rePhotosImageAlignment Documentation

Release 1.0

Axel Schaffland

Feb 17, 2017

CONTENTS:

CHAPTER

ONE

AAAP_RE_PHOTO MODULE

Mainscript for image alignment in the re.photos project. Two images which names are given as parameters at program start are loaded and scaled to the same size. In a two stage approach the user aligns the images. In the first stage the user draws rectangles on either of the two images. A probable corner point inside the rectangle meant by the user is automatically selected as well as a corresponding point in the second image. Alternatively the user can draw points directly. With four pointpairs a perspective transform is performed aligning the images roughly. In the second stage the user draws lines instead of point and as-affine-as-possible warping is used to tune the image alignment. After that the images are cropped to the maximal possible size.

aaap_re_photo.**init**()

Processes command line parameters, reads images, lines and points.

Returns

- **src_img** (*ndarray*) – Image which is warped.
- **dst_img** (*ndarray*) – Image to which the src_img is warped.
- **src_lines** (*list*) – List of lines in src_img read from line file. If line file not found or command line parameter *–line_file* not given an empty list is returned.
- **dst_lines** (*list*) – List of lines in dst_img read from line file. If line file not found or command line parameter *–line_file* not given an empty list is returned.
- **src_points** (*list*) – List of points in src_img read from point file. If point file not found or command line parameter *–point_file* not given an empty list is returned.
- **dst_points** (*list*) – List of points in dst_img read from point file. If point file not found or command line parameter *-point_file* not given an empty list is returned.
- **args** (*Namespace*) – Command line arguments.

aaap_re_photo.**ls**(*l*, *sf*)

aaap_re_photo.**main**()

First function to be called. Initializes programm and switches stages.

aaap_re_photo.**onMouse_stage_one**()

Mousecallback function for first stage user input. User input is drawn on resized images. Search for point and corresponding point is done on original sized images. Custom parameters are given as one tuple.

Parameters

- **event** (*int*) – Mouse event.
- **x** (*int*) – X-coordinate of mouse pointer.
- **y** (*int*) – Y-coordinate of mouse pointer.
- **flags** (*int*) – Type of mouse event.

- **img_d** (*ndarray*) – Resized image on which points are drawn.
- **img_d_clean** (*ndarray*) – Resized image to reset img_d if points are removed.
- **img_orig** (*ndarray*) – Unscaled image in which point is searched.
- **scale** (*float*) – Scale of img_d with respect to imig_orig.
- **points** (*list*) – List of points.
- **win_name** (*string*) – Name of window in which image is shown.
- **color** (*tuple*) – Color of drawn points.
- **img2_d** (*ndarray*) – Resized image for corresponding point on which points are drawn.
- **img2_d_clean** (*ndarray*) – Resized image for corresponding point to reset img_d if points are removed.
- **img2_orig** (*ndarray*) – Unscaled image for corresponding point in which corresponding point is searched.
- **scale2** (*float*) – Scale of img2_d with respect to imig2_orig.
- **points2** (*list*) – List of corresponding points.
- **win_name2** (*string*) – Name of window in which image2 is shown.
- **color2** (*tuple*) – Color of corresponding points.

aaap_re_photo.**onMouse_stage_two**()

Mousecallback function for second stage user input. User input is drawn on resized images. Search for line and corresponding line is done on original sized images. Custom parameters are given as one tuple.

Parameters

- **event** (*int*) – Mouse event.
- **x** (*int*) – X-coordinate of mouse pointer.
- **y** (*int*) – Y-coordinate of mouse pointer.
- **flags** (*int*) – Type of mouse event.
- **img_d** (*ndarray*) – Resized image on which lines are drawn.
- **img_d_clean** (*ndarray*) – Resized image to reset img_d if lines are removed.
- **img_orig** (*ndarray*) – Unscaled image in which lines is searched.
- **scale** (*float*) – Scale of img_d with respect to imig_orig.
- **lines** (*list*) – List of lines.
- **win_name** (*string*) – Name of window in which image is shown.
- **color** (*tuple*) – Color of drawn lines.
- **img2_d** (*ndarray*) – Resized image for corresponding lines on which lines are drawn.
- **img2_d_clean** (*ndarray*) – Resized image for corresponding lines to reset img_d if lines are removed.
- **img2_orig** (*ndarray*) – Unscaled image for corresponding line in which corresponding line is searched.
- **scale2** (*float*) – Scale of img2_d with respect to imig2_orig.
- **points2** (*list*) – List of corresponding lines.

- **win_name2** (*string*) – Name of window in which image2 is shown.
- **color2** (*tuple*) – Color of corresponding lines.

aaap_re_photo.**ps**(*p*, *sf*)

aaap_re_photo.**stage_one**(*src_img*, *dst_img*, *src_points*, *dst_points*, *src_lines*, *dst_lines*, *args*)
First stage. User drawn points are used for perspective alignment.

