Human Segmentation with NVIDIA-TX1

[1] Chun-Hsien Yu (余俊賢) [2] Tang Lee (李唐) [3] Chun-Wei Chen (陳俊偉)

[4] Meng-Ru Hsieh (謝孟儒) [5] Chiou-Shann Fuh (傅楸善)

[1, 2] Dept. of Electrical Engineering,
[3, 4, 5] Dept. of Computer Science and Information Engineering,
National Taiwan University, Taiwan
E-mail: [1] r04921119@ntu.edu.tw [5] fuh@csie.ntu.edu.tw

Abstract

The segmentation in color image is wide used in serval application. Many different methods for distinguish skin pixel from rgb world. The simplest, and often applied, method is called an "explicit skin cluster" classifier which expressly defines the boundaries of the skin cluster in certain color spaces. Though pixel-based segment like this may always result in fragment pieces and cannot easily segment a human body.

Moreover, another purpose is to implement segmentation on embedded systems such as board or digital camera. Here we pick NVIDIA-TX1 as our platform.

Keywords *Segmentation; explicit skin cluster classifier; skin model; embedded system; FCN;*

1. Introduction

There are many different methods to do segmentation of human body. The pixel-based segment is to judge whether the color range is in the skin pixel color or not. And there are exist serval method for that: parametric skin model and non-parametric skin model. The more common used is a method called "explicit skin cluster [2]." There are about 6 implementations of this: YCbCr [4], RGB [5], HSV1 [6], HSV2 [7], HSI [8] and rgb [9]. However, the output image result in fragile pieces and cannot easily distinguish the human itself.

With the deep learning model Fully Convolutional Networks, FCN [3], we can eventually approach this goal of object detection [18, 19, 20] and move on to the embedded system.

2. Method

There are three kinds of method that can detect skin color: parametric skin model, non-parametric skin model and explicit skin cluster.

2.1 Parametric Skin Model

Parametric model is created by training lots of skin data and the most used methods are Gaussian Model, GM and Gaussian Mixture Model, GMM [1]. Below is the introduction.

2.1.1 Gaussian Model

The skin distribution can be seen as normal distribution so it can be approximated by a single Gaussian to get the skin probability. Equ. 1 shows the single Gaussian probability density function. Here c is the skin probability we want to get, μ is the average and Σ is the covariance matrix. Both μ and Σ are get through training.

$$p\left(\frac{c}{skin}\right) = \frac{\pi}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp\left[-\frac{1}{2}(c-\mu)^T\Sigma^{-1}(c-\mu)\right] \quad (1)$$

2.1.2 Gaussian Mixture Model

To get more accurate distribution of skin color, Gaussian Mixture Model can approximate almost any kind of PDF by using serval GM to fit all sorts of skin distributions. Equ. 2 is the Gaussian Mixture Model we just mentioned and M stands for the number of GM, π means the weight of the ith GM and $\sum \pi = 1$.

$$p\left(\frac{c}{skin}\right) = \sum_{i=1}^{M} \frac{\pi}{(2\pi)^{\frac{n}{2}}|\Sigma_i|^{\frac{1}{2}}} exp\left[-\frac{1}{2}(c-\mu_i)^T\Sigma_i^{-1}(c-\mu_i)\right] \quad (2)$$

We know that the average and covariance modify the Gaussian Model and the other parameter (π, μ, Σ) can be calculated by Expectation Maximization (EM).

2.2 Non-parametric Skin Model

Non-parametric skin model use lots of skin data to detect the skin probability and the most common used method is Bayes classifier, Normalized Lookup Table (NLUT), Self-Organizing Map (SOM), and etc. We now introduce the common one: Bayes classifier.

2.2.1 Bayes Classifier

Before we undergo the process of Bayes Classifier, we need to collect the data of skin and non-skin to build the histograms and use all these information and Bayes decision rule detect skin color. The Bayes Rule is shown as Equ. 3.

$$p(skin|c) = \frac{P(c|skin)P(skin)}{P(c|skin)P(skin) + P(c|nonskin)P(nonskin)} \quad (3)$$

where *c* stands for the probability of skin color we want to get which also call posteriori probability. And to distinguish skin color and non-skin color probability can absorb through the histogram we have before. In the end we set a threshold to divide whether it is skin color or not.

