Protosc Tutorial

Tutorial Version 1.0, July 2021
Protosc version v1.03

Table of Contents

Introduction

Studies within the realm of cognitive neuroscience often use different categories of images to define their conditions. For example, a study might be interested in finding differential reaction times due the presentation of faces with a neutral versus an angry facial expression. Since any difference in these categories is a valid candidate to explain category related behavioral (or neural) differences, knowledge about the objective image differences between categories is crucial for the interpretation of the behaviors (or neural responses). However, natural images vary in many image features and not every feature is equally important in describing the differences between the categories. Here, we provide a methodological approach to find as many of the image features as possible, using machine learning as a tool, that have predictive value over the category the images belong to. In other words, we describe a means to find the features of a group of images by which the categories can be objectively, and quantitatively defined. Our method does not aim at providing a means for the best possible decoding performance; instead, our aim is to uncover prototypical characteristics of the categories. To facilitate the use of this method, we offer an open-source, MATLAB-based, toolbox that performs such an analysis and aids the user in visualizing the features of relevance. This text provides a guide for installing and using the toolbox.

For an example of application of the beta version of Protosc, see:
https://www.nature.com/articles/s41598-021-87881-w#citeas

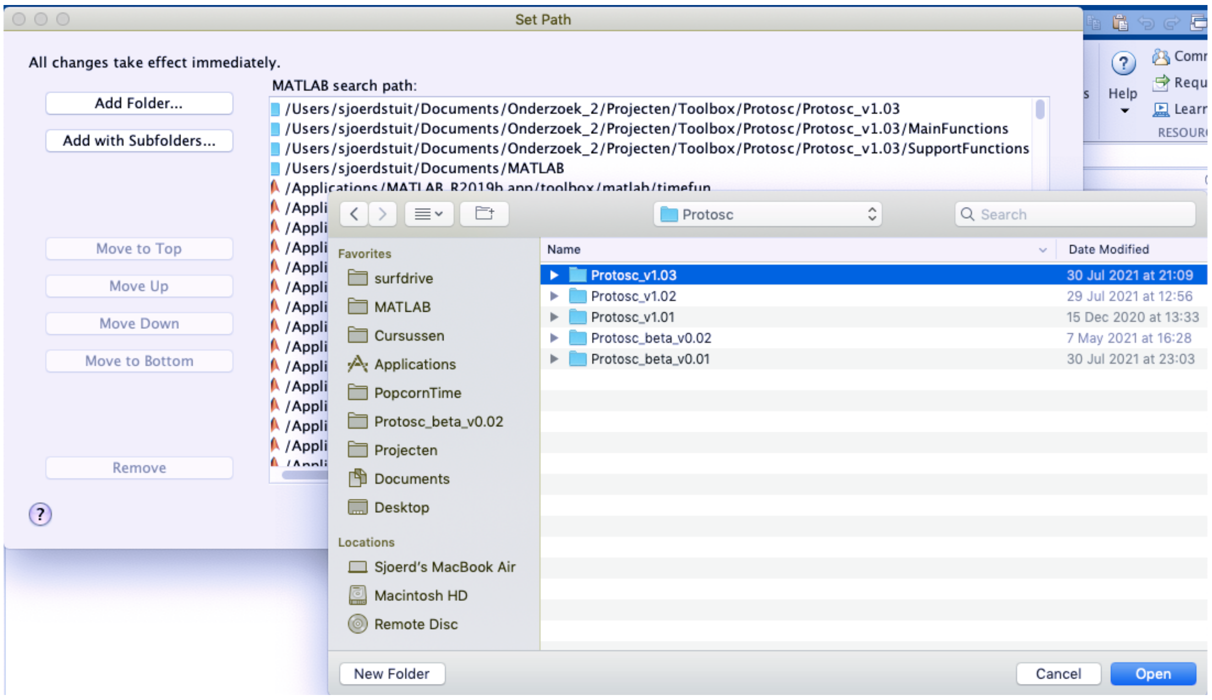
For an application of a previous version of the toolbox, see:
https://youtu.be/bN24w16-CZA

For an image set to try out the toolbox with, see:
https://www.kaggle.com/chetankv/dogs-cats-images

Download and Install

After downloading the zip file from the OSF files page (https://osf.io/f6nbu/files/), unzip the file. Next place the Protocs folder in the desired location on your computer. Go to Matlab 'set path' on the Home tab and add the Protosc folder with subfolders (see below). A total of 3 folders should be added. Then, save the path.

For a video on the installation process see: https://youtu.be/z23CKgpFJUU



All changes take effect immediately.

MATLAB search path:

```
/Users/sjoerdstuit/Documents/Onderzoek_2/Projecten/Toolbox/Protosc/Protosc_v1.03
/Users/sjoerdstuit/Documents/Onderzoek_2/Projecten/Toolbox/Protosc/Protosc_v1.03/MainFunctions
/Users/sjoerdstuit/Documents/Onderzoek_2/Projecten/Toolbox/Protosc/Protosc_v1.03/SupportFunctions
/Users/sjoerdstuit/Documents/MATLAB
/Applications/MATLAB_R2019b.app/toolbox/matlab/timefun
```