Parameters

- **src_img** (*ndarray*) – Image on which aaap-warping will be later performed.
- **dst_img** (*ndarray*) – Image which is only perspective transformed.
- **src_points** (*list*) – Points in src_img.
- **dst_points** (*list*) – Points in dst_img.
- **src_lines** (*list*) – Lines in src_img.
- **dst_lines** (*list*) – Lines in dst_img
- **args** (*Namespace*) – Parameters.

Returns

- **src_img** (*ndarray*) – Perspective transformed src_img.
- **dst_img** (*ndarray*) – Perspective transformed dst_img.
- **src_points** (*list*) – Perspective transformed points in src_img.
- **dst_points** (*list*) – Perspective transformed points in dst_img.
- **src_lines** (*list*) – Perspective transformed lines in src_img.
- **dst_lines** (*list*) – Perspective transformed lines in dst_img.
- **src_transform_matrix** (*ndarray*) – Perspective transform matrix of src image, lines and points.
- **dst_transform_matrix** (*ndarray*) – Perspective transform matrix of dst image, lines and points.
- **stage_one_success** (*bool*) – True if four point pairs where avaiable to perform perspective transform of src and dst images, lines and points, else False.

aaap_re_photo.**stage_two**(*src_img*, *dst_img*, *src_points*, *dst_points*, *src_lines*, *dst_lines*, *src_transform_matrix*, *dst_transform_matrix*, *stage_one_success*, *args*)
Second stage. User drawn lines are used for aaap-warping.

Parameters

- **src_img** (*ndarray*) – Image on which aaap-warping will be later performed.
- **dst_img** (*ndarray*) – Image to which src_img shall be warped.
- **src_points** (*list*) – Points in src_img.
- **dst_points** (*list*) – Points in dst_img.
- **src_lines** (*list*) – Lines in src_img.
- **dst_lines** (*list*) – Lines in dst_img
- **src_transform_matrix** (*ndarray*) – Perspective transform matrix of src image, lines and points from first stage.

- **dst_transform_matrix** (*ndarray*) – Perspective transform matrix of dst image, lines and points from first stage.
- **stage_one_success** (*bool*) – True if first stage was successful, else False.
- **args** (*Namespace*) – Parameters.

CHAPTER

TWO

IMAGE_AAAP MODULE

Functions for As-Affine-As-Possible Warping as described in 'Generalized As-Similar-As-Possible Warping with Applications in Digital Photography' by Chen and Gotsman.

image_aaap.**bilinear_point_in_quad_mesh**(*pts*, *X*, *P*, *qmSize*)

Express points in a quad mesh as the convex combination of there containing quads, using bilinear weights A = bilinearPointInQuadMesh(pts, X, P, qmSize)

Parameters

- **pts** (*ndarray*) – Points that are to be expressed as bilinear combinations of the quadmesh vertices
- **X** (*ndarray*) – The vertices of the quadmesh
- **P** (*ndarray*) – The connectivy of the quadmesh
- **qmSize** (*tuple*) – Size (rows/columns of quads) of the quadmesh, that is consctructed to cover some image plane.

Returns **Ascr** – A matrix that gives the weights for the points as combinations of the quadmesh vertices, i.e. A*X = pts.

Return type csc_matrix

image_aaap.**build_regular_mesh**(*width*, *height*, *grid_size*)

Creates quadratic meshgrid of given width, height and distance between grid points.

Parameters

- **width** (*int*) – Width of meshgrid.
- **height** (*int*) – Height of meshgrid.
- **grid_size** (*int*) – Distance between grid lines.

Returns

- **grid_points** (*ndarray*) – Array of points of the grid
- **quads** (*ndarray*) – quads spanning the grid
- **m** (*int*) – dimension of the grid

image_aaap.**construct_mesh_energy**(*grid_points*, *quads*, *deform_energy_weights*)

Create quadratic energy matrix for aaap deformation of quad mesh.

Parameters

- **grid_points** (*ndarray*) – Array of points spanning the mesh.
- **quads** (*ndarray*) – Index of grid points yielding quadrangulation of mesh.

- **deform_energy_weights** (*ndarray*) – Weighting affinity of warping. [alpha, beta, 0,0]. See paper for details.

Returns **L** – Sparse matrixs coresponding to aaap quadratic energy.

Return type csc_matrix

image_aaap.**deform_aaap**(*x*, *Asrc*, *pdst*, *L*, *line_constraint_type*)

AAAP/ASAP deform a quadmesh with line constraints

Parameters

- **x** (*ndarray*) – Geometry of the original quadmesh.
- **Asrc** (*csc_matrix*) – Matrix that express lines (sampled points on lines) as linear combinations of x.
- **pdst** (*list*) – Target positions of the lines (sampled points on them), each cell element corresponds to one line.
- **L** (*csc_matrix*) – AAAP/ASAP energy of the quadmesh.
- **line_constraint_type** (*int*) – Constraint type of each line.

Returns **y** – Geometry of the deformed quadmesh.

Return type ndarray

image_aaap.**sample_lines**(*src_lines*, *dst_lines*, *sample_rate*)

Samples points from line pairs.

