FexView: Manual

Filippo Rossi,
Institute for Neural Computation, University of California San Diego. Contact: frossi@ucsd.edu.

1. Introduction

FexView is a Matlab (www.mathworks.com) module for the visualization of facial expressions data, and it is included in the toolbox *FexMetrica* (www.github.com). Dr. Filippo Rossi wrote all the code included in *FexView* (Copyright (c) 2014 Filippo Rossi), with the exception of the functions cbfreeze.m and cbhandle.m, written by Carlos Adrian Vargas Aguilera.

1.2 Requirements

In order to run *FexView*, you need to have Matlab installed on your computer. The program was tested on OS X 10, and on Ubuntu 12.04 with Matlab v2013a, v2013b, and v2014a.

1.3 General Description

FexView can be used to display overlays on a 2D image of a face. The underlying image, or "template," can be any of the 9 provided. Alternatively, the user can import a new image. The process to convert that image into a "template" is described in **section 1.4**.

The building blocks for *FexView* are facial muscles. The location and texture of individual muscular fibers are defined for each "template," and the data are displayed on top of these pixels. Pixels within muscular region *X* are indicated in the header of the "template." Action Units are localized on a face based on the muscles that contract/relax in order to generate those action units. Emotions are defined on an image in terms of combination of Action Units. **Table 2** shows conversion from muscle to AUs, and from AUs to emotions.

The coordinates of facial muscles were derived from **[??]**. The correspondence of muscular activity with facial muscle was derived from the Facial Action Coding System **[??]**. However, we simplified this model in order to have only one muscle per Action Units. Additionally, there are several combinations of Action Units that can results in the expression of basic emotions. We used one AUs configuration per emotions, derived from the Emotional Facial Action Coding System **[??]**.

Action Unit	Muscle Used	Emotions
AU 01	*Frontalis pars medialis*	Fear, Sadness, Surprise
AU 02	*Frontalis pars lateralis*	Fear, Surprise
AU 04	*Corrugator*	Anger, Fear, Sadness
AU 05	*Levator palpebrae superioris*	Anger, Fear, Surprise
AU 06	*Orbitocularis oculi (pars orbitalis)*	Joy
AU 07	*Orbitocularis oculi (pars palpebralis)*	Anger, Fear
AU 09	*Nasalis alaris*	Disgust
AU 10	*Quadratus labii superioris*	
AU 12	*Zygomaticus major*	Contempt, Joy
AU 14	*Buccinatorius*	Contempt
AU 15	*Triangularis*	Disgust, Sadness
AU 17	*Mentalis*	
AU 18	*Incisivii labii*	
AU 20	*Risorius*	Fear
AU 23	*Orbicularis oris*	Anger
AU 24	*Orbicularis oris*	
AU 25	*Quadratus labii inferioris*	Fear, Surprise
AU 28	*Orbicularis oris*	

Table 1: **Muscles, AUs and emotions dictionary*.*** The relationships between muscles and Action Units and between combinations of Action Units and emotions were derived, respectively, from the Facial Action Coding System **[??]**, and from the Emotional Facial Action Coding System **[??]**.