2.3 Explicit Skin Cluster

It defines the boundaries of skin cluster in certain color space. It proposes a set of fixed skin thresholds in a given color space. Some color spaces permit searching skin color pixels in the 2D chromatic space, reducing dependence on lighting variation, others, such as the RGB space, address the lighting problem by introducing different rules depending on illumination conditions (uniform daylight, or flash).

Working within different color spaces, there are implemented the six different algorithms named for the color space adopted: YCbCr, RGB, HSV1, HSV2, HSI and rgb.

2.3.1 YCbCr

Chai and Ngan [4] develop an algorithm that exploits the spatial distribution characteristics of human skin color. A skin color map is derived and used on the chrominance components of the input image to detect pixels that appear to be skin. Working in the YCbCr space the authors find that the ranges of Cb and Cr most representative for the skin-color reference map were:

$$77 \leq Cb \leq 127 \text{ and } 133 \leq Cr \leq 173$$

2.3.2 RGB

Kovac et al. [5] work within the RGB color space and deal with the illumination conditions under which the image is captured. Therefore, they classify skin color by heuristic rules that take into account two different conditions: uniform daylight and flash or lateral illumination.

- Uniform daylight illumination

$$R > 95, G > 40, B > 20$$
$$Max\{R, G, B\} - \min\{R, G, B\} < 15$$
$$|R - G| > 15, R > G, R > B$$

- Flashlight or daylight lateral illumination

$$R > 220, G > 210, B > 170$$
$$|R - G| \leq 15, B < R, B < G$$

2.3.3 HSV1

Tsekeridou and Pitas [6] work within the HSV color space and select pixels having skin-like colors by setting the following thresholds:

$$V \geq 40$$
$$0.2 < S < 0.6$$
$$0° < H < 25° \ or \ 355° < H < 360°$$

The selected range of *H* restricts segmentation to reddish colors and the saturation range selected ensures the exclusion of pure red and very dark red colors, both of which are caused by small variations in lighting conditions. The threshold on *V* is introduced to discard dark colors.

2.3.4 HSV2

Starting from a training data set composed of skin color samples, Garcia and Tiziritas [7] compute the color histogram in Hue-Saturation-Value (HSV) color space, and estimate the shape of this skin color subspace. They find a set of planes by successive adjustments depending on segmentation results, recording the equations shown below which define the six bounding planes found in the HSV color space case, where $H \in [-180° \ 180°]$

$$V \geq 40$$
$$H \leq (-0.4V + 75)$$
$$10 \leq S \leq (-H - 0.1V + 110)$$
$$if \ H \geq 0: \quad S \leq (0.08(100 - V)H + 0.5V)$$
$$if \ H < 0: \quad S \leq (0.5H + 35)$$

2.3.5 HSI

Hsieh et al. [8] use the HSI color space system to design their color classification algorithm because it is stable for skin color under different lighting conditions. These rules apply to the intensity *I*, hue *H* and saturation *S*, and are detailed as follows:

$$I > 40$$
$$if \ 13 < S < 110:$$
$$0° < H < 28° \ and \ 332° < H < 360°$$

$$if\ 13 < S < 75: \quad 309° < H < 331°$$

The thresholds are empirically determined from the training set and the color system transformation from RGB to HSI is defined as follows:

$$I_1 = \frac{1}{3}(R+G+B);\ I_2 = \frac{1}{2}(R-B);$$
$$I_3 = \frac{1}{4}(2G-R-B)$$
$$I = I_1$$
$$S = \sqrt{I_2^2 + I_3^2};\ H = tan^{-1}\left(\frac{I_3}{I_2}\right)$$

