CS230

A Head Pose-based Controller for Assistive Photography

Oriana Peltzer
Department of Mechanical Engineering
Stanford University
peltzer@stanford.edu
SUNet ID: 06187734

Abstract

We develop a Head pose-based controller for a person to control the orientation of a camera without using their hands or arms.

1 Introduction and Related Work

Since the democratization of the camera, photography became an art, a tool for providing evidence, and a way to memorialize life experiences (Sontag, 2001). However, people with limited hand and arm mobility encounter more challenges when operating a camera, which prevents them from enjoying photography. A survey conducted in 2018 reports that people with motor impairments, while engaged with smartphone photography, do not capture or share photos as much as they would like due to the difficulty of operating the camera and the resulting poor photo quality (Mott et al., 2018).

To the extent of our knowledge, there is currently no widely available technical solution for operating a camera without using one's hands or arms.This project is the continuation of a project started in Winter 2020 for ENGR210 - Perspectives in Assistive Technology - to design a system for aiming a camera without using one's hands. The device is a mount that can be placed on the laptray of wheelchair users, and supports a phone or a camera. In an early prototype of this system (see Fig. 1), controlling the pan and tilt of the camera was ensured with a remote control. In this work, we wish to design vision-based software that would replace the remote control and completely remove the system dependency on hand use. We design a deep-learning-based controller for a user to change the pan and tilt of the camera by moving their head.

The project was originally proposed by Paul, a resident of Redwood City using a powered wheelchair. Wanting to enjoy photography, Paul created the ENGR210 Assistive Photography project to enable quadriplegic patients to take photos.

Abiyev and Arslan, 2020 recently proposed a head and eye-based controller for controlling a computer that does not require the user to wear any specific device. The algorithms for extracting eye blinking and head pose information each use Convolutional Neural Networks (CNNs). As their network is not publicly available, we will similarly train a CNN to infer head pose online.

2 Dataset and Features

The Biwi Kinect Head Pose Database (Fanelli et al., 2013) includes 15000 labeled images of 20 people, along with their head roll, pitch and yaw angles, and has been used to train and validate

CS230: Deep Learning, Winter 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

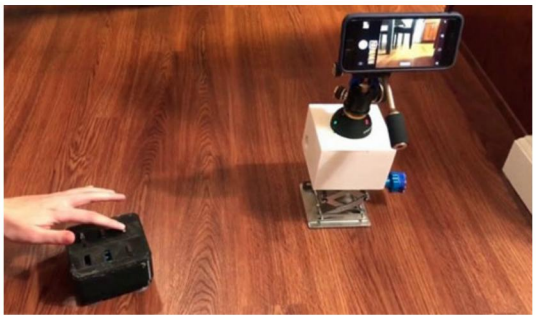


Figure 1: Early prototype of the camera mount. A user would control the pan of the camera and the shutter using a remote control (left). The camera's tilt and height adjustment is manual, which is not convenient. We wish to improve the device by providing a hand-free solution for aiming and taking photos.

learning algorithms (Chen et al., 2016; Liu et al., 2016). We selected four people at random in the dataset to use their samples as testing and validation data, and the rest is used as training data.

The metric we are currently using to use to evaluate the outcome of training is the Mean Squared Error for the pan and tilt vectors on the test dataset. As the user should always be facing the camera, we did not anticipate any difficulties due to the inexactness of taking the difference in between angular values.

3 Method

3.1 Head pose inference (CNN)

Our first goal is to employ a Convolutional Neural Network for inferring the pan and tilt values of head pose from image data. The output of the network is a two-dimensional vector in $[-\frac{\pi}{2}, \frac{\pi}{2}]^2$, as the user always faces the camera.

3.2 Camera control

Using the output of the CNN, the purpose of the controller is to change the camera's pan and tilt as the user moves their head. As the CNN's output may contain noisy data, we compute a smoothed head pose estimate using a Kalman Filter before feeding it into the controller.

4 Results

CNN Architecture

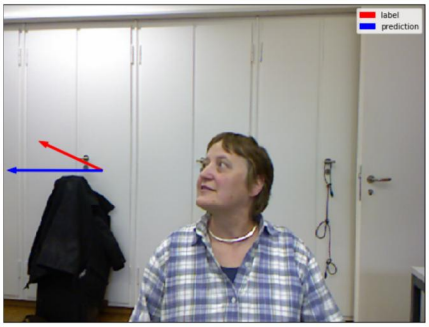
We chose to use Transfer Learning as shown in our architecture (Fig. 3). More specifically, we flatten the output of a pre-trained Resnet50 model, and feed it into to two dense layers with linear activation functions to obtain our output pitch and yaw values. We freeze the Resnet50 model and only train the weights and biases of the two dense layers. This results in 16,410 parameters to train.

Training results

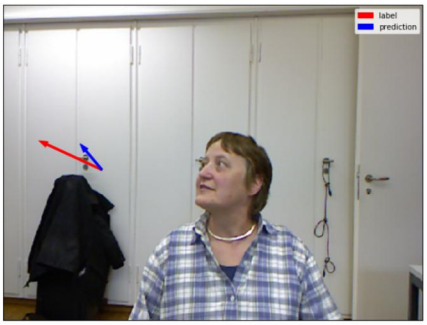
We started training by using data from 10 different people, which corresponds to about 50% of the BIWI dataset. After 14 epochs, while the training loss continues to decrease, the validation loss increases. To avoid overfitting, we decided to add the remaining training data to the training set. After 3 new epochs of training, we notice that the validation loss decreases.

After 40 epochs of training, we find that training and validation loss tend to stagnate. We also notice that the network's performance is not symmetrical, as left rotations are detected more accurately than right rotations. We therefore augmented the training dataset by flipping images and labels along the vertical axis.

