CPReader version 1.12r: using the data produced by Wings

David Houle (dhoule@bio.fsu.edu)

Eladio Marquez (emarquez@bio.fsu.edu)

March 2014

CPReader (© Eladio Marquez 2012-2014 / © The Mathworks, Inc. 1984-2013) is a companion program to Wings, which takes images of Drosophila wings, detects curves that fit an *a priori* template, and then fits spline curves to them, as shown in Houle et al. (2003) CPR reads the output of Wings, performs basic analysis of these shape data, and allows you to extract the information that you wish to analyze from them.

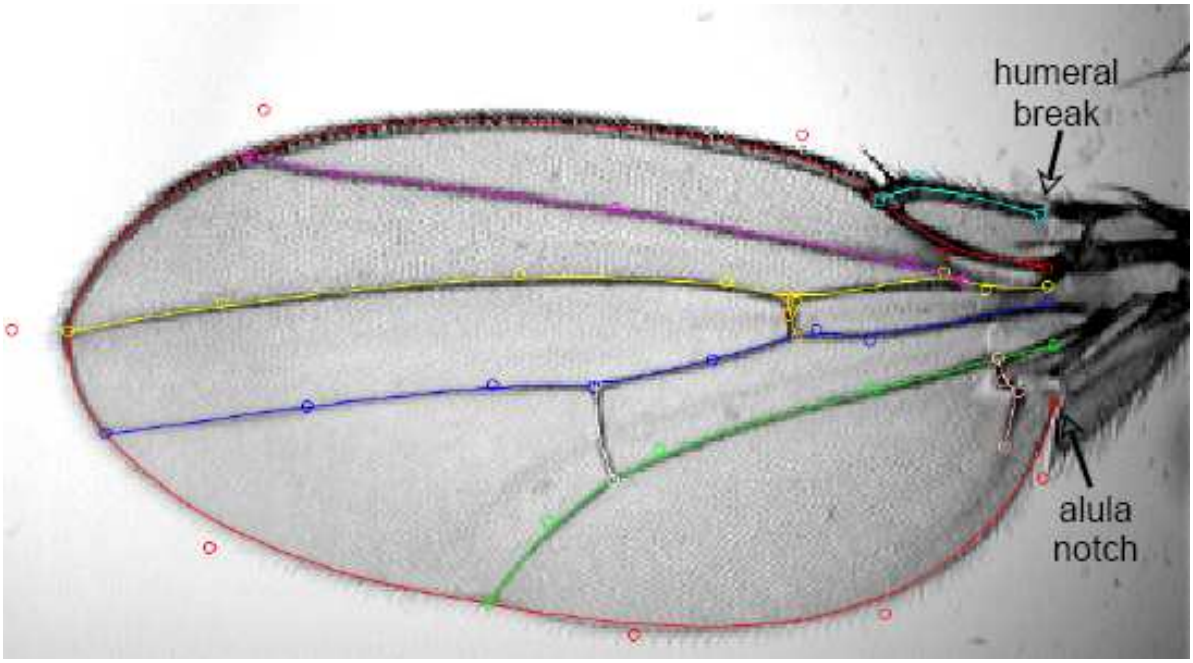

Figure 1. Nine B-splines fit to veins and edges of a *Drosophila* wing. Circles denote the positions of spline control points. The two starting landmarks that must be furnished to Wings are shown by the arrows. Starting landmark number 1 (in the ASC file) is the humeral break, and number 2 is the alula notch.

Wings4 is a JAVA program that will run on either Windows or Apple machines. CPR is a Windows program written in Matlab. Both CPR and Wings are available from http://bio.fsu.edu/~dhoule/Software/ . To run CPR, you must also download and install Matlab Component Runtime libraries (MCRInstaller.exe) for Windows, which can obtained from Eladio Marquez' personal website (http://www-personal.umich.edu/~emarquez/morph/). The hardware we use to image wings is described in Houle et al. (2003).

What's a spline, and why use them?

Splines are continuous curves, so they can represent the edges of objects. They are essentially equations that give the location of the curve at any point between its end points. For an object like a fly wing, the most obvious features are the veins and edges of the wing, which are curves, so a spline can capture those well.

Wings4 fits quadratic B-splines, one choice from the family of possible spline equations. The data used to compute our spline curves are the spatial coordinates (*x* and *y* here) of a set of *control points* (CP). For the wing in Fig. 1, CP are the small circles. Note that they are generally not on the curve, with the exception of those at the end of a spline curve. For example, there are 5 CP on the green curve in Fig. 1, one at each end, and three in the middle that are not precisely on the curve. The relationship between CP and the curves is discussed in the section B-splines in Wings below.