2.3.6 rgb

Gomez and Morales [9] use a constructive induction approach to determine the skin map. Starting with the three rgb components in a normalized form and a simple set of arithmetic operators, the authors produce a model for skin detection. The algorithm uses a Restricted Covering Algorithm (RCA) as its selective learner. The RCA searches for single rules in parallel. Among the different combination rules presented by the authors, we have chosen the one with the highest precision and success rate:

$$\frac{r}{g} > 1.185$$
$$\frac{r \cdot b}{(r+g+b)^2} > 0.107\ and \frac{r \cdot g}{(r+g+b)^2} > 0.112$$

where rgb are the normalized coordinates obtained as:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}$$

The other segment method with deep learning called Fully Convolutional Network, FCN, is a convolutional network that can efficiently learn to make dense predictions [14, 15] for per-pixel tasks like semantic segmentation [22, 23, 24, 25, 26, 27, 28] and local correspondence [21, 14]. It trained end-to-end for pixel-wise prediction and supervised pre-training. This method is efficient, both asymptotically and absolutely, and precludes the need for the complications in other works.

3. Fully Convolutional Network

Fully convolutional network (FCN) is a variant of convolutional networks, that can trained end-to-end, pixels-to-pixels, to produce correspondingly-sized output image of semantic segmentation of an arbitrarily sized input image. To our knowledge, the idea of extending a convnet to arbitrary-sized inputs first appeared in Matan et al. [29], which extended the classic LeNet [30] to recognize strings of digits. We adapt contemporary classification [10, 11, 12] networks (AlexNet [10], the VGG net [11], and GoogLeNet [12]) into fully convolutional networks and transfer their learned representations by fine-tuning [13] to the segmentation task.

3.1 Architecture

FCN is basically from a standard convolutional neural network (CNN), replacing all the several fully-connected layers (3 for AlexNet) with convolution layers. Thus, it's called fully convolutional network. While a standard CNN only take fixed-sized images as inputs, the fact that we learn filters in FCN instead of just nonlinear functions as in CNN makes it able to take arbitrary-sized images as inputs.

After the last convolutional layer, there is a deconvolution layer. A deconvolution layer can map course output to dense pixels, which is what we want, the heat map.

The scores are upsampled to the input dimensions by deconvolution layers within the net. Final layer deconvolutional filters are fixed to bilinear interpolation, while intermediate upsampling layers are initialized to bilinear upsampling, and then learned.

All the model are trained and tested with Caffe [16], a deep learning framework made with expression, speed, and modularity in mind.

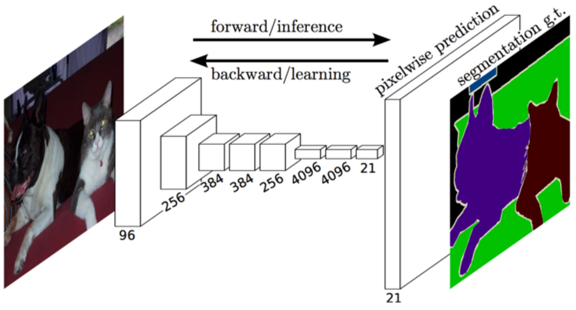


Figure: Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

3.2 Variants

FCN can be converted from different basic CNNs. 3 popular variants were tested as in the table below. Mean IU is the evaluation method of semantic segmentation. We can see that FCN-VGG16 performs best, with the longest forward time. Since our target platform is an embedded system with limited computational and memory resource, we chose FCN-AlexNet, the fastest variant.

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

Table 1: The comparison information for FCN.

4. Platform TX1

We choose NVIDIA TX1 as our platform for it contain GPU to undergo the process. It is a board that just released no more than a year. With its powerful functioning, it is now widely used in computer vision and deep-learning process. This is the reason why we discard other board.

