A study on robot tracking based on deep learning and monocular vision

Yiyang Chen^{a,b*}, Wanpei Chen^b, Lifeng Zhang^a

^aKyushu Institute of Technology, Kitakyushu, 804-0015, Japan

^bYangzhou University, Yangzhou, 225000, China

*Corresponding Author: chen.yiyang871@mail.kyutech.jp

Abstract

The human vision system can help people to perceive the distance to an object. However, for robots, it is a challenging task to use the vision system as efficiently as people because of its hardware and software constraints. Compared with the traditional way of binocular vision, using monocular vision has a simpler structure and faster operation speed. In this paper, the human recognition and distance measurement based on deep learning and monocular vision ranging algorithms are studied in details. The simulation result shows a reasonable accuracy of distance.

Keywords: monocular vision, deep learning, robots.

1. Introduction

Object tracking is an important research topic in computer vision with the rapid increase in numbers of great powered computers, high precision and cheap video cameras (4). This work target on person tracking based on the robot vision system. The Camshift algorithm is used to detect a person and the tracking works well when the circumstance lighting is enough, but in poor lighting conditions, the tracking performance was down. Therefore, we use the Deep Learning algorithm which is more accurate without using the redundant frames. Deep learning is a new research field of artificial intelligence in the learning process. It is similar to the human understanding of information through continuous abstraction of image data by finding the attributes or features (5). In tracking method, we use a monocular camera to measure distance in principle of Similar Triangle. Monocular vision can save a lot of costs and have a fast operation speed (1). In this paper, a simple construction, easy implementation and low computation cost method of monocular similar triangular

ranging are adopted which can detect a person body and tracking it at the same time.

2. Camera calibration

2.1 The purpose of camera calibration

The purpose of camera calibration is to obtain the internal and external parameters of the camera.

2.2 The preparation of camera calibration

In order to get theses coefficient, we should have the input of the image coordinates of all interior corners on the image and the spatial three-dimensional coordinates of all interior corners on the image of the calibration board. At the beginning, we prepare more than three calibration boards. Calibration pictures need to be taken in different positions, angles and postures. The calibration board needs to be a chessboard composed of black and white rectangular shapes with high accuracy.

Fig. 1. Calibration board.

2.3 The process of camera calibration

The first step is taking the photos of calibration boards in different positions, angles and postures. For each image, corner coordinates are extracted with `cvFindChessboardCorners` which only a rough coordinate, and then we call `cvFindCornerSubPix()` to get the exact corner

coordinates. When presented, it can be painted on the picture with `cvDrawChessboardCorners`. The final image corner coordinates are stored in N^2 matrix pointer `image_points`. We initialize the three-dimensional coordinates of the calibration points. It can be considered that the coordinates of the corresponding points on each graph are the same. The three-dimensional coordinates of the corner points are stored in the N^3 -dimensional matrix pointer `object_points`. The last step is to call the `cvCalibrateCamera` to calculate the internal and external parameter matrix of the camera.

3. Deep learning

3.1 The introduction of CNN

Convolutional Neural Network is one of the most important branches of deep learning, it has been widely used in detection, segmentation, object recognition and image fields. Convolutional Neural Network is a multi-layer neural network. Different from ordinary neural network, ordinary neural network consists of many neurons in each layer. Each neuron is connected to all neurons in the former layer and all neurons in the latter layer.

3.2 The process of object recognition based on SSD

Object recognition and location can be divided into two tasks. One is to find the object in the scene and give the bounding box information of the object. The other task is to identify the category of the object according to the object found. This process is usually called target detection.

The SSD method integrates the detection process into a network based on convolution network. The first part of the network uses standard networks for image classification, such as VGG, AlexNet, etc. After that, additional convolution layers are added to reduce the scale of the convolution layer feature map layer by layer. The feature maps of different convolution layers which used to predict object classification and bounding box migration, and multi-scale feature map detection is realized.

For each feature layer used for prediction, a series of convolution filters can be used to generate a series of fixed size predictions. For a feature layer with a size of $m*n$, the convolution operation of $3*3*p$ convolution kernel is used to generate a feature map with a size of P of $m*n$, which is used to predict the score of the category or coordinate offset from the default box. Because the scale of feature layer used to predict is different, SSD uses default box and

feature map cell to predict objects of different scales. As shown in Figure 2, graphs A is an image labeled with a ground truth box. Graphs B and C correspond to the scale which is $8*8$ and $4*4$. Each square is called a feature mapping unit. The dashed line frame in the graph is called a default box. Each feature mapping unit is responsible for predicting the object class and position offset of multiple default boxes. For the K default boxes at a given location, the fractions of C classes and the four position offsets relative to the real boxes need to be calculated. Therefore, for the $m*n$ feature map, there are $(c+4)*k*m*n$ outputs. SSD uses $3*3$ convolution to generate $m*n$ feature maps which number is $(c+4)*k$ to predict the default box of each feature mapping unit.