Protosc

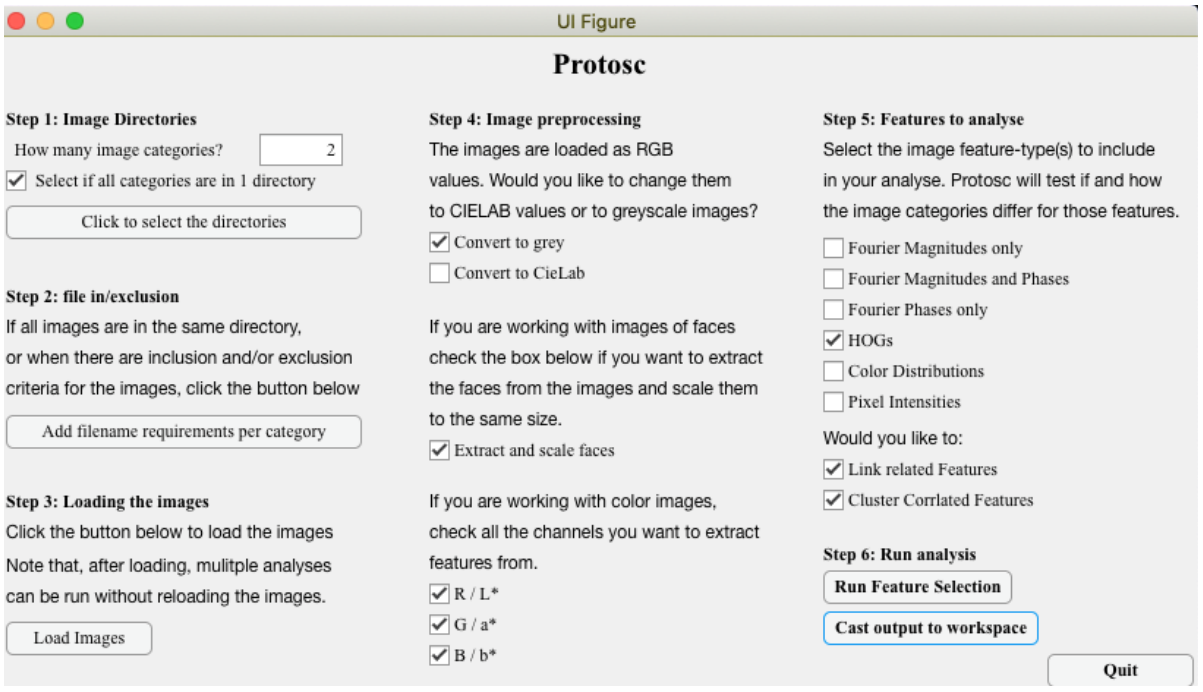
Name	Date Modified
Protosc_v1.03	30 Jul 2021 at 21:09
Protosc_v1.02	29 Jul 2021 at 12:56
Protosc_v1.01	15 Dec 2020 at 13:33
Protosc_beta_v0.02	7 May 2021 at 16:28
Protosc_beta_v0.01	30 Jul 2021 at 23:03

How to use the App

For a video on using the app see: https://youtu.be/kkwvvnSzSZ0

Starting up the app

To start the app, run `protosc_app.m` in the Matlab command window. The following window will open:


UI Figure
Click to select the directories
Add filename requirements per category
Load Images
Run Feature Selection
Cast output to workspace
Quit

Protosc

Step 1: Image Directories

How many image categories? 2

☑ Select if all categories are in 1 directory

Step 2: file in/exclusion

If all images are in the same directory, or when there are inclusion and/or exclusion criteria for the images, click the button below

Step 3: Loading the images

Click the button below to load the images Note that, after loading, mulitple analyses can be run without reloading the images.

Step 4: Image preprocessing

The images are loaded as RGB values. Would you like to change them to CIELAB values or to greyscale images?

☑ Convert to grey
☐ Convert to CieLab

If you are working with images of faces check the box below if you want to extract the faces from the images and scale them to the same size.

☑ Extract and scale faces

If you are working with color images, check all the channels you want to extract features from.

☑ R / L*
☑ G / a*
☑ B / b*

Step 5: Features to analyse

Select the image feature-type(s) to include in your analyse. Protosc will test if and how the image categories differ for those features.

☐ Fourier Magnitudes only
☐ Fourier Magnitudes and Phases
☐ Fourier Phases only
☑ HOGs
☐ Color Distributions
☐ Pixel Intensities

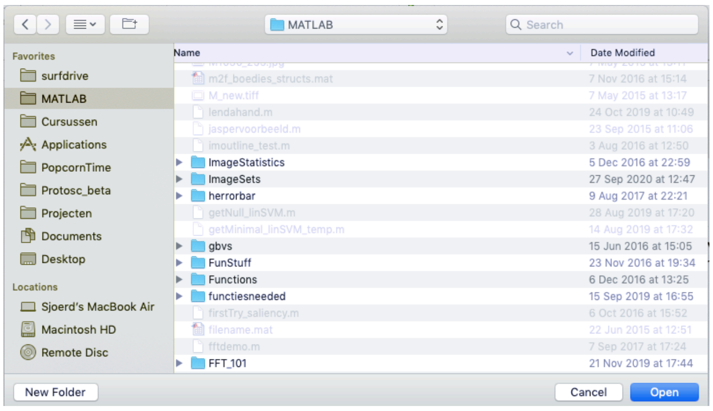
Would you like to:

☑ Link related Features
☑ Cluster Corrlated Features

Step 6: Run analysis

Step 1: Image Directories

First indicate how many categories of images you have. When clicking the button under step 1 of the app, one window will open (example below) for each category. However, if you check the box 'select if all categories are in 1 directory', only one window will open. Navigate to the location of the image for category 1 and press open. Do the same for the other categories (when applicable).



