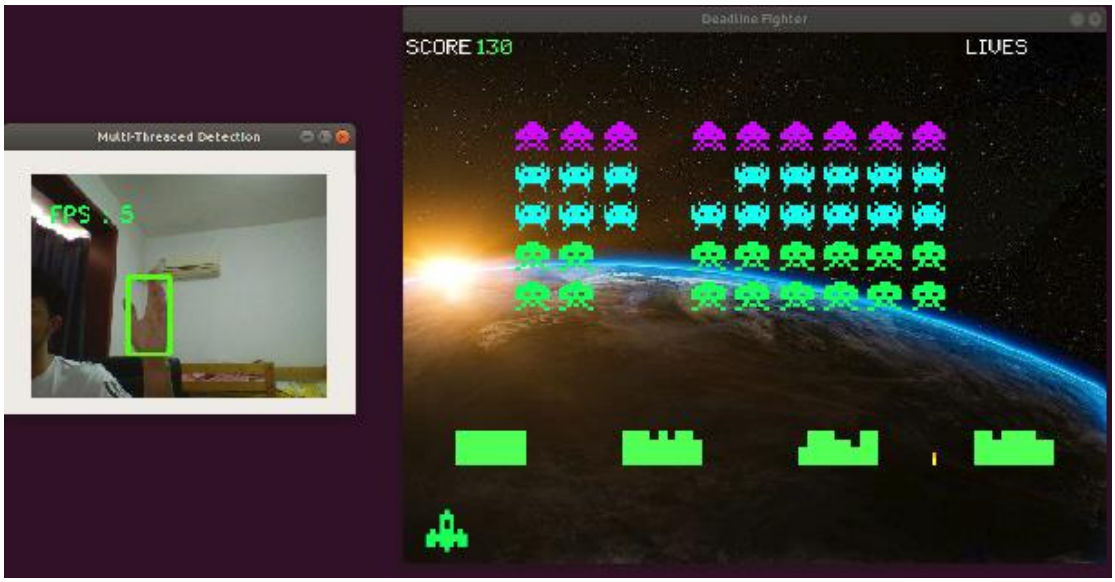
City University of Hong Kong

Department of Computer Science

CS4186 Computer Vision and Image Processing

Project Final Report

Hand (Pose) Detection and Tracking



55202829	Li Ruikang (Team Leader)
55591554	Ilias Batyrbekov (Developer)
55668691	Qu Yang (Developer)

Table of Contents

1 Introduction

1.1 Background

Hand, as one of the most flexible part of human body, is the ideal method which can be used as an input device for Human-Robot Interaction (HRI) as well as Human-Computer Interaction (HCI), for which can enhance Virtual Reality (VR) and Augmented Reality (AR) experience. The great potential market makes the hand detection and tracking becomes one of the hottest research topics nowadays. Although a more mature and stable method for detecting and tracking hand is glove-based sensing (also known as data gloves), which requires a glove-shaped device containing several sensors put on to the hand and transmit the signals (information about the hand) back to the computer. It has certain drawbacks as it may take complicated calibration and set up procedures. More importantly, the glove sensing device hampers the naturalness of interaction between humans and computers. Since the power of Computer Vision (CV) has grown in the high speed, more and more attention is drawn to the direct hand detection and tracking field for seeking more **natural and non-contact** solutions.

1.2 Problem Faced

The demand for directly using nature and non-contact method has put forward higher requirements on hand detection. The following five problems are the major concerns of using CV as a hand detection and tracking method.

1.2.1 High-dimensional Problem

To detect and track the hand, the parameters to be estimated of the hand's orientation as well as its location remains in a huge number. Since the hand is an articulated object, the natural hand need to use not less than six dimensions and may have more than 20 degrees of freedom (DOF).

1.2.2 Processing Speed Problem

Some applications using hand detection may require significant computational power. Besides, A real-time computer vision system still requires processing tons of data for even a single image sequence. Both of the requirements are highly dependent on hardware to be dedicated and expensive.

1.2.3 Self-blocking Problem

As the hand can freely rotate about 180 degrees (can easily show the front and back of the hand), the projection of the hand differs when the degree of rotation has changed, causing self-blocking of fingers, which causes the hand segmentation becomes extremely difficult. The self-blocking problem also affects the extraction of high-level features.

1.2.4 Uncontrolled Environments

As hand detection is considered to be applied in a variety of environments, which means the backgrounds together with the lighting condition is not restricted to a certain scene or value. In the computer vision field, only to extract a rigid object from an arbitrary background is still a challenging issue, to further detect the hand makes it become a more difficult problem.

