end-to-end trainable network. This model is evaluated on several optical flow datasets and achieves competitive accuracy.

4. **Deep Attention Networks for Egocentric Action Recognition**

   In order to handle potentially more complex action analysis from the egocentric videos, we extend our deep neural networks used for rat states analysis to be a two-stream architecture, which consists of an appearance stream and a motion-based stream. A spatial attention network is incorporated in each stream to help our model to focus on the most relevant spatial region of the frames to recognize actions. A temporal network is incorporated to better model the temporal structures of the videos for action recognition. Our model is evaluated on two egocentric action recognition datasets and achieves competitive performance.

8.1 Future Work

While we have made multiple contributions to enable the effective rat robot automatic navigation, much more can be done to further improve the rat robot system.

Extracting more information with the rat-mounted camera: Although the visual clues obtained from the video captured by rat-mounted camera (rat states and objects) are sufficient for our current navigation experiments, more information would be beneficial for future navigation requirements. For example, in the systems that use bird's eye cameras to monitor the rat actions, the information of the entire scene and the location of the rat in the scene are available. These additional information can facilitate more complex control, such as path finding and route planning [123].

One future direction of our work is to perform simultaneous localization and mapping (SLAM) [58] using the rat-mounted camera. The rat robot system are expected to construct and update a map of the environment during navigation, while simultaneously keeps track of its location within it. This task is challenging because the quick and unexpected motion of the rat head usually result in very shaky videos. It could be helpful to incorporate additional

hardware sensors and use some prior knowledge of the environment to help the localization and mapping process.

Utilizing the sensory ability of the rats: In our work, we only use the motion ability of the rats, and our rat cyborg perceives the environment purely relying on the computer vision algorithms. The sensory abilities of the rats, such as olfaction and vision, are not utilized. It would be interesting to use these abilities to facilitate the rat cyborg in search applications. One possible way is to use behavioral training to enable the rats to find objects (e.g. water, food) in the environment [123]. With the help of brain-computer interface, we could identify the neural signal that indicates when the rats see/smell particular objects. In this way, the rat can be considered as a powerful biological sensor, and the information read out from the implanted electrodes can be helpful in exploring the environments.

Researchers have studied the neural response of rats to flashing lights [24, 116, 111]. The critical flicker frequency (CFF) threshold for cells in the rat visual cortex is estimated in [111], which is defined as the frequency at which a flickering light is indistinguishable from a steady, non-flickering light. The experiments show that for flashing lights with frequency lower than CFF, the neurons have difference response histograms for different flickering frequency. Xu *et al.* [116] propose a rat robot that is able distinguish lights with high and low flashing frequencies, by analyzing features of neural signal recorded from primary visual cortex of the rat. These works can be combined in our rat cyborg system for better environment perception and more effective navigation.

Navigation in real world environments: Our rat cyborg has great potential for use in search-and-rescue applications, such as during earthquake remedy. This research is driven by these application requirements. However, our current navigation experiments are performed in the lab environment such as the multiple-armed maze and urban planning model, which are still relatively simple and preliminary comparing to real world environments. With the help of additional hardware sensors, the rat sensory capability, as well as more advanced action analysis method and control strategy, hopefully new rat cyborg could be built for search tasks in the real world environment in the future.

Bibliography

[1] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

[2] Chetan Arora and Vivek Kwatra. Stabilizing first person 360 degree videos. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1405–1413. IEEE, 2018.

[3] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[4] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[5] L. Bao, N. Zheng, H. Zhao, Y. Hao, H. Zheng, F. Hu, and X. Zheng. Flight control of tethered honeybees using neural electrical stimulation. In *IEEE/EMBS International Conference on Neural Engineering*, pages 558–561, 2011.

[6] Linchao Bao, Qingxiong Yang, and Hailin Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3534–3541, 2014.

[7] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2013.

[8] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 236–243, 2005.

[9] A. Bozkurt, R. Gilmour, D. Stern, and A. Lal. MEMS based bioelectronic neuromuscular interfaces for insect cyborg flight control. In *IEEE International Conference on Micro Electro Mechanical Systems*, pages 160–163, 2008.

[10] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, pages 25–36. 2004.

[11] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[12] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision (ECCV)*, pages 611–625. 2012.

