Personalized human video pose estimation

MATLAB code for propagating human pose annotation throughout a video, as detailed in the paper:

J. Charles, T. Pfister, D. Magee, D. Hogg and A. Zisserman "Personalized human video pose estimation", CVPR 2016.

The code has been tested to work in both Windows 7 and Linux and is also equipped to run across a CPU cluster.

Code is provided by James Charles (jjcvision@gmail.com).

What it does

Provided with:

- A video of a person,
- Dense optical flow (e.g. by using DeepFlow),
- A few initial pose annotations obtained either manually or automatically,

this code propagates initial pose annotations across the whole video to be later used for personalizing a generic pose estimator, such as [2].

What is included

- All stages of propagation, as detailed in [1] are included i.e. spatial matching, temporal propagation and self-evaluation.
- A script for setting up the training material to fine-tune the model of [2] on propagated pose annotations.
- Data for running the demo (360MB): demo_data.zip

Prerequisites

Install the following prior to running:

- MATLAB 2012a or later (may work on earlier versions but untested)
- MATLAB Image Processing Toolbox
- VLFeat 0.9.16 or later.
- Deepflow (used for running on your own video, but not required for the demo).

Self-contained packages

The following packages are also required but are included with this code so as to be self-contained:

- FLANN http://www.cs.ubc.ca/research/flann/
- liblinear https://www.csie.ntu.edu.tw/~cjlin/liblinear/

FLANN mex files are pre-compiled. If there are problems using these then please download, install and compile FLANN from: http://www.cs.ubc.ca/research/flann/. If you get the MATLAB error: `cannot open with static TLS` when running the demo, restarting MATLAB normally fixes the problem.

Setup

Compile mex files from within MATLAB:

```
>> compile
```

Setup MATLAB paths:

```
>> set_paths
```

Running the demo

First download the demo video, optical flow file and initial pose annotations (360MB): demo_data.zip

Extract the contents of the zip and retain the folder structure. Initial pose annotations were recovered using the automatic methods described in [1].

Set paths

Folder options in `./options/part_detector_options.m` need to be set according to your file system before running the demo:

```
folder.main = 'folder where you extracted the demo data';
```

Also, set the experiment folder to somewhere you would like to cache the output. You will require around 600MB. The default is in the current working director.

```
folder.experiment = 'folder where I want to store the cache';
```

Chose input video

There is the option of running on one of two demo videos. A very short video (10s) useful for testing the installation: `E7ULR-yfNnk_vshort`. And a longer video (1min) `E7ULR-yfNnk_cut` where more benefit of personalization is observable. In general, the longer the video, the greater the benefit from pose propagation. Set the chosen video in `demo.m`

```
videoname = 'E7ULR-yfNnk_vshort';
```

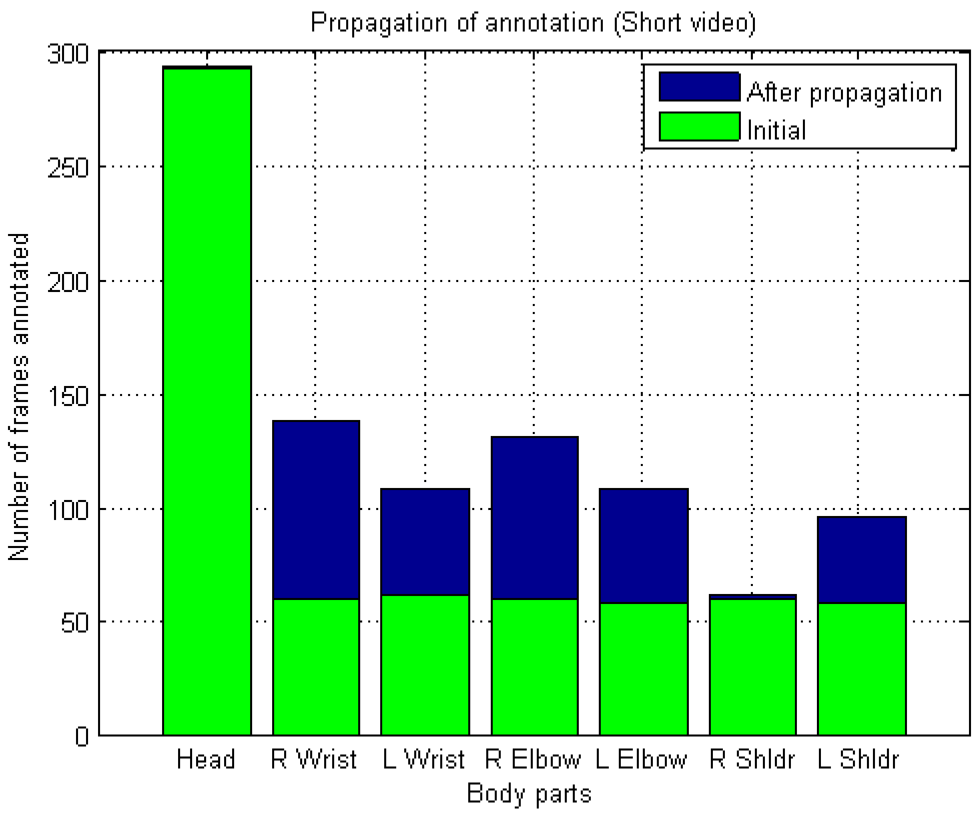
For best pose improvements on the longer video, we recommend setting the optical flow step size `opts.flow.stepsize` in `./options/propagation_options_youtube.m` to `30`. When using the shorter video you may leave this set on `3` for speed.