Fig. 2. SSD detection architecture.

The size of the default box directly determines the size of the object in the image. In order to make the detection result more accurate, all the detection boxes corresponding to the default box should cover the whole image. With m feature layers involved in detection, the size of default box in each feature map can be calculated by the following formula:

$$S_i = S_{\min} + \frac{S_{\max} - S_{\min}}{m-1}(i-1) \quad i \in [1, m] \quad (1)$$

Among them, $S_{\min}$ is the default frame scale at the lowest level of the feature layer to be predicted, and $S_{\max}$ is the default frame scale at the highest level. Each feature map unit has several default boxes. The width-to-height ratio is used to constrain the size of the default box. The default width-to-height ratio is $a_r = \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$, and the width and height of each default box are as follows:

$$w_i^a = s_i \sqrt{a_r} \quad (2)$$

$$h_i^a = \frac{s_i}{\sqrt{a_r}} \quad (3)$$

For the aspect ratio of 1, a scale s is added. Therefore, each feature graph unit has six prediction boxes, so that different sizes and shapes of objects can be predicted.

The number of default boxes is much larger than the number of real boxes, so we need to pair the default box with the real box to make the training labels up. SSD uses Jaccard similarity to match the default box and the real box.

In order to ensure that there is not much difference in the number of positive and negative samples, we need to set the similarity threshold in 0.5. And if the positive samples are bigger than 0.5, they are considered as the positive samples. As shown in the blue box and red box in Fig 2, the gray box in the figure is considered as negative sample. If the default box i and the j real box of the category p are matched, $x_{ij}^p = 1$, or $x_{ij}^p = 0$. According to the above matching

strategy, there must be $\sum_i x_{ij}^p > 1$. It means for the j real box, there are several default boxes matching it.

3.3 Program code and rendering

Here I mainly use target detection to detect people, so in the process of target detection, only the detected person is marked out. We changed the 'if True' in 97th line to 'if CLASSES[idx] = "person":'. So that the result of the target detection program can only show the detected person, and the effect of the detected person is much better than that of the traditional HOG+SVM method.

Fig. 3. Recognition result.

4. Ranging by monocular camera

4.1 Algorithm principle

We will use similar triangles to calculate the distance between the camera and a known object or target. Suppose that we have a target or object whose width is W . Then we put this goal at D distance from our camera. We take pictures of objects with cameras and measure their pixel width P . So we get the formula of the focal length of the camera.

$$F = (P * D) / W \quad (4)$$

For example, I put a standard 8.5 x 11 inch A4 paper (horizontal; $W = 11$) at a distance of $D = 24$ inches from the

camera and took a picture. I measured the pixel width of A4 paper in the photo to be $P = 249$ pixels. So my focal length F is:

$$F = \frac{(249px * 24in)}{11in} = 543.54 \quad (5)$$

When I continued to move my camera close to or away from objects or targets, I can use similar triangles to calculate the distance between objects and cameras.

$$D' = (W * F) / P \quad (6)$$

To be more specific, let's take another example. Suppose I move my camera three feet (or 36 inches) away from the target and take a picture of the A4 paper above. Through automatic graphics processing, I can get the A4 paper's pixel distance of 170 pixels in the picture. Substitute this into the formula to get:

$$D' = \frac{(11in * 543.45)}{170} = 35in \quad (7)$$

From the above explanation, we can see that in order to get the distance, we need to know the focal length of the camera and the size of the target object. These two known conditions are based on the formula (6). The distance D from the target to the camera is obtained, the P refers to the pixel distance, W is the width of A4 paper, F is the focal length of the camera.

Fig. 4. Detection effect of A4 paper.

In this picture, my camera is about 100cm away from the desktop. We can see that the distance in the picture is 96cm and the accuracy is still good.

