Project no. IST-34107
Project acronym: ARTTS
Project title: Action Recognition and Tracking based on Time-of-flight Sensors

Object Tracking Toolbox

Duration of the project: October 2006 – September 2009

Partners of the project consortium:

INB, University of Lübeck, Germany
IMM, Technical University of Denmark, Denmark
LAPI, University Politehnica Bucuresti, Romania
Swiss Center for Electronics and Microtechnology, Switzerland
SensoMotoric Instruments GmbH, Germany

Project website:

www.artts.eu

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

1 Introduction

The ARTTS Object Tracking toolbox is a library of MATLAB routines that provide functionality for tracking persons and objects in TOF images and for labelling ground-truth data. Here is an overview of the functionality included in the toolbox:

- Labelling upper body pose, positions of faces and positions of facial features in images
- Facial feature tracking
- Multiple person tracking
- Gait Analysis

2 Licence

The ARTTS Object Tracking toolbox (code and accompanying documentation, including this document) is provided under the following licence:

This work was developed within the ARTTS project (www.artts.eu), which is funded by the European Commission (contract no. IST-34107) within the Information Society Technologies (IST) priority of the 6th Framework Programme. Neither the European Commission nor the ARTTS Consortium can be held responsible for any consequence of using this code and the accompanying documentation.

3 Installation

Since some components in the Object Tracking toolbox require routines from the Image Processing toolbox, first make sure that the Image Processing toolbox has been installed and added to the MATLAB search path. Then, copy the directory `ot_toolbox` to a suitable location, then add this directory and the subdirectory `ot_toolbox/gait` to MATLAB's search path. The best way to do this is to include the corresponding `addpath` calls in your `startup.m` file.

4 Components

This section gives a more detailed overview of the components included in the toolbox. For each of the major components, we will give a description of the functionality, including links to relevant publications, a list of the routines that implement this functionality, and an example of how to use the routines.

For a detailed description of the routines, refer to the documentation that is included directly in the source code of the routines; this can be accessed from MATLAB by typing

`help` <routine name>

For example, the documentation for the routine read_artts_range can be displayed by typing

```
help read_artts_range
```

4.1 Labelling

This component provides utilities for labelling the positions of objects in TOF images. These labelled images can then be used to train detectors and compare detection results against a known ground truth.

The following routines are provided:

`label_nose`
: Label position of nose (or other facial features)

`label_faces`
: Label position of faces

`label_pose`
: Label upper body pose (positions of head, shoulders, and hands)

Here is an example of how to label facial features:

```
nose=label_nose('data/nose_db/nose_db.arttsidx');
```

In each image, click on the position of the nose. The labelled positions are returned as a 2×n matrix, where n is the number of images; each column of the matrix contains the position of the nose (row and column indexes) in the corresponding image.

Here is an example of how to label faces:

```
label_faces('data/nose_db/nose_db.arttsidx', 'faces.arttsidx');
```

In each image, drag the blue lines to enclose the face, then press the space bar to move to the next image. (See the documentation in `label_faces.m` for more details.) The labelled positions are saved in a new database index file `faces.arttsidx`.

Here is an example of how to label upper body pose:

```
label_pose('data/pose_db/pose_db.arttsidx', 'labelled_pose.arttsidx');
```

In each image, drag the nodes to the middle of the head, the shoulders, and the hands. (The red edge of the model represents the person's left arm, the green edge represents the right arm.) Press the space bar to move to the next image; see the documentation in `label_pose.m` for more details. The labelled positions are saved in a new database index file `labelled_pose.arttsidx`.

4.2 Facial feature tracking

This component provides functionality for tracking facial features using geometric features (the *generalized eccentricities*). These are combined with a very simple threshold-based classifier to produce a robust facial feature detector; typically,

this detector is used to track the position of the nose. For more details, see the following paper:

> Martin Böhme, Martin Haker, Thomas Martinetz, and Erhardt Barth. A facial feature tracker for human-computer interaction based on 3D Time-of-Flight cameras. International Journal of Intelligent Systems Technologies and Applications, 5(3/4):264–273, 2008.

