Package: coglyphr (via r-universe)

October 23, 2026

Title Compute Glyph Centers of Gravity from Image Data

Version 1.1.0

Description Computes the center of gravity (COG) of character-like binary images using three different methods. This package provides functions for estimating stroke-based, contour-based, and potential energy-based COG. It is useful for analyzing glyph structure in areas such as visual cognition research and font development. The contour-based method was originally proposed by Kotani et al. (2004) <https://ipsj.ixsq.nii.ac.jp/records/36793> and Kotani (2011) <https://shonan-it.repo.nii.ac.jp/records/2000243>, while the potential energy-based method was introduced by Kotani et al. (2006) <doi:10.11371/iieej.35.296>.

License GPL-3

Encoding UTF-8

URL https://github.com/mutopsy/coglyphr, https://mutopsy.github.io/coglyphr/

BugReports https://github.com/mutopsy/coglyphr/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.4.0)

Imports dplyr, sp

LazyData true

Suggests imager, png, jpeg, tiff, bmp, testthat (>= 3.0.0)

Config/testthat/edition 3

Repository https://mutopsy.r-universe.dev

Date/Publication 2025-12-24 09:29:22 UTC

RemoteUrl https://github.com/mutopsy/coglyphr

RemoteRef HEAD

RemoteSha a56eb7fe12ce73750ae16c91977df1e356b178a8

Contents

cog_contour 2
cog_potential 4
cog_stroke 6
draw_contour 8
draw_potential 9
draw_stroke 10
img_A 11
img_B 11

Index **12**

`cog_contour` *Compute Contour-Based Center of Gravity (COG)*

Description

Computes the center of gravity (COG) of a character-like binary image using its outer contour.

Usage

```
cog_contour(img, origin = c("bottomleft", "topleft"))
```

Arguments

`img`	An image input. Supported inputs are: • A file path to an image file (e.g., PNG, JPEG, TIFF, BMP). • A `cimg` object from the **imager** package (if **imager** is installed). • A numeric matrix `[h, w]` (grayscale) or numeric array `[h, w, ch]` (color). The image should be binary in the sense that background pixels take the value 1 (pure white), and all other pixels are treated as stroke (foreground) pixels.
`origin`	A character string indicating the location of the image origin. Use `"bottomleft"` (default) if the y-axis increases upward (Cartesian), or `"topleft"` if the y-axis increases downward (as in image arrays).

Details

In the contour-based method, the center of gravity (COG) is defined as the geometrical centroid of the convex hull (the smallest convex polygon) that encloses the stroke region of the character. The convex hull is estimated by tracing the outer contour of the glyph and computing the minimal convex polygon that wraps all stroke pixels, i.e., the foreground pixels whose intensity values are not equal to 1 (pure white). The centroid is then calculated as the arithmetic mean of the (x, y) coordinates of all pixels located within the interior of the convex polygon.

Mathematically, the contour-based center of gravity (G_x, G_y) is defined as the weighted mean of pixel coordinates within the polygon region, where each pixel contributes a value of 1 (unit mass) and background pixels are excluded. Specifically, let $p(x, y)$ be an indicator function such that

$p(x, y) = 1$ if the pixel (x, y) lies inside the convex polygon and $p(x, y) = 0$ otherwise. Then the horizontal and vertical components of the COG are computed as:

$$G_x = (\sum_{x=1}^{w}\sum_{y=1}^{h} p(x, y) \cdot x)/(\sum_{x=1}^{w}\sum_{y=1}^{h} p(x, y))$$

$$G_y = (\sum_{x=1}^{w}\sum_{y=1}^{h} p(x, y) \cdot y)/(\sum_{x=1}^{w}\sum_{y=1}^{h} p(x, y))$$

where w and h denote the width and height of the image, respectively.

This method was originally proposed by Kotani and colleagues (2004, 2011) and has been used in character analysis and font design to reflect the perceived shape of glyphs more robustly than simple stroke averaging.

Value

A list containing:

`statistics` A data frame with the following components:

- `center_x`, `center_y`: The (x, y) coordinates of the COG in pixel coordinates of the input image.
- `center_x_trim`, `center_y_trim`: The COG coordinates relative to the trimmed glyph region, excluding image margins.
- `center_x_std`, `center_y_std`: The standardized COG coordinates ranging from 0 to 1, based on the trimmed region's width and height.
- `margin_left`, `margin_right`, `margin_top`, `margin_bottom`: Margins between the glyph and the image boundary.
- `width_original`, `height_original`: Dimensions of the original image.
- `width_trim`, `height_trim`: Width and height of the trimmed glyph region, excluding margins.
- `area`: The number of pixels inside the convex hull (i.e., the total mass used to compute the COG).