2. Templates and "fexwimg" class

2.1 Description of "fexwimg" class

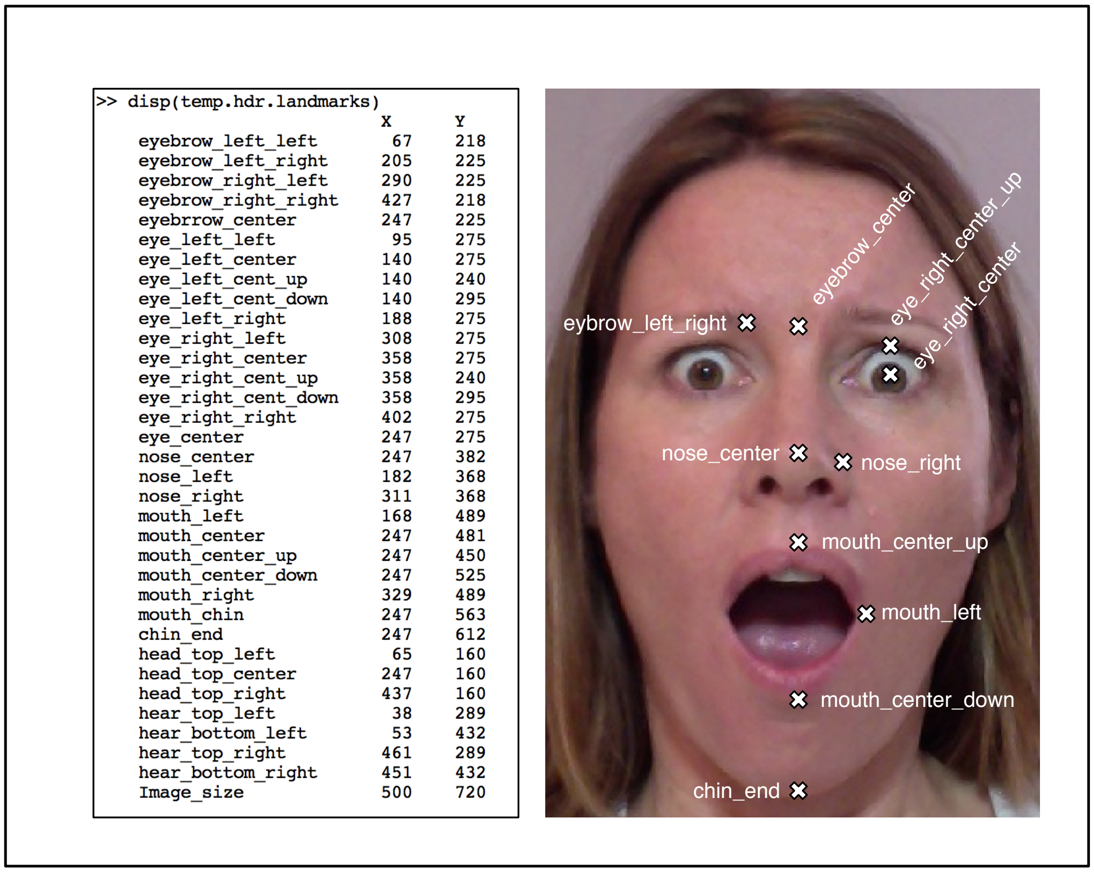
By "template" we refer to an instance of the class **fexwimg**. A **fexwimg** has three main properties:

(1) "name": A name given to the template (e.g. "anger");
(2) "img": The image;
(3) "hdr": The header of the image.

The header of the image is an instance of the class **fexwhdr**. The header of the image contains information about the morphology of the face in **img**. In particular, **fexwhdr** has the following properties:

- “path”: string indicating the path to the original image;
- “imsize”: vector with image size, [*Height, Width, Color Channels*];
- “format”: string indicating the format of the image;
- “landmarks”: dataset with coordinates for face landmarks;
- “muscles”: structure with coordinates and texture of facial muscles;
- “mask”: dataset with X and Y data for the perimeter of the face mask.

The properties “path,” “imsize,” and “format” are self-explanatory. Instead, **“landmarks”** is an instance of class **dataset**, and it is organized as follow. The property “ObsNames” contains the name of the landmarks (see **Fig. 1**).



```
>> disp(temp.hdr.landmarks)
                            X      Y
    eyebrow_left_left        67    218
    eyebrow_left_right      205    225
    eyebrow_right_left      290    225
    eyebrow_right_right     427    218
    eyebrrow_center         247    225
    eye_left_left            95    275
    eye_left_center         140    275
    eye_left_cent_up        140    240
    eye_left_cent_down      140    295
    eye_left_right          188    275
    eye_right_left          308    275
    eye_right_center        358    275
    eye_right_cent_up       358    240
    eye_right_cent_down     358    295
    eye_right_right         402    275
    eye_center              247    275
    nose_center             247    382
    nose_left               182    368
    nose_right              311    368
    mouth_left              168    489
    mouth_center            247    481
    mouth_center_up         247    450
    mouth_center_down       247    525
    mouth_right             329    489
    mouth_chin              247    563
    chin_end                247    612
    head_top_left            65    160
    head_top_center         247    160
    head_top_right          437    160
    hear_top_left            38    289
    hear_bottom_left         53    432
    hear_top_right          461    289
    hear_bottom_right       451    432
    Image_size              500    720
```

Figure 1: “Landmarks” property of fexwimg object. “Landmarks” is a dataset. The property “VarNames” is {“X”, “Y”}; the property “ObsNames” is a cell with elements {“eyebrow_left_left”, {“eyebrow_left_right”, …}. There is no need to include all the landmarks, but the naming has to follow the convention shown in the figure.

The strings with the names of the landmarks compose the “ObsNames” property of the **dataset**. Not all landmarks need to be specified; however, a large number of points usually guarantee more accurate localization of the muscular fibers. Some of these landmarks can be computed using the Emotient SDK, but at the moment, we didn't implement an automated procedure.

