An Embedded Vision-based Hand Tracking System on A Surgical Robot

Jiayi Chen *
jxc2077@case.edu

Haoyou Cheng †
hxc399@case.edu

Abstract

Abstract Da Vinci is a surgical system consisting of a multi-armed surgical robot and an operation station where a surgeon can move the robot's arms with controllers connected to his or her hands. Hand detection and tracking facilitates human-computer interactions and can give people more flexibility when doing work. We present an embedded vision-based hand tracking system on the Da Vinci surgical system based on a simple stereo camera. The hand tracking system can accurately and consistently detect and track hand movements across a variety of subjects. It also allows for basic hand gesture recognition. The system achieves these features through a pipeline based on SSD, a deep neural network for real-time object detection. The arms of the Da Vinci robot can move in the same manner with the hand movement tracked, in terms of direction and scaled distance. The grippers attached to the arms of the robot can open and close when a pinching gesture is detected. Our embedded hand tracking system gives surgeons more flexibility to work with the Da Vinci system as they can operate the robot without the controller or even the entire operation station. You can see all the work we have done in https://github.com/ElectricalHamster/H-DVRK/tree/main.

1 Introduction

Hand tracking has served as an important part in sign language recognition and human-computer interaction. Hand tracking is basically utilization of a tool to track hand motion and gestures in a real world environment by obtaining information from certain defined features such as points as fingertips and joints, lines as separate fingers. There are different tools to collect data of the features in hand tracking such as controllers with buttons, gloves[1], or markers[2], but a vision-based hand tracking with only cameras is simple in setups and burden-less for hands.

*Department of Electrical, Computer, and Systems Engineering, Case Western Reserve University

†Department of Electrical, Computer, and Systems Engineering, Case Western Reserve University

Recently, computer vision based object detection and gesture recognition with machine learning has been developed rapidly. Machine learning algorithms such as local orientation histogram, support vector machine (SVM) [3], and neural networks are widely used. Deep learning networks can acquire the characteristics of the target object easily and accurately from training dataset. One of the mainly used deep learning networks for object detection is Faster R-CNN [4]. Compared to Faster R-CNN, Single-Shot Multi-box Detector (SSD) is more efficient as it eliminates feature resampling and executes all computation in a single network which makes it easier to be trained and integrated into other systems[5].

The Da Vinci surgical system is a cutting-edge robot system to help with minimally invasive surgery. AS shown in Figure 1, the robot has two arms representing the left and right arm of a person. Each arm has a metal gripper attached to it. The gripper can open and close from 0 to 180 degrees. Combined with the movement of the arms and the act of the gripper, the robot can establish most hand activities needed in a surgery. However, despite its precision and versatility, there are limitations in its control setting. Surgeons currently work on manual operation devices and foot pedals, which can be cumbersome and may require them to divert attention away from the surgical field. Introducing a hand tracking system into the Da Vinci robot offers an opportunity to simplify the control interface, making it more intuitive. This intuitive means of control can lead to reduced surgical times and improved outcomes. Moreover, it will be easier for new users to operate the Da Vinci system, increasing its accessibility and usability.

