ADIP 2018 Fall Project Foreground Extraction- Group 4

Chang-Qi Zhang[a], Chen-Hung Hu[b] and Chien-Yu Lin[c]
Authors Student ID: 106368002 laboratory: lab107-1 advisor: Lih-Jen Kau[a]
Authors Student ID: 107368001 laboratory: lab107-1 advisor: Lih-Jen Kau[b]
Authors Student ID: 107368003 laboratory: lab107-1 advisor: Lih-Jen Kau[c]

***Abstract*— The foreground extraction methods from previous group. They either use a heavy computation method, Deeplab or manually assign region to achieve human based foreground extraction. In this final project we implement a semi-automatic foreground extraction software which can automatically generate foreground keypoints and provide UI for user select background keypoints.**

I. INTRODUCTION

Traditional image segmentation algorisms, for example, watershed and grabcut. The difference between this two method is watershed initial by assign both foreground and background keypoints, then based on those points to do gradient search, and grabcut initial by giving foreground rectangle region then use Gaussian mixture model, GMM to find foreground region. Dou to the watershed algorism require to assign foreground keypoints on the object which you want to extract, we have to integrate object detection method to get the region of the foreground object. Common object detection, YOLO only give a bounding box of objects. Those bounding boxes are usually not fit perfectly with the foreground that we expect. Therefore, we propose using Openpose to get accurate foreground keypoints on people. In this way, the runtime for our program will be much shorter than using machine learning segmentation technology, DeepLab, Mask RCNN, and SegNet.

II. PROCEDURE METHOD

Fig 1 shows the system structure for our semi-foreground segmentation method. Our system can automatically define the foreground keypoints when user click on openpose button. To achieve the function, we based on OpenPose[1] For background keypoints, user need to use our user interface to select the background keypoints. Once system get both OpenPose foreground keypoints and user define background then we apply Foreground/Background segmentation to extract foreground form the input image.

A. *Foreground/Background Segmenter*

B. *Openpose Foreground Keypoints Generator*

III. SOFTWARE ARCHITECTURE

Figure 3 shows the software structure for our program. All program use QT as main thread, it is able to do file control. In manual mode, it can read the mouse event for user define foreground and background keypoints. With intelligence mode, our program will active Openpose to get human body foreground keypoints.

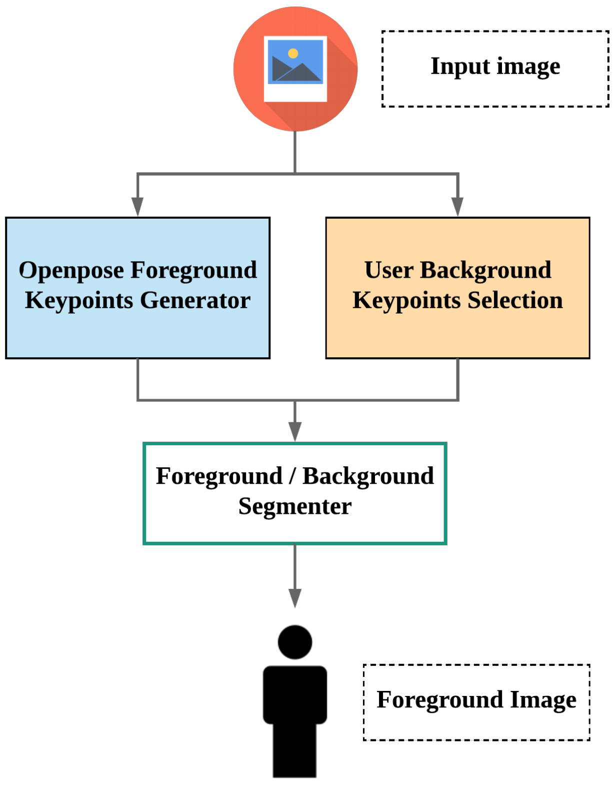
Input image
Openpose Foreground Keypoints Generator
User Background Keypoints Selection
Foreground / Background Segmenter
Foreground Image

Fig. 1: System structure

IV. RESULTS AND COMPARE

For evaluation the testing results, we used Dell Precision 5520 laptop with 8 cores Intel Xeon CPU E3-1505M v6 @ 3.00GHz, Quadro M1200 GPU and 32GB RAM. Running on Ubuntu 16.04 operating system.

