Viewpoint Invariant Person Detection in RGB-D Data

Alisha Rege
Computer Science Masters,
Stanford University.
Stanford, CA, USA
amr6114@stanford.edu

Abstract

Artificial Intelligence can play a key role in healthcare; however, due to patient confidentiality (HIPAA), we are unable to process this information without putting up some boundaries. This boundary comes in the form of RGB-D data; it prevents us from seeing a face or a distinguishing personal characteristic in videos. This project attempts to detect a person from any viewpoint in Stanford Health's RGB-D data. The goal is to create a detection system that will be able to identify a person from any view point. This will allow nurses and doctors to sense problems such as if a person suddenly fell or if the person has not moved in days. An SVM with HOG descriptor as features is used as a baseline. A 6-layer CNN classifier is proposed as a better system to classify the object.

Keywords
RGB-D, Computer Vision, CNN, SVM, HOG.

1. Introduction

With recent developments in the world of computer vision, its applications to the real world have become more apparent. Visual healthcare data can be exploited to improve patient experience as well as create a better healthcare environment. As detailed by the Institute for Healthcare Improvement [1], assessing healthcare situations can provide better services to the patients and help hospitals provide patients with the right care. However, the one road block that healthcare applications often face is the Health Insurance Portability and Accountability Act (HIPAA). This policy requires that insurance information and healthcare records be kept private and under protection. This thereby hinders the ability to collect RGB images that include distinguishing features such as facial characteristics. One way we can get around this is by using RGB-D data which is a form of media in which only depth is captured by the camera. This thereby anonymizes the subjects while allowing for data collection and processing. Figure 1 shows an example of a RGB-D camera capture.

In order to do statistical analysis of healthcare situations, the first task is to distinguish people from the surrounding objects. This classification can help distinguish the level of intensity for future projects. For example, if an object hasn't moved in days, it is not as important as a human not having moved for a couple of days. The proposed algorithm uses cropped images of objects as input. The convolutional neural network is composed of six layers that are in the same form as LeNet. [2] These layers output a label indicating whether the object is a person or not.

2. Related Work

Feature detection is an important part of object classification. Many previous papers detail different approaches to this problem. Multi-class Fruit Classification using RGB-D Data for Indoor Robots [3] uses scalable color descriptors, edge histogram detectors, and shape to classify the image. The scalable color descriptors characterize the color distribution inside the object region to identify the object. The edge detectors use different gradients to classify the number of edge types to separate the image features. And finally, Shape, a feature descriptor created by Professor Fei-Fei Li, is used to find measures such as compactness, symmetry, local convexity, and smoothness to identify the images. While these features are very useful, the dataset collected from the hospital does not have color so all of these descriptors cannot be used.

Another paper by Manuel Blum and etc. [4]uses a bag of words learning model to find features. Using SURF descriptors, this project classifies using K-means and predicts new models by comparing the number of features in each bin of K-means. Although a novel approach, this experiment does not perform as well as others because it lacks in-depth feature descriptors. With more descriptors, this project may have been able to achieve a greater success rate.

D2 features were also considered as detailed in CS 221 Final Report: 3D Shape Classification [5]. This method finds different distance measurements between vertices to classify the image type. The problem with this approach is that for each distinguishing feature, the distance has to be hard coded in and that reduces the scalability of this program.

Another aspect of the classification process is the algorithm. Previous projects in this field have developed different algorithms. Discounting the approaches that use RGB images for detection, two papers have demonstrated significant work in this area. One paper written by the Max

Planck Institute for Informatics [6] uses multiple Microsoft Kinects to get RGB-D data and combines it to classify the image. This approach, unlike the one outlined in this report, utilizes more than one point of view of the same room and will have larger computation time since it has