Spline data format

The Wings program will create a .cp file with data from each of the wings. The first part of a .cp file looks like this:

```
# Image: crossb1573.tif 286.5 79.5 288.0 125.0 Yunily 03/04/02 Wed_PM F39-21xM39-54v2
M 0.006985684.0    Resolution: 1.000000 1.000000
9
14
287.936387 123.536901
280.173680 147.223963
243.396758 173.045201
182.313180 178.688463
155.460223 171.009150
127.945452 168.800180
78.535084 149.530000
50.550815 113.646119
67.005519 77.287631
129.405146 54.736720
230.262215 65.291896
255.874894 82.691561
267.957581 89.166542
286.575417 92.234599
5
153.337939 171.337700
165.054074 155.883137
189.648997 137.097543
242.659892 118.950596
287.398487 111.165197
8
65.674047 133.003511
.
.
.
```

The first line of the cp file just copies over the information about this wing from the .ASC file. The second line (with just the number 9 on it) gives the number of spline curves to follow. The next line gives the number of control points on that spline (14 in this case). The next 14 lines

give the *x* and *y* coordinates. There follow additional blocks giving the number of coordinates, then the *x*, *y* pairs. Coordinates are in pixel units.

CPReader

CPR, short for CPReader, is our program for reading and working with spline data from .cp files. CPR incorporates the following functions:

1. Collection of data from many .cp files in different subdirectories into a single large matrix, and saving the data matrix to a standard format.
2. Procrustes superimposition of curves to make them comparable. Procrustes superimposition performs rigid scaling, translation and rotation to optimize fit of all curves to the mean curves, using a least-squares criterion. Scale information is saved for each individual as centroid size.
3. Extraction of pseudolandmarks and sliding semi-landmarks, which enrich wing shape information by sampling positions of points along veins in between landmarks. The number of pseudo/semi-landmarks is controlled by the user.
4. Extraction of univariate measures of wing shape, such as distances between landmarks, lengths along curves, areas enclosed by veins, and points along curves.
5. Principal components analysis (PCA) of shape data and visualization of specimens scores and deformations implied by PC axes. This useful for finding major outliers.
6. Visualization of shape deformations of individual specimens with respect to the Procrustes mean of the sample. CPR provides visualizations based on landmarks, semilandmarks, curves, animations, and interpolating splines. Splines can be visualized as grids or heat maps (i.e., *parrot plots*). These visualizations are better implemented in the program Lory (Marquez et al. 2012), also available from http://bio.fsu.edu/~dhoule/Software/ .

Setup

CPR executable, example files, and this guide are included in the distributed files cpr32.zip and cpr64.zip, which contain compiled versions for 32- and 64-bit Windows, respectively. To run CPR, simply unpack and run the .exe file.

CPR is created and maintained using Matlab®, and therefore the compiled version requires certain Matlab® libraries that are specific for each version. Current CPR releases have been compiled using Matlab® Compiler Runtime (MCR) libraries version 7.11, which are required to be installed in Windows prior to execute CPR. These libraries are available for download at http://www-personal.umich.edu/~emarquez/morph/. Due to licensing restrictions, the user needs to request login credentials from this site to gain access to the software (see site for further instructions).

CPR usage

Figure 2 depicts CPR's main interface. Labels in the figure correspond to CPR functions, which are described as follows.

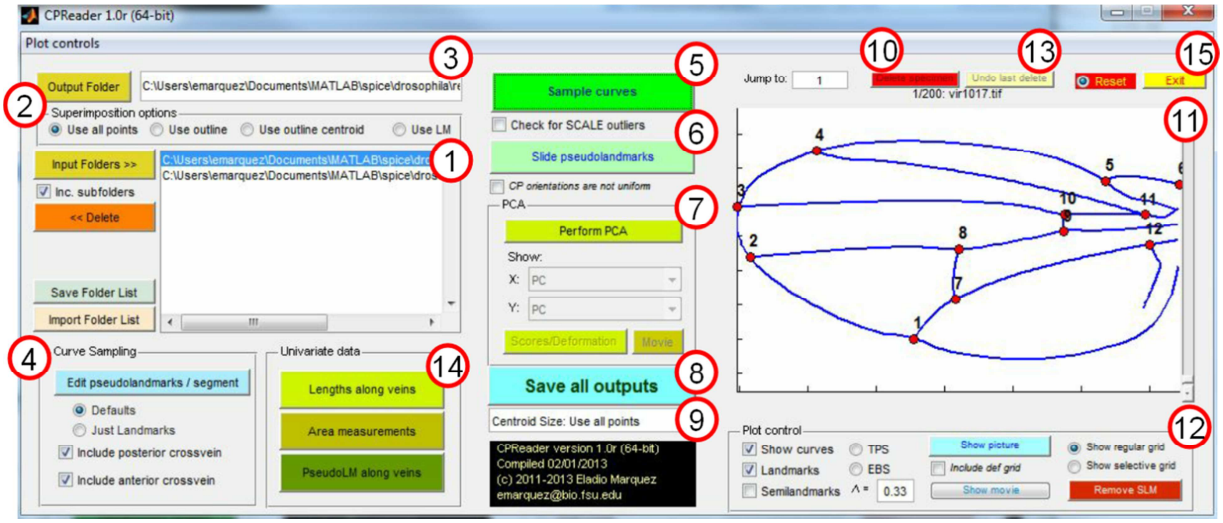

Figure 2. Main CPR interface. Two subdirectories containing CP files have been read in using program defaults.