Name	Date Modified
m2f_boedies_structs.mat	7 Nov 2016 at 15:14
M_new.tiff	7 May 2015 at 13:17
lendahand.m	24 Oct 2019 at 10:49
jaspervoorbeeld.m	23 Sep 2015 at 11:06
imoutline_test.m	3 Aug 2016 at 12:50
ImageStatistics	5 Dec 2016 at 22:59
ImageSets	27 Sep 2020 at 12:47
herrorbar	9 Aug 2017 at 22:21
getNull_linSVM.m	28 Aug 2019 at 17:20
getMinimal_linSVM_temp.m	14 Aug 2019 at 17:32
gbvs	15 Jun 2016 at 15:05
FunStuff	23 Nov 2016 at 19:34
Functions	6 Dec 2016 at 13:25
functiesneeded	15 Sep 2019 at 16:55
firstTry_saliency.m	6 Oct 2016 at 15:52
filename.mat	22 Jun 2015 at 12:51
fftdemo.m	7 Sep 2017 at 17:24
FFT_101	21 Nov 2019 at 17:44

Step 2: file in/exclusion

This step is required if all images come from a single directory. Otherwise it is optional. Clicking the button in this step will evoke the follow prompt in the Matlab command window:

```
For category 1
What should be in the filenames? (please separate multiple terms with * (e.g.
term1*term2):
```

Here you can indicate what should be part of the filenames for category 1. No quotation marks are required. Multiple filename requirements per category can be given by combining terms with an asterisks (for example: happy*frontal). After answering, Protosc asks if there is anything that should not be in the filenames for category 1. The process repeats for each category.

Step 3: Loading the images

After setting up the directories and filename requirements, the image can be loaded. Loading is kept as a separate non-automatized step so multiple analyses can be run on the same images without reloading them. Note though, that running the analysis without loading the images will cause the analysis to crash.

Step 4: Image preprocessing

Here you can choose to convert the images to grey-scale images or to the CIELAB color-space. For those working with images of faces, there is the option to extract only the face area from the images and scale all those face areas to images of the same size (See example of extracting the face area in the image below). Note that selecting this option means that only images from which a face can be extracted are selected for the analysis. Finally, you can indicate which channels (for color images) of the images you want to include.

On the left is an example image of a face, on the right is the same face but now the face area is extracted with the Viola-Jones algorithm. The rest of the image is discarded.

Step 5: Features to analyze

In Step 5 of the app you can indicate which feature types (e.g. HOG Features & Fourier Phases) to include in your analysis. Furthermore, you can indicate whether related features should be linked together and whether correlated features should be clustered (e.g. treated as a single feature). More info on the features, linking and clustering is displayed when placing the cursor over the respective terms.

Step 6: Run analysis

While running, progress is continuously reported in the Matlab command window. When complete, the main results are displayed in the command window. To access the output of the analysis, press the cast button under Step 6 of the app and the variable *out* will be added to your workspace.

How to use the autogenerated script

For a video on the autogenerate analysis script, see: https://youtu.be/7ZfkxZOqmOQ

Running the function `protosc_template(filename)` will create and open a m-file with a Protosc analysis. Compared to using the app, the script allows direct control over a wide range of settings for the analyses. Also, it is suitable for analyses where the categorical labels of the images are not easily set by the folder they are in, or when they can be separated by their filename.

Setting the directories

After generating the script, first indicate the location of your images by updating the following line to match your specific location on your computer:

```
imdir{1}                        =
'/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces/';
```

Note that this example is for when all images are in a single directory. If the images are in separate directories, the variable imdir can be extended as such:

```
imdir{1}                        = '/Directory_Category_1/';
imdir{2}                        = '/Directory_Category_2/';
imdir{n}                        = '/Directory_Category_n/';
```

Based on your operating system, different file separators might be required (/ for Mac, \ for Windows).

In- and exclusion criteria

Next, indicate the file name requirements per category:

```
mustinclude                     = {'_O.' '_C.'};
maynotinclude                   = [];
```

In this example, filenames for category one must include '_O.' and for category two they must include '_C.'. Similarly, filename criteria for exclusion can be given via the variable `maynotinclude`. Multiple filename requirements per category can be given via an asterisks. For example:

```
mustinclude                     = {'_O.*frontal' '_C.*frontal'};
```

When not having inclusion criteria, simply put

```
mustinclude                     = [];
```

Set the filename for autogenerated methods and results

After running the analysis, two text files will be created. One describes the methods used for the analysis, the second describes the results of the analysis.

```
methodsfilename                 = 'Methods';
resultsfilename                 = 'Results';
```

Choosing the feature types and additional settings

Next, choose the types of features you wish to include in your analysis:

```
get_FourierFeatures                     = 1;
get_Fourier_Mag_Pha                     = 0;
get_Fourier_Pha                         = 0;
get_HOGFeatures                         = 0;
get_ColDistributions                    = 0;
get_Pixels                              = 0;
```

Each of the above variable is used as a Boolean to indicate of the corresponding feature type should be included in the analysis. The script continues with a list of the main settings including a description of each settings. After adjusting the settings, the script is ready and the analysis can be run. The output of the analysis can be found in the variable out.

Features and Reference Frames

Protosc extracts multiple features from a feature space and tests which of those features can be used to separate the categories of images. This section will go through all the available feature spaces in Protosc and will explain how to keep track of what a particular feature represents in an image.

The feature spaces used by Protosc vary in their dimensions. For example, the Fourier magnitude features come from a 2-dimensional space while the Color distribution feature comes from a 1-dimensional space. Although a feature might be presented in a 2-dimensional space, some analyses (e.g. the feature selection algorithm) requires each example (image) to be represented by a 1-dimensional vector. This means the feature space are often transformed in the toolbox. To keep track of this transformation, a reference frame is created for each feature space. The extraction of features from images and the reference frame that comes with it is described below for each feature space separately.

Analyzing a single feature space

1) Fourier Magnitudes Only

The features

When extracting Fourier magnitudes without information on the phase, the images are resized to 200 by 200 pixels and the absolute values of the 2d FFT (FFT2.m from Matlab) of each image are extracted. The magnitude spectrum is subsequently down-sampled by taking the sum of all values corresponding to a particular spatial frequency and orientation range. Note that this down sampling serves two purposes: For one, it reduces the total number of features, meaning the analyses requires less processing power. Second, it disrupts the influence of phase information, meaning that contrast values also lose all indirect spatial specificity. As such, the spatial frequency analysis focusses only on global contrast differences between categories. Default settings down sample the spectrum into 24 equally sized spatial frequency bands and 16 orientation bands, resulting in 384 sections. Each section then holds the sum of the magnitudes that were within that area.