1.2.5 Rapid Hand Movement

As one of the most flexible parts of the human body, the hand can move like a speed of 5 meters per second and 300° per second for wrist rotation. However, even to achieve a 30 Hz frame rate is difficult for some algorithms.

1.3 Project Interests

The above information arose our interests in hand detection and tracking topic. In this project, we not only interested in the advanced method and technologies applied in the hand detection and tracking field but also have a strong motivation to put this method into practice. Therefore, the report will cover the research result (understanding) of the state of art methods for hand detection and tracking mainly in Section 2. In Section 3, an implementation report together with an operation instruction about hand detection-based *Space Invaders* will be shown.

2 State of Art Methods Research

2.1 Overview

There are mainly two ways of research directions for using the hand as an input device. The first aims to extract high-level information from the posture of hands, which is based on the method called **Gesture Classification**. The other aiming to capture the hand's three-dimensional motion is called **Pose Estimation System**. The remaining part of this section is to briefly discuss the above mentioned two research directions.

2.2 Gesture Classification

2.2.1 Hidden Markov Models (HMMs)

The Hidden Markov Models (HMMs) is a statistical model which are popularly applied in temporal pattern recognition, which contains the field of hand (gesture) recognition. It is also popularly used for dynamic gesture classification because HMMs are known to have high classification rates. Multiple methods containing HMMs can be applied to hand detection as well, such as combining the HMMs to detect pointing gestures together with Gaussian Process Regression to estimate pointing the direction to get a better result.

2.2.2 k-Nearest Neighbors (k-NN)

k-Nearest Neighbors (k-NN) also has a high classification rate in hand detection field, and its implementation is rather simple than HMMs. Apart from pure k-NN pose classifier, a histogram-based shape descriptor can be used to enhance feature extraction as well as the Average Neighborhood Margin Maximization (ANMM) is used for reducing dimensionality after applying k-NN as preprocessing method.

2.3 Pose Estimation System

2.3.1 Partial Pose Estimation

Partial Pose Estimation systems have the ability to deal with simple, low degree of freedom (DOF) tasks, including navigation or pointing. Some researchers viewed partial pose estimation as an extension of appearance-based systems, such as the fingertips or the palm and capture their three-dimensional motion. To be more specific, the partial pose estimation requires:

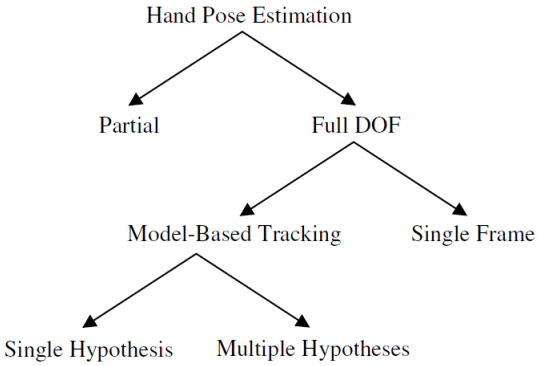


Figure 1. Different approaches to hand pose estimation

1. Hand localization
   There are two main approaches to achieve hand localization. The first one is called skin color segmentation which can be implemented within a short period. Classification-based detection can also be applied to locate the hand, in which a variety of hypotheses acted as classifiers are pre-defined and to detect the presence of the object.

2. Gesture classification
   Detailed information about gesture classification have been discussed in the above section, please refer to section 2.2.

3. Feature extraction
   The first step of the hand feature extraction is to take the landmarks on the hand's outline based on its stability and uniqueness. Then, a reliable algorithm to extract the hand feature is based on the distance between the hand position and the contour points.

2.3.2 Full Degree of Freedom (DOF) Pose Estimation

2.3.2.1 Single Frame

The meaning of single frame estimation is to use a single image or multiple images taken from different views at the same time to do hand detection.

1. Object detection
   Very similar to the hand detection method used in Section 2.3.1 hand localization.

2. Image database indexing
   Searching an enormous database of hand segment template may take a lot of time, therefore indexing techniques are applied to retrieve the nearest neighbor of the given input.

3. 2D-3D mapping
   Specialized Mapping Architecture (SMA) which is a machine learning architecture was used to map the invariant rotation and scale moments of the outline of hand.

2.3.2.2 Model-Based Tracking

Single Hypothesis

1. Optimization-based methods
   Standard optimization techniques are used to fit the model into the features being extracted. The fitting error can be minimized by using Genetic Algorithms and Nelder Mead Simplex.