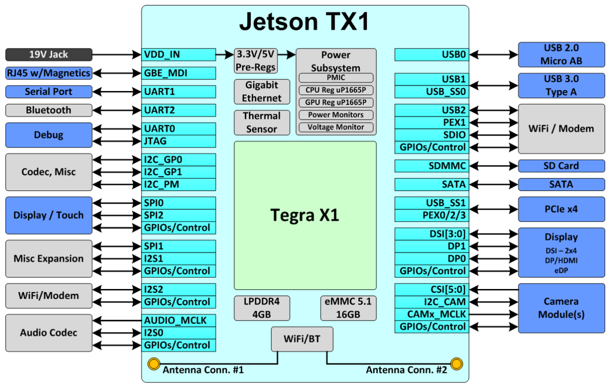
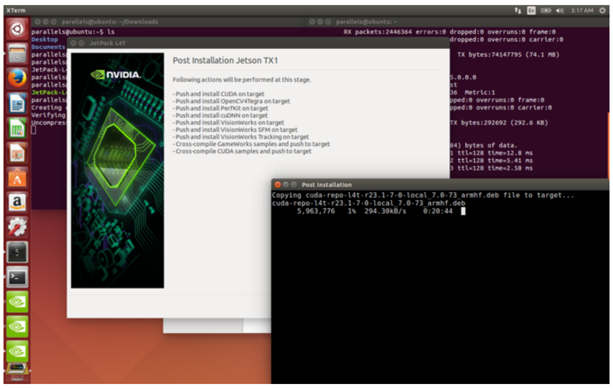


Figure: NVIDIA Jetson TX1

NVIDIA set up a social community site for developers like us to go to. The camera embedded do not fit the current OpenCV for the API isn't opened in previous version. So to turn on the camera we have to deal with the GStreamer and newer CUDA and of course we first need to upgrade the system.



```
parallels@ubuntu:~$ ls
Desktop
Documents
parallels
parallels
JetPack-L
parallels
parallels
JetPack-L
parallels
Creating
Verifying
Uncompres
```

```
RX packets:2446364 errors:0 dropped:0 overruns:0 frame:0
dropped:0 overruns:0 carrier:0

TX bytes:74147795 (74.1 MB)

5.0.0.0
st
36  Metric:1
pped:0 overruns:0 frame:0
pped:0 overruns:0 carrier:0

TX bytes:292692 (292.6 KB)

84) bytes of data.
1 ttl=128 time=12.8 ms
2 ttl=128 time=5.41 ms
3 ttl=128 time=2.58 ms
```

Post Installation Jetson TX1

Following actions will be performed at this stage.

- Push and install CUDA on target
- Push and install OpenCV4Tegra on target
- Push and install PerfKit on target
- Push and install cuDNN on target
- Push and install VisionWorks on target
- Push and install VisionWorks SFM on target
- Push and install VisionWorks Tracking on target
- Cross-compile GameWorks samples and push to target
- Cross-compile CUDA samples and push to target

Post Installation

```
Copying cuda-repo-l4t-r23.1-7-0-local_7.0-73_armhf.deb file to target...
cuda-repo-l4t-r23.1-7-0-local_7.0-73_armhf.deb
      5,963,776   1%  294.30kB/s    0:20:44
```

Figure: The udating process

The current version contains a complete GPU system and suitable CUDA but without accessing the hard drive we still cannot use the camera with OpenCV. That's why we still have to deal with GStreamer. GStreamer is a library for constructing graphs of media-handling components. The applications it supports range from simple Ogg/Vorbis playback, audio/video streaming to complex audio (mixing) and video (non-linear editing) processing.