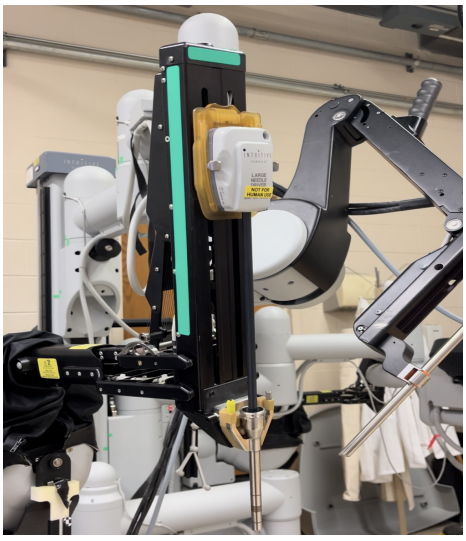

Figure 1: One Da Vinci arm with gripper

This project develops and integrates a robust hand tracking system into the

Da Vinci surgical robot. We employed Single Shot Detector (SSD) to accurately detect and track hand movement and gestures in real-time. The images for training comes from a well-known hand gestures dataset and testing is carried out in real-world Da Vinci lab. Recognition of each hand and the pinch gesture is implemented based on the position information of finger tips and wrists. All launching and kinematics of the Da Vinci robot are run by ROS-neotic, so we make system calls of launching and operation functions from our hand tracking system to control the robot. While the hand tracking system detects hand movement and recognizes the pinch gesture, the gripper attached to the robot arms closes and moves following the orientation of the hand tracking system.

2 Methods

2.1 Embedding Vision-based Hand Tracking System for DaVinci Robot

For the movement of the DVRK arm, we cannot simply transfer the arm in the format directly to the DVRK system. This causes excessive movement of the DVRK's arm. So we design an algorithm, as shown in algorithm 2.

The Embedding Vision-based Hand Tracking system we designed combined a deep learning detection model (codes on TensorFlow 1.15) and DVRK's ROS-noetic system.

A camera was setted from bottom up. When the user opens the camera and two hands appear into the frame, the hand tracking system will automatically start working. The hand tracking system has the ability of detecting 21 landmarks and the bounding box of each hand. After getting the landmarks and the bounding box, the algorithm will identify whether the current gesture is an opening gesture or a closing gesture. In the meanwhile, the coordinate of the wrist will be extracted. Then, the control side will transfer the gesture signal and the coordinate to DVRK side. The flow diagram is shown in figure 2.

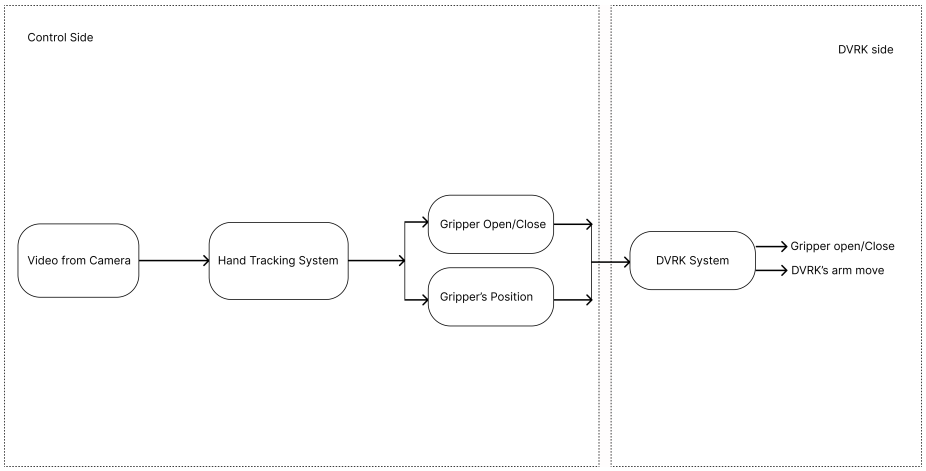

Figure 2: Whole system workflow

2.2 Hand Tracking System

In the dataset we used in this project, it contains 554,800 FullHD RGB images divided into 18 classes of gestures. Annotation for each kind of gesture contains labeled bounding boxes and 21 two-dimensional landmarks for each image.

In the preprocessing part, we resize every image to 384x384 pixels. In the meanwhile, we rescaled the values of landmarks and bounding boxes for each image because the sizes of images in this dataset are not uniform. After the processed images and corresponding landmarks correspond with bounding boxes, tfrecord file will be generated to facilitate training. The processing is shown in figure 3.