CPR functions:

1. *Data input panel*. When reading files, CPR searches entire folders for .cp files to read-in and it loads the most recent version of each wing's .cp file available in that folder. Users control which folders CPR looks into when it scans for .cp files and these folders are listed in the large box in the right side of the panel. **Input Folders** and **Delete** buttons are used to populate this list or to remove individual folders, respectively; whereas **Save Folder List** and **Import Folder List** can be used to write or read a text file containing directory names. This function is particularly useful when loading complex folder lists, as it avoids the need to load them individually with every use of the program. Folder lists contain text files with full path names written one per row. Alternatively, a list of folders can be created using any ordinary text editor, then imported into CPR. Checking the box **Inc. subfolders** instructs CPR to look for subfolders nested one level deep within *each* of the folders selected using the **Input button** and adds them to the folder list.

   COPYRIGHT NOTICE: **CPR's input folder interface implements Matlab® program uipickfiles.m written by Douglas Schwarz.**

2. *Superimposition options*. This panel allows users to choose which set of landmark points are used for superimposition. Landmark points that are used to determine alignment parameters are called guide points. The remaining points are superimposed using the

scaling, translation and rotation parameters estimated from the guide points. The choices of superimposition method are:

a. **Use all points** includes all landmarks and semi-landmarks as guides. This option weights all points equally and implements Generalized Procrustes Superimposition (GPS).

b. **Use outline** uses points on the outline curve as guides. This implements the recommended procedure in (Van der Linde and Houle 2009).

c. **Use outline centroid** uses all points for superimposition as in GPA, but calculates the centroid about which configurations are rotated using only outline points.

d. **Use LM** takes just the 12 landmarks (shown as numbered points in Figs. 2 and 3) as guides, and calculates centroid size using only those landmarks.

3. *Output destination*. The **Output Folder** button specifies the location for temporary and output files generated by CPR.
4. *Curve Sampling*. This panel allows users to control how curves are sampled. Clicking on **Edit pseudolandmarks/segments** brings up two screens in sequence where users can edit the number of pseudolandmarks to be acquired from each interlandmark segment; values of 0 are useful for excluding curve segments that are not reliably estimated in your data. Segment names are coded as vein names (OL=vein L1 plus the trailing edge of the wing, L2=vein L2, etc.) and location (P=posterior, A=anterior, p=proximal, d=distal, m=medial). Points sampled using this method will be placed equidistantly from each other and with respect to the flanking landmarks (hence the *pseudo-* prefix). Figure 3 maps segment labels on a schematic wing after acquiring the default number of pseudolandmarks. Click on **Defaults** to reset pseudolandmark counts to their default values (Clicking on the upper-right **Reset** button does **not** reset this selection). Clicking **Just landmarks** has the same effect as setting all segment counts to zero.

Warning: **To improve comparability of geometric data among specimen configurations, it is strongly advised to use the sliding function (see 6), which optimally positions individual configurations' pseudolandmarks along curves**.

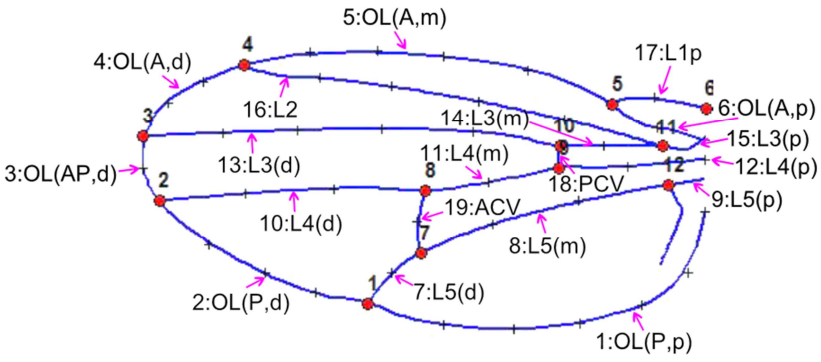

Figure 3. Labels assigned to interlandmark segments by CPR. Pseudolandmarks (represented as crosses along veins) are sampled independently and equidistantly over each of these segments. Shown pseudolandmarks correspond to default CPR values.

Crossveins. Checkboxes for including and/or excluding crossveins from the data are also provided; if both boxes are checked, CPR samples only landmarks and pseudolandmarks along longitudinal veins L1-L6. Note that this is different from setting ACV and PCV pseudolandmark counts to zero, which only affects the number of points sampled in between flanking landmarks, not the landmarks themselves.

5. *Curve acquisition*. Once you have made the above choices, click on the **Sample curves** button. This will read in all the .cp files in the selected subdirectories, sample semi-landmarks, superimpose the resulting data, and display the superimposed curves in the box on the right side of the interface.

   Clicking on **Check for SCALE outliers** prior to initiating curve acquisition prompts CPR to search for potential outliers in the SCALE parameter of .cp files. CPR uses Rosner's Outlier Test (Rosner 1983), which is valid for samples of size 25 of larger (Gibbons and Coleman 2001). Only the 25% most extreme entries are tested for outlier status. Keep in mind that multiple specimens may have the same SCALE parameter, and thus this algorithm tends to flag *correctly* entered SCALE values as outliers when a few distinct values are present in the data at different frequencies. CPR does not remove outliers automatically, but rather produces a list of potential outliers from which the user can select individual specimens for removal (Figure 4). To remove individuals, press **Select all** or customize your selection using the Ctrl and Shift keys, and then press **Exclude selected wings**. This procedure will not affect the actual .cp files, only the loaded data. Press **Cancel** to ignore all flags and to continue reading .cp files as normal. Note that correction of incorrect scale information in the .cp files may be preferable to removing outliers.