5. Experiment

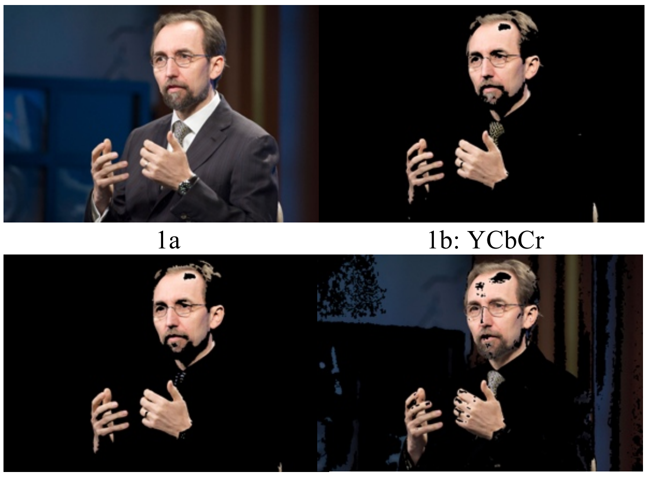


Figure 1. Experiment of skin color segmentation base on explicit cluster. 1a: origin image; 1b: YCbCr; 1c: RGB; 1d: HSV1. Note that YCbCr seem be more recall oriented, other more balanced in precision and recall oriented like RGB and HSV.

While implementing on embedding system, the problem we first encountered is the hardware problem. Though TX1 embed a GPU inside, the computation is still slow. It takes 5 sec for just a forward an image. Still, from the paper we knew that using VGG net will perform the best but the memory of GPU of TX1 is too small to store so we can just use the secondary one: AlexNet.

	FCN-AlexNet	FCN-VGG16
Mean IU	39.8	56.0
Forward time	50ms	210ms
Conv layers	8	16
Parameters	228M	539M
Capable with TX1	O	X

Table 2: Our own comparison on TX1.

In the beginning of using AlexNet, the final layer is always 0. At first, we thought that it might be the computing process too slow to meet the scoring layer so we try to sleep down the scoring process. However, it doesn't work until we figure out the pretrained AlexNet model with that last 2 layer are all 0 in parameter. So, we managed to run the FCN after training AlexNet on our own.

While the TX1 is still under development, there are some hardware cannot match up with the guilding. There is a community for developer to share. We got pretty much help for there.

Also we make some adjustment to get close to real-time process. While we now know the memory is precious to TX1, we move the pretraining work to the server and TX1 only focus on forwarding. And more we scale down the image for the input image and then scale back to build the heat map. With confidence, if it works fine the segment won't be too blurring or messy.

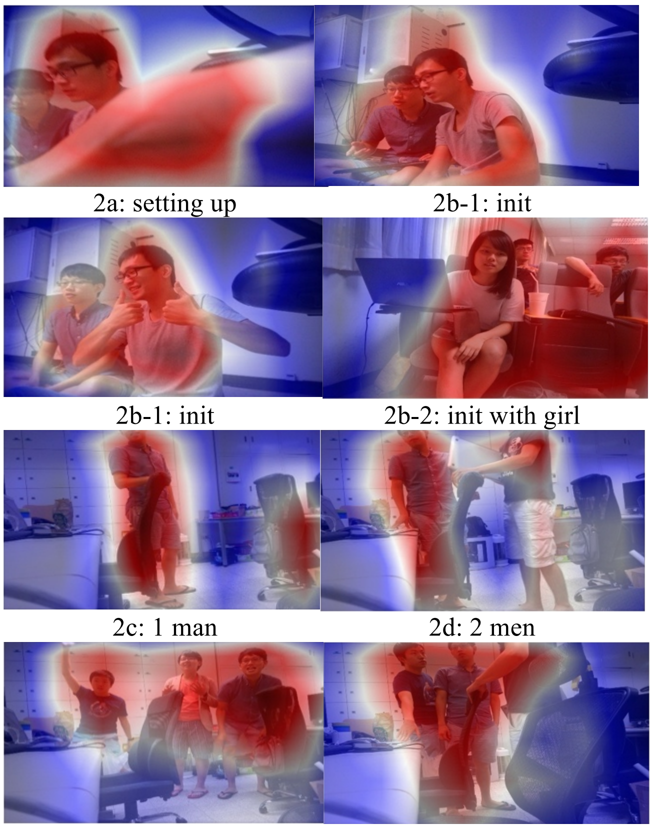

2a: setting up 2b-1: init
2b-1: init 2b-2: init with girl
2c: 1 man 2d: 2 men
2e-1: more men 2e-2: more men

Figure 2. Show the heat map of human segmentation. 2a-b: setting up the camera and tuning; 2c-e: showing performance of increasing people in the scene. Because that heat map predict the scale down image so when it scale back the range seems not that accurate.