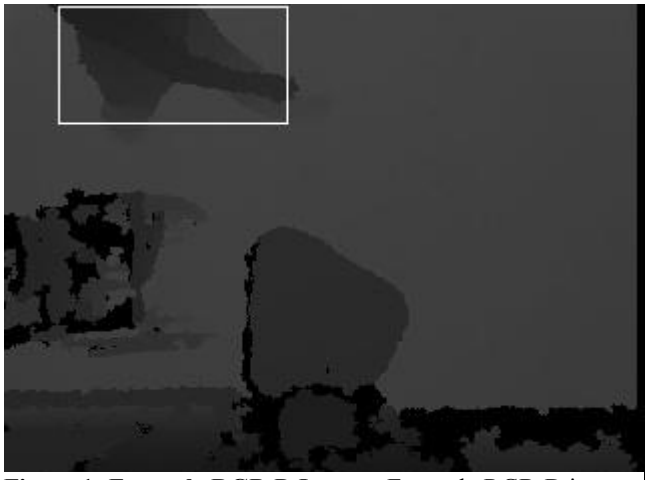

Figure 1: Example RGB-D Image - Example RGB-D image with example bounding box for person

more variables to consider. Another paper written by Chau Nguyen [7] uses cascaded classification to classify images. This approach uses both camera and depth images to classify the object for robots. This approach again uses multiple information points to form an analysis.

3. Dataset

The Healthcare in Artificial Intelligence Laboratory PAC has compiled a RGB-D dataset from Stanford's Lucile Packard Children's Hospital. This dataset has over 20 hours of film and captures the activity of the hospital over multiple days. The camera locations can be generalized to 3 key filming locations: the top down view of the room, the corner of a hallway, or the middle top of a wall of the room. Videos from every viewpoint were taken to train the dataset and videos from three cameras capturing different viewpoints were used to test the data.

A previous project by Alexandre Alahi [8] already finds bounding boxes for moving objects on the screen. The project originally focused on finding people, however in a situation such as healthcare with multiple objects moving, the program detects multiple objects as well as people. Figure 1 shows an example of the bounding box annotation.

After extracting the cropped images from the program, the images are hand-labelled to be person/not person. The annotations from the previous project serve a key purpose here. The videos were first converted to frames. Then, from each id, only 20 consecutive frames were labelled. These labels were fed into another script which outputted the cropped object frames into the files associated with their label. An example of the cropped image is shown in Figure 2. This serves as the ground truth for the project. A total of

Figure 2: Example of cropped image used for Training

approximately 2,000 images are labelled as person and a total of 1,500 are labelled as not person[1]. These cropped images are used to train the network. Four video[2] tracks are used to train the system.

4. Baseline

The baseline was configured by using the Histogram of Gradients [9] for the features and a Linear Support Vector Machine for the classifier. Bikramjot Hanzra's [10] work was used to implement the baseline.

4.1. Feature Descriptor: Histogram of Gradients

Known as one of the best feature descriptors in computer vision, histogram of gradients was used to extract feature points from the images. The HOG descriptor uses the Sobel Operator [11] to find x and y direction edge changes. The Sobel Operator a filter that is convolved over the image that looks like:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} and\ G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

These changes are then grouped into categories and compared across images. The gradient is quantized into 9 orientations. The image is divided into three by three subsquares and a histogram of direction is weighted with a magnitude. These vectors are used as the features for classification. Each image was resized to 56x56 pixels and then inputted into the skimage.feature [12] implementation of HOG.

4.2. Classification

The histogram of gradients was used as the features to the program. These features were then inputted into a linear SVM as detailed by the sklearn.svm [12] implementation. The logistic regression function was used to find the loss. The logistic regression function is as follows:

$$Loss = \log 1 - e^{-(w \cdot \phi(x))y}$$

L2 regularization by default was added to combat overfitting. Hence, the following equation was used for the SVM learning:

$$\min_{w} \frac{1}{2} \|w\|^2 + \lambda \|w\|_2^2$$

The SVM works by finding the best fit line between the features classification. It tries to minimize the logistic regression function with ever update.