Parameters

- **src_lines** (*list*) – List of lines in source image.
- **dst_lines** (*list*) – List of lines in destination image.
- **sample_rate** (*float*) – Distance between sampled line points.

Returns

- **p1** (*ndarray*) – List of points in src_img.
- **p2** (*list*) – List of points in src_img.

CHAPTER

THREE

IMAGE_AAAP_MAIN MODULE

Wrapper for As-Affine-As-Possible Warping as described in 'Generalized As-Similar-As-Possible Warping with Applications in Digital Photography' by Chen and Gotsman.

image_aaap_main.**aaap_morph**(*src_img*, *dst_img*, *src_lines*, *dst_lines*, *grid_size=15*, *line_constraint_type=2*, *deform_energy_weights=array([1., 0.01, 0., 0.])*, *n_samples_per_grid=1*, *scale_factor=1*, *show_frame=False*, *draw_grid_f=False*)

Warp src image as affine as possible under given line constraints.

Parameters

- **src_img** (*ndarray*) – Source image which will be warped to match destination image.
- **dst_img** (*ndarray*) – Destination image which is only scaled.
- **src_lines** (*list*) – User drawn lines in the source image.
- **dst_lines** (*list*) – User drawn lines in the destination image.
- **grid_size** (*int*) – Distance between grid lines in pixels. (Default value = 15)
- **line_constraint_type** (*int*) – 0: Fixed discretisation of lines. 1: Flexible discretisations, points can move on line but endpoints are fixed. 2: Flexible discretisation, all points including endpoints can move on line. (Default value = 2)
- **deform_energy_weights** (*ndarray*) – Weighting affinity of warping. [alpha, beta, 0,0]. See paper for details. (Default value = np.array([1,0.0100,0,0])
- **deform_energy_weights** – Weighting affinity of warping. [alpha, beta, 0,0]. See paper for details.
- **n_samples_per_grid** (*int*) – Number of line discretization points per grid block. (Default value = 1)
- **scale_factor** (*int*) – Scaling factor for first image and both line lists. Second image is not scaled since not used for computation. (Default value = 1)
- **show_frame** (*bool*) – True: Draw frame arround cropped area but do not crop. False: Crop images.
- **draw_grid_f** (*bool*) – True: Draw the aaap grid on the return images, False: Well...

Returns

- **src_img_morphed** (*ndarray*) – Warped and cropped source image.
- **dst_img_cropped** (*ndarray*) – Destination image cropped to same size as src_img.
- **src_img_cropped** (*ndarray*) – Source image cropped to evaluate warp.

CHAPTER

FOUR

IMAGE_DRAW_GRID MODULE

Function to draw a grid on an image. Used to show the grid deformation of aaap-morphing.

image_draw_grid.**draw_grid**(*img*, *grid_points*, *quad_indices*)

Draws line grid on given image.

Parameters

- **img** (*ndarray*) – Image on which to draw.
- **grid_points** (*ndarray*) – List of cornerpoints of the grid.
- **quad_indices** (*ndarray*) – List of indices of quads.

CHAPTER

FIVE

IMAGE_GABOR MODULE

Adapation of gabor_threads.py from openCV/samples/python/ Filters an image with a set of gabor filters.

image_gabor.**build_filters**()
Builds collection of gabor filters.

Returns **filters** – List of gabor filters.

Return type list

image_gabor.**get_gabor**(*img*)
Returns gabor filtered image of a given image.

Parameters **img** (*ndarray*) – Image to be filtered.

Returns **img** – The filtered image.

Return type ndarray

image_gabor.**process**(*img*, *filters*)
Filters image by several gabor filters from a list.

Parameters

- **img** (*ndarray*) – To be filtered image.
- **filters** (*list*) – Gabor filters.

Returns **accum** – Gaborfiltered image.

Return type ndarray

image_gabor.**process_threaded**(*img*, *filters*, *threadn=4*)
Starts threated gabor filtering of image.

Parameters

- **img** (*ndarray*) – To be filtered image.
- **filters** (*list*) – Gabor filters.
- **threadn** (*int*) – Number of threads for multiprocessing. (Default value = 4)

Returns **accum** – Gaborfiltered image.

Return type ndarray

CHAPTER

SIX

IMAGE_HELPERS MODULE

Collections of image processing function used throughout the project.

image_helpers.**adaptive_thresh**(*img*)

Thresholds given image adaptive.

Parameters **img** (*ndarray*) – Image to be thresholded.

Returns Thresholded grey image.

Return type ndarray

image_helpers.**do_scale**(*img1*, *img2*, *lines_img1*, *lines_img2*, *points_img1*, *points_img2*, *scale_img1*, *scale_img2*, *scale_factor*)

Scales an image, line and point pair. Uses a scale factor per image and one global factor.

Parameters

- **img1** (*ndarray*) – First image to be scaled.
- **img2** (*ndarray*) – Second image to be scaled.
- **lines_img1** (*list*) – First list of lines to be scaled.
- **lines_img2** (*list*) – Second list of lines to be scaled.
- **points_img1** (*list*) – First list of points to be scaled.
- **points_img2** (*list*) – Second list of points to be scaled.
- **scale_img1** (*float*) – Scale factor for first image/lines/points.
- **scale_img2** (*float*) – Scale factor for second image/lines/points.
- **scale_factor** (*float*) – global scale factor.

