Technical Documentation

SRILF 3D Face Landmarker

Federico M. Sukno, Paul F. Whelan

March, 2016

Version 1.0

TABLE OF CONTENTS

1. OVERVIEW 3

2. INSTALLATION 3

3. BASIC USAGE 3

3.1 Example 4

3.2 Generated output 5

4. CONFIGURATION MODELS AND OPTIONS 6

4.1 Configuration model 6

4.2 Accuracy vs speed 6

4.3 Occlusions 7

4.4 Visualization 7

4.5 Multi-threading 8

5. VISUALIZATION TOOL 8

6. SAMPLE SURFACES 8

7. LICENSE 9

8. REFERENCES 10

1. Overview

This document describes the use of the automatic landmarking software srilf3dFL version 1.0x and is an implementation of the Shape Regression with Incomplete Features (SRILF) algorithm. For technical details, please refer to [1]. The goal of this command-line application is to locate facial landmarks on a facial scan in a fully automatic manner.

Program name	srilf3dFL.exe
Inputs	Facial scan (PLY or OBJ format)
	Configuration file
Output	Landmark coordinates (text-like file, TXT or PTS formats)
External files needed	vcomp90.DLL
	blas_win32_MT.DLL
	lapack_win32_MT.DLL

2. Installation

This software uses a number of external libraries that are required for correct execution:

- Lapack and Blas: pre-compiled DLLs are provided together with the software. These libraries must be placed in the same folder as the executable.
- Microsoft Visual C++ Redistributable: These libraries must be installed on the computer. If they are not, an error will be generated when trying to execute srilf3dFL (typically the system will indicate that a DLL or command is missing). In that case, you need to install the required libraries, which are freely available on the web:
    - Microsoft Visual C++ 2008 SP1 Redistributable Package (x86) http://www.microsoft.com/downloads/en/details.aspx?familyid=A5C84275-3B97-4AB7-A40D-3802B2AF5FC2&displaylang=en

3. Basic usage

The software operates under Windows operating systems and provides a simple command-line interface. It consists of an executable file and a configuration file (that contains the model that will be used for landmark identification). At least two input arguments must be specified, namely the configuration file and the input file containing the mesh to process:

```
srilf3dFL.exe modelName.cfg inputMesh
```

The above command will run the algorithm and display the results (the identified landmarks) on a graphical window, without saving the results or indicating coordinate values. Further command-line options can be used to modify this behavior (see below).

The model used by srilf3dFL is specified by the configuration file. It is expected that the performance of the software will depend on the acquisition device and the type of faces used for training; hence it is recommended to use the appropriate model depending on the input data to be processed.

The input mesh must contain a triangulated surface of a facial scan, either in PLY or OBJ file formats. During execution, the program displays information about the input data and the progress of the different algorithmic steps, which can vary depending on the configuration file that is specified. For the most time consuming steps, an estimate of the execution time is also indicated. Note that the number of processing steps can vary depending on the configuration file that is used.

3.1 Example

A simple example to illustrate the basic use is provided below. As explained earlier, srilf3dFL takes two inputs: a configuration file with the model to use and the mesh to be landmarked. For this example we use the following:

- Model: srilf3dFL_HHLaser_HQ.cfg - This model was created using a dataset of high-quality facial surfaces captured with the hand-held laser scanner. The performance of the software depends heavily on the configuration model that is used and the model, in turn, depends on the training data. This dependence regards both the quality of the 3D scans as well as the accuracy of the annotations [2]. See section 4.1 for further details.
- Scan: in principle, a scan acquired under similar conditions to those of the specified configuration model. In this case we use one of the sample scans provided with the software.

The example was run using the following command:

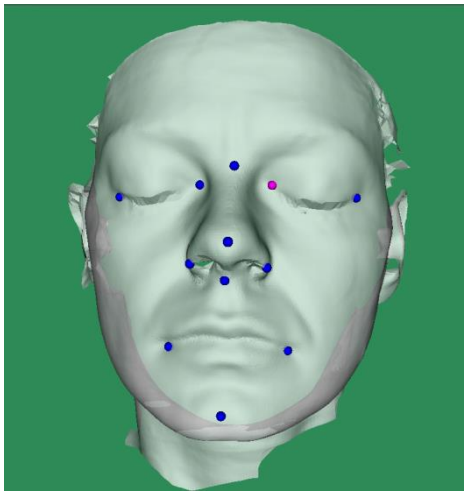
```
srilf3dFL.exe srilf3dFL_HHLaser_HQ.cfg SampleFace.ply
```

During execution, a number of status bars indicate the progress of the different steps of the program (see figure in the next page). Time is reported separately for the computation of descriptors and for the combinatorial search (landmark identification), but incrementally within these two blocks. In the example, the total run-time was 5.91s, which can be decomposed in 2.01s to compute all 5 descriptors plus 3.90s for the landmark search. Notice that, because APSC descriptors are used [3], the computation of the 1st descriptor is always the most time consuming (comparable to the computational load of a 3D Shape Context) and all subsequent ones are obtained at a marginal additional cost.