[13] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

[14] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005.

[16] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *European Conference on Computer Vision (ECCV)*, pages 428–441, 2006.

[17] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.

[18] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[19] Wenbin Du, Yali Wang, and Yu Qiao. Recurrent spatial-temporal attention network for action recognition in videos. *IEEE Transactions on Image Processing*, 27(3):1347–1360, 2018.

[20] Alireza Fathi, Ali Farhadi, and James M Rehg. Understanding egocentric activities. In *IEEE International Conference on Computer Vision (ICCV)*, pages 407–414, 2011.

[21] Alireza Fathi, Yin Li, and James M Rehg. Learning to recognize daily actions using gaze. In *European Conference on Computer Vision (ECCV)*, pages 314–327, 2012.

[22] Alireza Fathi and James M Rehg. Modeling actions through state changes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2579–2586, 2013.

[23] Z. Feng, W. Chen, X. Ye, S. Zhang, X. Zheng, P. Wang, J. Jiang, L. Jin, Z. Xu, C. Liu, F. Liu, J. Luo, Y. Zhuang, and X Zheng. A remote control training system for rat navigation in complicated environment. *Journal of Zhejiang University-Science A*, 8(2):323–330, 2007.

[24] Andrew T Fox, John R Smethells, and Mark P Reilly. Flash rate discrimination in rats: Rate bisection and generalization peak shift. *Journal of the experimental analysis of behavior*, 100(2):211–221, 2013.

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[26] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[27] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1764–1772, 2014.

[28] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 545–552, 2009.

[29] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50, 1988.

[30] Frederick Hayes-Roth, Donald A Waterman, and Douglas B Lenat. Building expert system. 1983.

[31] L. Hermer-Vazquez, R. Hermer-Vazquez, I. Rybinnik, G. Greebel, R. Keller, S. Xu, and J.K. Chapin. Rapid learning and flexible memory in "habit" tasks in rats trained with brain stimulation reward. *Physiology & behavior*, 84(5):753–759, 2005.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] R. Holzer and I. Shimoyama. Locomotion control of a bio-robotic system via electric stimulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1514–1519, 1997.

[34] Berthold K Horn and Brian G Schunck. Determining optical flow. In *Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[35] Yedid Hoshen and Shmuel Peleg. An egocentric look at video photographer identity. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[36] R. Huai, J. Yang, H. Wang, and X. Su. A new robo-animals navigation method guided by the remote control. In *International Conference on Biomedical Engineering and Informatics*, pages 1–4, 2009.

[37] Moustafa Ibrahim, Srikanth Muralidharan, Zhiwei Deng, Arash Vahdat, and Greg Mori. A hierarchical deep temporal model for group activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[38] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[39] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3D convolutional neural networks for human action recognition. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[40] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[41] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3128–3137, 2015.

[42] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[43] Fumi Katsuki and Christos Constantinidis. Bottom-up and top-down attention: different processes and overlapping neural systems. *The Neuroscientist*, 20(5):509–521, 2014.

[44] Kris M Kitani, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3241–3248, 2011.

[45] N. Kobayashi, M. Yoshida, N. Matsumoto, and K. Uematsu. Artificial control of swimming in goldfish by brain stimulation: confirmation of the midbrain nuclei as the swimming center. *Neuroscience letters*, 452(1):42–46, 2009.

[46] Johannes Kopf. 360 video stabilization. *ACM Transactions on Graphics (TOG)*, 35(6):195, 2016.

[47] Johannes Kopf, Michael F Cohen, and Richard Szeliski. First-person hyper-lapse videos. *ACM Transactions on Graphics (TOG)*, 33(4):78, 2014.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[49] Michael F Land and Mary Hayhoe. In what ways do eye movements contribute to everyday activities? *Vision research*, 41(25):3559–3565, 2001.

[50] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[51] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[52] Yin Li, Alireza Fathi, and James M Rehg. Learning to predict gaze in egocentric video. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3216–3223, 2013.

[53] Yin Li, Miao Liu, and James M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[54] Yin Li, Zhefan Ye, and James M Rehg. Delving into egocentric actions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 287–295, 2015.

[55] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision (ECCV)*, pages 614–629. Springer, 2016.

[56] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375, 2015.

