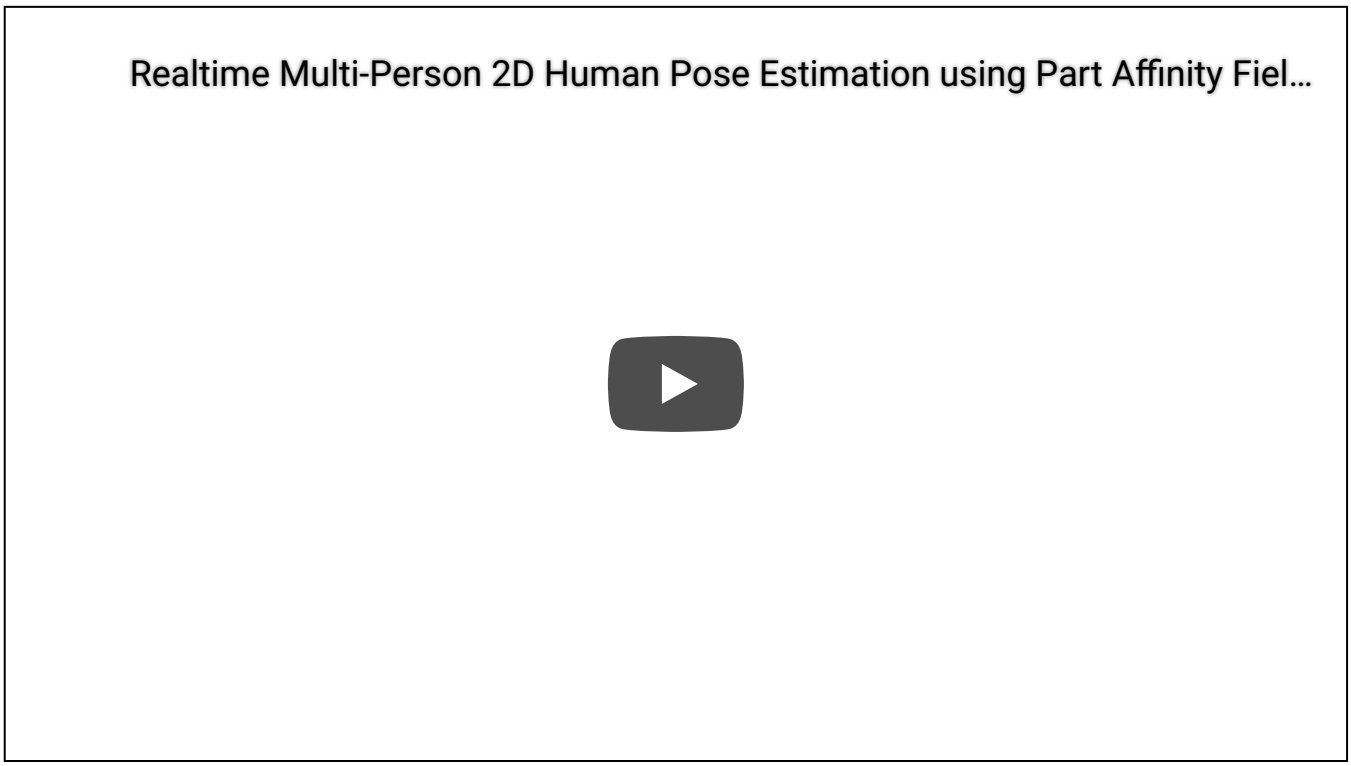
Human pose estimation using OpenPose with TensorFlow (Part 1)

Ale Solano

Oct 2, 2017 · 6 min read

This is the first part of a series of blog articles. Part 2 here.