Returns

- **img1** (*ndarray*) – Scaled first image.
- **img2** (*ndarray*) – Scaled second image.
- **lines_img1** (*list*) – Scaled first list of lines.
- **lines_img2** (*list*) – Scaled second list of lines.
- **points_img1** (*list*) – Scaled first list of points.
- **points_img2** (*list*) – Scaled second list of points.
- **scale_img1** (*float*) – Scale factor of first image times global scale factor.
- **scale_img2** (*float*) – Scale factor of second image times global scale factor.

image_helpers.**draw_circle**(*img*, *center*, *color=(255, 255, 255)*)
Draws circle on given image.

Parameters

- **img** (*ndarray*) – Image to be drawn on.
- **center** (*tuple*) – Center of circle
- **color** (*tuple*) – Color of the line. If no color given line is white. (Default value = (255,255,255))

image_helpers.**draw_frame**(*img*, *x_min*, *x_max*, *y_min*, *y_max*)
Draws a frame on a given image. Used to display cropping lines

Parameters

- **img** (*ndarray*) – Image on which frame is drawn
- **x_min** (*int*) – X coordinate of smaller point of rectangle
- **y_min** (*int*) – Y coordinate of smaller point of rectangle
- **x_max** (*int*) – X coordinate of bigger point of rectangle
- **y_max** (*int*) – Y coordinate of bigger point of rectangle

image_helpers.**draw_line**(*img*, *start*, *end*, *color=(255, 255, 255)*, *l_number=-1*)
Draws line and line number on given image.

Parameters

- **img** (*ndarray*) – Image to be drawn on.
- **start** (*tuple*) – Startpoint of line.
- **end** (*tuple*) – Endpoint of line.
- **color** (*tuple*) – Color of the line. If no color given line is white. (Default value = (255,255,255))
- **l_number** (*int*) – Linenumber. If no number given only line is drawn. (Default value = -1)

image_helpers.**draw_rectangle**(*img*, *start*, *end*, *color=(255, 255, 255)*)
Draws rectangle on given image.

Parameters

- **img** (*ndarray*) – Image to be drawn on.
- **start** (*tuple*) – Startpoint of rectangle.
- **end** (*tuple*) – Endpoint of rectangle.
- **color** (*tuple*) – Color of the line. If no color given line is white. (Default value = (255,255,255)

image_helpers.**get_crop_idx**(*crop_img*, *scale=400*)
Computes crop indices based on alpha channel. Searches biggest white rectangle in alpha channel.

Parameters

- **crop_img** (*ndarray*) – to be cropped image with alpha channel
- **scale** (*int*) – Downsample image by img size / scale to speed up (Default value = 400)

Returns Cropindices [x_min, y_min, x_max, y_max]

Return type list

image_helpers.**lce**(*img*, *kernel=11*, *amount=0.5*)

Local Contrast Enhancement by unsharp mask. From the value channel of the image in hsv color space a gaussian blured version is subtracted.

Parameters

- **img** (*ndarray*) – BGR-Image which is enhanced
- **kernel** (*int*) – Size of the gaussian kernel. (Default value = 11)
- **amount** (*float*) – Strength of the contrast enhancment. (Default value = 0.5)

Returns **img_bgr** – Contrast enhanced np.float32 BGR-Image, values between 0, 255.

Return type ndarray

image_helpers.**line_intersect**(*a1*, *a2*, *b1*, *b2*)

Compute intersection of two lines. All input points have to be float. Returns startpoint of second line if lines are parallel.

Parameters

- **a1** (*ndarray*) – Startpoint first line.
- **a2** (*ndarray*) – Endpoint first line.
- **b1** (*ndarray*) – Startpoint second line.
- **b2** (*ndarray*) – Endpoint second line.

Returns intersection point.

Return type ndarray

image_helpers.**pint**(*p*)

image_helpers.**scale**(*img1*, *img2*, *lines_img1*, *lines_img2*, *points_img1*, *points_img2*, *scale_factor=1*)

Upscales the smaller image and coresponding lines/points of two given images. If scale factor is given all entities are scaled by this factor. Aspect ratio is preserved, blank space is filled with zeros.

Parameters

- **img1** (*ndarray*) – Image 1.
- **img2** (*ndarray*) – Image 2.
- **lines_img_1** (*list*) – Lines in image 1.
- **lines_img_2** (*list*) – Lines in image 2.
- **points_img_1** (*list*) – Points in image 1.
- **points_img_2** (*list*) – Points in image 2.
- **scale_factor** (*int*) – Scaling factor for image/line/point pair. (Default value = 1)

Returns

- **img1** (*ndarray*) – If img1 is bigger returns img1 else scaled img1.
- **img2** (*ndarray*) – If img2 is bigger returns img2 else scaled img2.
- **lines_img_1** (*list*) – Lines in image 1, scaled if img1 is scaled.
- **lines_img_2** (*list*) – Lines in image 2, scaled if img2 is scaled.
- **points_img_1** (*list*) – Points in image 1, scaled if img1 is scaled.

