Sub-region Detector Algo Summary

Scott White
August 29, 2016

Input Parameters

```
def find_regions(
        src_img,
        target_img,
        bg_colors=None,
        pre_erode=0,
        dilate=2,
        min_area=0.5,
        max_area=2.0
):
```

src_img	3-D NumPy array of pixels in HSV (source image)
target_img	3-D NumPy array of pixels in HSV (target image)
bg_colors	List of color names to use for background colors. If None, the dominant color in the source image will be used
pre_erode	# of erosion iterations performed on masked image prior to any dilation iterations
dilate	# of dilation iterations performed on masked image
min_area	minimum area cutoff percentage (compared to target image) for returning matching sub-regions
max_area	maximum area cutoff percentage (compared to target image) for returning matching sub-regions

HSV Color Ranges

- In OpenCV:
  - Hue: 0 -> 180
  - Saturation: 0 -> 255
  - Value: 0 -> 255
  - Red wraps around hue range

*** Note: OpenCV will read 16-bit / channel RGB TIFF images, but will downsample each color to 8-bit. I guess technically, converting to HSV will result in a slight further loss since the hue range is not fully 8-bit.

- isd_lib defines 9 color ranges:
  - 3 major colors (spanning 40 hue values)
    - Red
    - Green
    - Blue
  - 3 minor colors (spanning 20 hue values)
    - Yellow
    - Cyan
    - Violet
  - 3 monochrome colors
    - White
    - Gray
    - Black

Summary of find_regions

Determine candidates

- Get bg_colors from src_img (if bg_colors not specified)
- Get color profile from target_img (# of px in each color range)
- Find feature colors from color profile ***
- Create masks (src & target) using feature colors
- Erode masks, followed by dilation, then fill holes ***
- Filter target mask by largest blob

Filter candidates

- Get area of largest blob in target mask
- Filter blobs in src_img by size given min & max area %

Summary of find_regions

What's a feature color?

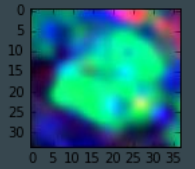
A feature color is any color within the 9 defined HSV color ranges found in the target image having > 10% of the total non-background area

The cutoff is termed 'prevalence' in the function get_common_colors() and can be changed (default is 10%...actually 0.1)

Really, the non-feature colors are just an extension of the set of background colors

Given a target & bg colors:

```
bg_colors = ['black', 'blue']
```



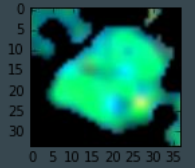
The color profile is:

```
red:      48
yellow:    0
green:   320
cyan:    378                    Total non-bg area: 772
blue:    443 (bg)
violet:   22                    Feature colors > 77px
black:    43 (bg)
gray:      4
white:     0
```

The feature colors are:

```
['green', 'cyan']
```

And the resulting target mask:



Summary of find_regions

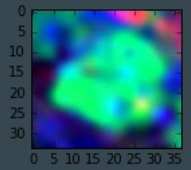
Erode, dilate, fill holes, find largest

The erosion is done prior to dilation to remove “bridges” in simple structures near each other

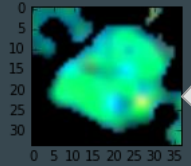
Dilation iterations help merge components in complex structures or regain lost eroded areas

Filling holes just ensures the blob is a solid object

Find largest blob to isolate the object of interest...hopefully

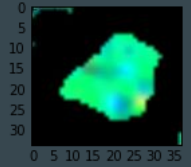


Original target

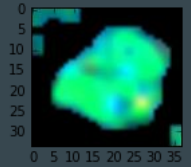


Extract feature colors

...bridge to adjacent cell

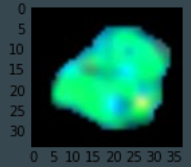


2 erosions remove bridge



2 dilations restore size

...but still have unwanted stuff



Find largest blob

Summary of find_regions

Really a 2 stage process:

1. Find candidates
    - Intentionally liberal, finds anything w/ color profile of target
    - Seems to work well, and nice that it avoids searching through image space
    - Does rely on a “recipe” of erosions, dilations, etc.
    - Recipes could be just a few presets determined by some details of the object of interest...proximity, complexity, etc.
2. Filter candidates
    - Currently based on area threshold of binary mask, and works OK-ish
    - This stage has the most potential for improvement
    - Could/should be replaced by more elegant method(s) including some type of machine learning

Ideas for improvements...

- Build an inventory of features & their properties
  - Must take into account development stage b/c of variation
  - Once assembled, straight-forward to create OOP model
- Use feature inventory to make a few stage 1 presets
  - Do feature objects tend to “touch”...need erosion
  - Is feature object simple & homogeneous (e.g. a single cell)...need less dilation
  - Or a complex object with different components (e.g. proximal acinar tubule)...needs more dilate iterations
- Replace 2nd stage with something more elegant
  - Utilize color and/or shape information
  - Possibly some ML algorithm using those plus feature inventory properties

Feature Inventory

Structure properties for IHC images:

- Name of structure
- Species
- Development stage
- Mean % of pixels for each protein *
- SD of pixels for each protein *
- Mean # of pixels for entire structure
- SD of total structure pixels
- Some proximity metric of how close features are from each other
- Possibly some metric of homogeneity of structure components

* These need to be based on properly segmented contours with backgrounds removed & not just bounding boxes for regions