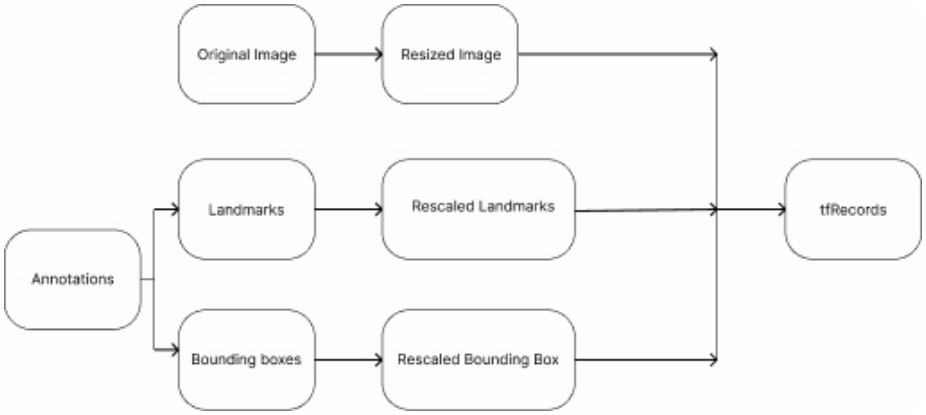

Figure 3: flow diagram of preprocessing

2.3 Hand Detection Model

In our research, we modified the traditional SSD one-stage detection method and added a new branch to detect landmarks. Due to the complexity and large size of the original SSD model, we simplified the model and added augmentation, such as rotation and flipping, during the training process to ensure the model has good generalization ability. The backbone is shown in figure 4. In the training process, we select five stages of the model and name them feature1 through 5.

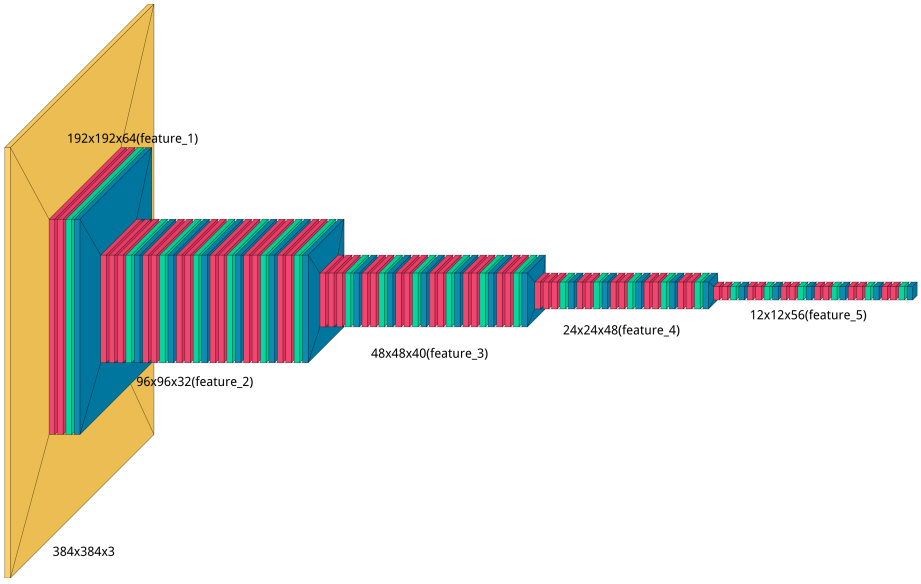

Figure 4: Backbone of detection model

After getting these five features are the input of the bounding box branch and landmark branch. The algorithm for calculating branch losses is shown in algorithm 1.