   Note: Since centroid size calculations are adjusted for the SCALE parameter, it is useful to test for outliers for this value at least once per data set.

6. *Sliding semilandmarks*. We adopt the term *pseudolandmark* to denote points equidistantly sampled along a curve, whereas *semilandmark* refers to points whose locations along the curve have been adjusted according to an optimality criterion. Optimal sliding is important because pseudolandmarks are not generally comparable across configurations with variable shapes. Sliding assumes that pseudolandmark position *along* the curve is arbitrary, while their position *on* the curve is not. Optimization proceeds iteratively by allowing points to slide along wing veins toward each point's mean, whereupon all configurations are re-superimposed using GPS; iterations stop when the variance of Procrustes residuals reaches a minimum. Landmarks and semilandmarks are then re-aligned using the method specified in (2).

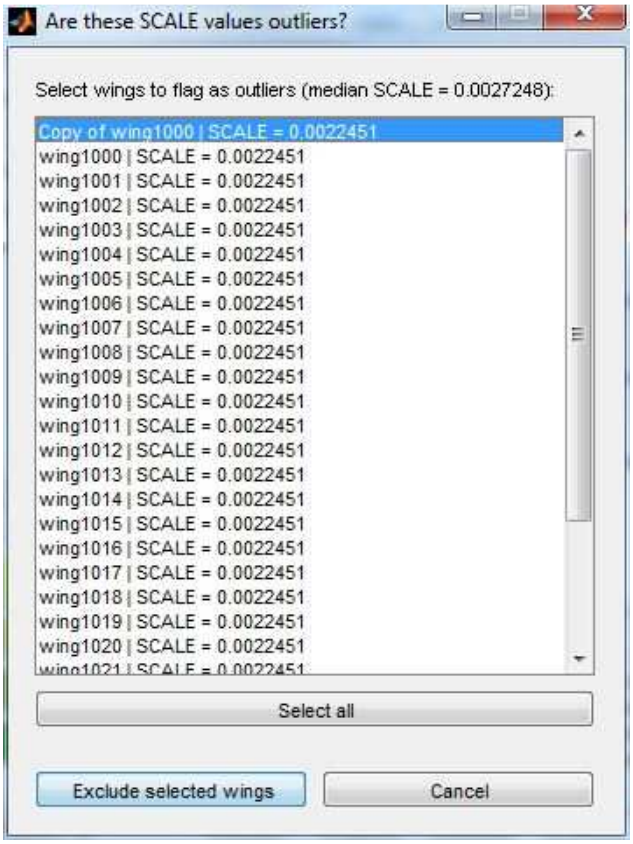

Figure 4. SCALE outlier flags generated by CPR. A median SCALE value is provided for comparison. Note that in this case, none of the shown values are actual outliers, but rather represent an "outlying" minority among all loaded values. In this case, it would be appropriate to proceed by clicking **Cancel**, ignoring the message.

Prior to sliding pseudolandmarks, determine if all of the loaded configurations are in the same orientation; that is whether some configurations are *reflections* of others. CPR will detect orientation and reflect configurations to the orientation found for the first configuration read in the data if the box **CP orientations are not uniform** is checked. Unlike superimposition algorithms implemented in CPR, sliding algorithms are *not* invariant to reflections. Consequently, in those cases sliding algorithms will converge to the wrong solution or fail to converge at all.

Note: Sliding is a computationally intensive step that can potentially take a long time. Data sets of hundreds of configurations can finish in minutes, but if many thousands of individuals are included, this step can take days. We recommend doing a PCA both before and after this step. A PCA analysis on pseudolandmark (i.e., pre-slid) data will still enable the detection of most outliers, so that you do not have to repeat this very time-consuming step.

7. *Principal Component Analysis (PCA).* Functions in this panel perform a PCA on the loaded data set and provide basic tools to explore deformations associated to and the distribution of scores along PCs. Pressing **Perform PCA** prompts CPR to run the analysis anew (taking into account any specimen removal done since any previous run), and opens a window with a plot of specimen scores on the first two PCs (Fig. 5a); score projections on other PCs can be also plotted by selecting the appropriate X and Y axes

under **Show**. PC selection menus also provide the percentage of the total variance captured by each PC.

To aid in the identification of outliers, CPR computes the Euclidean distance between the origin and each specimen in the PCx and PCy plane and shows specimen labels for points whose distance to the origin is larger than the mean distance (Fig. 5a).

To visualize eigenvectors as shape deformations, CPR offers two alternatives: (1) First, pressing **Scores/Deformation** uses the open window to toggle between plots of scores between PCs assigned to the *x* and *y* axes, and the deformation implied the eigenvector scored *on axis X* of the plot (Fig. 5b). Parameters used to build this plot are set in the *Plot control* panel (see 13); (2) second, pressing **Movie** opens another window which shows the eigenvector scored on the *x* axis of the PC plot as an animated linear deformation. Both static and animated deformations represent inferred changes between the grand mean over all loaded configurations and the absolute maximum score over the PC being represented.