4.2 Calculate the distance between man and camera

In the first part, we have calculated the distance between A4 paper and the camera. In the specific application, I need to calculate the distance between human and the camera to realize the judgment of the distance between the robot and the target. The main idea here is to detect people by deep learning, and then calculate the distance between people and cameras according to their

estimated height and camera focal length. The reason for choosing height here is that people's height changes slightly in different directions, and our camera height is fixed, so we choose height.

Table 1. Ranging results.

Times	Result	Real distance
1	270cm	273cm
2	344cm	351cm
3	412cm	419cm

5. Robot moving control

5.1 The hardware and software of mobile robot

To realize the tracking function, the motion of the object must include the real-time information and effectiveness of the motion information between the moving object, the camera, and the robot.

Mobile robot control system is ubuntu 16.04 with ROS indigo desktop. Vision camera acquisition system is Logicoool C270 web camera. The robot has 4 USB interfaces to connect with the camera and monitor. It also has a micro USB for lidar.

Fig. 5. The robot, camera and lidar.

5.2 Control the robot by sending Twist message

ROS uses geometry_msgs/Twist message type to issue motion commands. Control commands are used by base controller nodes. The full name of /cmd_level is 'command velocities'. Base controller node subscribes to /cmd_level topic and converts the motion command into motor signal through PID control algorithm.

A mobile robot operates on a two-dimensional platform (such as a floor). We only need the x-axis velocity and z-axis angular velocity in the Twist message. This is because this type of robot will only move forward and backward on the x-axis and will only rotate around the z-axis. In other words, the linear velocity of the Y and Z

axes is always zero, and the robot will not rotate on other axes.

We want the robot to move in a straight line at a speed of 0.1m/s. if we publish the topic of a Twist message on the command line of the terminal, the message section is written in the following form: '{linear: {x: 0.1, y: 0, z: 0}, angular: {x: 0, y: 0, z: 0}}'. And this Twist message will send to the robot and used by other ROS nodes.

We can get the distance by deep learning and similar triangles algorithm. So I set two critical values. One is 290, and another is 220. If the distance we have ranged bigger than 290cm, the program will send Twist message to robot to make it move forward, if the distance is smaller than 220cm, the program will control the robot to stop.

Fig. 6. Experimental screenshot.

6. Conclusions

This paper proposed a method of robot tracking based on deep learning and ranging by a monocular camera. We used a normal web camera instead of a depth camera to save the costs and accelerate the operation speed. Deep learning can detect people effectively and make the detection box very stable. A stable detection box can help us to range the distance precisely. Also this method has the demerit that we need know the height before the distance ranging. The future work is to solve this problem by improving the ranging distance algorithm.

References

- (1) Deonceaux C, and Kachi-Akkouche D : "Robot

- obstacle detection with monocular vision based on motion analysis[C]”, Intelligent Vehicles Symposium, 2004 IEEE, 527-532, 2004
- (2) Kasai T, Onoguchi K Sangyo : “Detection System for Vehicle Platooning using Multi-information Map[C]”, Intelligent Transportation Systems (ITSC), IEEE, 1350-1356, 2010
 - (3) Figat J, Kasprzak W : “NAO-mark vs QR-code Recognition by NAO Robot Vision[M]”, Progress in Automation, Robotics and Measuring Techniques, Springer International Publishing, 55-64, 2015
 - (4) Afef SALHI, Ameni YENGUI JAMMOUSSI : “Object tracking system using Camshift, Meanshift and Kalman filter”, International Journal of Electronics and Communication Engineering, Vol. 6, No. 4, 2012
 - (5) Kai Yu, Lei Jia, Yuqiang Chen : “Deep learning’s yesterday, today and tomorrow[J]”, 50(9):1799-1804, 2013
 - (6) Jain A, Kemp C C. EL-E : “an assistive mobile manipulator that autonomously fetches objects from flat surfaces[J]”, Autonomous Robots, 28(1):45-64, 2010
 - (7) Levine S, Wagener N, Abbeel P : “Learning Contact-Rich Manipulation Skills with Guided Policy Search[J]”, 156-163, 2010
 - (8) Russakovsky O, Deng J, Su H, et al : “ImageNet Large Scale Visual Recognition Challenge[J]”, International Journal of Computer Vision, 115(3):211-252, 2015
 - (9) Qing-li, HE Ke-zhong : “Research on key techniques and development of outdoor intelligent autonomous mobile robot[J]”, 22(6): 519-526, 2000
 - (10) MALM H, HEYDEN A : “A new approach to Hand-Eye calibration[A]”, Proceedings of International Conference on Pattern Recognition, IEEE, 525-529, 2000