Since srilf3dFL was called without parameters, landmark coordinates were displayed on the screen, indicating also whether their location was FOUND (based on vertices with high-scoring descriptors or "candidates") or if they were INFERRED (based on model-statistics). A graphic window is also automatically opened to show the located landmarks on the input mesh (see figure to the right).

Landmarks are color-coded to indicate where they were found (blue) or inferred (magenta). In this example, only one landmark was inferred: endocanthionL (left inner-eye corner). Again, this is also indicated in the information made available on the screen.

```
SRILF 3D Face Landmarker, rev 1.0
(C) Dublin City University - All rights reserved.
Provided exclusively for research use under CC BY-NC-ND licence.
Type srilf3dFL -help for detailed information.

Reading LaserData\SampleFace.ply
        70050 Verts and 139460 Triangles
Preprocessing
------------------------------------------------------

Computing descriptors (1 of 5)
[##################################################] 1.85398 s

Computing descriptors (2 of 5)
[##################################################] 1.86016 s

Computing descriptors (3 of 5)
[##################################################] 1.94978 s

Computing descriptors (4 of 5)
[##################################################] 2.00597 s

Computing descriptors (5 of 5)
[##################################################] 2.01211 s

Landmark identification
*[##################################################] 3.90007 s

Face3D-id   x-coord    y-coord    z-coord    name     found/inferred
-------------------------------------------------------------------
L13     -23.32      43.31      -21.39     nasion     FOUND
L01     -24.33      9.00       -2.75      pronasale    FOUND
L28     -25.53      -57.48     -32.11     pogonion    FOUND
L17     -37.82      38.02      -35.45     endocanthionR    FOUND
L16     -7.47       38.49      -33.78     endocanthionL    Inferred
L24     -48.18      -30.05     -38.40     cheilionR    FOUND
L23     0.73        -30.22     -36.49     cheilionL    FOUND
L03     -39.83      2.64       -21.82     alareR    FOUND
L02     -8.98       1.83       -20.37     alareL    FOUND
L06     -25.53      -3.94      -19.17     subnasale    FOUND
L15     -72.96      34.17      -47.13     exocanthionR    FOUND
L14     28.81       36.00      -42.87     exocanthionL    FOUND
```



3.2 Generated output

If called with only two arguments (config file and input mesh), the resulting landmarks are displayed on top of the input mesh in a new graphic window that is opened at the end of the landmark identification process. A third parameter can be specified so that the positions of the identified landmarks are written to a file:

```
srilf3dFL.exe modelName.cfg inputMesh outputFile
```

If doing so, the program will not display the results nor create a graphic window. It is possible to obtain the landmark locations simultaneously written to a file, to the screen and displayed in a graphic window by adding -display after the output filename:

```
srilf3dFL.exe modelName.cfg inputMesh outputFile -display
```

In case the output is written to a file, two formats are possible, which are automatically selected depending on the extension of the file:

- Text format (.TXT extension): plain text format, using a separate line for each landmark. Each of these lines indicates: i) the landmark index (following Face3D project definitions[1]), ii - iv) the x, y and z coordinates identified for the landmark, v) the name of the landmark and vi) whether the landmark position has been determined based on a descriptor candidate (Found) or inferred from model statistics (Inferred).
- Landmark format (.PTS extension): plain text format compatible with Landmark 3.0[2], using a separate line for each landmark. Each of these lines indicates: i) the landmark index and ii - iv) the x, y and z coordinates identified for the landmark. The resulting file can be imported directly in the aforementioned software.

[1] http://www.face3d.ac.uk/
[2] http://institute-for-data-analysis-and-visualiz.software.informer.com/

When the results are saved into a file, it is also possible to display them into the same graphic window generated by the landmarker by using vtkViewMeshLmk.exe (see Section 5)

4. Configuration models and options

This section provides a description of the configuration model that is provided by default (including its scope and limitations) and the command-line options that can be specified to control some aspects of the landmarker. Executing `srilf3dFL.exe -help` provides a condensed summary of the latter.