Hey! Take a look at this:



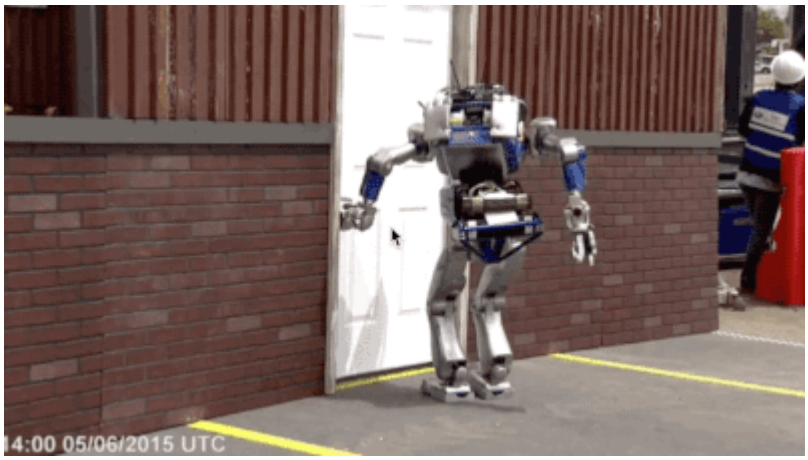
This is happening.

Show me a picture of yourself and I will know immediately what you're doing with your body. No need for a Kinect, your smartphone camera is enough. Applications are endless.

Just imagine yourself trying up new clothes without going to the shop, just using a mirror and your smartphone. Or raising your hand waiting for a self-driving taxi to pick

you up. Or feeling tired at home and taking a sit on the air just to see how a robotic chair appears under your butt.

This also could mean that a humanoid robot could predict your movements and beat you in a Kung Fu battle. But don't worry, we're not on that state yet.



State of the art Terminator

It's called OpenPose and, according to its Github readme, "OpenPose is a library for real-time multi-person keypoint detection and multi-threading written in C++ using OpenCV and Caffe".

CMU-Perceptual-Computing-Lab/openpose

openpose - OpenPose: A Real-Time Multi-Person Keypoint Detection And Multi-Threading C++...

github.com

Deep learning (what else) is the magic trick here. The OpenPose library is built upon a neural network and has been developed by Carnegie Mellon University with astounding help of COCO and MPII datasets. Because, you know, the power of deep learning is proportional to the quality, vastness and availability of datasets.

There's just so much I could do with this. However, it's built in Caffe. And I don't feel so comfortable with Caffe. There are plenty of Deep Learning libraries, yet the mainstream library (and the one that I use the most) is TensorFlow. Porting OpenPose to TensorFlow would make this amazing tool more accessible to the world. And easier for me.

1. Test OpenPose on a video

First thing first: let's try and enjoy this.

Just after discovering the power of OpenPose, I decided to test it out myself. But everything just fell apart when I noticed I needed CUDA to build OpenPose. The fact is that I do have an NVIDIA GPU (940 MX) inside of my laptop… but I can't access it with CUDA. Why?

1. Drivers don't work on Ubuntu 16.04 because my laptop is sort of designed for Windows only.
2. I can't install CUDA on Windows since I don't have a Visual Studio license.

Solution: use cloud services. I am a recent graduate of the Deep Learning Foundations Nanodegree at Udacity and, during the course, they provide us with a 100$ credit to use it on an Amazon EC2 instance, so we had (and still have) access to a machine with a powerful GPU and (yes!) CUDA already installed.

```
$ git clone https://github.com/CMU-Perceptual-Computing-
Lab/openpose.git
```