`points` A data frame of (x, y) coordinates representing the contour points of the convex polygon.

References

Kotani, A. (2011). Contour-based evaluation method of center of gravity on characters and its application to font development. *Memoirs of Shonan Institute of Technology*, **45**(1), 23–33. https://shonan-it.repo.nii.ac.jp/?action=repository_view_main_item_detail&item_id=368

Kotani, A., Asai, Y., Nakamura, Y., Otuka, M., Mituyama, Y., & Onoye, T. (2004). Contour-based evaluation method of center of gravity on "LCFONT." *IPSJ SIG Technical Report*, **115**, 63–70. https://ipsj.ixsq.nii.ac.jp/records/36793

Examples

```
data(img_A) # load example image from the package
result <- cog_contour(img_A, origin = "bottomleft")

result$statistics # summary data frame
result$points # contour polygon vertices (x, y, angle)
result$origin # image origin specification
```

`cog_potential` *Compute Potential Energy-Based Center of Gravity (COG)*

Description

Calculates the center of gravity (COG) of a character-like binary image based on potential energy.

Usage

```
cog_potential(img, origin = c("bottomleft", "topleft"))
```

Arguments

`img`	An image input. Supported inputs are: • A file path to an image file (e.g., PNG, JPEG, TIFF, BMP). • A `cimg` object from the **imager** package (if **imager** is installed). • A numeric matrix `[h, w]` (grayscale) or numeric array `[h, w, ch]` (color). The image should be binary in the sense that background pixels take the value 1 (pure white), and all other pixels are treated as stroke (foreground) pixels.
`origin`	A character string indicating the location of the image origin. Use `"bottomleft"` (default) if the y-axis increases upward (Cartesian), or `"topleft"` if the y-axis increases downward (as in image arrays).

Details

In the potential energy-based method, the center of gravity (COG) is defined as the weighted mean of the coordinates of all pixels within the convex hull of the stroke region, where the weight at each pixel is determined by the potential induced by all other stroke pixels. The method assumes that each stroke pixel contributes a unit mass and exerts an attractive force on all other pixels within the convex polygon, inversely proportional to their distance, modeling a simplified gravitational interaction. To avoid excessive computation, unintended influence from remote regions, and to restrict the analysis to a perceptually relevant area, the potential is computed only within the convex polygon (i.e., the same region used in the contour-based COG calculation), rather than across the entire image.

Let S be the set of all stroke pixels, and let R be the set of all pixels within the convex polygon region. The potential at each pixel $(x, y) \in R$ is defined as:

$$p(x,y) = \sum_{\substack{(x',y')\in S \\ (x',y')\neq(x,y)}} \frac{1}{\sqrt{(x-x')^2+(y-y')^2}}$$

That is, the potential at each point in R is the sum of the inverse distances to all stroke pixels in S, excluding the case where $(x', y') = (x, y)$. Pixels outside the convex polygon are assigned a potential value of zero and do not contribute to the COG calculation.

Then, the center of gravity is computed as:

$$G_x = \left(\sum_{(x,y)\in R} p(x,y) \cdot x \right) / \left(\sum_{(x,y)\in R} p(x,y) \right)$$

$$G_y = \left(\sum_{(x,y)\in R} p(x,y) \cdot y \right) / \left(\sum_{(x,y)\in R} p(x,y) \right)$$

In other words, the COG corresponds to the weighted mean of pixel coordinates in the convex region, where weights are given by their potential values induced by the distribution of stroke pixels.

This method was originally proposed by Kotani et al. (2006) and has been used in character analysis and font design to reflect the perceived shape of glyphs more robustly than simple stroke averaging, and to further improve upon the contour-based COG by incorporating the spatial distribution of strokes within the convex polygon, thereby aligning more closely with the subjective impression of a character's center.

Value

A list containing:

`statistics` A data frame with the following components:

- `center_x`, `center_y`: The (x, y) coordinates of the COG in pixel coordinates of the input image.
- `center_x_trim`, `center_y_trim`: The COG coordinates relative to the trimmed glyph region, excluding image margins.
- `center_x_std`, `center_y_std`: The standardized COG coordinates ranging from 0 to 1, based on the trimmed region's width and height.
- `margin_left`, `margin_right`, `margin_top`, `margin_bottom`: Margins between the glyph and the image boundary.
- `width_original`, `height_original`: Dimensions of the original image.
- `width_trim`, `height_trim`: Width and height of the trimmed glyph region, excluding margins.