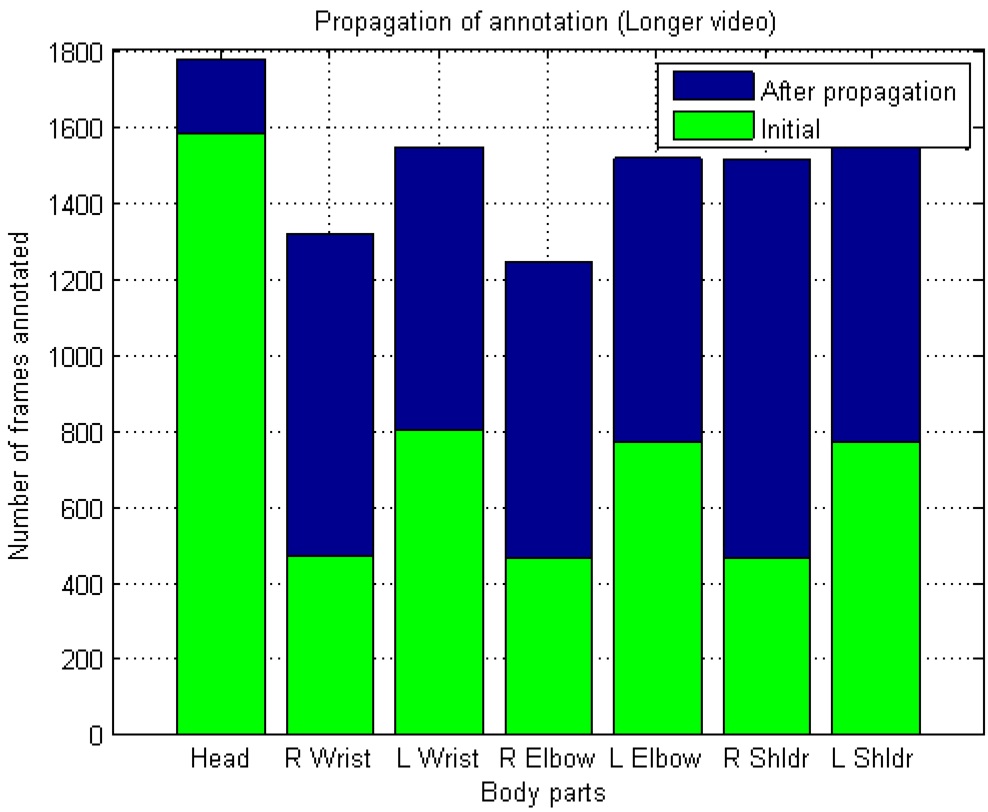
Running the demo

Run the demo:

```
>> demo
```

First, a visualization of the initial pose estimates on the demo video will play. Afterwards, pose propagation will run for one iteration and then a visualization of propagated poses will be displayed. Depending on the demo video used, one of the following propagation coverage graphs will be displayed:





Using multiple CPUs

Using one cpu, the short demo will complete in approx. 1 hour, the longer video taking approx. 4-6 hours. On a 12 core machine the longer demo video will complete in approx. 30 minutes. Multiple CPUs can be requested by changing the variable `num_cpus` in `demo.m` provided you have the `Parallel Processing Toolbox` installed. Memory requirements for running the longer demo is around 1.5GB and will increase with the number of cpus requested if running on one machine.

Running on your own video

Resize the video so that the person has shoulder width approx. `100px` wide.

Copy your video into the `videos` folder within the demo, e.g. if your video is called `myvideo.avi`, place this in the folder `./demo_data/videos/`.

Compute dense optical flow across the whole video (we recommend using DeepFlow from http://lear.inrialpes.fr/src/deepflow/). Dense optical flow needs to be stored in a `.mat` containing the array `flow` and `minmax`.

A function called `collect_optic_flow` for producing a flow file in this format is provided but requires Deepflow to be installed. Set the path to the compiled binary of Deepflow in `propagation_options_youtube.m`

```
opts.deepflowstatic = '/DeepFlow_release2/deepflow2-static';
```

Compute the required dense optical flow file

```
>> collect_optic_flow('myvideo','youtube');
```

The array `flow` is a `uint8` array of size `frame_height x frame_width x 2 x N`, where `N` is the number of video frames and the 3rd dimension holds the `u` and `v` flow vector fields, respectively.