The reference frame

For gray scale images, the reference frame consists of a 200 by 200 matrix containing 384 wedges, one for each spatial frequency and orientation combination. The central point in the reference frame reflects the lowest spatial frequency with the frequency increasing outwards. Orientation is reflected by the radial angle with spatial frequencies along the vertical axis reflecting horizontal edges in the images, 45 degrees clockwise from vertical reflects 45 degrees counterclockwise from horizontal edges in the images, and so on. When using color images and without linking the features, the reference frames for each used color channel are concatenated. For example, an analysis with unlinked features and 3 color channels results in a 200 by 600 matrix. When linking the features, color channels are stacked in the 3rd dimension.

The function for extracting this feature type and the corresponding reference frame is
`protosc_Stims2Fourier_Magnitude_Features_Segmented`

2) Fourier Magnitudes and Phases

The features

When extracting Fourier magnitudes *with* phases, the images are downscaled to 25x25 pixels (using the default settings) before extracting both the phase and the associated magnitude to avoid extremely large feature spaces. The phases and magnitudes of a color channel are always linked together during feature selection.

The reference frame

For gray scale images, the reference frame consists of a 25 by 25 by 2 matrix where each point reflects one spatial frequency and orientation combination. The first channel (the 3rd dimension of the matrix) contains Fourier magnitudes and the second channel contains the corresponding phases. The central point in the reference frame reflects the lowest spatial frequency with the frequency increasing outwards. Orientation is reflected by the radial angle with spatial frequencies along the vertical axis reflecting horizontal edges in the images, 45 degrees clockwise from vertical reflects 45 degrees counterclockwise from horizontal edges in the images, and so on. When using color images and without linking the features, the reference frames for each used color channel are concatenated. For example, an analysis with unlinked features and 3 color channels results in a 25 by 75 by 2 matrix. When linking the features, color channels are stacked in the 3rd dimension, resulting in a 25 by 25 by 6 matrix.

Function for extracting this feature type and the corresponding reference frame:
`protosc_Stims2Fourier_Magnitude_Phase_Features`

3) Fourier Phases Only

The features

When extracting Fourier phase information only, the images are also downscaled to 25 by 25 pixels (using the default settings) before extracting both the phases to avoid extremely large feature spaces.

The reference frame

For gray scale images, the reference frame consists of a 25 by 25 matrix where each point reflects one spatial frequency and orientation combination. The central point in the reference frame reflects the lowest spatial frequency with the frequency increasing outwards. Orientation is reflected by the radial angle with spatial frequencies along the vertical axis reflecting horizontal edges in the images, 45 degrees clockwise from vertical reflects 45 degrees counterclockwise from horizontal edges in the images, and so on. When using color images and without linking the features, the reference frames for each used color channel are concatenated. For example, an analysis with unlinked features and 3 color channels results in a 25 by 75 matrix. When linking the features, color channels are stacked in the 3rd dimension, resulting in a 25 by 25 by 3 matrix.

Function for extracting this feature type and the corresponding reference frame:
`protosc_Stims2Fourier_Phase_Features`

4) HOGs

The features

HOG values reflect the presence of edges, per orientation, for each location of an image (Dalal & Triggs, 2005). As such, HOG values can capture the structure within an image. Before extracting the HOG features, the images are resized to 200 by 200 pixels to ensure they are all the same size. The default settings that the toolbox uses the creates HOG features for 9 unsigned (meaning dark to light edges and light to dark edges are pooled together) orientations in non-overlapping 10x10 pixel sections of the image. We chose to use non-overlapping section for simplicity when visualizing results. The default resolution (10x10) was chosen to be able to capture small structural aspects of the images.

The reference frame

The reference frame consists of a matrix of 20 by 20 cells, each cell covering 10 by 10 pixels of the image, with the same spatial organization as the original image. This means the upper left element in the plane corresponds to HOG features in the upper left 10 by 10 pixels in the image. The progression continues downward till the 20th cell, than start again in the second column.

Function for extracting this feature type and the corresponding reference frame:
`protosc_Stims2HOG_Features`

5) Color Distributions

The features

When extracting color distributions, the probability of values in that layer for falling within one of 25 bins is estimated for each layer of the image. We suggest to first convert the images to cie l*a*b*. Note that with this method, the linking of features from different layers is meaningless and therefore they will not be linked together in the feature selection analysis irrespective of the settings given.

The reference frame

The reference frame consists of a 3 by 25 matrix where each row reflect a different color channel (from top to bottom: R/L*, G/a*,B/b*) and each column represents a bin. Note that during visualization the matrix is made square.

Function for extracting this feature type and the corresponding reference frame:
`Protosc_ValueDistributions_Features`

6) Pixel Intensities

The features

To avoid an extremely large feature-space when extracting pixel intensities, the images are first down-sampled to 25x25 pixels. Other than that no transformations are made besides converting the 2- or 3-dementional image to a 1-dementional vector required for the feature selection procedure.

The reference frame

For gray scale images, the reference frame consists of a 25 by 25 matrix with the same spatial organization as the original image, meaning the upper left element in the plane corresponds to the upper left pixel in the image. When using color images and without linking the features, the reference frames for each used color channel are concatenated. For example, an analysis with unlinked features and 3 color channels results in a 25 by 75 matrix. When linking the features, color channels are stacked in the 3rd dimension, resulting in a 25 by 25 by 3 matrix.