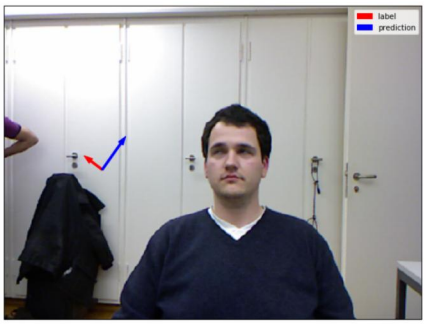
(a) 14 epoch training progress



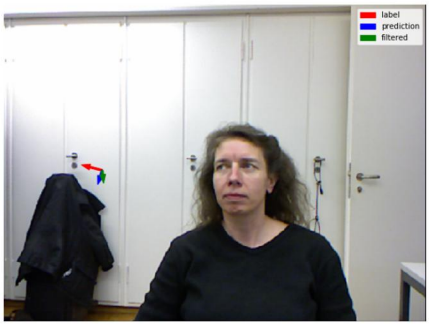
(b) Training example after 10 epochs



(c) Training example after 14 epochs



(d) Validation example after 14 epochs



(e) Unsuccessful validation example after 48 epochs, with filter

(f) Successful validation example after 48 epochs, with filter

Figure 2: Sample from the Biwi Kinect Head Pose Database, along with pitch and yaw label and prediction after different epochs of training.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 480, 640, 3)]	0
tf.__operators__.getitem (Sl	(None, 480, 640, 3)	0
tf.nn.bias_add (TFOpLambda)	(None, 480, 640, 3)	0
resnet50 (Functional)	(None, 15, 20, 2048)	23587712
global_average_pooling2d (Gl	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 8)	16392
dense_1 (Dense)	(None, 2)	18

Total params: 23,604,122
Trainable params: 16,410
Non-trainable params: 23,587,712

Figure 3: Transfer Learning model summary

After 48 epochs, the network's performance is still erratic. While some rotations are accurately captured, other rotations are not detected. See Fig. 2 for visualizations of the training results.

Network Training Challenges

We used a CPU, instead of a GPU, to train the network. This posed limitations in terms of time required to train the network. We should use a GPU to further improve training results in future work.

The training dataset was limited in terms of number of people, background and setting. Generating more training data should improve the learning performance and avoid overfitting.

Filtering

We employ a Kalman Filter to compute head pose estimates in real time. As images stream in sequentially, the variation in between two consecutive head poses is limited. To eliminate part of the random noise from the Network output, we model a person's head pose as a discrete time Markov Process.

Let $X(t)$ be the head pan and tilt angles at time index t. Let $Y(t)$ be the measurement of head pose corresponding to the output of the CNN. We adopt the following simplified model:

$$X(t+1) = X(t) + w(t),\ Y(t) = X(t) + v(t)$$

where the process noise $w(t) \sim \mathcal{N}(0, Q)$ and measurement noise $v(t) \sim \mathcal{N}(0, R)$ are Gaussian White Noise. The model does not include a control input, as we suppose that the person's head movements are unknown a-priori.

Q captures the covariance in between consecutive measurements. In practice, we compute Q as being the statistical covariance in between consecutive measurements in the (sequential) dataset. Similarly, we compute R as being the statistical covariance of the error in between measurement and label over validation data.

We find that while the filter improves the smoothness in between consecutive measurements, it is not sufficient to compensate for the consistent errors made by the Neural Network during training. A video of the filtered values can be found at https://drive.google.com/file/d/1aKjbKJnDej1e-Y0LSPaoRJ4xsUvH84US/view?usp=sharing .

Future Work and Anticipated Challenges

CPU time To be used as a controller, the system must generate predictions in real time. We anticipate a required frequency of at least 10Hz for fluent use with a human operator. However, due to the high number of parameters in the resnet50 model, we may need a GPU to ensure that the feedforward step does not take too much time.

BIWI dataset bias Most of the images from the BIWI head pose dataset have been taken from the same indoor location, and in the same lighting conditions. In order for the controller to perform well, we must ensure that our model does not fail in different locations and settings.

Intended performance and data distribution In addition, we anticipate that the distribution of head rotations in our camera control application will differ from the training and validation distributions. More precisely, the user is unlikely to show high pan and tilt head angles. Instead, the most important control inputs for the camera will be centered around zero (the user faces the camera). This will require a more in-depth evaluation our model in this specific context. In particular, the Mean Squared Error Loss may not be best suited to low pitch-and-yaw examples. In Chen et al., 2016, a few additional terms are provided in their loss function addition to the squared error, regularizing their model and making it more robust to outliers.

5 Acknowledgements

We thank the CAs for their valuable feedback and advice for constructing the model.

References

Abiyev, R. H., & Arslan, M. (2020). Head mouse control system for people with disabilities. *Expert Systems*, *37*(1), e12398.

Chen, J., Wu, J., Richter, K., Konrad, J., & Ishwar, P. (2016). Estimating head pose orientation using extremely low resolution images. *2016 IEEE Southwest symposium on image analysis and interpretation (SSIAI)*, 65–68.

Fanelli, G., Dantone, M., Gall, J., Fossati, A., & Van Gool, L. (2013). Random forests for real time 3d face analysis. *International journal of computer vision*, *101*(3), 437–458.

Liu, X., Liang, W., Wang, Y., Li, S., & Pei, M. (2016). 3d head pose estimation with convolutional neural network trained on synthetic images. *2016 IEEE international conference on image processing (ICIP)*, 1289–1293.

Mott, M. E., E, J., Bennett, C. L., Cutrell, E., & Morris, M. R. (2018). Understanding the accessibility of smartphone photography for people with motor impairments. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–12.

Sontag, S. (2001). *On photography*. Macmillan.