The property **"muscles"** is a structure, s.t. each field is named after the muscles in **Table 1**. All fields contain two subfields, "**indx**" and "**texture**." The field **"indx"** is a vector of linear indices with the location of a muscle on the image. The field "**texture**" is a vector with the texture of the muscles at each pixel.

The field **"mask"** is a **dataset**, with variable names "X" and "Y." "X" and "Y" are coordinates that describe an ellipsis or a double ellipsis, which masks the face in the image. A mask can be defined using a set of six parameters, two for the center of the face, two for the upper and lower major semi-axes, one for the minor semi-axis, and one for the rotation. When upper and lower semi-axes have the same value, the mask is elliptical. Instead, when they have different value, the mask is a double ellipsis.

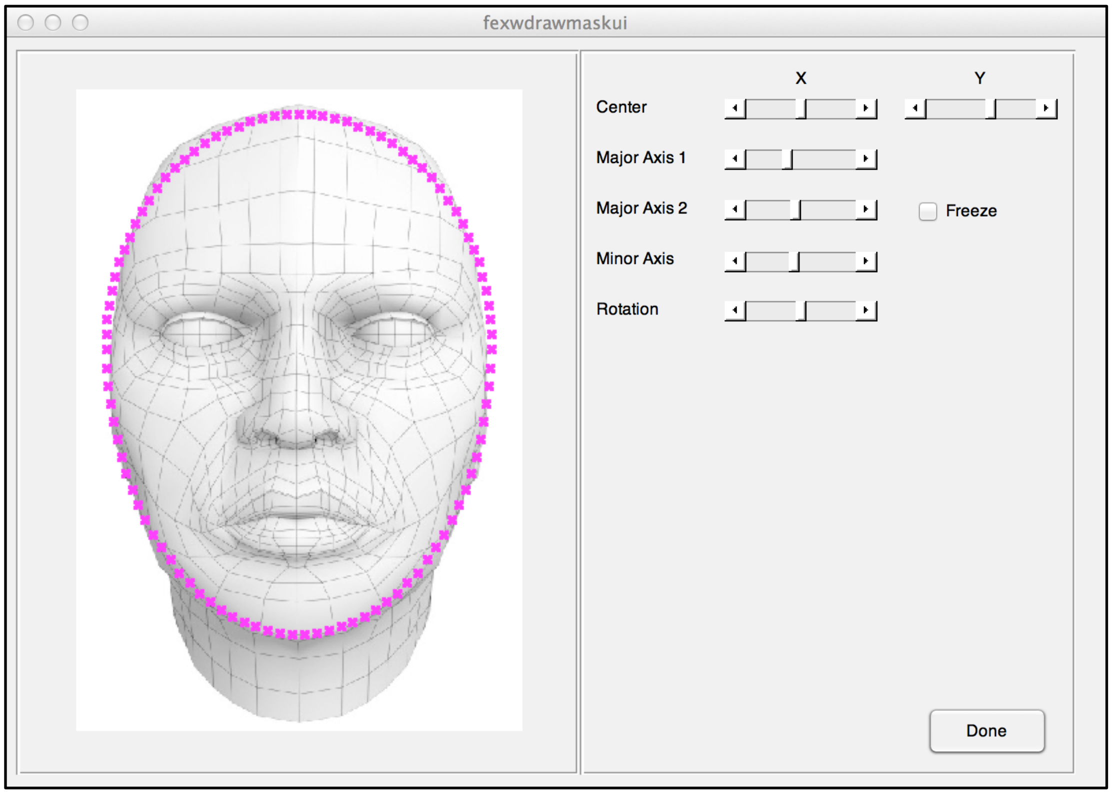


Figure 2: **User interface for draw mask.** Note that the ellipsis is not displayed until both *"Major Axis 1"*, and *"Minor Axis 2"* are larger than 0. Additionally, if *"Minor Axis" > "Major Axis 1"* or *"Minor Axis" > "Major Axis 2"*, the minor semi-axis is set to *min(Major Axis 1, Major Axis 2)*. Finally, if "Freeze" is checked, *Major Axis 2* is set to the value of *Major Axis 1*.

The class **fexwimg** has two methods associated with **"mask,"** namely **"drawmask,"** and **"getmask."** The method **"drawmask"** will open a user interface, which helps the user drawing a mask of the face in the image (see **Fig. 2** for an example, and the caption for further details).