As Figure 2 shows, the human segmentation scores out every part of the image and it tells the probability of them not just a binary yes, no result, so the output remain complete not fragment.

FCN model doesn't need to see the whole human body. You can only show it partial limb and the result will contain this, Figure 3.

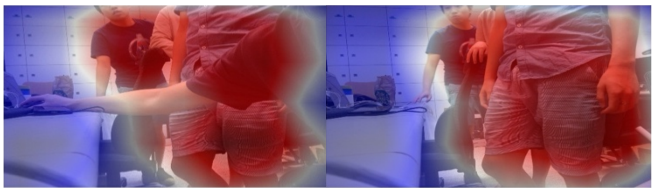

3a: hand 3b: leg

Figure 3. Show the partial limbs to predict. 3a: only show arms; 3b: show legs to predict. Although the result is not that precisely it still tells a lot of information.

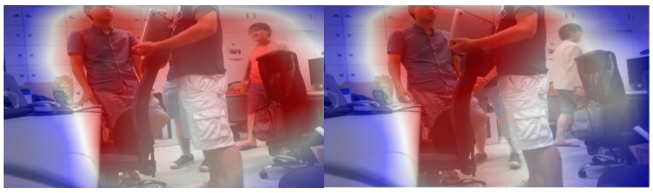

4a: man with face 4b: man without face

Figure 4. Showing the difference of poses. 4a: the man with turning face has higher score; 4b: the man without face has lower score but still predict success.

The reason why Figure 3 and 4 can show the prediction of human body is that the pretrained data no matter VGG net, AlexNet or GooLeNet they all track out many human feature before. That is the reason why the original pixel-based, judging if the color location in skin, is now not so popular. After learning from those features FCN can sure predict almost everything if we have enough data.

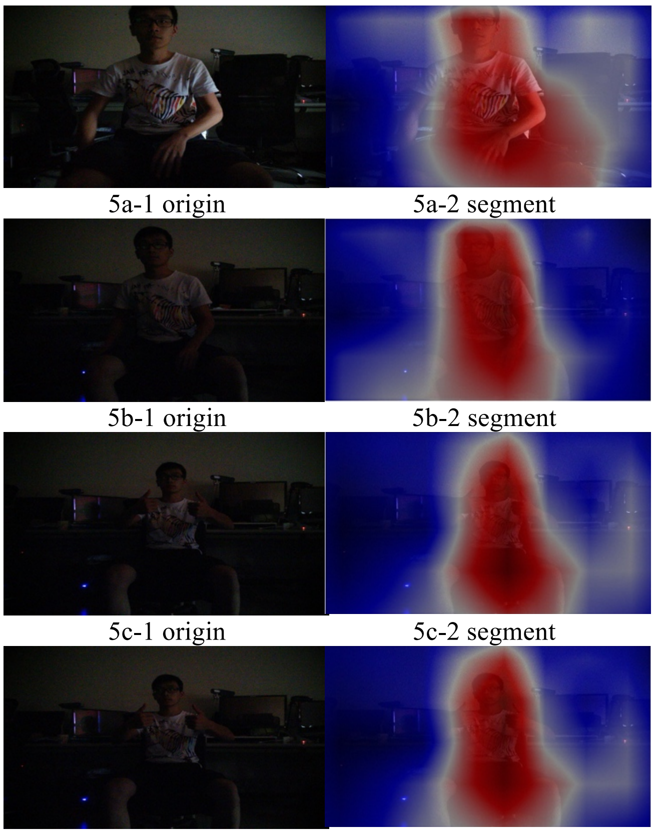
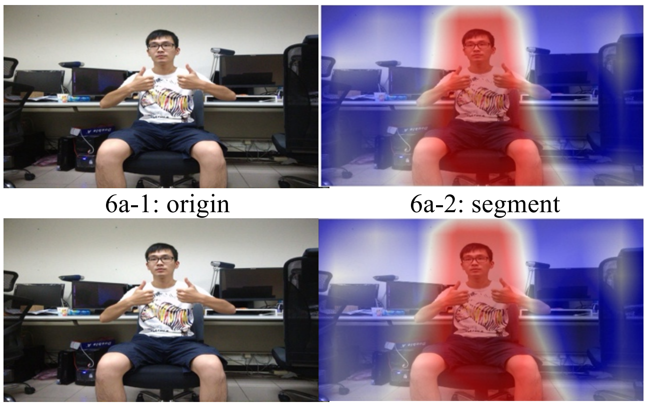