`potentials` A data frame containing the (x, y) coordinates and the normalized potential value for each pixel within the convex hull. The potentials are normalized so that their sum equals 1.

References

Kotani, A., Tanemura, Y., Mitsuyama, Y., Asai, Y., Nakamura, Y., & Onoye, T. (2006). Potential energy-based center of gravity evaluation of characters. *The Journal of the Institute of Image Electronics Engineers of Japan*, **35**(4), 296–305. doi:10.11371/iieej.35.296

Examples

```
data(img_A) # load example image from the package
result <- cog_potential(img_A, origin = "bottomleft")

result$statistics # summary data frame
head(result$potentials) # pixel coordinates with normalized potential values
result$origin     # image origin specification
```

`cog_stroke` *Compute Stroke-Based Center of Gravity (COG)*

Description

Computes the center of gravity (COG) of a character-like binary image using its stroke region.

Usage

```
cog_stroke(img, origin = c("bottomleft", "topleft"))
```

Arguments

`img`	An image input. Supported inputs are: • A file path to an image file (e.g., PNG, JPEG, TIFF, BMP). • A `cimg` object from the **imager** package (if **imager** is installed). • A numeric matrix `[h, w]` (grayscale) or numeric array `[h, w, ch]` (color). The image should be binary in the sense that background pixels take the value 1 (pure white), and all other pixels are treated as stroke (foreground) pixels.
`origin`	A character string indicating the location of the image origin. Use `"bottomleft"` (default) if the y-axis increases upward (Cartesian), or `"topleft"` if the y-axis increases downward (as in image arrays).

Details

In the stroke-based method, the COG is defined as the arithmetic mean of the (x, y) coordinates of all pixels that belong to the stroke region, i.e., the foreground pixels whose intensity values are not equal to 1 (pure white). This approach assumes that each stroke pixel has unit mass and contributes equally to the center calculation. The image is assumed to be binary, where the background pixels have a value of 1, and all other pixels are treated as part of the glyph.

Mathematically, the stroke-based center of gravity (G_x, G_y) is defined as the weighted mean of pixel coordinates within the stroke region, where each pixel contributes a value of 1 (unit mass) and background pixels are excluded. Specifically, let $p(x, y)$ be an indicator function such that $p(x, y) = 1$ if the pixel (x, y) belongs to the stroke region, and $p(x, y) = 0$ otherwise. Then the horizontal and vertical components of the COG are computed as:

$$G_x = \frac{\sum_{x=1}^{w} \sum_{y=1}^{h} p(x,y)\, x}{\sum_{x=1}^{w} \sum_{y=1}^{h} p(x,y)}$$

$$G_y = \frac{\sum_{x=1}^{w} \sum_{y=1}^{h} p(x,y)\, y}{\sum_{x=1}^{w} \sum_{y=1}^{h} p(x,y)}$$

where w and h denote the width and height of the image, respectively.

Value

A list containing:

`statistics` A data frame with the following components:

- `center_x`, `center_y`: The (x, y) coordinates of the COG in pixel coordinates of the input image.
- `center_x_trim`, `center_y_trim`: The COG coordinates relative to the trimmed glyph region, excluding image margins.
- `center_x_std`, `center_y_std`: The standardized COG coordinates ranging from 0 to 1, based on the trimmed region's width and height.
- `margin_left`, `margin_right`, `margin_top`, `margin_bottom`: Margins between the glyph and the image boundary.
- `width_original`, `height_original`: Dimensions of the original image.
- `width_trim`, `height_trim`: Width and height of the trimmed glyph region, excluding margins.
- `area`: The number of pixels in the stroke region (i.e., the total mass used to compute the COG).

`strokes` A data frame of (x, y) coordinates representing the stroke region (i.e., non-white pixels).

`origin` The origin specification used for the returned coordinates.

Examples

```
data(img_A) # load example image from the package
result <- cog_stroke(img_A, origin = "bottomleft")

result$statistics # summary data frame
head(result$strokes) # stroke pixel coordinates
result$origin # image origin specification
```

draw_contour *Visualize Contour-Based Region and Center of Gravity (COG)*

Description

Visualizes the character region enclosed by the convex polygon computed using cog_contour. Optionally overlays crosshair lines at the computed center of gravity (COG) to aid interpretation.