- **points_img_2** (*list*) – Points in image 2, scaled if img2 is scaled.
- **scale_factor_img1** (*float*) – Scale factor by which first image/lines/points were scaled.
- **scale_factor_img1** (*float*) – Scale factor by which second image/lines/points were scaled.
- **x_max** (*int*) – After x_max one image is padded with zeros in x direction.
- **y_max** (*int*) – After y_max one image is padded with zeros in y direction.

image_helpers.**scale_image_lines_points**(*img*, *lines*, *points*, *scale_factor*)
Scales an image, points and lines in this image by a given scale factor.

Parameters

- **img** (*ndarray*) – To be scaled image.
- **lines** (*list*) – To be scaled lines.
- **points** (*list*) – To be scaled points.
- **scale_factor** (*int*) – Factor by which image, lines and points are scaled.

Returns

- **img** (*ndarray*) – Scaled image.
- **lines** (*list*) – Scaled lines.
- **points** (*list*) – Scaled points.

image_helpers.**set_verbose**(*verbose*)
Sets the global verbosity function.

Parameters **verbose** (*bool*) – If true verbose output, false no verbose output.

image_helpers.**show_image**(*img*, *name='img'*, *x=1000*, *y=1000*)
Resizes image and displays it in window with given name. Constraints both sides of image by given length constraints.

Parameters

- **img** (*ndarray*) – Image to be displayed.
- **name** (*string*) – Name of openCV window in which image is displayed. (Default value = 'img')
- **x** (*int*) – Max size of image in x direction. (Default value = 1000)
- **y** (*int*) – Max size of image in y direction. (Default value = 1000)

Returns

- **img_d** (*ndarray*) – Scaled image.
- **scale** (*float*) – Scale by which the image was scaled.

image_helpers.**statistic_canny**(*img*, *sigma=0.33*)
Edge detection on color image depending on image properties.

Parameters

- **img** (*ndarray*) – Image on which to detect edges.
- **sigma** (*float*) – Standard deviation. (Default value = 0.33)

Returns Edge image.

Return type ndarray

image_helpers.**unsharp_mask**(*img*, *sigma=1*, *amount=0.8*)
Sharpends given image via unsharp mask.

Parameters

- **img** (*ndarray*) – Image to be sharpened.
- **sigma** (*float*) – Sigma of Gaussian kernel for bluring. (Default value = 1)
- **amount** (*float*) – Amount of sharpening. (Default value = 0.8)

Returns **img** – Sharpened Image.

Return type ndarray

image_helpers.**vprint**(**a*, ***k*)

image_helpers.**weighted_average_point**(*point1*, *point2*, *alpha*)
Return the average point between two points weighted by alpha.

Parameters

- **point1** (*tuple*) – First point multiplied by 1 - alpha
- **point2** (*tuple*) – Second point multiplied by alpha
- **alpha** (*float*) – The weight.

Returns Weighted point

Return type tuple

CHAPTER

SEVEN

IMAGE_IO MODULE

Writes and saves point and line lists from and to json files

image_io.**read_lines**(*filename*)
Reads lines from files.

Parameters **filename** (*string*) – Name of file.

Returns

- **src_lines** (*list*) – List of lines in src image. Empty list if unable to open file.
- **dst_lines** (*list*) – List of lines in dst image. Empty list if unable to open file.

image_io.**read_points**(*filename*)
Reads points from files.

Parameters **filename** (*string*) – Name of file.

Returns

- **src_points** (*list*) – List of points in src image. Empty list if unable to open file.
- **dst_points** (*list*) – List of points in dst image. Empty list if unable to open file.

image_io.**write_lines**(*src_lines*, *dst_lines*, *filename*)
Writes linelists to file.

Parameters

- **src_lines** (*list*) – List of lines in src image.
- **dst_lines** (*list*) – List of lines in dst image.
- **filename** (*string*) – Name of file.

image_io.**write_points**(*src_points*, *dst_points*, *filename*)
Writes pointlists to file.

Parameters

- **src_points** (*list*) – List of points in src image.
- **dst_points** (*list*) – List of points in dst image.
- **filename** (*string*) – Name of file.

CHAPTER

EIGHT

IMAGE_LINES MODULE

Functions to find interesting lines near user drawn lines as well as corresponding lines.

image_lines.**center_of_line**(*p1*, *p2*)

image_lines.**get_corresponding_line**(*img1*, *img2*, *line1*, *psd=15*, *max_lines_to_check=60*, *template_size=200*)

Return a corresponding line in a second image given a line in a first image. Find max_lines_to_check lines in patch arround line1 in img2. Compare correspondence of found lines by template matching. Transform template in img2 such that found line in img2 and given line in img1 allign. Compute sum of squared differences and weight templates/lines.