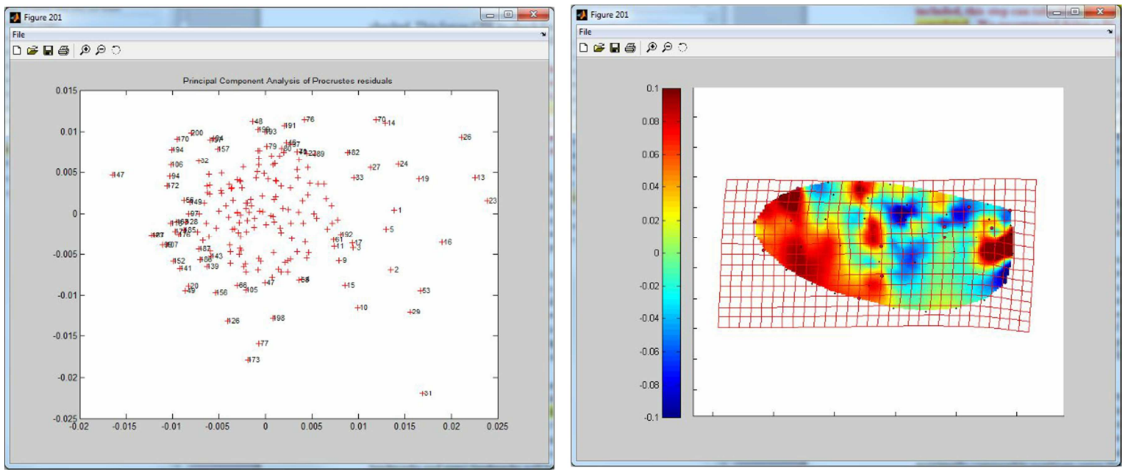

Figure 5. (a) Principal component scores on PC*x* and PC*y*, where *x* and *y* are selected by the user, and (b) deformation implied by PC*x*.

8. *Data output*. CPR does not automatically save processed data. To output landmark and semilandmark data to a physical file, click **Save all outputs**. This writes two tab-delimited text files to the directory specified in (3), named **`Output_Landmarks.dat`**, which contains data for the 12 landmarks described above, and **`Output_Landmarks+Veins+Outline.dat`**, which contains data for all landmarks and semilandmarks in the analysis. Each data file includes headers, specimen labels read from .cp files, landmark/semilandmark coordinates as columns x_1, y_2,…, x_n,y_n (with n = number of landmarks), and a column of centroid sizes (CS) computed using the landmarks and semilandmarks included in the file (see *CPR output format* for further details). A message box is shown upon successfully writing files to the output directory.

Note: **If CPR finds older files with the same name as standard output files in the destination directory, old files are overwritten without warning.**

9. *Output choices for centroid size*. The last column of the data output is configurations' centroid size (see *CPR output format*); this menu allows users to select which landmarks/semilandmarks to include in this centroid size calculation. This selection affects only the output file **`Output_Landmarks+Veins+Outline.dat`**), which includes the entirety of the data, whereas the centroid size for the landmarks-only data set is always computed from landmarks only. The three options offered in the pull menu are:

    a. **Use all points**: CPR uses all landmarks and pseudo-/semilandmarks to calculate centroid size (default);

    b. **Use aligned points**: CPR uses only points selected as guides for superimposition (see 2) to calculate centroid size;

    c. **Use unaligned points**: CPR uses only points not selected as guide for superimposition to calculate centroid size.

    Note that all three options produce the same result when all points are selected as superimposition guides.

10. Plot *navigation*. Use the **Jump to:** box to display curves or shape deformations of individual configurations. This is useful to visualize curves for potential outliers detected by inspecting PC scores, and to find the name of the corresponding .cp file. You can return to Wings to check and, if necessary, edit these individuals. Entering 0 in this box will show all loaded individuals, superimposed according to the algorithm chosen in (2).
11. *Plot navigation by slide bar*. Use the slider to browse through configurations. Moving the slider completely to the bottom will show all loaded individuals, superimposed according to the algorithm chosen in (2).
12. *Plot control*. This panel allows users to control a number of visualization options for plots displayed in the main interface and PCA-related plots. The program Lory (Marquez et al 2012), also available from http://bio.fsu.edu/~dhoule/Software/, offers more advanced controls and is recommended to produce visualizations with better quality, particularly those with interpolated deformations. This panel also provides with a tool for removing individual pseudo/semilandmarks from the current data set. The following list describes the panel components:
    a. **Show curves/Landmarks/Semilandmarks**: Self-explanatory; these features can be included jointly or individually in visualizations.
    b. **TPS/EBS/λ**: Deformations between the mean shape and individual configurations depicted as interpolating thin-plate splines (TPS) or elastic-body splines (EBS). Parameter λ is the Poisson parameter used in EBS interpolation (Marquez et al. 2012). Interpolations are shown as inferred local expansions and contractions based on interpolated Jacobian determinants, which we refer to as parrot plots after the colors in the visualization.

