Detecting the Use of Hand Hygiene Dispensers

Research Sample

1 Introduction

1.1 Overview

PAC (Partnership in AI-Assisted Care) has several ongoing projects across various areas of healthcare. One project is related to hospital hand hygiene and infection control.

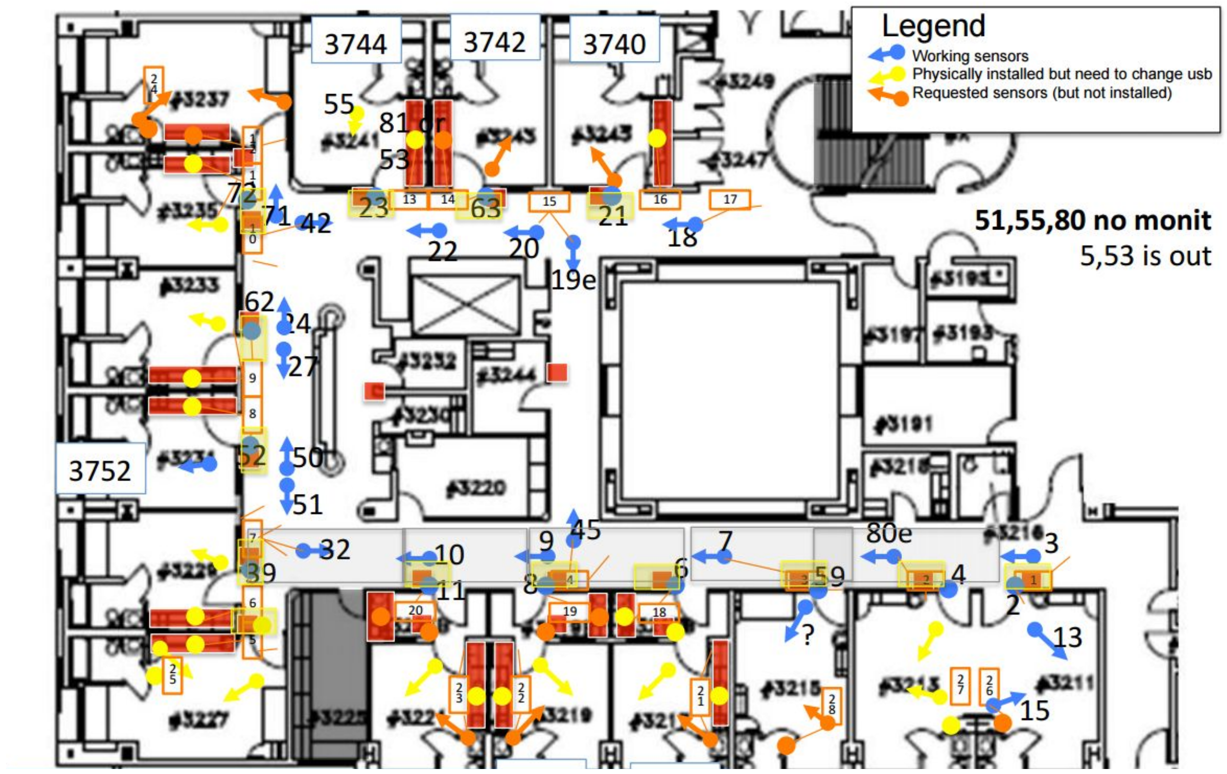
The goal of the hand hygiene project is to measure how often people are washing their hands and determine if a person is about to make physical contact with a patient. You would be surprised how many hospital infections are caused by this. The end goal is to enforce this in real-time and notify nurses before a contaminated person enters a patient room.

Before we can enforce proper hand hygiene, the first step is to determine whether a person uses a hand hygiene soap or alcohol-gel dispenser. These dispensers are typically attached to the wall outside and inside patient rooms. We can treat this as a binary classification task where the input is a single image and the output is a binary label indicating dispenser usage or not (1=person used dispenser, 0=no person or person did not user dispenser). PAC currently has 32 depth sensors (i.e., Microsoft Kinect) installed at Lucile Packard Children's Hospital at Stanford. In this research sample, we will use this data to train and analyze a machine learning model.