Function for extracting this feature type and the corresponding reference frame:
`protosc_Stims2Pixels_Features`

Analyzing multiple feature spaces at once

When selecting multiple feature spaces to extract features from, and when using all those spaces to find the features whose values have predictive value over the category the image belongs to, Protosc will concatenate the different feature spaces. When using the app or the autogenerate script, and assuming default settings, the order of the features in the data matrix used for feature selection is as follows:

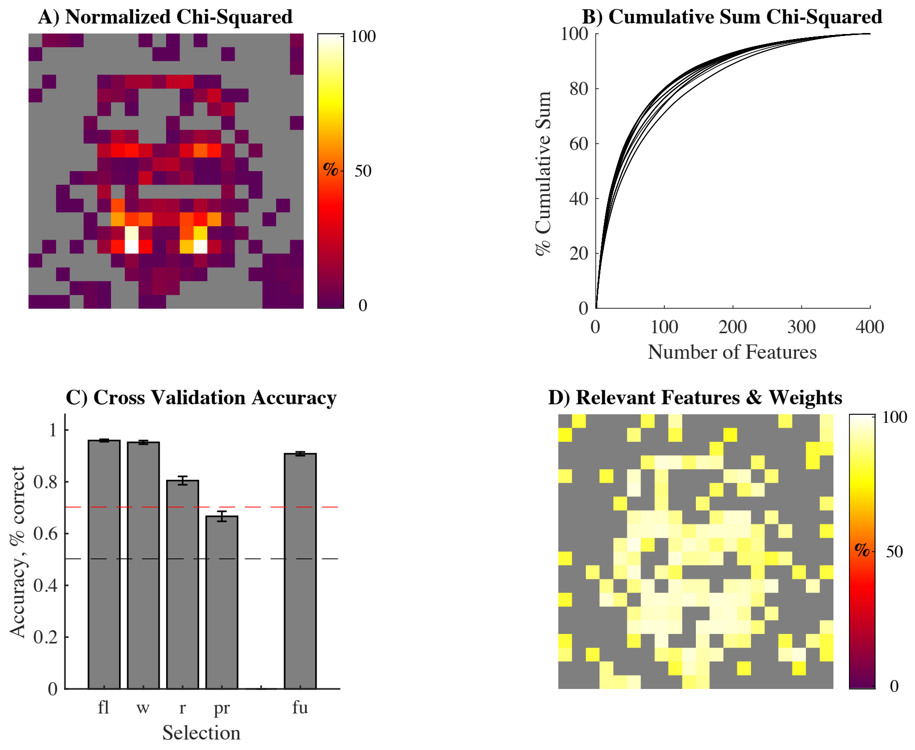
Fourier magnitudes (24 spatial frequency bands x 16 orientation bands = 384 features if linked, otherwise 384 features per color channel), Fourier magnitudes and phases (25 rows x 25 columns = 625 features if linked, otherwise 625 features per color channel), Fourier phases (25 x 25 = 625 features if linked, 625 features per color channel), HOG features (20 rows x 20 columns = 400 features), Color distributions (25 features per color channel) and Pixel Intensities (25 rows x 25 columns= 625 features if linked, otherwise 625 features per color channel).

Thus, when combining Fourier magnitudes with HOG features and using grey scale images, the first 384 columns in the data matrix and in the variable `out.MainFindings.All_feature_Weigths` (see Inspecting the output struct) will refer to the Fourier magnitudes and the next 400 columns will refer to the HOG features. The reference frames for the used features will be combined in the visualizations of the features.

Another example, now using color images and unlinked features: When combining Fourier phases and Pixel Intensities for RGB images, the first 625 features are the Fourier phases for the R channel, the next 625 features are the Fourier phases for the G channel, followed by 625 features Fourier phases for the B channel. Then 625 pixel intensity features the R channel, 625 pixel intensity features for the G channel, and finally 625 pixel intensity features for the B channel. Six reference frames will be combined: 3 for the Fourier phases and 3 for the pixel intensities.

Inspecting the data visualizations

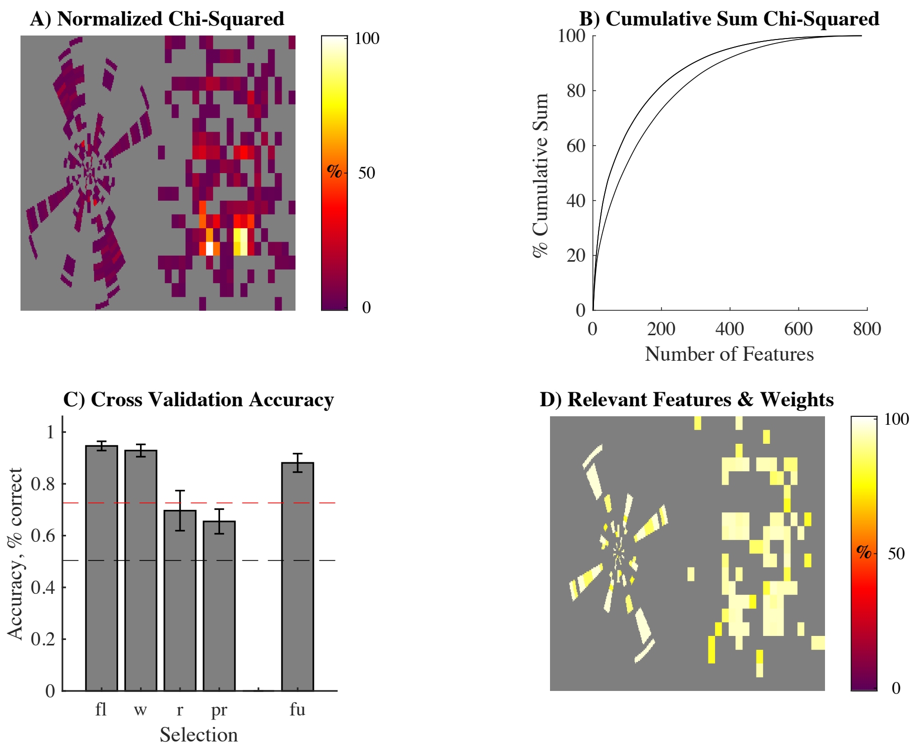
Protosc visualizes the following information by default: A) the chi-squared values, normalized between 0 and 100, for each feature presented in the corresponding reference frame. B) the percentage cumulative sum scores of the chi-squared values, C) the cross-validation accuracies per method of feature selection and D) the significant features and their respective weights presented in the corresponding references frame. See an example based on comparing happy to angry faces of the Nimstim data set in the figure below.