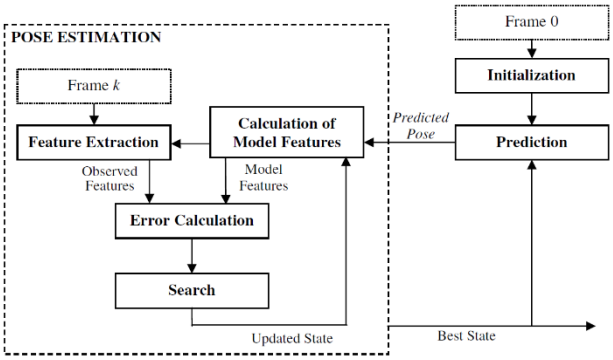


Figure 2. Model-based tracking

2. Physical force models
   Using physical force models is another approach to fitting the model to the extracted features. Solving dynamic equations derived from the kinematic model can be applied to updating the parameters inside the model. Noted that limiting the finger movement significantly enhances the robustness of hand tracking.

Multiple Hypothesis

1. Particle filters
   This technique is popularly used when using Monte Carlo simulation to implement recursive Bayesian filters. To be more specific, particle filers use easy-to-sample density's weighted samples to represents an arbitrary probability density.

2. Tree-based filter
   Grid-based filtering is an alternative approach to Bayesian filtering implementation. The distinctive feature of the Tree-based filter is that it supports the single frame pose estimation as well as initialization by traversing the tree.

3. Template database search
   The template database is generated artificially for hand tracking. Each search contains a variety of kept hypotheses together with their neighborhood using beam search technique to get the best matching or establish a new hypothesis.

3 Practice – Space Invaders Based on Hand Detection

3.1 Overview

The great motivation of putting hand detection and tracking knowledge into practice has inspired our team members to come up with a lot of creative ideas. Since the three of us are all crazy computer games' fans, we also noticed the great development of kinetic applications applied in the video games field, such as Xbox one Kinect and PlayStation Move. Therefore, we fixed our project to be a hand detection and tracking practice applied to computer games. Space Invaders is a very popular computer game ever since being released in 1978 and was chosen among a variety of games because of its simplicity (Spaceship moves along with 1D space, only has shooting action) and playability, which we thought it is a great project for beginners. The following section will cover the detailed information of the implementation procedure, the operating instructions as well as the reflection after we finished the implementation.

3.2 Implementation

3.2.1 Hand Tracking API Extraction

Hand Tracking is originally a project from GitHub where the user could use a trained TensorFlow model to track the movement of multiple hands. In this project, to get better performance, a multiple thread hand tracker is used and the number of hands is set to one.

At first, the game operation is defined that ship will follow the movement of the hand. However, it is later found out that there exists inaccuracy in the detection of movement due to the error of model. Thus, a new operation setting is that ship will move and shoot according to the location of the hand. Concrete details will be discussed in section 3.4.

Three events, ***eleft***, ***eright*** and ***eup*** which represent movement, are delivered into space invader game object for communication. Hand tracking will detect the location of the hand and set the related event to be true. Space invader game will receive the event.

3.2.2 Transplantation to Space Invaders

Space Invaders is a game where the player moves in 1-dimensional space, thus a Ship (the character that player controls) can either increase or decrease the magnitude of position in that space. Additionally, the ship can shoot the invaders, and it requires an additional event to take place. Thus, we needed to efficiently communicate the left, right, up movements of the hand.

Communication was accomplished by using the threading library of Python, namely Thread and Event. To run the game and hand detection simultaneously we initiate a "SpaceInvaders" object and pass it three python threading events, for moving left, right, and to top. The thread responsible for hand tracking sets event, when the hand is on the far right, far left, and top corner.
When running on CPU (6 physical cores) in the MacOS environment (with 4 worker threads), 16-20 frames were captured by hand tracker per second. Since the delay of setting an event is negligible, input frequency was only 16-20 Hz. Therefore, to make a better user experience frequency of gameplay (number of

times in game objects are updated) was changed from 1Hz to 0.5 Hz, thus allowing the input to be “smoother”.

Since the hand detector process has to be a parent process for starting up worker threads, the game has to be a child thread. Therefore, some functionality of the game that required accessing hardware from child thread was not activated because some operating systems (such as macOS) do not allow it. This problem could be solved by running games in a separate process and setting up Pipe for inter-process communication, but it caused significant input lag. Consequently, a multithreaded approach was chosen.