The following routines are used to train and evaluate the detector:

`box_classifier`
: Trains a simple “box” classifier on labelled training images

`nose_detector`
: Evaluates the facial feature detector and returns the position of the facial feature in one or several images

`show_nose_detections`
: Applies the facial feature detector to a set of test images and visualizes the result

Here is an example of how to use the facial feature detector:

```
% Read training images
images=read_artts_db('data/nose_db/nose_db.arttsidx');

% Load positions of noses
noses=load('data/nose_db/nose.txt');

% Train classifier
classifier=box_classifier(4, images, noses, default_feat_opts());

% Run classifier on training images and display results
show_nose_detections(images, classifier, 0, default_feat_opts());
```

In this example, the detector is applied to the same images that were used to train it. For a real test, the detector would of course have to be trained on new, unseen images.

4.3 Gait analysis

This component does gait tracking and analysis. The main file `TrackGait` is a script-file that takes the user through every step in the process, from raw input collected by the `TOFrecorder`, through the tracking performed by `runFrame`, which finds the pose in a given frame, to the actual gait analysis done by `gaitAnalysisTreadmill`. For more details, see the following paper:

> Rasmus R. Jensen, Rasmus R. Paulsen and Rasmus Larsen. Analysis of gait usig a treadmill and a Time-of-flight camera. In Dynamic 3D Imaging – Workshop in Conjunction with DAGM, volume 5742 of Lecture Notes in Computer Science, pages 154-166, 2009.

The following routines are used for recording and analysing gait sequences:

`compile_gait_mex`
: Routine that compiles the gait analysis MEX files. Execute this once before using the gait analysis routines.

`TrackGait`

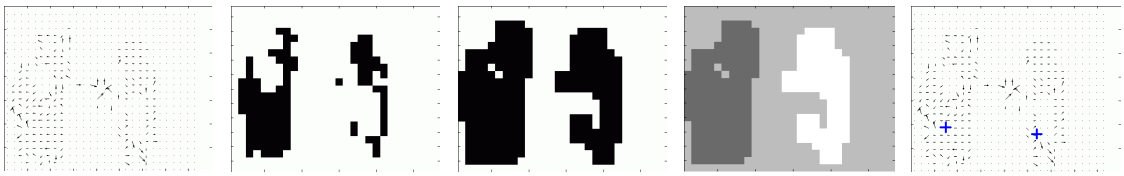
Script file that takes the user through every step in the gait analysis, from selecting the input image sequence in the data folder to actually performing the gait analysis. When started, the script prompts the user for the input image sequence and the directory in which the results (videos and diagrams) should be saved. The script can be run as a whole or cell-wise. The input data file requires the first frame to be without the subject, and the subject should walk from left to right. A sample input file is contained in the subdirectory 'data'.

`TOFrecorder`

GUI that works as a recorder for a connected TOF camera. Films sequences and stores them in the format used by `TrackGait`. The subject should walk from left to right in the frame.

4.4 Multiple person tracking (LAPI)

This component tracks multiple persons in a TOF image sequence. The first step is motion estimation, using an improved version of the block-matching algorithm (with zero-motion detection) that is less sensitive to noise. Then, to determine the precise number of moving objects and their location, the algorithm follows these steps: a) construct a binary image (0: background, 1: moving object) from the motion vectors; b) erode this binary image; c) dilate the previously obtained image; d) label the connected components; and e) determine the centres of the previously obtained objects. These steps are depicted in Figure 1. An example (a frame) is shown in Figure 2.



a) b) c) d) e)

Figure 1: a) binary image; b) erosion; c) dilation; d) labeling; e) determine objects' centres

Figure 2: Multiple person tracking example

Due to the high sensitivity of the motion estimation, the tracking is practically never lost while the person is present in the scene.

For more details, see the following paper:

Şerban Oprişescu, Mihai Ciuc, Vasile Buzuloiu. Histogram and motion based intrusion detection and tracking algorithms for ToF cameras. Proceedings of the IEEE International Symposium on Signals, Circuits & Systems (ISSCS), Iasi, Romania, 2009.

The following routine implements the multi-person tracking:

`mult_track`

Tracks multiple persons in an image sequence. The routine takes the name of the image sequence as input and returns a tracking matrix containing the persons' coordinates in each frame.

The function adapts to a changing number of persons in the scene. Each person is identified in the scene by a yellow marker in the form of a star or cross to indicate whether the person is approaching or leaving the camera, respectively.

Here is an example of how to use this routine:

```
track_pers=mult_track('data/movie_2_pers.mat');
```