Parameters

- **img1** (*ndarray*) – Destination image in which line is already found
- **img2** (*ndarray*) – Source image in which the corresponding line is searched
- **line1** (*list*) – Line in img1 for which a corresponding line needs to be found
- **psd** (*int*) – Patchsizedivisor (Default value = 15)
- **max_lines_to_check** (*int*) – Number of lines which correspondence to given line is evaluated. (Default value = 60)
- **template_size** (*int*) – Size of template with which correspondence of line is determined. (Default value = 200)

Returns The correspoding line.

Return type list

image_lines.**get_line**(*p1*, *p2*, *img*, *psd=70*)

Search in image for nearest most similar line of a given line. Search in H, S, and V for line in area around given line. Select line with similar rotation and many supporting edge pixels.

Parameters

- **p1** (*ndarray*) – Start point of user drawn line.
- **p2** (*ndarray*) – End point of user drawn line.
- **img** (*ndarray*) – Image in which nearest line in searched.
- **psd** (*iny*) – Patch size divisor: Area around line in which similar line is searched. Higher value = Smaller area (Default value = 70)

Returns **[p1,p2]** – List of two points of computed most similar line

Return type list

image_lines.**get_patch**(*img*, *p1*, *p2*, *psd=70*)

Return image patch arround a given line. Return horizontal patch, a rectangular mask with same slope as the line and offset between min point of patch and image

Parameters

- **img** (*ndarray*) – Image from which the patch is generated
- **p1** (*tuple*) – First point of the given line
- **p2** (*tuple*) – Second point of the given line
- **psd** (*int*) – Patchsizedivisor (Default value = 70) determining the patch size i.e. the size arround the given line.

Returns

- **patch** (*ndarray*) – Horizontal patch
- **mask** (*ndarray*) – Zero matrix of size patch with one rectangle determining the actual patch arround the line.
- **offset** (*ndarray*) – Offset between min point of patch and min point of image.

image_lines.**get_theta**(*p1*, *p2*)

Computes gradient angle of line.

Parameters

- **p1** (*ndarray*) – First point of line.
- **p2** (*ndarray*) – Second point of line.

Returns **theta** – Gradient angle.

Return type float64

image_lines.**get_transformed_patch**(*patch*, *p11*, *p12*, *p21*, *p22*)

Rotates and translates given patch such that second line is mapped to a first horizontal line.

Parameters

- **patch** (*ndarray*) – Patch to be translated and rotated.
- **p11** (*tuple*) – First point of first line.
- **p12** (*tuple*) – Second point of first line.
- **p21** (*tuple*) – First point of second line.
- **p22** (*tuple*) – Second point of second line.

Returns **patch** – Transformed patch.

Return type ndarray

image_lines.**get_weighted_lines**(*img*, *mask*, *p1_o*, *p2_o*, *range_theta=0.2617993877991494*)

Returns sortet list of lines in proximity of a given line. Lines are sorted by line quality.

Parameters

- **img** (*ndarray*) – Image in which lines are searched.
- **mask** (*ndarray*) – Mask applied to image to limit search area.
- **p1_o** (*ndarray*) – First point of original line.
- **p2_o** (*ndarray*) – Second point of original line.

- **range_theta** (*float*) – Limits how far new lines are rotated from original line. (Default value = 1/12.*np.pi)

Returns **best_lines** – Array of lines in descending order.

Return type ndarray

image_lines.**lim_line_length**(*p1_h*, *p2_h*, *p1_o*, *p2_o*)

Limits length of line h to linesegment o. Computes a line segment of line h found by hough transform to the length of user drawn line segment o by computing the normals at start and end point of user drawn line segment and their intersection with hough line.

Parameters

- **p1_h** (*ndarray*) – First point on hough line.
- **p2_h** (*ndarray*) – Second point on hough line.
- **p1_o** (*ndarray*) – Startpoint of user drawn line segment.
- **p2_o** (*ndarray*) – Endpoint of user drawn line segment.

Returns

- **z1** (*ndarray*) – Startpoint of line segment of hough line.
- **z2** (*ndarray*) – Endpoint of line segment of hough line.

image_lines.**line_detect**(*img*, *mask=None*, *sigma=0.33*, *magic_n=10*, *min_theta=0*, *max_theta=3.141592653589793*)

Line detection on given image via Canny and Hough. Lines are limited to have an angle between min_theta and max_theta.

Parameters

- **img** (*ndarray*) – Image in which to detect lines.
- **mask** (*ndarray*) – Matrix of image size. Detection of lines only where mask is nonzero. (Default value = None)
- **sigma** (*float*) – Sigma for adpative Canny edge detection. (Default value = 0.33)
- **magic_n** (*int*) – Offset for adaptive Canny edge detection. (Default value = 10)
- **min_theta** (*float64*) – Minimum line angle for hough line detection. (Default value = 0)
- **max_theta** (*float64*) – Maximum line angle for hough line detection. (Default value = np.pi)

Returns **lines** – Detected lines in Hesse normal form.