1.2 Background

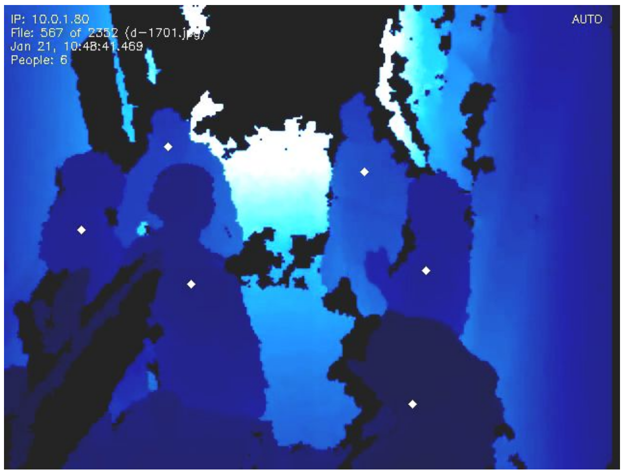
Below is a floor plan of the hospital wing where we have our sensors. Blue arrows indicate sensor location and direction. Blue dots indicate top-down view sensors (on the ceiling, facing the floor).

Note: We do not refer to the video cameras as cameras. Instead, we refer to them as ***sensors***. Cameras are typically interpreted as RGB color video streams. In this research sample, we will not use color video.



Legend
Working sensors
Physically installed but need to change usb
Requested sensors (but not installed)
51,55,80 no monit
5,53 is out
3744
3742
3740
3752

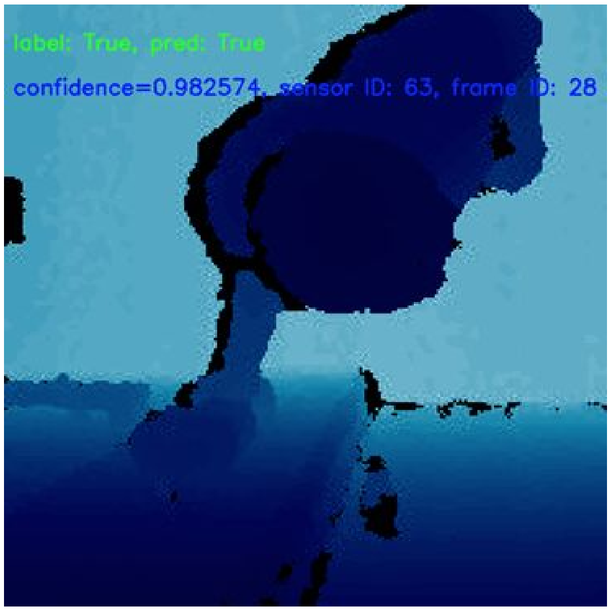
We have a mix of side view and top view sensors.

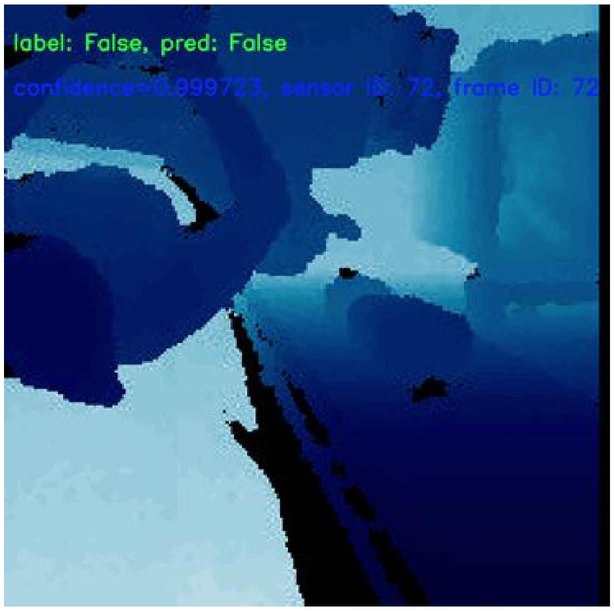


Notice how these images have a blue color and is not a normal picture. These are called ***depth images***. See Section 2.1 for more details.