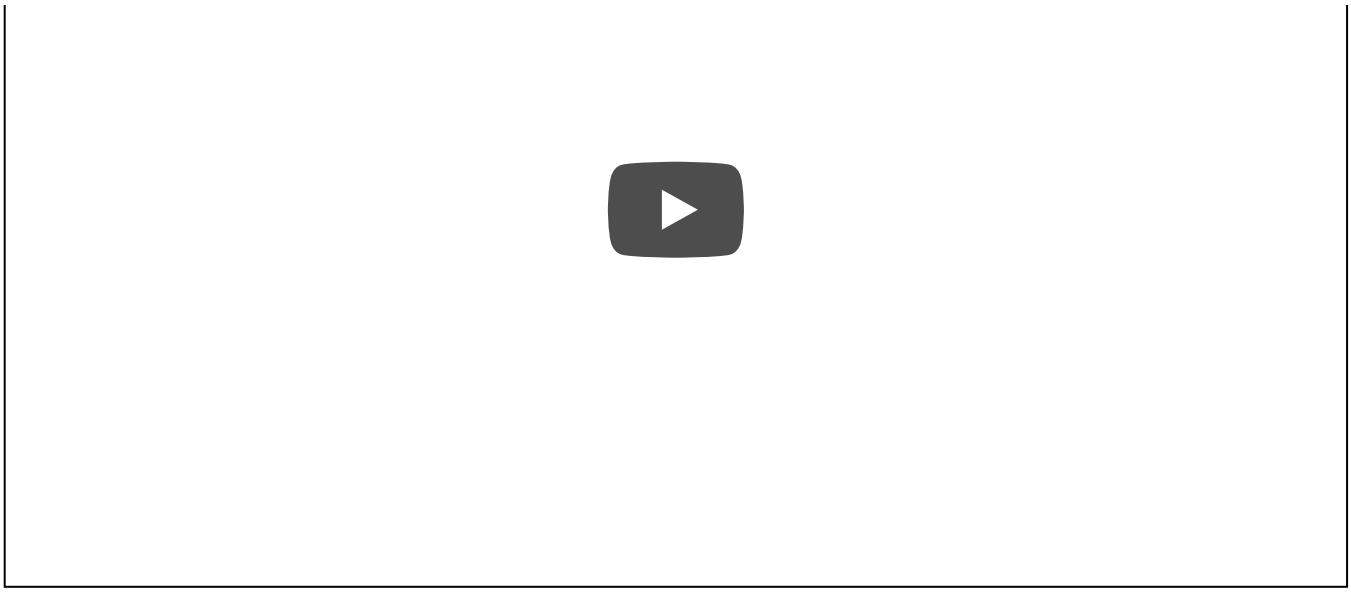
In order to make the installation of the OpenPose library, I followed the Script Compilation steps beginning in step 1.2, since CUDA was already installed. The next steps worked unbelievable well, though it seems that they have come up with an easiest installation method in the last two weeks. I have not tested it yet.

So, now that I had everything set up, it was time to see the magic happen. Following the documentation, I ran OpenPose on a movie clip.

```
./build/examples/openpose/openpose.bin --video clockwork.avi --
write_video output/result.avi
```

And this is what I got:

OpenPose test on A Clockwork Orange



Great! It works!

Note: actually, the first clip I tried was a WWE fight between The Undertaker and The Heartbreak Kid, Shawn Michaels. It looked like a great experiment to check if OpenPose could follow their movements, but I forgot the crowd factor. The output video was full of tiny skeletons due to the dozens (hundreds, sometimes, depending on the camera shot) of people in the background.

2. OpenPose models in TensorFlow

We could be testing OpenPose on thousands of videos and GIFs and never get tired, but we should focus on our goal: port it to TensorFlow so we are able to use this technology in amazing applications.

Once we know everything is set up properly, the next step is to convert the models in a TensorFlow format.

OpenPose gathers three sets of trained models: one for body pose estimation, another one for hands and a last one for faces. And each set has several models depending on the dataset they have been trained on (COCO or MPII).

So let's begin with the body pose estimation model trained on MPII. We need two files: one that describes the architecture of the model (.prototxt) and one that stores the variables values of the model (.caffemodel). And, in order to run the modelos on TensorFlow, we need three checkpoint files (.ckpt). That's we are seeking now.

With these two files, and TensorFlow GPU installed, we can use the next GitHub repository to obtain its equivalent files in TensoFlow.

ethereon/caffe-tensorflow

caffe-tensorflow - Caffe models in TensorFlow

github.com

The key script here is `convert.py`, which takes those two files as arguments along with paths for the new couple of files.

```
convert.py pose_deploy_linevec_faster_4_stages.prototxt --caffemodel
pose_iter_160000.caffemodel --data_output_path data/output/path.npy
--code_output_path code/output/path.py
```

This will return a Python class that describes the architecture (*graph*) of the model and an .npy file storing the values of the TensorFlow variables. Hey! Those are not .ckpt files! How can I use them in TensorFlow? I have been working on it.

alesolano/npy2ckpt

npy2ckpt - Convert deep learning models from .npy format to .ckpt format

github.com

This GitHub repo aims to convert from .npy to .ckpt. I developed it myself for these kind of issues so, if you use it and find something strange, tell me! *TODO: improve the documentation and choose a license.*

Cool! Finally we can run an OpenPose neural network on TensorFlow! What are we waiting for? Let's test it.