Return type ndarray

image_lines.**weight_lines**(*patch_p*, *lines*, *p1_o*, *p2_o*, *max_delta=0.05*, *number_of_lines=10*)

Weights hough lines by similarity to edges. Generates line segments with length according to user drawn lines and compares segments with edges in edge image. Select best matching line. Only lines with similar orientation to user drawn line are regarded. Only best ten lines fullfilling above criterium are considered.

Parameters

- **patch_p** (*ndarray*) – Edgeimage.
- **lines** (*ndarray*) – List of lines.
- **p1_o** (*ndarray*) – Startpoint of user drawn line.
- **p2_o** (*ndarray*) – Endpoint of user drawn line.

- **max_delta** (*float*) – max angle between given line and lines to be weighted. (Default value = 0.05)
- **number_of_lines** (*int*) – number of lines generated by first step (Default value = 10)

Returns

- **line_segs** (*ndarray*) – best lines.
- **weights** (*ndarray*) – weights of the best line segments.

CHAPTER

NINE

IMAGE_MORPHING MODULE

Functions to moprh an image with src and dst quad mesh. Corners of the two quad meshes are mapped to each other, area inbetween is interpolated witch perspective transform.

image_morphing.**morph**(*src_img*, *points_old*, *points_new*, *quads*, *grid_size*, *processes=1*)

Returns morphed image given points of old and new grid and quadindices. Sourceimage is divided in stripes which are morphed parallel.

Parameters

- **src_img** (*ndarray*) – The image which will be morphed.
- **points_old** (*ndarray*) – Positions of grid points in src_img.
- **points_new** (*ndarray*) – Positions to where the old points are moved.
- **quads** (*ndarray*) – List of quad indices.
- **grid_size** (*int*) – Distance between grid lines.
- **processes** (*int*) – Number of multiprocessing.Processes which are spawend. (Default value = 1)

Returns **img_morh** – The morphed src_img.

Return type ndarray

image_morphing.**morph_process**(*src_img*, *s_x_min*, *shared_dst*, *dst_shape*, *points_new*, *points_old*, *quads*)

Multiprocessing process to morph image stripes.

Parameters

- **src_img** (*ndarray*) – Stripe of the original image which is to be morphed.
- **s_x_min** (*int64*) – Offset of image stripe. Used to determine where to add stripe in shared_dst.
- **shared_dst** (*SynchronizedArray*) – Shared destination image in which morphed image is saved.
- **dst_shape** (*tuple*) – Size of the destination image.
- **points_new** (*ndarray*) – Corner locations of the quad grid in the destination image.
- **points_old** (*ndarray*) – Corner locations of the quad grid in the source image.
- **quads** (*ndarray*) – Indices of corners of the quads constituiting the mesh.

image_morphing.**to_numpy_array**(*mp_arr*)

Create numpy array from multiprocessing array.

Parameters **mp_arr** (*SynchronizedArray*) – Multiprocessing input array.

Returns Numpy output array.

Return type ndarray

CHAPTER

TEN

IMAGE_PERSPECTIVE_ALIGNMENT MODULE

Functions to transform images, list of lines and list of points to match another image with a perspective transform matrix. The later is computed from two lists of points, which should be matched.

image_perspective_alignment.**perspective_align**(*img_1*, *img_2*, *points_img_1*, *points_img_2*, *lines_img_1*, *lines_img_2*, *alpha=None*)

Aligns the two images with the best matching perspective transform given the two point lists. Points and lines in their list are transformed as well.

Parameters

- **img_1** (*ndarray*) – Image 1
- **img_2** (*ndarray*) – Image 2
- **points_img_1** (*list*) – marked points in image 1
- **points_img_2** (*list*) – coresponding points in image 2
- **lines_img_1** (*list*) – marked lines in image 1
- **lines_img_2** (*list*) – coresponding lins in image 2
- **alpha** (*float*) – 0 = align img_2 to img_1, 1 = align img_1 to img_2, 0.5 align img_1 and img_2 to mean and points and lines acordingly. If alpha is None the smaller image is transformed to bigger one. (Default value = None)

Returns

- **img_1** (*ndarray*) – perspective transformed img_1
- **img_2** (*ndarray*) – perspective transformed img_2
- **points_img_1** (*list*) – perspective transformed points_img_1
- **points_img_2** (*list*) – perspective transformed points_img_2
- **lines_img_1** (*list*) – perspective transformed lines_img_1
- **lines_img_2** (*list*) – perspective transformed lines_img_2

image_perspective_alignment.**transform_lines**(*lines*, *transform_matrix*)

Transforms a list of lines given a transformation matrix.

Parameters

- **lines** (*list*) – The list of lines given as lists.
- **transform_matrix** (*ndarray*) – The transformation matrix.

Returns **lines** – Transformed lines in float32

Return type list

image_perspective_alignment.**transform_points**(*points*, *transform_matrix*)
Transforms a list of points given a transformation matrix.

Parameters

- **points** (*list*) – the list of points given as touples
- **transform_matrix** (*ndarray*) – the transformation matrix

Returns **point_array** – transformed points in int32

Return type list

CHAPTER

ELEVEN

IMAGE_SAC MODULE

Functions to find interesting points near user drawn points as well as corresponding points.

image_sac.**getCorespondingPoint**(*img1*, *img2*, *point*, *template_size_s=101*)
Search for coresponding point on second image given a point in first image.