A) gives an overview of the degree of differences, in Chi-Squares normalized between zero and one, between the categories for each feature. Please note that this reflects a statistical comparison between samples of the images; the results for the features most relevant for *predicting* which category an image belongs to may differ.
B) shows, via its steepness, how fast the chi-squared scores add up. For example, a relatively flat (diagonal) line indicates all feature hold about the same amount of information while a very fast rise followed by a flat line shows only a few features hold all the information. There is one curve for each fold.
In C), the black dashed line shows the mean chance performance while the red dashed line shows the 95th percentile of the chance distribution. In terms of interpretation, we can see, for this example, that a random selection of features already performs rather well, meaning there are many relevant differences between the categories. However, we also see that the pseudo random selection does not perform above the 95th percentile of the chance distribution. This suggests that, when the random selection can't include features already selected by the filter or wrapper selection, decoding does not work. This indicates that the filter or wrapper selection are finding the most relevant features and few to none relevant features are likely remaining.
In D) only the significant features are shown, with the color reflecting their associated mean performance (the figure shown here is decoding happy versus angry facial expressions).

The visualization described above can be recreated with the function `protosc_all_figures.m`

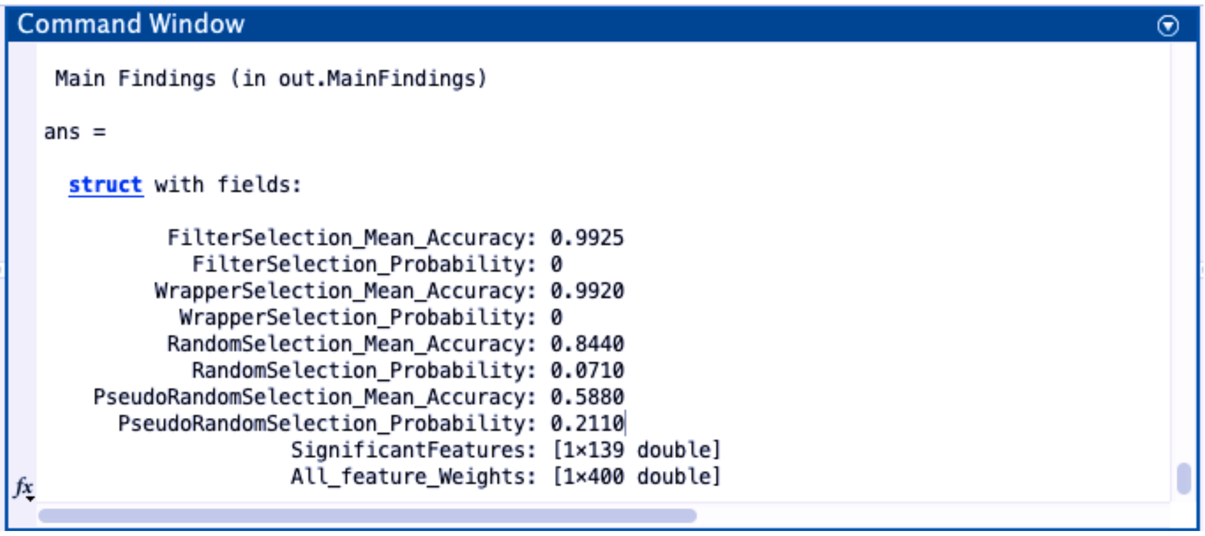
As noted in the previous section, when analyzing multiple feature spaces at once, the reference frames for the feature spaces are combined. In the example below, Fourier magnitude features are combined with HOG features. The Fourier reference frame is presented on the left in A and D, HOGs on the right.



Inspecting the output struct

For a video on the output variable, see: https://youtu.be/qz2hCvz1IYE

The output and results of the analyses are given in the format of a struct. The standard name for the struct is 'out'. To facilitate ease of use, the most important information is given in the sub-variable `MainFindings`, whose content can be displayed in the command window via the command `out.MainFindings`. The following is an example of the returned output:



```
 Main Findings (in out.MainFindings)

ans =

  struct with fields:

          FilterSelection_Mean_Accuracy: 0.9925
            FilterSelection_Probability: 0
         WrapperSelection_Mean_Accuracy: 0.9920
           WrapperSelection_Probability: 0
          RandomSelection_Mean_Accuracy: 0.8440
            RandomSelection_Probability: 0.0710
    PseudoRandomSelection_Mean_Accuracy: 0.5880
      PseudoRandomSelection_Probability: 0.2110
                    SignificantFeatures: [1×139 double]
                    All_feature_Weights: [1×400 double]
```

The output shows the fraction correct and the associated probability that the accuracy reflects chance performance based on a permutation test of each of the four feature selection procedures of Protosc. Next, it provides a vector of indices of significant features and a vector of the estimated weight of each feature. This example is based on a HOG features using default settings and therefore has a total of 400 features. Well over 25% of the features are estimated to be significant for decoding the classes.