The flow vectors are scaled (per frame) between 0 and 255 according to the minimum and maximum values within a frame.

The array `minmax` is a `double` array of size `2 x 2 x N`. Rows represent `u` and `v` respectively and the columns are the min and max flow values respectively, per frame.

These arrays are saved in `./demo_data/optical_flow/` to a file called `myvideo.mat`

Next, produce initial body joint annotations (2D locations of 7 upper body joints) and save to a MATLAB structure called `detections.manual`:

- `detections.manual.locs` is a `2 x 7 x N` array representing pose annotations for a selection of `N` frames. A pose for each frame is given by a set of `x` and `y` (row 1 and row 2) body joint locations for the head, right wrist, left wrist, right elbow, left elbow, right shoulder and left shoulder, respectively, per column (person centric).
- `detections.manual.frameids` = `1 x N` array indicating the annotated frame IDs.

Save this struct:

```
>> save('./demo_data/initialisation/myvideo.mat','detections');
```

If a body joint is unknown for a particular frame, then this can be indicated by setting the `x` and `y` coordinates in `detections.manual.locs` to `-999`

Finally, initial poses can be propagate for one iteration:

```
>> detections = propagate('myvideo','youtube',...
    './demo_data/initialisation/myvideo.mat',1,1,1);
```

Pose output is visualized with:

```
>> show_skeleton('./demo_data/videos/myvideo.avi',...
    1,detections.manual.frameids,detections.manual.locs,0);
```

Settings

A number of useful parameter settings can be set for adjusting speed of execution and performance. These can be found in the option files `./options/part_detector_options_youtube.m` and `./options/propagation_options_youtube.m`.

We recommend experimenting with the following:

- `opts.model.forest.numtrees` for setting the number of trees in the random forest part detector. Lower numbers will reduce training time, ensure this number is a multiple of 2.
- `opts.model.rotations` for setting rotation augmentation during part detector training, setting less rotations will reduce memory requirements during training.
- `opts.flow.numneighbours` step size used for selecting frames to temporally propagate (i.e. set to 1 to select every frame or 2 to select every 2nd frame, etc.). Higher values will increase speed.
- `opts.flow.stepsize` temporal window used to propagate forward and backwards in time. In [1], this was set to 30.

Personalizing the ConvNet of [2]

Personalization is achieved by fine-tuning the ConvNet of [2] (Caffe model) on the propagated pose estimates.

The function `fusion.setupFinetuningCropped` is designed to simplify the task of setting up the training data so the model can be fine-tuned in Caffe.

Various options should be set correctly prior to running this function and a detailed guide is provided:

Fine-tuning Caffe model of [2]

Download the model and source code of [2] from: https://github.com/tpfister/caffe-heatmap and install Caffe.

Set the fine-tuning options in `./options/propagation_options_youtube.m` and provide absolute paths.

Set filename to generic ConvNet model

```
opts.cnn.finetune.model_filename = 'caffe-heatmap-flic.caffemodel';
```

Set folder to where training data will be generated

```
opts.cnn.finetune.ramdisk_folder = '/mnt/ramdisk/data/';
```

Set folder to where snapshots and training scripts are created

```
opts.cnn.finetune.main_save_folder = '/fusion_training/';
```

Set the root folder of the Caffe install

```
opts.cnn.finetune.cafferoot = '/caffe-heatmap/';
```

The `demo.m` script shows an example use of the function `fusion.setupFinetuningCropped`. In `demo.m` set `isFineTune` to `true` and re-run the script. You will then be signaled to run the generated `train.sh` bash script.

Model snapshots are saved in the snapshot folder within `opts.cnn.finetune.main_save_folder`.

Note. Body joint locations which are missing from frames are indicated by having the value `-999`. This occurs in frames where all body joints have not been successfully propagated, resulting in parts of poses. Therefore, the Euclidean loss function for [2] should be altered so as to provide zero loss for these missing body joint locations.

Citation

If you use this code then please cite:

```
@InProceedings{Charles16,
   author       = "Charles, J. and Pfister, T. and Magee, D. and Hogg, D. and
Zisserman, A.",
   title        = "Personalizing Human Video Pose Estimation",
   booktitle    = "IEEE Conference on Computer Vision and Pattern Recognition",
   year         = "2016",
}
```

References

[1] J. Charles, T. Pfister, D. Magee, D. Hogg and A. Zisserman "Personalized human video pose estimation", CVPR 2016.

[2] T. Pfister J. Charles and A. Zisserman "Flowing ConvNets for Human Pose Estimation in Videos", ICCV 2015.