1.3 Classification Task

Our goal is to classify a top-view depth image as "using dispenser" or "not using dispenser." Some examples are below. On the left, the person is using a dispenser. On the right, they are not using the dispenser.




label: False, pred: False
confidence=0.999723, sensor ID: 72, frame ID: 72

Your task is to train a classifier using the data from the hospital wing. Whether you train a model for each sensor, or a single model is up to you. The goal is to maximize classification accuracy over the entire hospital unit.

You are responsible for creating a training/validation set split and optimizing for classification accuracy. You should apply any machine learning techniques you know (data augmentation, regularization, traditional and deep learning methods). You can use any existing code online from research papers, blogs, or public Github repositories.

We have a hidden test-set which we use to benchmark your final model. This is purposely kept hidden to ensure proper generalization to new examples. Once you have trained a model which achieves good validation accuracy, please save the model to disk and keep the loading and inference code ready.

Please create a training script which does all model training and saves the model. For validation and test set evaluation, it is best to keep this in a separate file which loads the saved model. This will be useful once we deploy your model to production. Please use Github to version your code (tutorial).

2 Data

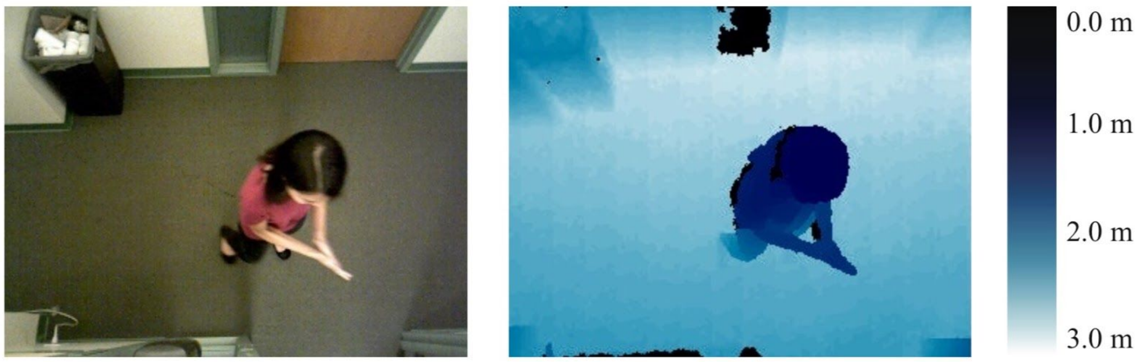
We have collected and annotated 113,379 top-view images with binary classification labels. The data is split into images and labels. Please have 5 GB free on your machine. The tar file is 2.3 GB but once you decompress it, it will consume an additional 2.3 GB. You do not need to train on all the images.

Download the data: https://aicare.stanford.edu/jan2018.tgz (2.3 GB; gzipped tar)

2.1 Depth Images

The use of computer vision technologies in healthcare environments has been limited due to strict privacy laws protecting personally identifiable information. To comply with privacy regulations, we use depth images instead of standard color photos. These depth images are collected with special depth sensors instead of standard cameras.

A depth image can be interpreted as a de-identified version of a standard color photo. In color photos, each pixel represents a color, often encoded as combination of red, green, and blue (RGB) values as unsigned integers. In depth images, each pixel represents how far away the “pixel” is, often encoded in real world metres as floating point numbers. While we lose color information in depth images, this allows us to comply with privacy regulations.



The figure above shows a color image and its corresponding depth image. For visualisation purposes, the depth image is artificially colored blue using a color gradient. Notice that while color information is lost in a depth image, one is still able to understand the semantics of the scene (e.g., a person is washing their hands). This is because depth images convey volumetric 3D information while color images convey colored appearance.

2.2 Folder Structure

Once you have decompressed the tar file, you will see 17 folders, each with a two digit number as the filename. Each of these folders corresponds to a different sensor. Inside each sensor folder, you will see two more folders with names 0 and 1. These correspond to the labels. The label 1 denotes a positive example (i.e., someone is using the dispenser) and the label 0 denotes a negative example (i.e., no person in the image or a person not using the dispenser).