First possible matching corners are searched, then template matching at the possible corner locations is done.

Parameters

- **img1** (*ndarray*) – Image in which point was found.
- **img2** (*ndarray*) – Image in which corresponding point is searched.
- **point** (*ndarray*) – Location of found point in img1.
- **template_size_1** (*int*) – Size of scaled template for template matching.

Returns **point** – The corresponding point.

Return type ndarray

image_sac.**getPFromRectangleACorespondingP**(*img1*, *img2*, *point1*, *point2*)
Wrapper for getPointFromRectangle and getCorespondingPoint.

Parameters

- **img1** (*ndarray*) – Image in which point is searched.
- **img2** (*ndarray*) – Image in which corresponding point is searched.
- **point1** (*tuple*) – First corner of rectangle in which point is searched.
- **point2** (*tuple*) – Second corner of rectangle in which point is searched.

Returns

- **returnPoint1** (*tuple*) – Found point.
- **returnPoint2** (*tuple*) – Corresponding point.

image_sac.**getPointFromPoint**(*img*, *point*)
Returns most fitting point near a given point in an image.

Parameters

- **img** (*ndarray*) – Image in which point is searched.
- **point** (*tuple*) – Point near which the best point/corner is searched.

Returns **point** – Best point.

Return type tuple

image_sac.**getPointFromRectangle**(*img1*, *point1*, *point2*)
Computes point of interest in a rectangle defined by two given points.

The best corner is returned weighted by the distance to the center of the rectangle.

Parameters

- **img1** (*ndarray*) – Image in which corner point of interest is searched.
- **point1** (*tuple*) – First corner of rectangle.
- **point2** (*tuple*) – Opposite corner of first corner of rectangle.

Returns **returnPoint** – The best point of interest.

Return type tuple

image_sac.**get_and_pre_patch**(*img*, *point*, *size_half*)
Get a patch from an image and preprocess it.

Algorithm returns biggest possible template limited by the image size.

Parameters

- **img** (*ndarray*) – Image from which to get the patch.
- **point** (*ndarray*) – Centerpoint of the patch.
- **size_half** (*int*) – Half of the patchsize.

Returns

- **subimage** (*ndarray*) – Template as copy.
- *ndarray* – Offset of lowest corner of patch with respect to image origin.
- *ndarray* – Differences between theorethical template size and practical template size limited by size of the image.

CHAPTER

TWELVE

LER PACKAGE

12.1 Submodules

12.2 ler.image_ler module

Find height, width of the largest rectangle containing all 0's in the matrix.

The algorithm for *max_size()* is suggested by @j_random_hacker [1]. The algorithm for *max_rectangle_size()* is from [2]. The Python implementation [3] is dual licensed under CC BY-SA 3.0 and ISC license.

[1]: http://stackoverflow.com/questions/2478447/find-largest-rectangle-containing-only-zeros-in-an-nn-binary-matrix#comment5169734_4671342

[2]: http://blog.csdn.net/arbuckle/archive/2006/05/06/710988.aspx

[3]: http://stackoverflow.com/a/4671342

Changed by axschaffland@uos.de to return not max_size but position of largest empty rectangle.

class `ler.image_ler.`**`Info`**(*start*, *height*)
Bases: `tuple`

`height`
Alias for field number 1

`start`
Alias for field number 0

class `ler.image_ler.`**`TestCase`**(*methodName='runTest'*)
Bases: `unittest.case.TestCase`

`test`()

`ler.image_ler.`**`area`**(*size*)

ler.image_ler.**max_rectangle_size**(*histogram*)
Find height, width of the largest rectangle that fits entirely under the histogram.

```
>>> f = max_rectangle_size
>>> f([5,3,1])
(3, 2)
>>> f([1,3,5])
(3, 2)
>>> f([3,1,5])
(5, 1)
>>> f([4,8,3,2,0])
(3, 3)
>>> f([4,8,3,1,1,0])
(3, 3)
>>> f([1,2,1])
(1, 3)
```

Algorithm is "Linear search using a stack of incomplete subproblems" [1].

[1]: http://blog.csdn.net/arbuckle/archive/2006/05/06/710988.aspx

ler.image_ler.**max_size**(*mat*, *value=0*)
Find height, width of the largest rectangle containing all *value*'s.

For each row solve "Largest Rectangle in a Histrogram" problem [1]:

[1]: http://blog.csdn.net/arbuckle/archive/2006/05/06/710988.aspx

12.3 Module contents

CHAPTER

THIRTEEN

SPQR PACKAGE

13.1 Submodules

13.2 spqr.image_spqr module

spqr.image_spqr.**main**()

spqr.image_spqr.**qr_solve**(*A_data*, *A_row*, *A_col*, *A_nnz*, *A_m*, *A_n*, *b_data*)
Python wrapper to qr_solve.<

13.3 Module contents

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

l

s