5a-1 origin 5a-2 segment
5b-1 origin 5b-2 segment
5c-1 origin 5c-2 segment
5d-1 origin 5d-2 segment

Figure 5. Show FCN perform segmentation in darkroom. 5a: near camera sitting; 5b: far from camera sitting; 5c-d: almost same pose but segment drifting but the main part is still correct. In the dark room the light is weak the prediction is affect by the main object sometimes it can segment more precisely compare to Figure 6.



6a-1: origin 6a-2: segment

6b-1: origin 6b-2: segment

Figure 6. Show a bright environment affect segmentation. As we can see, it predicts the chair back as a person sitting. Doesn't affect much sometimes its useful for prediction.

From Figure 5 and 6, we change the environment setting from the light. The first insight, the darker image the worse segmentation. However, it is not that directly.

In Figure 5a and 5b, we can see the nearer the man the brighter the body. Though that is an accident, we still find something interesting. The brighter image predicts more noise but why is that? The reason can be narrow down to scaling. When scaling, yes of course every feature scales down and including noise. So the darker one can perform well than brighter one.

In Figure 5c, 5d, 6a and 6b, they are almost the same photo except the light issue. First we talk about same environment. Closely look, we can find out the shape of 5c and 5d are different around the head. It can refer to the main feature and the noise it detected. The counterpart 6a and 6b don't have this kind of issue. Then we compare these 2 groups. The segmentation of darker is smaller than the brighter. It is more tightly and more accurate. It is because the other stuff in the scene are almost disappear, too dark almost black.

After Figure 5 and 6, we can know that if we need more tightly segmentation, we can use this method to quickly take to photo in different photometry just like taking a HD image.

6. Conclusion

The state-of-art method to process image issue is using deep learning. With CNN we can take out image features and undergo some process to learn image information and with RNN we can absorb the temporal information to deal with time sequencing problem such as learning hobbies or logical issue like question and answering.

TX1 is now the outstanding board for sure while it is still under a lot of improvements. The NVIDIA itself rapidly update the functions on TX1 and willing to face all kind of developing issue. So it is still quite an excitement and challenging to use such a leading embedded system.

The FCN model, we focused on, is a potentially general architecture for deep learning, image segmentation, captioning, question and answering and etc.

Moving on to the embedded system is quite a challenge. We cannot expect there will be enough memory or storage so we have to make a choice between performance and processing time. Like this experiment, we know that VGG16 performs the best and FCN can do pixel-based, but we need to cost down the processing time. Taking the secondary AlexNet, only do forwarding on TX1 and scale down the image to do prediction are all our penalty to balance between.

7. Future Work

The current contemporary classification networks we took is AlexNet. The community suggest that even though we cannot pretrain the data on the board, TX1, we can train them on server and take feature back only. That gives me an idea to try using VGG net and more others also suggest to use SegNet [17]. SegNet training for both the basic and extended variant was performed until convergence, yielding approximately 50 epochs. Inference takes 0.12s per image. The community say that it fit TX1 well so we will try this later on.

We have release our arrangement on community and try to excite other developer to work on the same stuff with us. Segmentation is quite a huge topic it should and will get lots of attention.