Algorithm 1 Calculate Branch Loss

procedure CALC_BRANCH_LOSS($predictions, targets, weights, w, epsilon$)
  input: predictions, targets – float tensors $[batch\_size, num\_anchor, 10]$
  input: weights – float tensors $[batch\_size, num\_anchor, 10]$
  input: $w, epsilon$ – float numbers
  if $use\_branch\_wing\_loss$ **then**
    $x \leftarrow predictions - targets$
    $c \leftarrow w \cdot (1.0 - \log(1.0 + \frac{w}{epsilon}))$
    $absolute\_x \leftarrow |x|$
    $losses \leftarrow \begin{cases} w \cdot \log(1.0 + \frac{absolute\_x}{epsilon}) & \text{if } w > absolute\_x \\ absolute\_x - c & \text{otherwise} \end{cases}$
    $losses \leftarrow losses \cdot weights$
    $loss \leftarrow \text{sum}(losses, \text{axis} = 2)$
  else
    $abs\_diff \leftarrow |predictions - targets|$
    $abs\_diff\_lt\_1 \leftarrow abs\_diff < 1.0$
    $loss \leftarrow \text{sum}(\begin{cases} 0.5 \cdot abs\_diff^2 & \text{if } abs\_diff\_lt\_1 \\ abs\_diff - 0.5 & \text{otherwise} \end{cases} \cdot weights, \text{axis} = 2)$
  end if
  return $loss$
end procedure

The branches of bounding box and landmark use MAE(Mean Absolute Error) loss function. Use this method can help us get precision of branch in current step. The precision is defined as follows:

$$\text{Precision} = (1 - \text{MAE}) = 1 - \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{1}$$

where n is the number of observations. y_i is the actual value for the i-th observation. $\hat{y}_i$ is the predicted value for the i-th observation.

In the training process, we add all loss from each branch to a dictionary. This set of loss values $\{\text{loss}_1, \text{loss}_2, \ldots, \text{loss}_n\}$ and their corresponding weights $\{\text{var}_1, \text{var}_2, \ldots, \text{var}_n\}$, the total loss L is calculated as:

$$L = \sum_{i=1}^{n} \text{Precision}_i \times \text{loss}_i + \log(1 + \text{var}_i) \tag{2}$$

where:

$$\text{Precision}_i = \frac{1.0}{\max(\min(\text{var}_i, 100.0), 0.0)} \tag{3}$$

The total loss function L is iterated continuously to make L as small as possible

2.4 Detection and Analysis of Gripper Index and Closure States

2.4.1 Gripper Closure States

The figure 5 and figure 6, we've drawn landmarks of the wrist, the tip of index finger and the tip of thumb. In the meanwhile, we also drawn the line from the tip of index finger to the wrist and the line from the tip of thumb to the wrist.

The point of the wrist can be regarded as the connection of the gripper to the DVRK's arm, and the point between the index finger and the tip of the thumb can be regarded as the end of the two ends of the gripper.

It's easy to find that there is an angle between the two lines. It is worth noting that when the user shows the open and closed gestures, the Angle between the two is different.

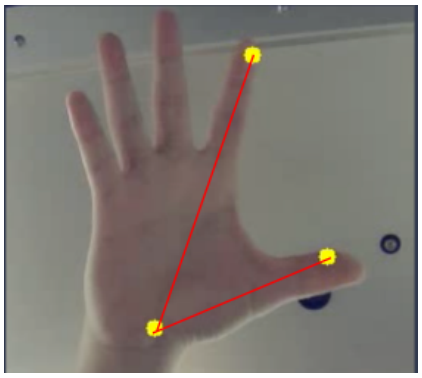

Figure 5: Hand Opening Gesture

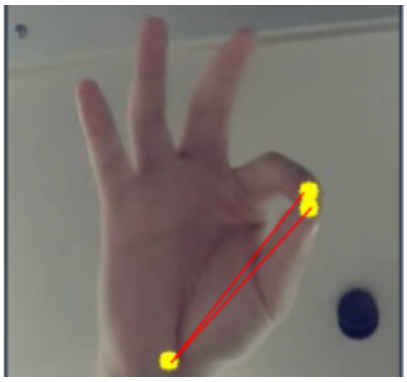

Figure 6: Hand Closing Gesture

Cause we can easily get the positions of the wrist (wrist_position), thumb tip (thumb_tip), and index tip (index_tip), we define the vectors from the wrist to the thumb and index finger as follows:

$$Vector_{\text{wrist\_to\_index}} = wrist_{\text{position}} - thumb_{\text{tip}} \tag{4}$$

$$Vector_{\text{wrist\_to\_index}} = wrist_{\text{position}} - index_{\text{tip}} \tag{5}$$