2.3 Compressed Numpy Files

Inside each of the 0 and 1 folders, you will see a list of .npz files. This is a compressed numpy file. We do not use .jpg images for two reasons: (i) we are not using color images but instead use a single-channel floating point matrix and (ii) we use better compression algorithms instead of jpg compression.

2.3.1 Loading in Python

```
>>> import numpy as np
>>> A = np.load('20170121_210810_614.npz')
>>> A.files
['x']
>>> depth_map = A['x']
>>> depth_map.shape, depth_map.dtype
(240, 320), dtype('float16')
```

2.3.2 Format

As mentioned earlier, images are stored as depth maps and not normal RGB images. In the example above in Section 2.3.1, depth_map is a 240x320 matrix containing real values. To visualize the image, you need to convert it to an RGB or grayscale image.

2.3.3 Visualization

If you try to visualize the 240x320 matrix directly from the numpy file, it will likely break your image viewer (e.g Matplotlib, OpenCV, PIL). It contains real values and we must convert it to a proper integer-image format.

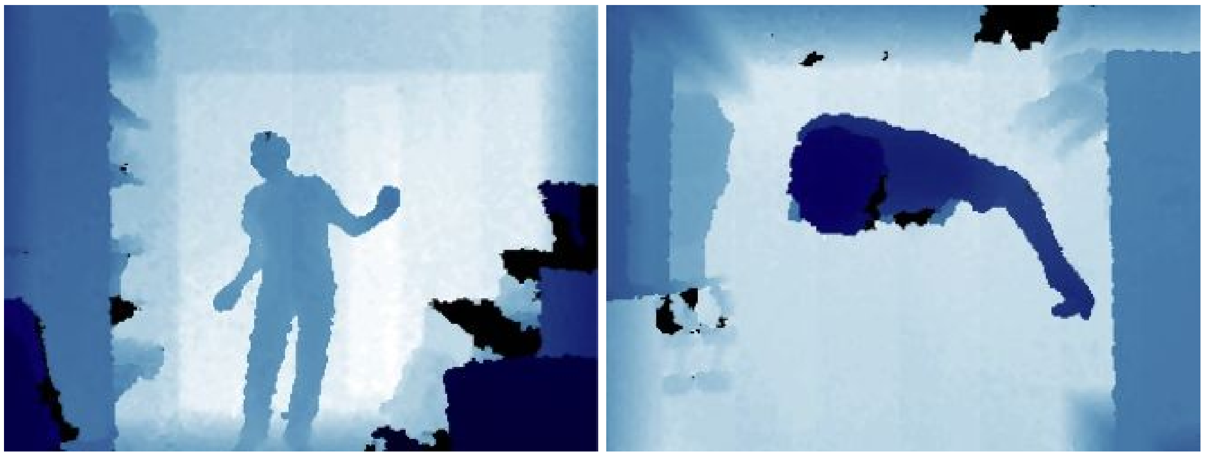
The following Python function takes depth_map which is a 2D matrix and returns a 3D matrix (i.e. RGB image). The image is colored as blue/ocean by default but you can try other OpenCV color maps. For an example in C++, see this webpage. You do not have to use the code snippet below. Instead, you can normalize each depth map by the min/max to produce an image in the range 0 to 1, inclusive.

Raw: https://pastebin.com/u89uKdar

```
import numpy as np
import cv2
def depth_map_to_image(depth_map):
    img = cv2.normalize(depth_map, depth_map, 0, 1, cv2.NORM_MINMAX)
    img = np.array(img * 255, dtype=np.uint8)
    img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)
    img = cv2.applyColorMap(img, cv2.COLORMAP_OCEAN)
    return img

   def main():
       data = np.load('20170121_210810_614.npz')
       depth_map = data[’x’].astype(np.float32)
       ocean = depth_map_to_image(depth_map)
       cv2.imshow("Image", ocean)
       cv2.waitKey(0)
```

The resulting images will look like:



When processing the depth maps (e.g. computing means, data augmentation), you should use the raw depth maps and not the RGB-ocean images! The depth maps are real values and contain more information than the (falsely) colored integer image!