REFERENCES

[1] Zhe Cao and Gines Hidalgo and Tomas Simon and Shih-En Wei and Yaser Sheikh, OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields
[2] Shih-En Wei, Varun Ramakrishna, Takeo Kanade and Yaser Sheikh, Convolutional Pose Machines

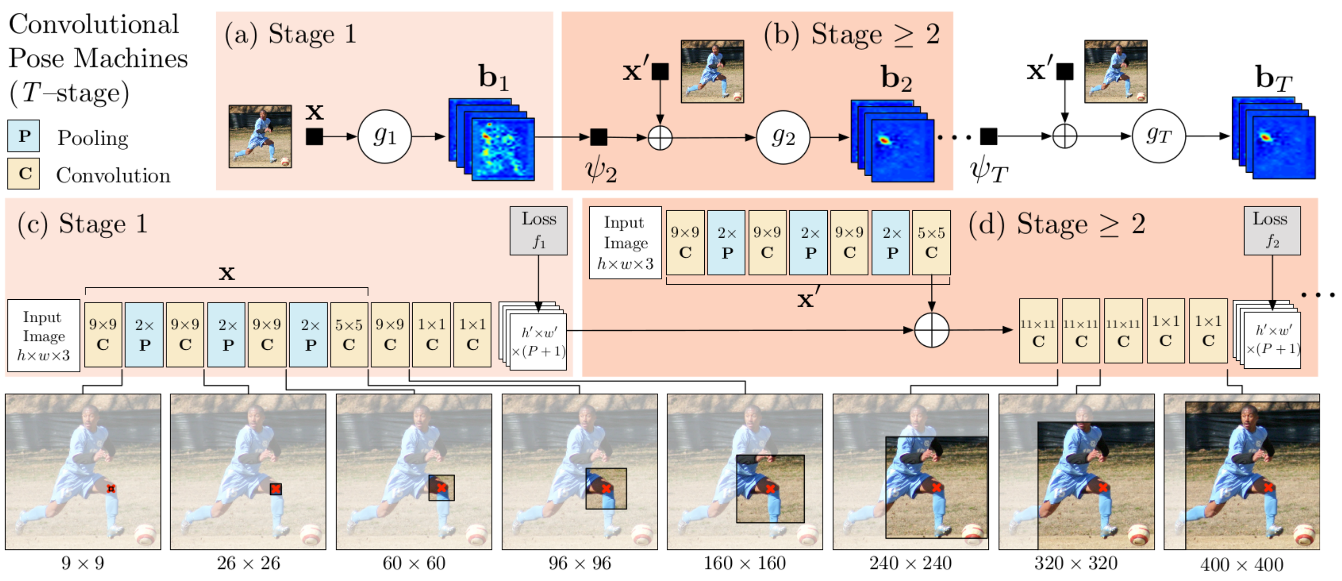

Fig. 2: Openpose network architecture[2]

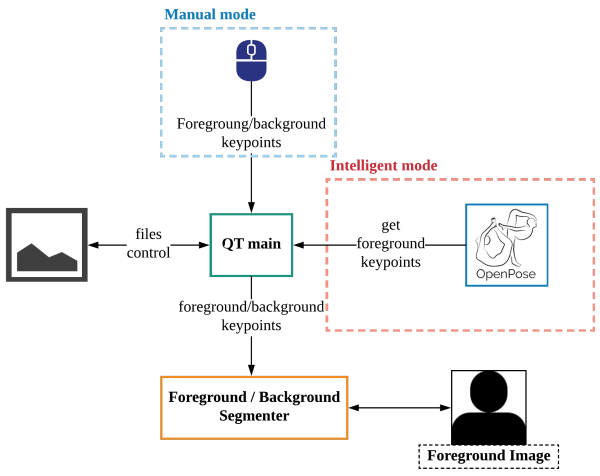
Manual mode
Foregroung/background keypoints
Intelligent mode
files control
QT main
get foreground keypoints
OpenPose
foreground/background keypoints
Foreground / Background Segmenter
Foreground Image

Fig. 3: Software structure

IMG	Precision	Recall	MAE	Runtime(ms)
1	0.9151	0.8036	0.1287	4415
2	0.9346	0.5528	0.1010	1764
3	0.9845	0.9980	0.0091	1329
4	0.9832	0.9582	0.0246	1622
5	0.9705	0.9944	0.0186	1260
6	0.8739	0.9999	0.0598	1123
7	0.8231	0.7952	0.1654	1688
8	0.8125	0.9391	0.1315	1485
9	0.5943	0.9994	0.0940	1674
10	0.9031	0.9133	0.0899	1376
11	0.9326	0.8997	0.0389	1542
12	0.9308	0.8498	0.0469	1783
13	0.9597	0.9591	0.0183	1527
14	0.6748	0.9508	0.1202	1666
15	0.7340	0.7223	0.1123	2098
16	0.9710	0.6350	0.1504	2121
17	0.9900	0.9999	0.0077	1775
18	0.9595	0.9148	0.0214	1605
19	0.9220	0.7015	0.0449	1703
20	0.9212	0.9865	0.0030	1632
21	0.9027	0.9627	0.0486	2398
22	0.9229	0.8287	0.1099	1779
23	0.9569	0.9937	0.0146	1631
Avg.	0.8944	0.8851	0.0678	1782

TABLE I: Testing result with test dataset.