The dot product of these two vectors is :

$$Dot\ Result = Vector_{\text{wrist\_to\_index}} \cdot Vector_{\text{wrist\_to\_index}} \tag{6}$$

The norms of each vector are calculated as:

$$\|\text{wrist\_to\_index}\| = \sqrt{\text{wrist\_to\_index} \cdot \text{wrist\_to\_index}} \tag{7}$$

$$\|\text{wrist\_to\_index}\| = \sqrt{\text{wrist\_to\_index} \cdot \text{wrist\_to\_index}} \tag{8}$$

The cosine of the angle between these two vectors is as follow:

$$\cos(\theta) = \frac{\text{Dot Result}}{\|\text{wrist\_to\_index}\| \times \|\text{wrist\_to\_index}\|} \tag{9}$$

The angle θ in radians is then computed using the arccosine function:

$$\theta = \arccos(\cos(\theta)) \tag{10}$$

And converted to degrees:

$$\text{angle\_in\_degree} = \frac{180}{\pi} \times \theta \tag{11}$$

Finally, we determine if the thumb and index finger are considered closed based on a threshold angle:

$$\text{is\_closed} = \text{angle\_in\_degree} \leq \text{threshold} \tag{12}$$

2.4.2 Gripper Closure States

We judge the left and right hands by the direction of the vector from the tip of the thumb to the wrist. The formula is defined as follows:

$$Hand\_Side = Position_{(\text{tip of thumb})} - Position_{(\text{wrist})} \tag{13}$$

In other words, if the value of the vector is positive, it means that the hand is left. Otherwise, the hand in the current bounding box is the right hand.

2.5 DVRK's Arm Movement

For the detection model, we can get the position of the point from the adjacent frame, and it is easy to calculate the displacement of the corresponding point in the two adjacent frames. Figure 7 and Figure 8 are two adjacent frames. The current distance can be obtained by directly making the difference between the positions of points detected by the model.

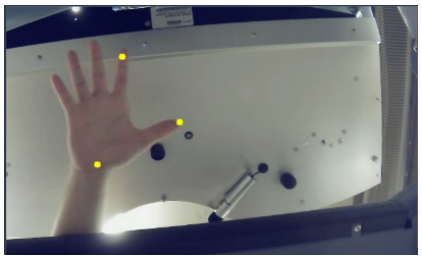

Figure 7: Last Position of Hand in Last Frame

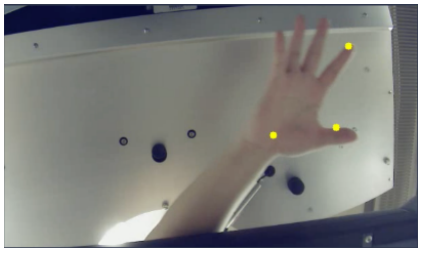

Figure 8: Last Position of Hand in Current Frame

For the movement of the DVRK arm, we cannot simply transfer the arm in the format directly to the DVRK system. This causes excessive movement of the DVRK's arm. So we design an algorithm, as shown in algorithm 2.

Algorithm 2 Moving Da Vinci Robotic Arm Based on Hand Detection

Initialize:
Get the initial position of the DVRK arm
Define *frame_size* as $(frame\_width, frame\_height)$
Detect hand initially to establish a starting reference for the hand
while true **do** ▷ Continuous tracking and movement loop
    Calculate distance between the wrist's position in the last frame and the current frame
    Scale the measured distance to match the real-world dimensions of the DVRK arm's operating environment
    Update the DVRK arm's position by transferring the new scaled position
    if exit_condition **then** ▷ Check for a condition to exit the loop
        break
    end if
end while

Algorithm 2 can perfectly solve the problem of DVRK's arm movement. However, it is worth noting that in the scale distance, the coefficient of scale should be very careful (try to set the initial value of this coefficient to be large enough).