4.1 Configuration model

The configuration model srilf3dFL_HHLaser_HQ.cfg is provided with the software. It has been created from a training set with the following specifications:

- Facial surfaces obtained from a hand-held laser scanner that fuses data from multiple views, therefore obtaining a full coverage of the facial surface.
- Surfaces had a resolution between 1mm and 2mm, with no significant holes or missing parts.
- The subjects that were scanned were mainly Western-European, had their eyes closed (because of laser safety) and were instructed to pose with neutral facial expression.

For optimal performance, input surfaces should meet the above specifications. This means, for example, that performance using low resolution meshes or under marked facial expressions might be suboptimal. In particular, the software automatically checks the smoothness of the input surfaces with respect to the parameters in the configuration model, issuing the following warning message in case of miss-match:

```
***** QUALITY MISSMATCH *****
The input data does not seem to fulfill the characteristics
expected by the configuration file. This might be due to
a missmatch on acquisition devices between the model
and the input mesh. It is strongly recommended to cancel
operation and try again with the appropriate model

Continue anyway (Y/N) ?
```

In case the input surfaces do not meet the requirements of the provided model, it is recommended to either: i) Pre-process the input surfaces appropriately (if possible); or ii) Contact us to check if there is a more suitable model for your needs. Models for the FRGC and Bosphorus databases are already available at: http://fsukno.atspace.eu/Data.htm

4.2 Accuracy vs speed

The software is configured to provide a trade-off between accuracy and speed (with some bias to the former). There are a number of parameters to modify some aspects related to this:

`-findL LL`	Specifies the minimum number of landmarks that should be found based on candidates (i.e. not inferred). The default setting is 4, but it is known that the accuracy is higher when more landmarks are identified. A key element of the SRILF algorithm is that it automatically identifies as many landmarks as possible in a case-by-case basis; using –findL to modify the minimum target can potentially increase the run-time in scans that are particularly difficult

	(which depends on the configuration model that is used).
`-mres MR`	Specifies the mesh resolution to be used (in mm). If the input mesh has a resolution that is higher than the value specified by MR, an umbrella operator is applied so that only a fraction of the input vertices are used for calculations. Smaller values (higher resolution) should produce the best accuracy while larger values produce a speed-up but can potentially impair localization accuracy. Values between 1mm and 2mm are recommended.
`-maxTime SS`	Specifies the maximum run-time to determine if the combinatorial search can make an additional attempt upon non-satisfactory results (in seconds). When the combinatorial search does not reach a solution with at least 4 landmarks (or the value specified by –findL), the algorithm lowers the threshold for candidate selection and re-runs the combinatorial search (this is why using –findL can increase the run-time). However, if –maxTime is specified, the re-run will only happen up to the specified run-time. Notice that the specified time is therefore approximate since it will not be checked until the necessity of a re-run is evaluated. If not specified, the default maximum run-time is 3600 (1 hour).

4.3 Occlusions

Because landmarks are expected to be characteristic points of the input surface, the resulting coordinates are forced to match the nearest vertex of the input mesh. This behavior ensures that the detected landmarks lie on the input surface (which is especially relevant to "inferred" points), but is a problem in the presence of occlusions or missing parts of the surface. In such cases, it is typical that the combinatorial search needs several attempts to find a plausible solution, which considerably increases execution time. Thus, when the combinatorial search exceeds some predefined run-time, occlusion handling is automatically enabled, in a similar fashion to specifying –occ2 at the command line. The following 3 flags can be used to modify the default operation described above:

`-occ DD`	Landmarks are allowed to be off-surface, i.e. they can be floating in the empty space. This is allowed up to DD mm away from the nearest vertex on the input surface.
`-occ2`	Same as –occ but only if a first run of the combinatorial search without occlusions fails to find a suitable solution.
`-noOcc`	Disable all occlusions handling. Identified landmarks are always restricted to lie on the surface. If this is not possible, no solution is returned.

4.4 Visualization

Once processing is completed, the identified landmarks can be either saved to a file (if an output file is specified) or displayed on the screen. The following flags can be used to modify this:

`-display`	Force the information to be displayed on the screen, even if saved into a file.
`-nodisplay`	Disable on-screen visualization. This flag should not be used without specifying an output file.