[57] J. Lin, C. Yu, J. Jia, S. Zhang, Y. Wang, W. Chen, and X. Zheng. Using dlPAG-evoked immobile behavior in animal-robotics navigation. In *International Conference on Computer Science and Education*, pages 1295–1298, 2010.

[58] Haomin Liu, Guofeng Zhang, and Hujun Bao. Robust keyframe-based monocular slam for augmented reality. In *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*, pages 1–10. IEEE, 2016.

[59] Jun Liu, Gang Wang, Ling-Yu Duan, Kamila Abdiyeva, and Alex C Kot. Skeleton-based human action recognition with global context-aware attention lstm networks. *IEEE Transactions on Image Processing*, 27(4):1586–1599, 2018.

[60] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[61] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[62] Heiko J Luhmann, Joseph P Huston, and Rüdiger U Hasenöhrl. Contralateral increase in thigmotactic scanning following unilateral barrel-cortex lesion in mice. *Behavioural brain research*, 157(1):39–43, 2005.

[63] Minghuang Ma, Haoqi Fan, and Kris M Kitani. Going deeper into first-person activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[64] Humberto Milani, Heinz Steiner, and Joseph P Huston. Analysis of recovery from behavioral asymmetries induced by unilateral removal of vibrissae in the rat. *Behavioral neuroscience*, 103(5):1067, 1989.

[65] Wanli Ouyang, Xiaogang Wang, Xingyu Zeng, Shi Qiu, Ping Luo, Yonglong Tian, Hongsheng Li, Shuo Yang, Zhe Wang, Chen-Change Loy, et al. DeepID-Net: Deformable deep convolutional neural networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2403–2412, 2015.

[66] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1310–1318, 2013.

[67] G. Paxinos. *The rat nervous system*. Academic Press, 2004.

[68] George Paxinos and Charles Watson. *The rat brain in stereotaxic coordinates: hard cover edition*. Academic press, 2006.

[69] Y. Peng, Y. Wu, Y. Yang, R. Huang, C. Wu, X. Qi, Z. Liu, B. Jiang, and Y. Liu. Study on the control of biological behavior on carp induced by electrophysiological stimulation in the corpus cerebelli. In *International Conference on Electronic and Mechanical Engineering and Information Technology*, volume 1, pages 502–505, 2011.

[70] Yuxin Peng, Yunzhen Zhao, and Junchao Zhang. Two-stream collaborative learning with spatial-temporal attention for video classification. *arXiv preprint arXiv:1711.03273*, 2017.

[71] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2847–2854, 2012.

[72] 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.

[73] Xiaofeng Ren and Chunhui Gu. Figure-ground segmentation improves handled object recognition in egocentric video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 6, 2010.

[74] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1164–1172, 2015.

[75] J.N.J. Reynolds, B.I. Hyland, and J.R. Wickens. A cellular mechanism of reward-related learning. *Nature*, 413(6851):67–70, 2001.

[76] Nicholas Rhinehart and Kris M. Kitani. Learning action maps of large environments via first-person vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[77] Nicholas Rhinehart and Kris M. Kitani. First-person activity forecasting with online inverse reinforcement learning. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[78] Jason Tyler Rolfe and Yann LeCun. Discriminative recurrent sparse auto-encoders. *arXiv preprint arXiv:1301.3775*, 2013.

[79] R. Romo, A. Hernández, A. Zainos, C.D. Brody, and L. Lemus. Sensing without touching: psychophysical performance based on cortical microstimulation. *Neuron*, 26(1):273–278, 2000.

[80] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[81] H. Sato, S. Kolev, N. Goehausen, M.N. Nyi, TL Massey, P. Abbeel, and MM Maharbiz. Cyborg beetles: The remote radio control of insect flight. In *IEEE Sensors*, pages 1–4, 2010.

[82] Wolfram Schultz. Getting formal with dopamine and reward. *Neuron*, 36(2):241–263, 2002.

[83] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.

[84] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[85] Bing Shuai, Zhen Zuo, and Gang Wang. Quaddirectional 2D-recurrent neural networks for image labeling. *IEEE Signal Processing Letters*, 22(11):1990–1994, 2015.

[86] Bing Shuai, Zhen Zuo, Gang Wang, and Bing Wang. DAG-Recurrent neural networks for scene labeling. *arXiv preprint arXiv:1509.00552*, 2015.