c. **Show regular grid/Show selective grid**: In addition to parrot plot visualizations, CPR overlays a deformation grid based on the same interpolation function (i.e., TPS or EBS) used for parrot plots. Grids are computed by evaluating these functions at specific points in the plane of the configuration. Regular grid points (default) are placed equidistant from each other, whereas selective grid points are placed such that sampling density is proportional to the magnitude of local variation (see **Plot control** menu options for further explanation). Unselecting both grid buttons removes the grid from the visualization.
d. **Show picture**: this brings up a window where a TIF file for the wing, if present in the same input folder as the corresponding .cp file, is shown with an overlay of curves, control points, and, if **Include def grid** is checked, landmarks, semilandmarks and a deformed regular grid.
e. **Remove SLM**: Pressing this brings up a numeric list of currently loaded pseudo/semilandmarks (starting at number 13, since 1-12 correspond to landmarks), which can be selected individually or in group for deletion. Clicking **Delete** immediately removes these landmarks are renumbers the remaining ones while preserving their order. When this function is being used, points in the visualization panel are changed from symbols to numbers which correspond to the numbers in the list.

13. *Individual specimen deletion.* Use **Delete specimen** to remove the currently shown individual from the loaded data (corresponding physical files will remain untouched). **Undo last delete** recovers the last specimen deleted using the **Delete specimen** button.
14. *Univariate data*. CPR provides three kinds of univariate variables that can be extracted from curve and landmark data. In the current version, CPR offers only one way to compute these variables, which are automatically saved to the output folder (3) upon clicking the appropriate button. Data sets are saved as tab-delimited .dat files with a standard file name. A message box is produced upon successfully saving a set. These options are described as follows:
    a. **Lengths along veins**: this estimates the lengths, in pixel units, along the 18 interlandmark vein segments of the wing, which are returned as corresponding variables labeled L1,…, L18. Length along the veins is calculated from the full spline fits, so it is NOT the straight-line distance between landmarks. In addition to the data file, an image file named **`Output_vlength_legend.png`** is automatically saved. This describes the position in the wing of each of the 18 lengths. Data file is named **`Output_vLength.dat`**, which also includes headers and individual labels extracted from .cp files (see *CPR output format*).
    b. **Area measurements**: this estimates the area, in squared pixel units, of 8 vein- and crossvein-bound regions in the wing, which are returned as corresponding variables labeled A1,…, A8. In addition to the data file, an image file named **`Output_area_legend.png`** is automatically saved. This describes the position in

the wing of each of the 8 areas. Data file is named **`Output_area.dat`**, which also includes headers and individual labels extracted from .cp files (see *CPR output format*).

c. **PseudoLM along veins**: this calculates points equidistantly placed along veins and cross-veins (i.e., pseudolandmarks), and saves a separate output file for each curve. Pressing this button brings up an input window where users can select the number of points to sample along each vein. Sets of knots corresponding to distinct veins are saved in separate files, named **`BsplinePLM_X.dat`**, where X equals vein denominations (i.e., L1-L6, ACV, PCV; see Fig. 3). Data files also include headers and individual labels extracted from .cp files (see *CPR output format*).

15. *Reset*. The **Reset** button erases all currently loaded data and restores options to their state upon startup. It is not required to click this button prior to loading a new data set, unless the user wishes CPR to forget previously selected options. The number of pseudolandmarks sampled per segments is *not* deleted when pressing this button, but instead it has its own dedicated reset function (see 4).
16. *Plot controls menu*: the **Plot controls** menu contains an array of customizable options for deformation visualizations, as well as links to save current plots to image files. These and more options are implemented in Lory (Marquez et al 2012; also available from http://bio.fsu.edu/~dhoule/Software/), and users are advised to use that software for better results. Following is a description of each of the items in this menu:
    a. **Modify deformation magnification**: Users can enter a multiplier value in the range [-5,5] which CPR applies to the shape difference between sample mean and individual configurations (default = 1).
    b. **Modify regular grid extent**: Users can change the coverage of regular grid overlaid on a warped configuration. The entered value is the proportion of the absolute maximum value over the *x* and *y* axes of the configuration (default = 1.2, i.e., grid is 20% bigger in both directions than the configuration is in its longest direction).
    c. **Modify regular grid density**: Users can change the density of intersection in a regular grid by adjusting the number of internal intersections along the long axis of the configuration (default = 30).
    d. **Modify vector magnification**: Deformation vectors (i.e., landmark-wise differences between individuals and reference) are often difficult to see. This parameter allows changing the length of these vectors without altering the true magnification of the deformation. This is provided as a visualization aide, but should be used with caution since it results in a misrepresentation of the actual magnitude of the deformation (default = 0 = true magnitude of vectors).
    e. **Modify selective grid parameters**: Selective grids provide an alternative to regular grids in positioning points for evaluating deformation interpolations. They