[1] The numbers are uneven due to the regularization of the data. If two images were considered to be too similar, they were hand removed from the training set so as to avoid overfitting.

[2] Video from both the hallway and room viewpoint each and 2 videos from the top down view because these are vastly different from the other two views.

5. Convolutional Neural Network Implementation

5.1. Data Input

The dataset mentioned previously was used for this implementation. All images were scaled to 56x56 to be inputted into the network. This number was chosen because it was the average size of a square bounding box surrounding a person in the dataset. Placeholders were made for the inputs x and y, the respective input images and labels. The size of x was 50x56x56x3 as the batch size was 50 images each. The size of y was 50x2 as the one hot encoding system was used to indicate which class it belonged to.

5.2. Network Layers

A 6-layer neural network was used to train the CNN on TensorFlow [13]. After assessing the variety of networks available (LeNet, Google Net, etc.), the LeNet architecture [2] was chosen. The task to identify the image as person or not person in gray scale data is not as complex as identifying an image in RGB data due to the one dimension input of the data. The other architectures (Google Net, AlexNet, NIN) perform better; however, they are typically used to learn more complex images. In order to avoid the possibility of overfitting and given the project's time constraints, it was deemed necessary to use LeNet to learn the images. The LeNet architecture consists of three fundamental layers: the convolutional layer, the max-pooling layer, and the fully connected layer. The following architecture was used:

1. Input (56 x 56 x1)
1. Convolutional Layer & RELU
2. Pooling Layer
3. Convolutional Layer & RELU
4. Pooling Layer
5. Fully-Connected Layer
6. Output layer

5.2.1. Convolutional Layer with RELU

The convolutional layer of this architecture implements a 5x5x3 filter that computes the dot product for every pixel. The first layer of convolution has a bias of 32 whereas the second layer has a bias of 64. The biases serve as a form of regularization. The following formula is used to calculate the value at every pixel (x_{ij}^{ℓ}) and then the sigmoid function is used to provide non-linearity:

$$x_{ij}^{\ell} = \sum_{a=0}^{5-1}\sum_{b=0}^{5-1} \omega_{ab}\, y_{(i+a)(j+b)}^{\ell-1}$$

$$y_{ij}^{\ell} = \sigma(x_{ij}^{\ell})$$

5.2.2. Max-Pooling Layer

A max-pooling layer is implemented between the convolutional layers to reduce the spatial size of the image as well as the parameters to avoid overfitting. Max-pooling is applied to the image to down sample it two times (thereby taking the max of the pixels and downsizing the image).

5.2.3. Fully Connected Layer

The fully connected layer then takes the input and connects it to every neuron it has. This layer takes 14x14x64 Inputs and then outputs 1024 values.

5.2.4. Output Layer

Finally, the output layer takes these 1024 values and interprets the classes. This output layer gives two values: a vector detailing the likelihood that it belongs to each class (person/not person).

5.3. Optimization

This program uses the Adam Optimizer [14] to find the best weights for the neural network. This optimization program is the best currently available. It adjusts the parameters of the update based on the momentum of the change.

5.4. Classification and Loss function

The softmax with cross entropy function was used to calculate the loss of misclassification. The formula for softmax is as follows:

$$\varphi_i = \frac{e_{\eta_i}}{\sum_{j=1}^{k} e_{\eta_j}}$$

where φ_i is the probability distribution of the class i. This is then inputted into the cross-entropy loss which is:

$$L_i = target * -log(\varphi_i) + (1 - target) * -log(1 - \varphi_i)$$

6. Metrics of Measurement

Two metrics of measurement were used to calculate the effectiveness of the program: accuracy and mean average precision (mAP). Other forms of measurement such as MOTP and MOTA [15] were ruled out because they were typically used for video applications and tracking rather than classification.