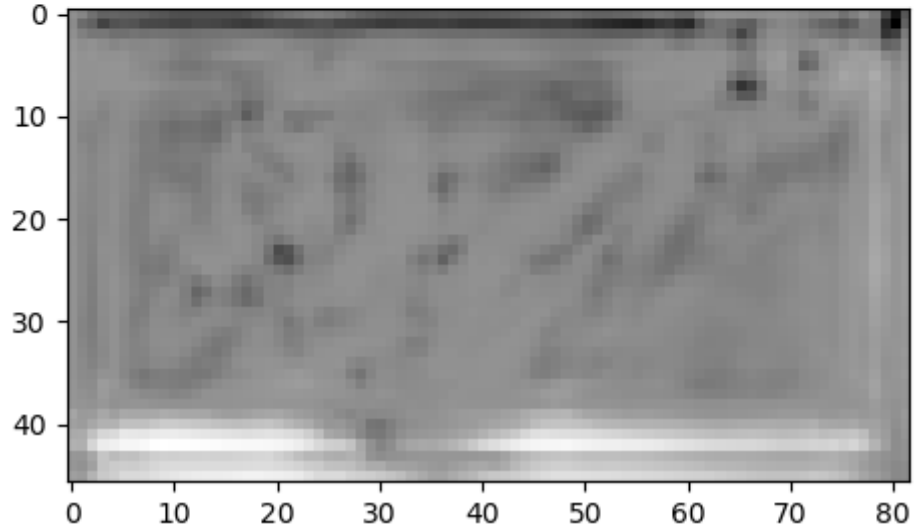
```
import tensorflow as tf
import cv2

### GRAPH ###
tf.reset_default_graph()

saver = tf.train.import_meta_graph('/tmp/openpose_model.ckpt.meta')
```

```
inputs = tf.get_default_graph().get_tensor_by_name('Placeholder:0')
body = tf.get_default_graph().get_tensor_by_name('concat_stage7:0')

print('The input placeholder is expecting an array of shape {} and
type {}'.format(inputs.shape, inputs.dtype))

### IMAGE ###
img = cv2.imread('chuck-norris-kick.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
res_img = cv2.resize(img, (656, 368))
prep_img = res_img.reshape([1, 368, 656, 3])

### SESSION ###
with tf.Session() as sess:
    saver.restore(sess, '/tmp/openpose_model.ckpt')

    output_img = sess.run(body, feed_dict={
            inputs: prep_img
        })

print(output_img.shape)
```

And the result is an image with 57 layers of depth. Mmmm… This is not exactly what I want. Why 57 layers?

- 18 layers for body parts location
- 1 layer for the background
- 19 layers for limbs information in the x direction
- 19 layers for limbs information in the y direction

Here I show the input image and the combination of the 18 first layers of the output, corresponding to the joints of the body.



It's easy to notice that OpenPose has correctly identified Chuck's two hands along with his left hip and left foot. The other joints are not that obvious to see. With the information of the other 39 layers it would generate a perfect skeleton for the lovely Mr. Norris.

Is there a way to combine joints and limbs? Of course. OpenPose is not just a set of trained neural networks, it's an entire library. And one of the functions inside this library is `bodyPartConnectorCaffe.cpp`. And, yes, it's again programmed with Caffe, but its conversion to TensorFlow is not that simple.

3. OpenPose library in TensorFlow

This is the final step of all. However, it's the most tricky. It's so tricky that it's not finished yet. Not even started.

I'm just researching this right now, and it seems that the developers of OpenPose have made the work as easiest as possible. The Caffe code is perfectly isolated and there are only a few C++ files that depend on Caffe. These files are:

```
core/maximumCaffe.cpp
core/nmsCaffe.cpp
core/netCaffe.cpp
core/resizeAndMergeCaffe.cpp
pose/poseExtractorCaffe.cpp
pose/bodyPartConnectorCaffe.cpp
```

So I have two options now:

- Build the entire library in Python from scratch.
- Porting those 6 files to TensorFlow using the TensorFlow C++ API.

I guess I'll go with the second option. Something is for sure: I'll need to deal with Bazel. Wish me luck.

And… if anybody is into helping or is currently working on this, just get in touch with me!! See you!

TensorFlow Openpose Deep Learning Robotics Virtual Reality

About Help Legal