are described in Marquez et al (2012) and fully implemented in Lory (available from http://bio.fsu.edu/~dhoule/Software/), which also allows users to save values interpolated at grid nodes. Starting from the Delaunay tessellation array defined by a configuration's semilandmarks and/or landmarks, nodes defined at the center of these triangles are dropped if they contain redundant information with respect to their neighboring nodes, or kept if they are in a region where local change or variation is being detected. Clicking on this menu item brings up input fields for three parameters:

i. **Number of tessellation passes (>= 1)**: A tessellation "pass" is a round of triangulation based on nodes defined in previous round. The first round is based on the original semilandmarks and/or landmarks, whereas subsequent rounds are based on the circumcenters of triangles found in previous passes. Keep in mind that each pass can take a considerable time to process;

ii. **Threshold as within-wing %AREA change (> 0)**: As described in Marquez et al (2012), local area changes are measured using the determinant of the Jacobian of the interpolant function (i.e., TPS or EBS) evaluated at each triangulation node. Point neighborhood is determined using an extra triangulation pass (i.e., neighborhood of points after a one-pass tessellation is defined using a second pass, and so on). This parameter determines the selection threshold for the proportional difference in area between neighboring points in a single configuration, so that a node is dropped if this difference is smaller than the threshold (default = 0.1; use a positive value ≈ 0 to keep all tessellation points);

Threshold as between-wing %AREA change (> 0): This parameter controls the selection of nodes compared across all specimens, and defines a cutoff for the proportional difference in local change in area between each configuration and the sample mean for the change at the same node. Nodes that differ from the mean area change by less than the specified threshold are dropped from the grid (default = 0.01; use a positive value ≈ 0 to keep all tessellation points).Note that the two thresholds are applied to grid node selection in tandem, so that the final set of removed nodes is the *union* of the sets chosen by both criteria. In order to retain all nodes from every tessellation pass, make sure to set both parameters to a positive value ≈ 0.

f. **Modify parrot plot resolution**: User can adjust the resolution of the color grid by changing the number of pixels along the long axis of the image. Although a higher pixel count means higher resolution (i.e., pixels are drawn smaller), it also has a cost in processing speed. Minimum resolution is 3, whereas entering 0 prompts CPR to not drawing a color grid (default = 150).

g. **Change color scheme**: Each color scheme specifies how local deformation values are translated into colors. Available schemes available in CPR are Matlab® standard colormap schemes (default = "jet").
h. **Change colormap breadth**: *Colormap breadth* refers to the minimum and maximum local change in area (in base-2 log scale) to be mapped to the extreme colors in a colormap. Values are set independently and can be symmetric or asymmetric, although often a symmetric scale is easier to interpret. Leaving one or both input fields empty prompts CPR to choose limits automatically, which is sometimes a good first approximation when choosing the most appropriate colormap limits (default = [-0.3,0.3].
i. **Save plot as bitmap**: Available formats/extensions are 24-bit JPEG/jpg, 24-bit uncompressed TIFF/tif, 24-bit compressed TIFF/tif, and 16-bit Windows Bitmap/bmp.
j. **Save plot as vector graphics**: Available formats/extensions are PDF/pdf, Color EPS Level 1/eps, BW EPS Level 1/eps, Enhanced Metafile/emf.
k. **Save Segment Structure**: A *segment structure* is a connectivity matrix automatically generated by CPR which describes how landmarks and semilandmarks are connected to draw a typical wing. Matrix entries are 1 when there is a link between two points, and 0 otherwise. This matrix is internally used when producing animations but is included also as an output as it can be used in other visualization programs (e.g., Matlab® Gplot).
l. **Play a sound when done reading data/sliding pseudolandmarks**: When checked (default), Matlab® attempts to play a WAV file from the Windows sounds folder upon completion of data acquisition (i.e., Sample curves button). If the folder `C:\Windows\Media` is not found, or if none of the compatible files is available, no sound is produced (note that success of data acquisition is unaffected by this). Compatible sound files are `tada.wav`, `chord.wav`, `chimes.wav`, and `ding.wav`.

CPR output format

CPR output files are tab-delimited text files, with one wing per line. They include a header with variable names in the first line and labels read from CP files for each specimen. Here is an example:

Header:

CPFile File O1x O1y O2x O2y Perp Date Time Tags Scale Sex x1 y1 . . . x49 y49 CS

Typical entry:

wing1000 \path\anc1000.tif 540 151 558 239 Jeff 19/03/05 1:30PM w1118 1261.10 F -0.0601 -0.0972 . . . -0.0317 -0.0388 2.951

Meaning of the listed variables (columns) in sequence:

1. CPFile: name of the cp file (as assigned by WINGS).

2. File: full physical path to splined image.
3. O1x: the x coordinate in pixels of orientation landmark 1 (humeral break, see Fig. 1).
4. O1y: the y coordinate in pixels of orientation landmark 1.
5. O2x: the x coordinate in pixels of orientation landmark 2 (alula notch, see Fig. 1).
6. O2y: the y coordinate in pixels of orientation landmark 2.
7. Perp: the 6th column of information from the ASC file. In the Houle lab, this is the name of the person that imaged the wing.
8. Date: the date the image was recorded.
9. Time: the 8th column of information from the ASC file. In the Houle lab, this is the time of day the wing was imaged.
10. Tags: the 9th column of information from the ASC file. In the Houle lab, this is the genotype of the fly.
11. Scale: the number of mm/pixel in the image file.
12. Sex: sex of the fly imaged.

x1, y1, x2, y2, . . . x*n*, y*n*: Any number (*n*) of pairs of x and y coordinates corresponding to landmarks and semi-landmarks. The first 12 pairs are the Type 1 landmarks formed by vein intersections and the humeral break. The remaining *n*-12 pairs are semilandmarks.