3.3 Install Guide

1. conda create --name spaceinvader
2. conda activate spaceinvader
3. conda install pygame (version 2 is recommended)
4. conda install tensorflow==1.4.0-rc0
5. cd hand_detection_spaceinvader
6. python3 spaceinvader_t1.py

We have also test Unbuntu 18.04. If you encounter camera opening problem, you may try followings instructions:

1. Add sudo before command line
2. Add QT_X11_NO_MITSHM=1 to environment

sudo vim /etc/environment
QT_X11_NO_MITSHM=1
source /etc/environment

3.4 Hand Operating Instructions (How to Play)

Once the user starts to run the code, two window prompts will appear as the below picture. The left window is a demonstration of the detected hand box in the camera. The right one is the interface of the space invader game.

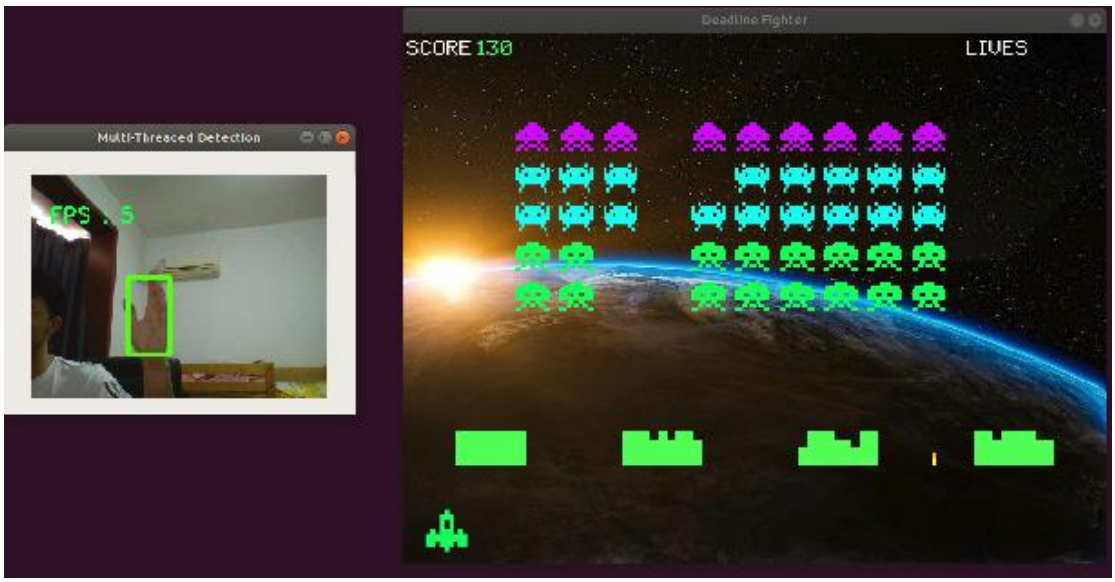


Figure 3. Demonstration of space invader based on hand detection and tracking

There are two types of operations that are set up in this special space invader.

1. Shooting
   As the commander of the spaceship, **moving your hands up to the TOP HALF of the detection box** is the command for shooting to destroy the evil space invaders!

2. Moving
   Moving your hands to the LEFT or RIGHT CORNER to control the position of the spaceship. To be more specific, if the user's hand is at the left corner, the spaceship will be moving left and vice versa.

3.5 Limitation of the Application

The hand detection model file we use is downloaded from Github. After testing, we found that there exists some systematic error

1. Although the model performs well in hand detection with still photos, it may not perform very well when using the camera. The detection quality may drop heavily in a changing light environment. For example, it performs badly in sunlight. Moreover, sometimes it mistakenly recognizes the face as hand.

2. Under the correctly recognized circumstances, the shape and location of the detected box may variate when the hand is still. It results in relative inaccuracy and error during the tracking of hand movement. For example, if the captured box moves to the right a little bit while the user does not intend to, the program may detect the movement and move the ship.

3. With the effect of first and second limitation, we have to use more frames to track the hand. Since the hand tracking part has a comparatively low fps (frame per second) around 5, it leads to a delay in space invader game. Average time delay is around 0.5 seconds

4 Reference

1. Erol, A., Bebis, G., Nicolescu, M., Boyle, R., & Twombly, X. (2007). Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding, 108*(1), 52-73.
2. Suarez, J., & Murphy, R. (2012). Hand gesture recognition with depth images: A review. *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication,* 411-417.
3. https://github.com/victordibia/handtracking
4. https://leerob.io/blog/space-invaders-with-python
5. CS4186 Lecture Notes