Image processing for Bacticam

May 2020

1 Code Documentation

1.1 Modes

1. **debug_mode**
   If True, outputs from each operation are written to "Debug/ " and its corresponding folder. False by default.

2. **validation_mode**
   If True, additional validation is used when finding the contour. Note: Only needed in some extreme cases. False by default.

3. **color_balance**
   If True, the final output is processed with color balance before writing. True by default.

1.2 Image Balance

1. **def imageBalance(img, balance_level=1)**

   (a) **def autoBrigthnessContrast(img, clip_hist_percent)**
   Returns an image with balanced brightness and contrast of an input (img), based on given percentage value.

   (b) **def simplestColorBalance(img, percent)**
   Returns an image with balanced colors of an input (img), with given intensity percentage.

1.3 Identify Agar Plate

1. **def identifyPlate(img)**
   Returns the ellipse values, defining the outer contour of the agar plate. The ellipse is defined as ((centerPointX, centerPointY), (width, height), view angle).

1.4 Identify Compartment Edges

1. **def findCompartmentEdges(img, ellipse)**

   (a) **def skeletonizeLines(img, ellipse)**
   Returns a binary image of the compartment edges skeleton.

   (b) **def houghLinesP(skel, org)**
   Approximates lines of the skeleton given by *skeletonizedLines()*. Returns a line representation of the compartment edges as an array $[[(line1\_x1, line1\_y1), (line1\_x2, line1\_y2)], [(line2\_x1, line2\_y1), (line2\_x2, line2\_y2)]]$, as well as the intersection point between the two lines as (x, y).

1.5 Identify Rotation

1. **def IdentifyRotation(img, ellipse, lines, center_point)**

   (a) **def sortLines(lines)**
   Return lines in sorted order, with line1 being the one closest to origo.

   (b) **def ellipse_polyline(ellipse, n=5000)**
   Converts the ellipse from *identifyPlate()*, to an array of n points.

   (c) **def ellipseLineIntersection(ellipse_points, line1, line2)**
   Returns an array of the intersection points between the lines and the ellipse.

   (d) **def findCompartment(img)**
   Returns an image of only the red agar plate compartment, using color space segmentation based on HSV-values. HSV-values are currently set to segment red pixels.

   (e) **def identifyOrientation(segmentation, intersection_pt)**
   Returns the intersection points from *ellipseLineIntersection()* ordered based on the orientation of the red compartment. The function checks pixels on lines between each intersection points clockwise to calculate the path with the highest red mean value.

   (f) **def sortEllipsePoints(ellipse_points)**
   Returns ellipse points in clockwise order, from a static start point.

   (g) **def sortArc(ellipse, ellipse_points, landmark_pts)**
   Returns ellipse points in a clockwise order based on the agar plates' rotation.

   (h) **def sortPoints(landmark_pts, sorted_ellipse_pts, center_point)**
   Returns an array of all key points found sorted to match a reference.

      i. **def sortLinePoints(line)**
      Returns points of both lines of equal n length.

   (i) **def refPoints()**
   Returns the reference key points needed to match the input image in the *imageRegistration()*.

1.6 Image Registration

1. **def imageRegistration(file_set, ellipse, key_points, dst)** Returns the final outputs using Image Registration on the entire file set (i.e., all images of the same agar plate id). Each image is masked before processed through *unwarp()*. The homography for each image is calculated to match the reference image based on the input key_points.

   (a) **def unwarp(img, src, dst)** Returns the warped image and the homogpraghy. The homography is calculated to match the warp the input image.

1.7 Other

1. **def imageResize(img, height=600, width=600)**
   Resizes an input (img) to given size (height, width).

2. **def fillImage(height, width, color)**
   Returns a black image of given size(heigh, width). Used mainly in debug_mode.

3. **def map(x, in_min, in_max, out_min, out_max)**
   Maps brigthness and contrast values in *autoBrigthnessContrast()*

4. **def autoCanny(img, sigma)**
   Approximates suitable threshold values for Canny Edge Detection based on the conditions (color, brightness, contrast etc.) of the input (img). Returns a binary image. Used in *identifyPlate()*

5. **def getMean(parts)**
   Splits a line in exact n number of points.

6. **def lineIntersection(p1, p2)**
   Calculates the intersection points between two lines.

7. **def maskPlate(out, None, mask)**
   With the Region of interest defined in *identifyPlate(img)*, A binary mask is applied to remove any background noise.

1.8 Functions used in validation mode

1. **def identifyPlateWithValidation(img)**
   Returns an ellipse as in *identifyPlate()*, but with an additional validation when finding the contour. If a contour is too far away from it circular representation derived from *houghCircle()*, brightness and contrast are automatically adjusted. The process is repeated until a valid contour is found.

   (a) **def houghCircle(input_img)**
   Returns a circular representation of the outer contour of the agar plate.

   (b) **def apply_brightness_contrast(img, brightness=255, contrast=127)**
   Returns an image with adjusted brightness and contrast based on input parameters.

      i. **def apply_mask(matrix, mask, fill_value)**

      ii. **def apply_threshold(matrix, low_value, high_value)**