CS: centroid size in mm

Large data sets in CPR

CPR retains a great deal of information about each individual, and consequently runs rather slowly. For data sets of thousands to tens of thousands of individuals we use a stripped down version of CPR called *CPRlite* that runs orders of magnitude faster, but does not provide many of the functions in CPR. Contact us at dhoule@bio.fsu.edu if you would like a copy.

B-splines used in Wings

Splines are piecewise polynomial functions parameterized with the spatial locations of points. We use a parameterization based on control points, which are not necessarily on the curve that is being represented. Knots are points on the curve that are the mean position of adjacent pairs of control points. Piecewise means that the full curve is a combination of n short curves that are interpolated based on the subset of the control points that are nearest the section of curve being reconstructed. Assume that we have a curve running generally left to right. The control points are numbered sequentially from left to right from 0 to n+1. Wings uses a quadratic parameterization that uses the three closest control points to generate the curve. For example, consider the ith knot that is the mean of control points i-1 and i. To the left of this knot, the three control points used for interpolating the curve are i-2, i-1 and i. To the right of this point, we use points i-1, i, and i+1. Choosing a higher number of control points to represent a curve allows it to have a larger number of changes in curvature, that is to be more wiggly. The number of control points for each curve on the wing was chosen so that the resulting curve fits well over a variety

of wings. Simple curves, such as the cross-veins, are represented by just three control points, while the more complex outline of the wing takes 14 control points.

Given the set of control points $\left\{\left(c_{x0}, c_{y0}\right), \left(c_{x1}, c_{y1}\right), \cdots, \left(c_{x(n+1)}, c_{y(n+1)}\right)\right\}$, the corresponding knots are

$$k_{x(i)} = \left(c_{x(i-1)} + c_{x(i)}\right) \Big/ 2$$
$$k_{y(i)} = \left(c_{y(i-1)} + c_{y(i)}\right) \Big/ 2.$$

The *i*th curve segment is runs from knot *i* to knot *i*+1. The location of the point $\left(x_{i\square p}, y_{i\square p}\right)$ that is proportion *p* of the length between knot *i* and knot *i*+1 is

$$x_{i\square p} = c_{x(i-1)} w_1 + c_{x(i)} w_2 + c_{x(i+1)} w_3$$
$$y_{i\square p} = c_{y(i-1)} w_1 + c_{y(i)} w_2 + c_{y(i+1)} w_3, \text{ where}$$
$$w_1 = \frac{(1-p)^2}{2}$$
$$w_2 = \frac{1}{2} + p - p^2$$
$$w_3 = \frac{p^2}{2}.$$

Each segment of curve can be reconstructed to the degree desired by sampling *p* in the interval $p = 0$ to $p = 1$, with the extreme values 0 and 1 returning the *i*th and *i*+1th knots respectively.

Wings' output returns the first and last knots (the actual ends of the curves) instead of the 0^{th} and *n*+1th control points. This is convenient as the ends of many of the curves are often Type I landmarks that can be used directly in geometric morphometric analyses. To actually compute curves (semi-landmarks in geometric morphometrics jargon) from Wings output, these end knots can either be converted to their corresponding control points as

$$c_{x0} = 2k_{x1} - c_{x1}$$
$$c_{y0} = 2k_{y1} - c_{y1}, \text{ and}$$
$$c_{x(n+1)} = 2k_{x(n)} - c_{x(n)}$$
$$c_{y(n+1)} = 2k_{y(n)} - c_{y(n)}$$

or the weights of the 1^{st} and last curves adjusted to compensate for the combination of knots and control points.

Authorship and support

Wings was written by Kim van der Linde, supported by NSF grants DEB 019129 and 0950002, and by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 RR021813. The underlying spline fitting algorithms were originally written by Feng Lu and supported by NSERC. CPR was written by Eladio Marquez, supported by NIH R01 GM094424, and NSF DEB-0950002.

Cited references

Rosner, JB. 1983. Percentage points for a generalized ESD many-outlier procedure. Technometrics 25:165-172.

Gibbons RD, Coleman DD. 2001. Statistical Methods for Detection and Quantification of Environmental Contamination. John Wiley & Sons.

Houle, D., J. Mezey, P. Galpern, and A. Carter. 2003. Automated measurement of Drosophila wings. BMC Evol. Biol. 3:25.

Marquez, E. J., R. Cabeen, R. P. Woods, and D. Houle. 2012. The Measurement of Local Variation in Shape. Evolutionary Biology 39:419-439.

Van der Linde, K., and D. Houle. 2009. Inferring the nature of allometry from geometric data. Evolutionary Biology 36:311-322.