Interpreting performance per approach

As noted above, Protosc uses 4 methods of feature selection simultaneously. Two of those methods, the filter approach and the wrapper approach, are expected to result in significant decoding performance if the categories of images are (easily) separable by their features. The two other approaches, random feature selection and pseudo random feature selection, are *not* expected to result in significant decoding if there are only a few features that separate the categories by their features. If random feature selection results in significant decoding, the categories can be separated by many different features and the exact combination of features is not very relevant. If pseudo random feature selection results in significant decoding, it means that there are still many relevant features besides the features selected via the filter and the wrapper approaches.

Significant features

The four approach are combined to find features that are reliably associated with above chance performance. The indices of these features can be found in `out.MainFindings.SignificantFeatures`. There, the features are given in order of importance (from highest associated decoding performance to lowest). To see where in the reference frame a feature comes from, and thus, what it represents, use the function `protosc_find_feature(out,featureindex)` and supply the out variable with the feature you want to highlight in the reference frame.

All feature weights

The vector `out.MainFindings.All_feature_Weigths` contains the estimated weight of each feature, irrespective of its significance.

The out variable beyond the MainFindings

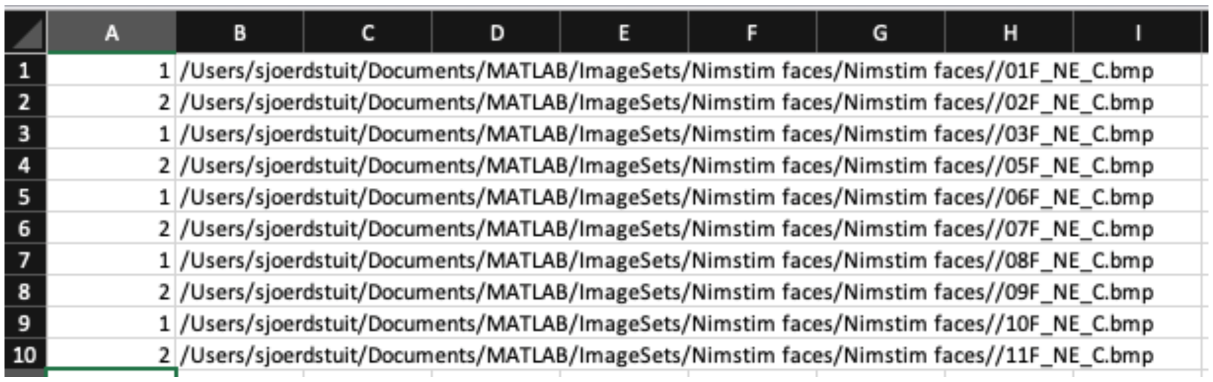
Many aspect of the analysis are stored in the out variable. What follows is a list of relevant variables:
`out.AllData` contains a matrix of all the data used for the analysis
`out.settings` contains a struct with all the settings used for the current analysis.

`out.codeinfo` contains a struct with information concerning the file from which the analysis was ran and date of the analysis.
`out.datainfo` contains a struct with information on the data used for the analysis such as the number of examples in each category.
`out.Results` is a nested struct where information on the progress of the analysis is stored.
The most relevant sub-variables are the following:
`out.Results.MaxModelParams` gives the number of features included per fold of the analysis
`out.Results.Filter`, `out.Results.Wrapper`, `out.Results.RandomModel` and `out.Results.PseudoRandomModel` each provide detailed information of the respective feature selection method. Within each of these you can find the F1 score, the accuracies and used features per fold.

Analyses based on behavioral data

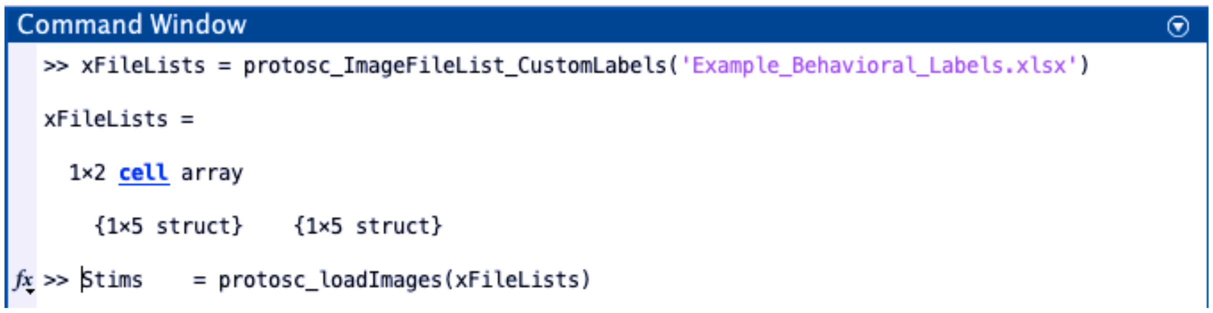
Although the standard functionality of this toolbox is built for decoding predetermined image categories, categorization of the images can also be done by participants in an experiment. For example, the task for a participant may be the indicate if a neutral face appears more happy or more sad. The experimenter can then place all the images responded to with 'more happy' in one directory and all the images responded to with 'more sad' in another directory and test which features best predict the response of the participant. However, creating the two directories per participant is not the most efficient way. To assign behavior-based responses to images and test for features that have predictive value for the given responses, the user can create an excel file containing one column of numerical labels and one column of filenames and supply it to `protosc_ImageFileList_CustomLabels.m`.

The excel file should contain numerical labels in column A and filename including the full path of the file in column B. See an example below (the example below is from the file Example_Behavioral_Data.xlsx given with the toolbox).