3 Results and Discussion

According to our training model, for bounding box, mAP is 87.5%. This result is not very high, but it is good enough for our experiment. For landmark, the MAE value is around 0.07. Throughout the model, landmark's accuracy rate is quite good. The rendering of hand tracking system is shown in figure 9. Through the method shown in Figure 9, we can easily get the landmarks required by the DVRK system. It's shown in figure 10.

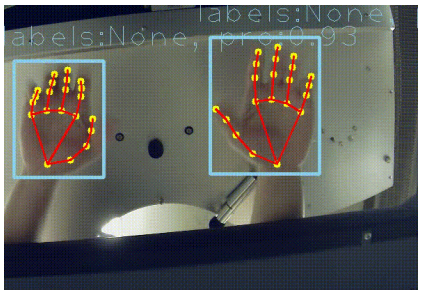

Figure 9: Whole landmarks detection for two hands

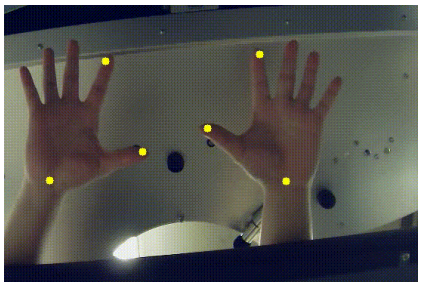

Figure 10: Required landmarks detection for two hands

After that, it is only necessary to process these data through the above algorithm 1 and algorithm 2, and then transmit it to the DVRK system. Videos of the code running successfully in DVRK system can be seen on the project's GitHub home page.

4 Conclusion

In conclusion, our project has presented a feasible embedded vision-based hand tracking system on the Da Vinci surgical robot. We have successfully developed a hand tracking system that can consistently and accurately detects hand movements and recognize certain hand gestures in real-time with a single stereo camera. We have also integrated the hand tracking system into the Da Vinci robot, whose arms and attached grippers can establish the same move with the detected hand activity.

We have shown that our hand tracking system can detect both hands and correctly identify the left and right hand with high accuracy. Recognition of the pinching gesture is also achieved in a high rate. This solidifies the robustness of our hand tracking system and its effectiveness for use with the Da Vinci robot. Moreover, the integration of our hand tracking system into the Da Vinci surgical system ensures compatibility and safety with proper criterion for launching and moving of the robot.

There are several aspects of our system need further actions. The depth information of the images taken from the current stereo camera is very unstable. We were not able to utilize that to properly track hand movements in the z direction. Calibration or refinement of the stereo camera will realize hand tracking in the z direction with our approach. Additionally, while real-time hand tracking and gesture recognition is achieved with our system, there is a delay after integration of the system to the DA Vinci robot. It is mainly because of the reaction time of the computer to take information from our hand tracking time and input it to the robot system. That needs a large-scale refinement of both software and hardware.

Overall, our project establishes the application of human-robot interaction.

By integration of a computer vision based hand tracking system into the robot control setting, we aim to provide a more accessible and efficient system for high quality surgeries.

References

[1] Dipietro, L., Sabatini, A. M., and Dario, P. A survey of glove-based systems and their applications. *IEEE Trans. Sys., Man, and Cybernetics C*, 38(4): 461–482, 2008.

[2] Zhao, W., Chai, J., and Xu, Y.-Q. Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data. In *Proc. Eurographics Symposium on Computer Animation*, 2012, pp. 33–42.

[3] Hsieh, C.-C., and Liou, D.-H. Novel Haar features for realtime hand gesture recognition using SVM. *Journal of RealTime Image Processing*, 10(2): 357–370, 2012.

[4] Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6): 1137–1149, 2017.

[5] Liu, W., Anguelov, D., Erhan, D., Szegedy, S., Fu, C.-Y., and Berg, A. C. SSD: single shot multibox detector. In *Computer Vision—ECCV 2016*, vol. 21, pp. 21–37, 2016.