6.1. Accuracy

In order to optimize the program the accuracy was used to calculate how correct the program was. The formula to calculate it is as follows:

$$\%accuracy = 1 - \frac{\left|Label_{correct} - Label_{predicted}\right|}{Label_{correct}} * 100$$

6.2. Mean Average Precision (mAP)

The other metric for measuring the correctness of the program was the mean average precision. This is the average of precision values. The sklearn metrics [12] implementation was used to find this value. The formula is as follows:

$$mAP = \frac{1}{numentries} \sum_{c=1}^{numentries} AP(c)$$

$$AP(c) = \frac{\sum_{k=1}^{n} P(k) x\, rel(k)}{\sum_{k=1}^{n} rel(k)}$$

Classifier	Training mAP	Training Accuracy	Testing mAP	Testing Accuracy
SVM w/ HOG descriptors	.702	.609	.542	.532
CNN over cropped images	.977	.971	.693	.590
CNN over entire image	.723	.624	.560	.540
CNN with double representation	.985	.981	.705	.650
CNN with unequal representation	.985	.904	.693	.608
CNN with Histogram Equalization	.988	.982	.715	.667
CNN on precise cropped images	.999	.999	.912	.890

Figure 3: Results

7. Experiments

Figure 3 details the results of this project. The original setup of the system was to delete images from the over represented class to make the representations equal, then to use softmax regression with cross entropy to calculate the loss, and finally these unchanged cropped images were fed into the neural network cropped. Many different transformations were tried on the CNN network.

Evaluation of algorithm and Epoch. The epoch was tweaked to find the optimal value. Figure 5a shows the epoch vs mAP correlation and Figure 5b shows the correlation between epoch and accuracy for the CNN network. One can see that as the epoch increases, the accuracy of the training set decreases; however, as the epoch surpasses 410, the system overfits the data for the training set and the testing accuracy starts to decrease. This shows that the system is performing well. Hence for all future manipulations, the epoch was set to 410. Furthermore, Figure 5c shows the ROC curve with an AUC of 75%; this means it is doing fairly well. This was later improved on.

Entire Image vs Cropped Images. Instead of classifying on the cropped images, the system was tested to determine if it could perform well on uncropped images. The accuracy and mAP were lower than the original system by approximately 10%. This is due to the fact that the entire image contains examples of both classes. Since most of the image is not a person for training and the features may differ based on where the person is standing, the implementation does not do well. If the cameras were non-movable, training on the entire image may make sense; however, a different model would have to be made for every sensor to properly capture the background and identify the characteristics of a person entering the scene. Additionally, another class or many more training examples may have to be added such that it does not categorize any movement in the scene as a person.

Class Representation. The system originally removes cases from the positive human cases to make the representations of the classes equal. Two variations were tested to increase accuracy: doubling the representation of the non-human class to make the representations equal and keeping the representations unequal. The results are outlined in Figure 3. The best outcome was when representation from the non-person class was doubled rather than taking away from the positive human class. This is logical because this means that the relevant feature data for person orientations was not taken away.

Image Manipulation: Histogram Equalization. [16] After looking at the input to the system, the images were also tweaked to examine the output of the system. First, the images were contrasted by using Histogram Equalization. This causes the contrast to increase in the images by smoothing out the buckets sizes for each pixel intensity. Figure 4 shows how an image is affected by this. The person that was barely detectable in the first image now becomes more pronounced. This change also makes the accuracy and mAP increase by approximately 3% and 8% respectively and thereby makes it a better system.

Classification and Loss. No experiments were done here because the classification and loss functions were ideal for the task (in the implementations provided by Tensor Flow). sigmoid cross entropy with logits was not considered due to the fact that the labels are mutually exclusive.

Better Training Set. Finally, the case of a thoroughly vetted training set was taken. A total of 1,000 images (500 positive/500 negative) were meticulously hand-labelled as positive or negative. Any images that did not capture the

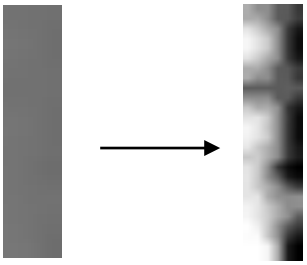

Figure 4: Histogram Equalization – left-hand side is the original image and the right-hand side is the histogram equalization.