3 Deliverables

The purpose of this research sample is to expose you to some of the day-to-day tasks you may face while working with us on research problems. This research sample is purposely made vague to give you maximum flexibility in how you solve the problem.

With that said, you are given total control on how you solve the problem. Whether you train one model or 17 models (one per sensor) and whether you use convolutional networks or not, is entirely up to you.

We ask you to submit: (i) your code and (ii) a short 1-2 page write-up on what you've done.

We will primarily be evaluating you based on your write-up. Your classification accuracy is only a small piece of what we are looking for. If you have to pick between improving your accuracy score or improving the write-up/analysis, we recommend the latter.

3.1 Code

Please use Github or Bitbucket and submit the link to us when you are done. If you are using a private repo, you can add Albert (Github and Bitbucket: **ahaque**) as a read-only collaborator.

Allowed Languages: Python 3 or C++; Jupyter notebooks are allowed.

Code style and comments: Please do not spend too much time reformatting your code or adding tens of lines of comments just for us. Simply write your code as you normally would. It is better to focus on the write-up (see below).

3.2 Writeup

We've provided a template below for you to follow when preparing your write-up. For each question, we have also included some example answers to help guide you as you complete the problem. The answers below might be good things to try!

You do not have to answer each question below. They serve more as guidelines to help you structure both your write-up and solution to the problem. We highly recommend for you to come up with your own questions/analysis and include them in your write-up. You do not need to notify us beforehand. Remember, this research sample is open-ended. You can include plots/graphs, figures, or tables if desired. If you do answer the questions below, please explain why (except the overview).

3.2.1 Template

Your write-up should be **at most 2 pages**, single spaced. You do *not* have to use latex.

1. **Overview [1 paragraph]**
   - What was your best model? e.g. SVM or CNN
   - What was your best validation set classification accuracy? e.g. 72% accuracy
   - What were the specifications of your training/validation set? e.g 90/10 split
   - What were the hardware specifications of the machine you used for training? e.g. Macbook Pro 2.9 GHz quad core
2. **Data [1-2 paragraphs]**
   - Did you subsample the data? Did you use all the data?
   - Did you resize or crop the images?
   - Did you apply any data augmentation techniques?
   - What was the distribution of labels/images, per sensor? What about time of day?
   - Did you do any other data pre-processing?
3. **Methods [1 paragraph]**
   - What machine learning methods did you try? e.g. logistic regression, SVM, CNN
   - What features did you use? e.g. HOG, depth maps, ocean images
   - What were the settings or hyperparameters of the models you tried? e.g. learning rate=0.1
   - Why did you use certain model configurations? e.g. 2 layer neural network
4. **Results & Discussion [1-2 paragraphs]**
   - For each model you tried, what training/validation accuracy did you get?
   - What was the accuracy for each sensor?

- Were there any images or sensors which your model struggled with?
- Does the time of day or month have any effect on accuracy?
- What other metrics did you use to evaluate your model?

5. **Future Work [1 paragraph]**
   - If you had more time, what future work would you do? e.g. what models?
6. **References [optional]**

4 Resources

In this section, we have included some links that may be useful for you.

4.1 Software

- Installing OpenCV (Mac)
- Jupyter Notebook Tutorial: The Definitive Guide
- Git/Github tutorial
- Stanford CS 231N: Python Numpy Tutorial
- An introduction to machine learning with scikit-learn
- Learning PyTorch with Examples

4.2 Reading

Haque et al. Towards Vision-Based Smart Hospitals: A System for Tracking and Monitoring Hand Hygiene Compliance, 2017. [pdf]

Mehra et al. Depth-Based Activity Recognition in ICUs Using Convolutional and Recurrent Neural Networks, 2017. [pdf]

Ma et al. Measuring Patient Mobility in the ICU Using a Novel Noninvasive Sensor, 2017. [pdf]