The camera issue should be fix this month or two. The current resolution we use is 640 x 480 for the time issue. It can be adjusted to higher resolution. To combine with the method, we mention before the dark-bright segment, we can assume to have more accurate segmentation through the higher image.

nvgstcapture-1.0 -m 1 --image-res=10	
(0) : 176 x 144	(1) : 320 x 240
(2) : 480 x 480	(3) : 640 x 480
(4) : 800 x 600	(5) : 720 x 480
(6) : 1280 x 720	(7) : 1280 x 960
(8) : 1600 x 1200	(9) : 2048 x 1536
(10) : 2240 x 1680	(11) : 2560 x 1920
(12) : 4096 x 3072 (only for t11x)	

Table 3: The resolution that we can used in TX1. The higher the more details we can get.

With the improvement of the technology, in the near future we may have a camera with GPU as well. And at that time we can test and train our segmentation on it to produce a self-learning segmentation camera which can auto-picture and auto-focus on the right thing we focus on.

8. References

[1] M.-H Yang and N. Ahuja, "Gaussian Mixture Model for Human Skin Colour and its Applications in Image and Video Databases", SPIE/EI&T Storage and Retrieval for Image and Video Databases (San Jose, January 1999), pp. 458-466.

[2] M. Jones, J. Rehg, "Statistical Color Models with Application to Skin Detection", IEEE Conference on Computer Vision and Pattern Recognition, CVPR '99 (1999), pp. 274-280.

[3] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[4] D. Chai and K. N. Ngan, Face segmentation using skin colour map in videophone applications, IEEE Transactions on Circuits and Systems for Video Technology 9 (4) (1999) 551-564

[5] J. Kovac, P. Peer and F. Solina, 2D versus 3D colour space face detection, 4th EURASIP Conference on Video/Image Processing and Multimedia Communications, Croatia, 2003, pp. 449-454.

[6] S. Tsekeridou and I. Pitas, "Facial feature extraction in frontal views using biometric analogies", Proc. of the IX European Signal Processing Conference, vol. I, 315-318 (1998).

[7] C.Garcia and G. Tziritas, Face detection using quantized skin colour regions merging and wavelet packet analysis, IEEE Transaction on Multimedia 1 (1999) 264-277.

[8] I-S. Hsieh, K-C. Fan, and C. Lin, A statistic approach to the detection of human faces in colour nature scene, Pattern Recognition 35 (2002) 1583-1596.

[9] G. Gomez and E. F. Morales, "Automatic feature construction and a simple rule induction algorithm for skin detection, Proc. Of the ICML workshop on Machine Learning in Computer Vision, A. Sowmya, T. Zrimec (eds), 31-38 (2002).

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[11] K. Simonyan and A. Zisserman. Very deep convolu- tional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842,2014.

[13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional acti- vation feature for generic visual recognition. In ICML, 2014.

[14] J. Long, N. Zhang, and T. Darrell. Do convnets learn corre- spondence? In NIPS, 2014.

[15] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In Com- puter Vision–ECCV 2014, pages 834–849. Springer, 2014.

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Gir- shick, S. Guadarrama, and T. Darrell. Caffe: Convolu- tional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014.

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv:1511.00561v2 [cs.CV], 2015.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition, 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.

[20] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014.

[21] P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with convolutional neural networks: a comparison to SIFT. CoRR, abs/1405.5769, 2014.

[22] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In NIPS, pages 2852–2860, 2012.

[23] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013.

[24] Y. Ganin and V. Lempitsky. N4 -fields: Neural network nearest neighbor fields for image transforms. In ACCV, 2014.

[25] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In ECCV. Springer, 2014.

[26] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In European Conference on Computer Vision (ECCV), 2014.

[27] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. Toward automatic phenotyping of developing embryos from videos. Image Processing, IEEE Transactions on, 14(9):1360–1371, 2005.

[28] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In ICML, 2014.

[29] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker. Multidigit recognition using a space displacement neural network. In NIPS, pages 488–495. Citeseer, 1991.

[30] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to hand-written zip code recognition. In Neural Computation, 1989.