`-opacity OP`	Set the opacity of the surface used to visualize the identified landmarks. Values range from 0 (fully transparent) to 1 (no transparency). If not specified, the default value of 0.75 is used.

4.5 Multi-threading

By default, most time-consuming parts are ran in parallel with as many threads as cores are present in the system. The following flag allows limiting the maximum number of threads to a specific value.

`-omp N`	Specifies the number of threads to use. The landmarker uses OpenMP for multi-threading.

5. Visualization tool

In order to simplify the visualization of results when they have been saved into a file, an additional command line tool is provided. Its syntax is as follows:

```
vtkViewMeshLmk.exe [flags] inputMesh landamrkFile [landmarkFile2]
```

The square brackets denote optional parameters. Thus, the input mesh and at least one file containing landmarks are required. The input mesh must be in PLY or OBJ format. Landmark files are accepted in .TXT format (as generated by srilf3dFL), .PTS format (Landmark 3.0 files) and .DILM (Dimensional Imaging annotation files).

Additionally, it is possible to indicate a second file containing landmarks. The landmarks from the first file will be displayed in blue and the ones from the second file will be displayed in red. This can be useful to visually compare automatic landmarks with ground truth annotations.

It is also possible to control the opacity of the mesh using the optional flag `-opacity OP` (to set the opacity to `OP`). Values must be between 0 (full transparency - no surface will be displayed) and 1 (no transparency). If not specified, the default value of 0.75 is used.

6. Sample surfaces

A number of surfaces are provided in Samples/ folder to serve as examples that can give an immediate idea of how the software works. These surfaces were captured with the same technology and under compatible settings with the provided model, and illustrate the accuracy that can be expected from the software under normal operating conditions.

Additionally, the Occlusions/ folder contains some more challenging examples where facial surfaces were captured under partial occlusions. When running these examples, you should keep in mind the remarks provided in Section 4.3: if srilf3dFL is used without specifying occlusions handling, the landmarker will firstly attempt to identify facial landmarks under the assumption that there are no occlusions and, after a few unsuccessful attempts, will automatically enable occlusions handling. When we know beforehand that there are occlusions in the input surface, using `-occ` flag can considerably shorten the run-time.

7. License

The SRILF 3D Face Landmarker is provided exclusively for research use under a Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) licence[3]. In brief, this implies that you are free to share (copy and redistribute the material in any medium or format) under the following terms:

- **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **NonCommercial** — You may not use the material for commercial purposes.
- **NoDerivatives** — If you remix, transform, or build upon the material, you may not distribute the modified material.
- **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material.

Visualization and other partial functionality is based on VTK (Copyright (c) 1993-2008 Ken Martin, Will Schroeder, Bill Lorensen. All rights reserved). See www.vtk.org for further information.

Development of this software was supported by the Wellcome Trust, through the Face3D[4] project and the European Commission, through the SP-MORPH[5] project. The joint DCU/RCSI team responsible for the underlying research was composed by the authors listed below.

If using this software in your research, please cite the following publication, which describes the implemented algorithm:

> F. Sukno, J. Waddington, P. Whelan (2015): 3D Facial Landmark Localization with Asymmetry Patterns and Shape Regression from Incomplete Local Features; IEEE Transactions on Cybernetics, 45(9):1717-1730, 2015.

For more information, please contact Federico Sukno (federico.sukno@gmail.com) or Paul Whelan (paul.whelan@dcu.ie).

[3] https://creativecommons.org/licenses/by-nc-nd/4.0/
[4] www.face3d.ac.uk
[5] www.cipa.dcu.ie/face3d/SP_MORPH_Project.htm

8. References

[1] F.M. Sukno, J.L. Waddington and P.F. Whelan. Compensating inaccurate annotations to train 3D facial landmark localization models. In Proc. Workshop on 3D Face Biometrics, Shanghai, China, 2013.

[2] F.M. Sukno, J.L. Waddington and P.F. Whelan, Asymmetry Patterns Shape Contexts to Describe the 3D Geometry of Craniofacial Landmarks. In Computer Vision, Imaging and Computer Graphics -- Theory and Applications. Communications in Computer and Information Science, Volume 458, 2014.

[3] F.M. Sukno, J.L. Waddington and P.F. Whelan. 3D Facial Landmark Localization with Asymmetry Patterns and Shape Regression from Incomplete Local Features. IEEE Transactions on Cybernetics, 45 (9): 1717-1730, 2015.