The method **"getmask"** converts the coordinates of the perimeter of mask (namely the **dataset** in **mask**) into a matrix of the same size of the image, with value "0" for pixel outside the face, and value "1" for within-face pixels.

2.2 List of templates and procedure to create a template

FexView contains 12 predefined templates. Templates #01 - #04 are models of the face. Templates #05 - #12 are images of posed facial expressions; specifically: anger, disgust, joy, sadness, surprise, contempt, fear, and neutral expression.

FexView does not comprise yet an automated procedure to import user-supplied images as instances of **fexwimg**. Instead, the user needs to import an image manually. The following code will generate a new template.

```
1. image_path = 'image/path/new_image.jpg';
2. marks = load('landmarks/path/marks.mat);
3.
4. NewImg = fexwimg('path',image_path,'landmarks',marks);
5.
6. NewImg.drawmask();
7. NewImg.test();
```

The code in the box entails:

(1) Indicate the path to the desired image [*line 1*];
(2) Import the landmarks dataset described in section 2.1 [*line 2*];
(3) Initialize the "fexwimg" object [*line 4*];
(4) Draw the mask [*line 6*];
(5) Test whether the template was generated correctly [*line 7*].

3. Overlays

3.1 Overview of "fexwoverlay" class

Given a template, the class “**fexwoverlay**” plots values of interest on action units, muscles, or emotions. The class has the properties and methods displayed in **Table 2**.

Properties	`data`	[**dataset**] data to be displayed.
	`template`	[**fexwimg**] base image for the overlay.
	`handles`	[**structure**] structure of handles for the various layers of the image.
	`side`	[**string**] indicates whether to display AUs (or emotion) bilaterally, or unilaterally.
	`bounds`	[**vector**] 2-component vector with bounds for the data.
	`combine`	[**string**] procedure used to combine the various AUs (‘mean’,’median’, or ‘max’).
	`smoothing`	[**structure**] name and parameters for smoothing the overlay.
	`colmap`	[**string**] name of one of Matlab color map.
	`colbar`	[**structure**] information on how to and whether to insert a colorbar.
	`background`	[**boolean**] indicates whether to include a background color for the entire face.
	`optlayers`	[**structure**] indicates transparency of the overlay, intensity of fiber texture, and brightness of the image.
Methods	`update`	Updates the properties. If an image was generated, it will also update the image.
	`makeoverlay`	Generates overlay data (this method is used internally, or in combination with “show”).
	`coldataidx`	Converts “data” into a set of indices for a 256- color map.
	`show`	Shows or updates the image.
	`saveo`	Save the image, or the “fexwoverlay” object.
	`list`	List muscles, action units, or emotions.
	`maskingo`	Return a mask for the overlay.

Table 2: **Overview of “fexwoverlay” properties and methods.** List of properties and methods included in the “**fexwoverlay**” class.

3.2 Construct “fexwoverlay” and the “data” property

An instance of **fexwoverlay** is constructed by calling *fexwoverlay*. There are three legal construction syntaxes:

```
1. OvObj = fexwoverlay;
2. OvObj = fexwoverlay(data);
3. OvObj = fexwoverlay(data,'VarName',VarVal, ...);
4.
5. OvObj = fexwoverlay('VarName',VarVal, ...); % Not legal!
```

When you use the method at [line 1], fexwoverlay opens a user interface (see **Section 4**). When specifying the argument "data" as in [line 2], the property "data" is set to the data you entered, while all other properties are set to their default values (elaborated momentarily). The construction using syntax at [line 3] let you specify further properties from the list in **Table 2**. Note that the syntax at [line 5] is not legal.

The dataset property is a **dataset** object with "VarNames" properties set to the names of the muscles, Action Units or emotions to be displayed. The naming of the variables needs to be consistent with that used in *FexMetrica*. In order to get a list of available channels, you can use the method "list." For example, the code: `fexwoverlay.list('AU')` will display naming for the Action Units. All columns of "data" with unrecognized names are ignored.

Instead of a **dataset**, the property "data" can be initialized to a cell of string containing strings. For example, data can be set to {'AU1', 'AU2', AU3}. In this case, equally spaced values are generated to associate a color to each of the facial expression channels.

Two important caveats to the constructor procedure are:

(1) Channel types (e.g. muscles, AUs, or emotions) don't mix;
(2) Only one emotion can be displayed.

The constructor can be used to specify further properties (syntax at [line 3]). These properties can also be set using the "update" method, and they are discussed in the next section.