Usage

```
draw_contour(lst, show_cog = TRUE, plot_image = TRUE, cimg = TRUE)
```

Arguments

lst	A list returned by cog_contour, containing a data frame of polygon points and computed statistics.
show_cog	Logical. If TRUE (default), draws horizontal and vertical red lines through the COG to indicate its location.
plot_image	Logical. If TRUE (default), plots the reconstructed image. If FALSE, returns the image object without displaying it.
cimg	Logical. If TRUE (default) and **imager** is available, returns a cimg object. If FALSE, always returns a raster object.

Details

By default (cimg = TRUE), the function returns a cimg object if the **imager** package is available. If **imager** is not installed or if cimg = FALSE, a base R raster object created by as.raster() is returned.

Value

If cimg = TRUE and **imager** is installed, returns a cimg object. Otherwise, returns a raster object (class "raster").

See Also

cog_contour

Examples

```
data(img_A) # load example image from the package
result <- cog_contour(img_A)
draw_contour(result, show_cog = TRUE, cimg = FALSE)
```

`draw_potential` *Visualize Potential-Based Center of Gravity (COG) and Potential Field*

Description

Visualizes the normalized potential field and center of gravity (COG) computed by `cog_potential`. Each pixel's potential is shown as grayscale intensity, where darker pixels indicate higher potential. Optionally overlays crosshair lines at the computed COG to indicate its position.

Usage

```
draw_potential(lst, show_cog = TRUE, plot_image = TRUE, cimg = TRUE)
```

Arguments

`lst`	A list returned by `cog_potential`, containing a data frame of normalized potentials and computed statistics.
`show_cog`	Logical. If `TRUE` (default), draws horizontal and vertical red lines through the COG.
`plot_image`	Logical. If `TRUE` (default), plots the generated image. If `FALSE`, returns the image object without displaying it.
`cimg`	Logical. If `TRUE` (default) and **imager** is available, returns a `cimg` object. If `FALSE`, always returns a raster object.

Details

By default (`cimg = TRUE`), the function returns a `cimg` object if the **imager** package is available. If **imager** is not installed or if `cimg = FALSE`, a base R raster object created by `as.raster()` is returned.

Value

If `cimg = TRUE` and **imager** is installed, returns a `cimg` object. Otherwise, returns a raster object (class `"raster"`).

See Also

`cog_potential`

Examples

```
data(img_A)
result <- cog_potential(img_A)
draw_potential(result, show_cog = TRUE, cimg = FALSE)
```

draw_stroke *Visualize Stroke Region and Center of Gravity (COG)*

Description

Visualizes the stroke region of a character-like binary image using the result from cog_stroke. Optionally overlays crosshair lines at the computed center of gravity (COG) position.

Usage

```
draw_stroke(lst, show_cog = TRUE, plot_image = TRUE, cimg = TRUE)
```

Arguments

lst	A list returned by cog_stroke, containing stroke pixel data and computed statistics.
show_cog	Logical. If TRUE (default), draws horizontal and vertical red lines through the COG to visualize its position.
plot_image	Logical. If TRUE (default), plots the image. If FALSE, returns the image object without plotting.
cimg	Logical. If TRUE (default) and **imager** is available, returns a cimg object. If FALSE, always returns a raster object.

Details

By default (cimg = TRUE), the function returns a cimg object if the **imager** package is available. If **imager** is not installed or if cimg = FALSE, a base R raster object created by as.raster() is returned.

Value

If cimg = TRUE and **imager** is installed, returns a cimg object. Otherwise, returns a raster object (class "raster").

See Also

cog_stroke

Examples

```
data(img_A)
result <- cog_stroke(img_A)
draw_stroke(result, show_cog = TRUE, cimg = FALSE)
```

img_A *Example Image A*

Description

A binary example image representing the letter "A". This image is used for demonstrating center-of-gravity calculations in the coglyphr package.

Usage

```
data(img_A)
```

Format

A cimg object of size 500 × 500 × 1 × 1 (width × height × depth × spectrum).

Source

Created for illustration in the coglyphr package.

img_B *Example Image B*

Description

A binary example image representing the letter "B". This image is used for demonstrating center-of-gravity calculations in the coglyphr package.

Usage

```
data(img_B)
```

Format

A cimg object of size 500 × 500 × 1 × 1 (width × height × depth × spectrum).

Source

Created for illustration in the coglyphr package.

Index

∗ **datasets**
    img_A, 11
    img_B, 11

cimg, *11*
cog_contour, 2, *8*
cog_potential, 4, *9*
cog_stroke, 6, *10*

draw_contour, 8
draw_potential, 9
draw_stroke, 10

img_A, 11
img_B, 11