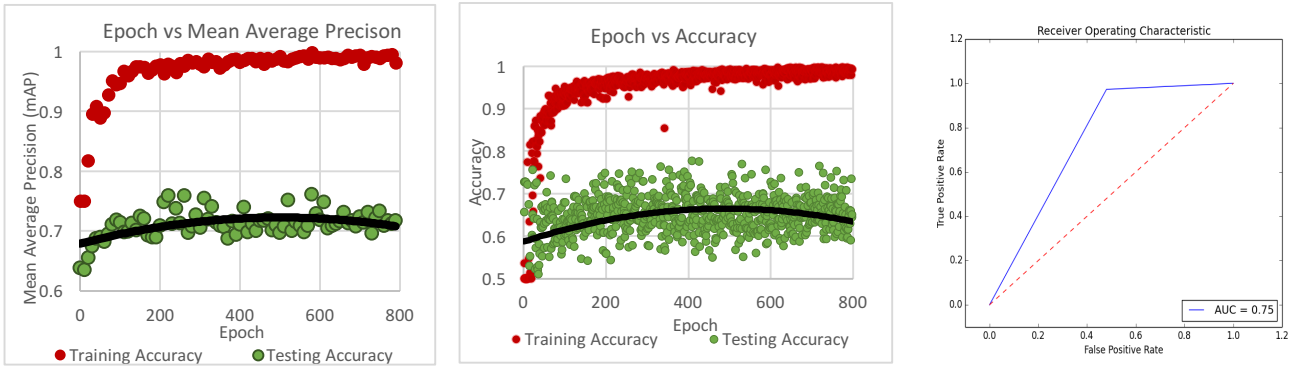

Figure 5: (a) Epoch vs Mean Average Precision (b) Epoch vs Accuracy (c) ROC curve - as the epoch increases, the training images achieve higher accuracy/mAP because they are being trained on more. However, the overfitting causes the testing curve to decrease after 410 epochs. The ROC curve for the maximum epoch shows that the training works fairly well.

entire human or were not on the human properly were removed. This relabeling caused the accuracy and mAP to shoot up to 89% and 91% respectively. This showed that the original system was working properly; however, either an increase in training data or a decrease in bad examples would cause the system to work better.

Viewpoint To better see the results, Figure 6 shows how each viewpoint representation affected the results. The viewpoint that has more representation (mid-wall) tested better because more cases were encompassed. On the other hand, the hallway which had less training examples did worse.

8. Visual analysis

The CNN did a great job of learning what a person is not. With the exception of one image, the neural network was able to classify all non-person images properly. The following examples illustrate the results.

Figure 7b came out positive as there were many images in the training set that accounted for the case of a person in the edge of the camera viewpoint. Due the activity levels at the hospital many people were entering and leaving the scene. However, there were two noteworthy cases presented themselves as problematic for the classifier. The first, depicted in Figure 7a, was the case of improper bounding box. This can be corrected simply by expanding the cropping system to include a padding. This means that some of the environment will be part of the image but it would provide for a more robust system. The second, depicted in Figure 7c, was the lack of training examples for certain objects; in this picture, the intravenous depicted was never in the training set and hence was never trained in the no class. This could easily be solved by adding more examples in the training set.

Viewpoint	Number of Training Images	Testing mAP
Top-Down	544Y/377N	.71
Mid-top Wall	767Y/383N	.81
Hallway	374Y/284N	.64

Figure 6: Viewpoint Training Examples vs mAP

9. Conclusion

After experimentation, it was decided that the best system will be a convolutional neural network that has histogram equalization done on the input images, the representation from all classes should be equal by over-sampling from the underrepresented classes, and if possible have the best training set with minimal noise.