	A	B
1	1	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//01F_NE_C.bmp
2	2	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//02F_NE_C.bmp
3	1	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//03F_NE_C.bmp
4	2	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//05F_NE_C.bmp
5	1	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//06F_NE_C.bmp
6	2	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//07F_NE_C.bmp
7	1	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//08F_NE_C.bmp
8	2	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//09F_NE_C.bmp
9	1	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//10F_NE_C.bmp
10	2	/Users/sjoerdstuit/Documents/MATLAB/ImageSets/Nimstim faces/Nimstim faces//11F_NE_C.bmp

Supplying the excel file as input to `protosc_ImageFileList_CustomLabels` creates a file list that can be subsequently used to load the associated images. See an example below.



```
>> xFileLists = protosc_ImageFileList_CustomLabels('Example_Behavioral_Labels.xlsx')

xFileLists =

  1×2 cell array

    {1×5 struct}    {1×5 struct}

fx >> Stims      = protosc_loadImages(xFileLists)
```

Furthermore, the analysis does not need to be based on a singular images. For example, the toolbox was used in Stuit et al., (2021), where participants selected one of two images with an eye movement. Since two images per trial were used, the feature differences for each image pair were calculated and assigned a label that reflected which image the participant selected with an eye movement. The following is an example to create feature differences based on 2 images:

```
% Example image feature differences:
% Create settings variable
settings = protosc_Settings;
% Get filelist for images on the left
FileLists_1 = protosc_ImageFileList_CustomLabels(xlsFilename_1);
% load the images
[Stims_1,settings] = protosc_loadImages(FileLists_1,settings);
% convert the images to features
[FourierFeatures_1,refGrid_FourierFeatures] =
protosc_Stims2Fourier_Magnitude_Features_Segmented(Stims_1,settings);

% Get filelist for images on the right
FileLists_2 = protosc_ImageFileList_CustomLabels(xlsFilename_2);
% load the images
[Stims_2,settings] = protosc_loadImages(FileLists_2,settings);
% convert the images to features
[FourierFeatures_2,~] =
protosc_Stims2Fourier_Magnitude_Features_Segmented(Stims_2,settings);

% subtract one from the other per category (since the features are in a
% cell-type variable)
for category = 1:size(FourierFeatures_1,2)
    FourierFeatures_differences{category} = FourierFeatures_1{category} -
    FourierFeatures_2{category};
end


AllData = protosc_Features2DataMatrix(settings,FourierFeatures);
settings.ReferenceMap = protosc_SetRefGrids(settings,refGrid_FourierFeatures);
out = protosc_FeatureSelection(AllData,settings);
```

Autogenerated Files

Protosc can generate three files after running the analyses. The first file is an excel file containing the features values for the significant features associated with each of the images used in the analysis so they can be used as covariates in subsequent analysis outside of Protosc. The file is generated via: `protosc_report_feature_table(out,filename)`

The second file is a text file that contains a report on the feature extraction approaches used and the settings for the feature selection analysis. The file is generated via:
`protosc_report_methods(out,filename,get_FourierFeatures,get_Fourier_Mag_Pha,get_Fourier_Pha,get_HOGFeatures,get_ColDistributions,get_Pixels)`

Note that all inputs after `filename` are booleans reflecting the features used in the analysis.

The last file is a text file reporting the statistical results of the analysis (e.g. feature significance and model performance comparisons), including basic interpretations. The file is generated via: `protosc_report_results(out,filename)`

Examples for customization of analysis

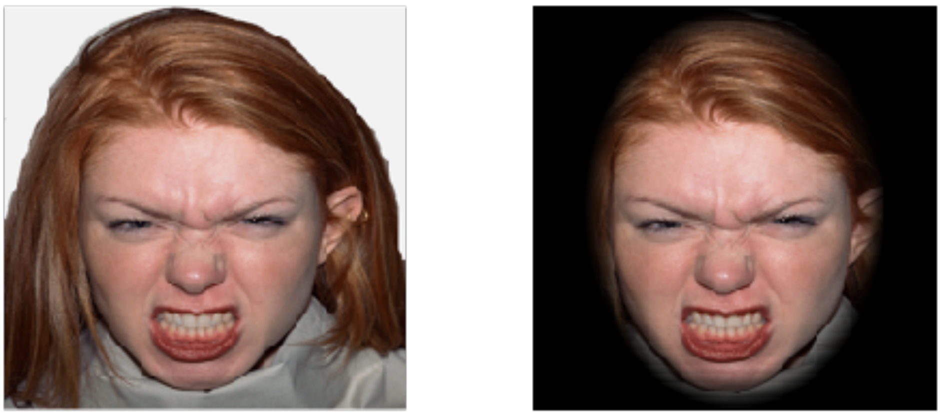
Directly supplying images without reading them from files

```
% The stims variable which contains the images needs a cell for each
% category and in each cell the
% images is placed as follows:
% Stims{category}(example_number).im = the example image
% Here is an example to fill the Stims variable yourself with greyscale
% 100x100 white noise images:
for category = 1:2
    for image_number = 1:10
        Stim{category}(image_number).im = rand(100);
    end
end
```

Adding an aperture to the images in Stims

```
% The following creates an oval aperture with a smooth edge
% width is set a 75% of the image width, height is left at full height
oval =
protosc_im_smoothedge_oval(round(size(Stims{1}(1).im,2)*.75),size(Stims{1}(1).im,1)
);
% go through all categories in Stims
for category = 1:size(Stims,2)
    % go through all examples in the category
    for example = 1:size(Stims{category},2)
        % go through all channels in the image
        for channel = 1:size(Stims{category}(example).im,3)
            % create an updated version of the Stims variable.
             updated_Stims{category}(example).im(:,:,channel) =
             Stims{category}(example).im(:,:,channel).*oval;
        end
    end
end
```

The figure below shows the one of the original images on the left, and the same image but updated with the oval on the right:



Other functions for creating an aperture are:

`protosc_im_circle` (for a circular aperture)

`protosc_im_smoothedge_circle` for a circular aperture with a smooth edge