[87] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[88] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[89] Suriya Singh, Chetan Arora, and CV Jawahar. First person action recognition using deep learned descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[90] Hyun Soo Park, Jyh-Jing Hwang, Yedong Niu, and Jianbo Shi. Egocentric future localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[91] Chao Sun, Nenggan Zheng, Xinlu Zhang, Weidong Chen, and Xiaoxiang Zheng. Automatic navigation for rat-robots with modeling of the human guidance. *Journal of Bionic Engineering*, 10(1):46–56, 2013.

[92] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014.

[93] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[94] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[95] S.K. Talwar, S. Xu, E.S. Hawley, S.A. Weiss, K.A. Moxon, and J.K. Chapin. Behavioural neuroscience: Rat navigation guided by remote control. *Nature*, 417(6884):37–38, 2002.

[96] Damien Teney and Martial Hebert. Learning to extract motion from videos in convolutional neural networks. *arXiv preprint arXiv:1601.07532*, 2016.

[97] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. IEEE, 2015.

[98] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[99] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

[100] Francesco Visin, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron Courville, and Yoshua Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.

[101] M. Wan, X. Huang, H. Ni, and B. Kong. A new rat navigation method based on CC2431. In *International Conference on Computer Research and Development*, volume 2, pages 262–265, 2011.

[102] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176, 2011.

[103] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79, 2013.

[104] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013.

[105] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[106] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision (ECCV)*, pages 20–36. Springer, 2016.

[107] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[108] Y. Wang, XC Su, RT Huai, and M. Wang. A telemetry navigation system for animal-robots. *Robot*, 28(2):183–186, 2006.

[109] Andreas Wedel, Daniel Cremers, Thomas Pock, and Horst Bischof. Structure-and motion-adaptive regularization for high accuracy optic flow. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1663–1668, 2009.

[110] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1385–1392, 2013.

[111] Elizabeth F Wells, Geula M Bernstein, Brian W Scott, Patrick J Bennett, and Julie R Mendelson. Critical flicker frequency responses in visual cortex. *Experimental brain research*, 139(1):106–110, 2001.

[112] W. Wenbo, G. Ce, S. Jiurong, and D. Zhendong. Locomotion elicited by electrical stimulation in the midbrain of the lizard gekko gecko. *Intelligent Unmanned Systems: Theory and Applications*, pages 145–153, 2009.

[113] Z. Wu, Y. Yang, B. Xia, Z. Zhang, and G. Pan. Speech interaction with a rat. *Chinese Science Bulletin*, 2014.

[114] Huijuan Xu and Kate Saenko. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision*, pages 451–466. Springer, 2016.

[115] Jia Xu, Lopamudra Mukherjee, Yin Li, Jamieson Warner, James M Rehg, and Vikas Singh. Gaze-enabled egocentric video summarization via constrained submodular maximization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2235–2244, 2015.

[116] Kedi Xu, Yi Qu, Kang Lin, Xiaoxiang Zheng, and Yueming Wang. A BMI-based flashing light recognition system on free-moving rats. In *IEEE International Conference on Information Science and Technology (ICIST)*, pages 640–643. IEEE, 2014.

[117] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81, 2015.

[118] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29, 2016.

[119] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Describing videos by exploiting temporal structure. In *IEEE International Conference on Computer Vision*, pages 4507–4515, 2015.

[120] Ting Yao, Tao Mei, and Yong Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[121] Alfred L Yarbus. *Eye movements during perception of complex objects*. Springer, 1967.

[122] Ryo Yonetani, Kris M. Kitani, and Yoichi Sato. Recognizing micro-actions and reactions from paired egocentric videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[123] Yipeng Yu, Gang Pan, Yongyue Gong, Kedi Xu, Nenggan Zheng, Weidong Hua, Xiaoxiang Zheng, and Zhaohui Wu. Intelligence-augmented rat cyborgs in maze solving. *PloS one*, 11(2):e0147754, 2016.

[124] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[125] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537, 2015.

[126] Yang Zhou, Bingbing Ni, Richang Hong, Xiaokang Yang, and Qi Tian. Cascaded interactional targeting network for egocentric video analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1904–1913, 2016.