3.2 Other properties and "update" method

Properties other than "data" can be set when initializing a **fexwoverlay** instance, or using the method "update."

```
1. OvObj = fexwoverlay(data,'combine','max','colmap','cool');
2.
3. OvObj.update('combine','max','colmap','cool');
```

For example, the code in [line 1] and [line 3] in the box above can be used to set the option “combine” and “colmap.” Additionally, when using “update,” the image of the overlay is updated automatically – assuming that the image exists.

The sub-sections below describe defaults and possible arguments for each of the properties that can be specified using the constructor or “update.”

3.2.1 Template

The argument template can be a string, an integer between 1 and 12, or a fexwimg object. When the user wants to display an overlay that is not between the12 included in *FexView*, “template” must be a fexwimg. A list of available names for the 12 templates is provided in the **Table 3** (note that this argument is case insensitive). The **default** value is “1.”

Number	Names
1	‘1’,‘template_01’
2	‘2’,‘template_02’
3	‘3’,‘template_03’
4	‘4’,‘template_04’
5	‘5’,‘template_anger’,’anger’
6	‘6’‘template_disgust’’disgust’
7	‘7’,‘template_joy’,’joy’
8	‘8’,‘template_sadness’,’sadness’
9	‘9’,‘template_surprise’,’surprise’
10	‘10’,‘template_contempt’,’contempt’
11	‘11’,‘template_fear’,’fear’
12	‘12’,‘template_neutral’,’neutral’

Table 3: List of available templates.

3.2.2 Side

The argument “side” is a string, with optional values “left,” “right,” and “both.” The **default** is “both.”

3.2.3 Bounds

The argument “bounds” is a two-component vector, which determines the data to be displayed. For instance, suppose that data is {AU1:-1.00, AU2:2.00, AU4:3:00}, if you set bounds to [0,10], the overlay will have a 256-dimension color map for values between 0 and 10; value smaller than 0 (e.g. AU1 in the example) won’t be display. By **default**, the property “bounds” is set to:

- [0, max(data)] when data is between 0 and infinity.
- [0,min(data)] when data is between negative infinity and 0.
- [-ValMax, ValMax] when data includes both positive and negative entry. ValMax is the largest absolute value in data.

3.2.4 Combine

The argument “combine” is a string, with optional values “mean,” “median,” and “max.” The **default** is “median.” Note that “median” requires substantially more time than the other combination methods.

3.2.5 Smoothing

The argument “smoothing” can be a string in which case it is one of the possible kernel functions. Alternatively, smoothing is a structure, with fields “kernel,” “size,” and “param.” The field “param” is used only when “kernel” is set to “Gaussian,” “Log,” and “Laplacian.” **Table 4** summarizes the possible options and defaults.

kernel	**size (pixels)**	**param**
None	***	***
Gaussian	**10**	**2.5**
Log	10	2.5
Motion	10	0
Average	10	***
Disk	10	***
Laplacian	***	0.2

Table 4: **Option for “smoothing” properties.** The mark “***” indicates that field “size” or field “param” is not used of the specific filter kernel. The values in the table show the defaults used for each kernel. All kernels except “laplacian” have a size parameter, which can be set to any strictly positive value (note that setting “size” to 1 is the same as setting kernel to “none”). For “Gaussian” and “log” kernels, the field “param” is the standard deviation (as such, it is a positive value). For “laplacian,” the field “param” is the shape and its support is [0.0, 1.0]. For “motion” it is the angle of rotation counterclockwise (0 is horizontal motion).

By **default** smoothing is performed using a Gaussian kernel, 10^2 pixels in size, and standard deviation set to 2.5 (namely the “smoothing.param” field). AN example of code that can be used to change the smoothing kernel is given below.

```
update(OvObj,'smoothing',struct('kernel','gaussian','size', 20,
    'param',5));
```

For more details on smoothing, consult the documentation associated with Matlab function “fspecial.m.”

3.2.6 Colmap

The argument “colmap” is a string, and the optional values are all the color maps defined in Matlab. The **default** is “jet.” Alternatively, “colmap” can be a K × 3 customary color map.

3.2.7 Colbar

The argument “colbar” is a Boolean value, set by **default** to ‘false.’ When “colbar” is set to ‘true,’ the image will comprise a colorbar position on the bottom of the image. In order to change properties of the colorbar, the user can manipulate directly the handle for the colorbar store in the “handles” property. Note that a colorbar is generated only of the property “data” contains values and not only names of facial expressions channels.