This system is highly competent with the current setup and could be used to solve many problems to come. One of the immediate future steps is using these annotations to find whether the person in the image is a doctor, nurse, or patient. The RGB-D data will be combined with trajectories and skeleton analysis. Another useful future application is using this system to find anomalies in patient patterns such as falling down, or not moving.

Acknowledgments

Special thanks to the Artificial Intelligence Vision Lab for providing guidance as well as the dataset. Thanks to Stanford's Lucile Packard Hospital for providing the dataset for this experiment.

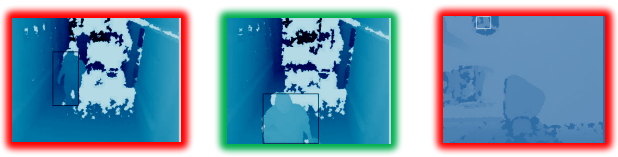

Figure 7: (a) Example of Improperly labelled image due to bounding box **(b)** Example of properly labelled image **(c)** Example of improperly labelled image due to uniqueness.

References

[1] Institute for Healthcare Improvement, "Vision, Mission, Values," 10 December 2016. [Online]. Available: http://www.ihi.org/about/pages/ihivisionandvalues.aspx.

[2] Y. L. e. al, *LeNet-5, convolutional neural networks,* n/a: http://yann.lecun.com/exdb/lenet/, n/a.

[3] A. K. S. A. S. a. A. Z. L. Jiang, "Multi-class fruit classification using RGB-D data for indoor robots," IEEE Int. Conf. Robotics and Biomimetics (ROBIO), Shenzhen, 2013.

[4] J. S. J. W. M. R. M. Blum, "A learned feature descriptor for object recognition in RGB-D data," *Robotics and Automation (ICRA),* pp. 1298-1303, 14 May 2012.

[5] T. Du, P. Xu and R. Hu, "CS 221 Final Report: 3D Shape Classification," Stanford CS 221, Stanford, 2015.

[6] W.-C. Chiu, U. Blanke and M. Fritz, "Cross-Modal Stereo by Using Kinect," Max Planck Institute for Informatics, 2 September 2011. [Online]. Available: https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/object-recognition-and-scene-understanding/cross-modal-stereo-by-using-kinect/. [Accessed 10 December 2016].

[7] C. Nguyen, "Object Detection Using Cascaded Classification Models," Cornell University, Ithica, 2010.

[8] Y. B. L. J. a. P. V. A. Alahi, "A sparsity constrained inverse problem to locate people in a network of cameras," Proceedings of the 16th International Conference on Digital Signal Processing (DSP), Santorini, Greece, 2006.

[9] N. Dalal and B. Triggs, *Histogram of Gradients,* France: Conference on Computer Vision and Pattern Recognition , 2005.

[10] B. Hanzra, "Object-Detector," GitHub, 23 July 2015. [Online]. Available: https://github.com/bikz05/object-detector. [Accessed 10 December 2016].

[11] I. Sobel and G. Feldman, *Sobel Operator,* Stanford: Stanford University, 1968.

[12] J. L. S. J. N.-I. F. B. J. D. W. N. Y. E. G. T. Y. a. t. s.-i. c. Stéfan van der Walt, *scikit-image: Image processing in Python,* PeerJ: http://dx.doi.org/10.7717/peerj.453, 2014.

[13] A. A. e. Martín Abadi, *TensorFlow: Large-Scale machine learning on heterogenous systems,* n/a: Tensorflow.org, 2015.

[14] D. P. a. B. J. Kingma, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980 [cs.LG], n/a, 2014.

[15] K. E. A. S. R. Bernardin, "Multiple object tracking performance metrics and evaluation in a smart room environment," Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV2006, Graz, Austria, 2006.

[16] G. Bradski, "opencv_library," Dr. Dobb's Journal of Software Tools, n/a, 2008.