3.2.8 Background

Background property is a Boolean value, which determines whether the image will be shown with or without a background color within the face. By **default**, the background argument is set to ‘false.’

3.2.9 Optlayers

The “optlayers” argument is a vector, comprising one, two, or three components. Once you provide the argument using the constructor or the method “update,” the property “optlayers” has three fields:

- `optlayers.overlay`: transparency of the overlay on the face template – a number between 0 and 1 (default: 0.1). For `optlayers.overlay = 1`, no overlay is displayed on top of the template.
- `optlayers.fibers`: transparency of the overlay on the muscular fibers. For `optlayers.fibers = 1`, the texture of the muscles is not displayed. By default, this value is set to 0.7.
- `optlayers.brightness`: the brightness of the colors. This is a number between -1 and 1, which is set by default to 0.

When specifying the "optlayers" argument, the user can enter one, two or three values. When only one value is entered, that value is interpreted as the "overlay" field; if two values are entered, they are interpreted as the "overlay" and "fibers" field, and so on.

3.3 Other methods

The "fexwoverlay" object from *FexView* contains six methods besides the already discussed "update": "show," "makeoverlay," "coldataidx," "list," "maskingo," and "saveo."

3.3.1 Show

The method "show" needs to be called in order to create the image. "Show" does not take any argument. After the image is generated, there is no need to call the method "show" again, because the method "update" will call "show automatically" in order to refresh the data from an existing figure.

3.3.2 Makeoverlay

The method "makeoverlay" generates the matrices with the overlay data. Generally, there is no need to call this function, because it is automatically used by "update" and "show," unless the dataset stored in the property "data" has more than one row. When "data" has multiple rows, the indexing between colors and values in "data" is generated using all values, but only the first row is displayed. However, you can call explicitly >> `OvObj.makeoverlay(row_number)` in order to show data from the desired row.

3.3.3 Coldataidx

The method "coldataidx" transform the value from "data" into index corresponding to a color map. This function is automatically called by the method "makeoverlay"; however, it can also be used in isolation. Legal syntaxes are shown in the box below.

```
[ColDataIdx,BaseColor] = OvObj.coldataidx(Y)
[ColDataIdx,BaseColor] = OvObj.coldataidx(Y,repval)
[ColDataIdx,BaseColor] = OvObj.coldataidx(Y,repval,bnds)
```

The input comprises “Y,” which is either a vector or a matrix with values. “Y” can be obtained with the command: `>>double(OvObj)`. “Y” has one column per facial expression channel in “data.” The argument “repval” is a Boolean term (default set to ‘false’), which determine whether to repeat values across columns of “Y.” The argument “bnds” indicates the bounds of the colormap, and it is a 2-component vector [upper,lower]. By default, “bnds” is set to OvObj.bounds (see **Section 3.2.3**). If you provide the argument “bnds,” the property “bounds” is updated.

The output “ColDataIdx” is a matrix/vector of the same size of “Y,” which replaced values in “Y” with indices on the color map in “colmap.” The output “BaseColor” indicate the index for the color used for the background of the image (in case the property “background” is set to true). “BaseColor” is set to “0” whenever data in “Y” comprises only positive, or only negative values. “BaseColor” is set to the median color from “colmap” when the data comprises both positive and negative values.

3.3.4 List

The method “list” is called issuing the command: >> `ObObj.list(string)`. This returns a cell with the names of facial expression channels. The input argument “string” can be any of the following value: {‘all’, ‘muscles’, ‘aus’, ‘emotions’}, or it can be empty (same as ‘all’).

3.3.5 Maskingo

Same as “getmask” method from “fexwimg,” described in **Section 2.1**.

3.3.6 Saveo

The method “saveo” can be used to save a “fexwoverlay” instance, or to save an image. The method “sevo” takes three optional arguments, “format,” “dpi,” and “name.”

Format indicates the extension of the file saved. For images, the extension can be set to: '-dpdf,' '-djpeg,' '-dpng,' '-dtiffn,' or '-dbmp' to save an image. For any other values, “saveo” will save the overlay instead of an image (default is ‘-djpeg’). **Dpi** indicate the resolution of the image – an integer with minimum value 150 (default is 300). **Name** is a string with the name/path of the saved file. When the argument “name” is left empty, the file is saved to the default directory, with name “fxwo-dd:mm:yy:HH:MM:SS.”

3.4 Examples

[… …]

4. User Interface

[Working on updating the UI]

5. References

[… …]