Preprocessing for Automatic Pattern Identification in Wildlife: Removing Glare

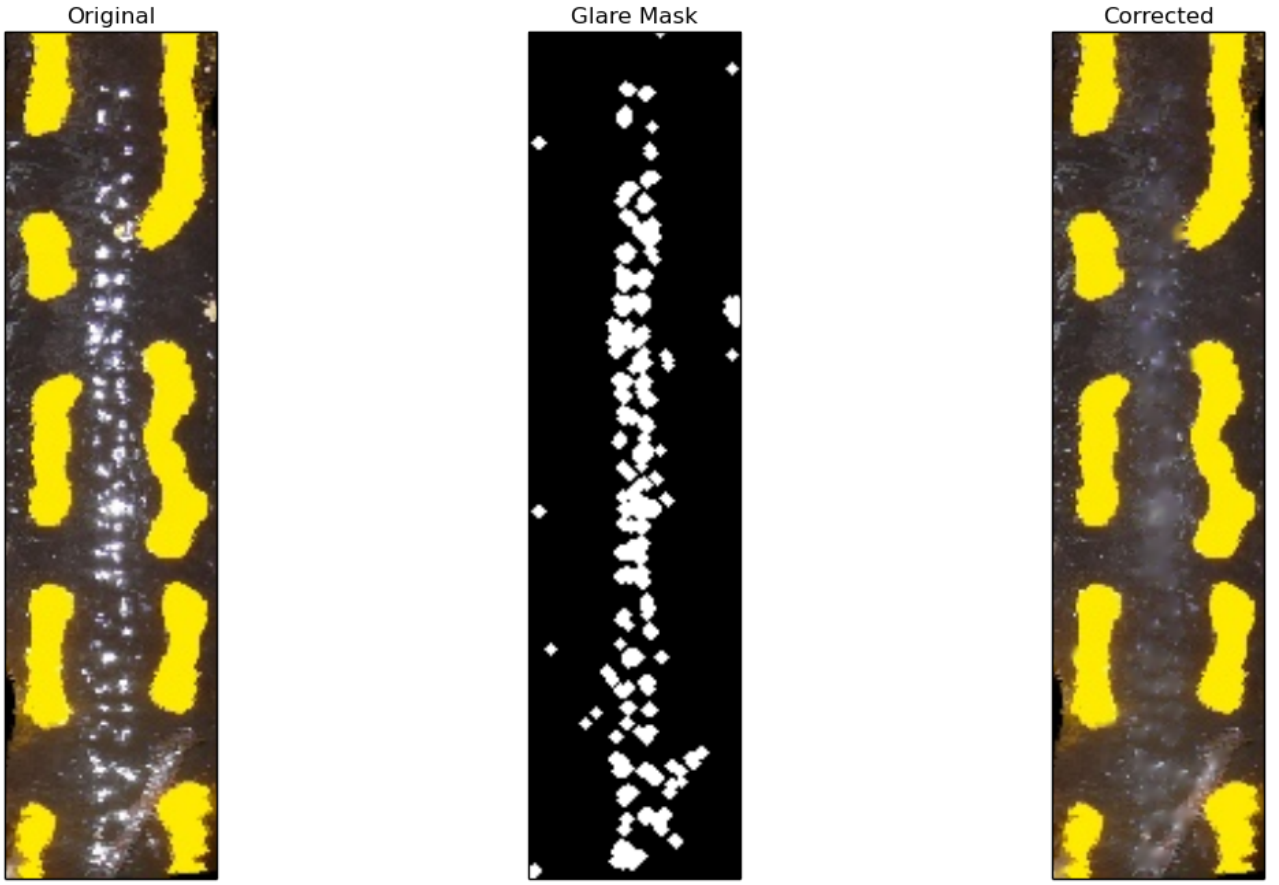
A common problem for automatic photo-identification (http://www.amphident.de/en/blog/computer-assisted-photo-identification-an-overview.html) of individual animals are specular reflections on the photograps. In particular, species that live in wet areas, like amphibians, or that have a strongly reflecting surface, like fish, snakes or e.g. beetles can produce a significant amount of glare on the photos. At the stage of photo shooting, a polarizing filter (https://en.wikipedia.org/wiki/Polarizing_filter_(photography)) in front of the camera lens can efficiently reduce the reflections. However, once the photo is taken and glare is contained, digital post-processing can also efficiently remove glare in the photos.

In this demonstration, we will remove the specular reflections and glare from a pattern of *Salamandra salamandra*. The basic procedure consists of 3 steps:

1. Decompose the original image into a color, saturation and brightness component.
2. Find particularly bright areas in the image.
3. Inpaint these ares with the surrounding pixels.

Below, an exemplary original and the corrected image are shown. There is a significant amount of glare on the original image, which was effectively removed in the corrected image.

```
subplot(1,3,1); _imshow(image_in, "Original");
subplot(1,3,2); _imshow(glare, "Glare Mask");
subplot(1,3,3); _imshow(corrected, "Corrected");
```



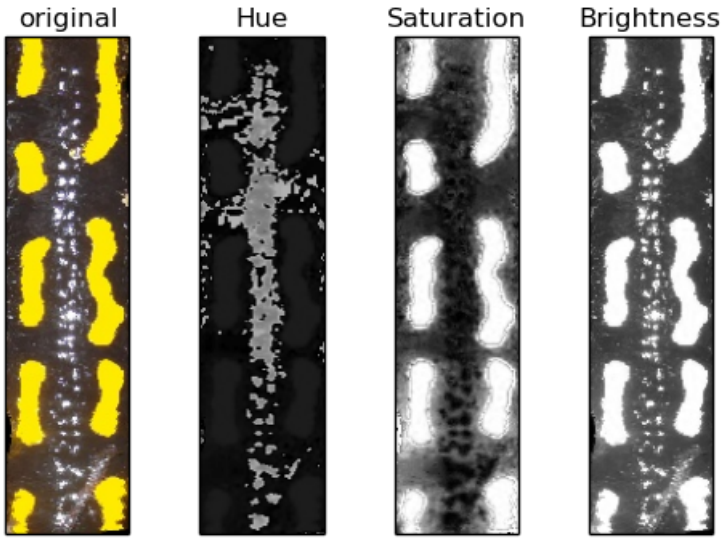
These are some standard imports and a function to easily show images:

```
%matplotlib inline

from pylab import *
import cv2
import os, sys
```

```
def _imshow(I, theTitle=None):
    if len(I.shape) == 3:
        imshow(I)
    else:
        if I.min() == 0 and I.max() == 1:
            imshow(I, cmap="gray")
        else:
            imshow(I, cmap="gray",vmin=0,vmax=255)
    xticks([])
    yticks([])
    if theTitle:
        title(theTitle);
```

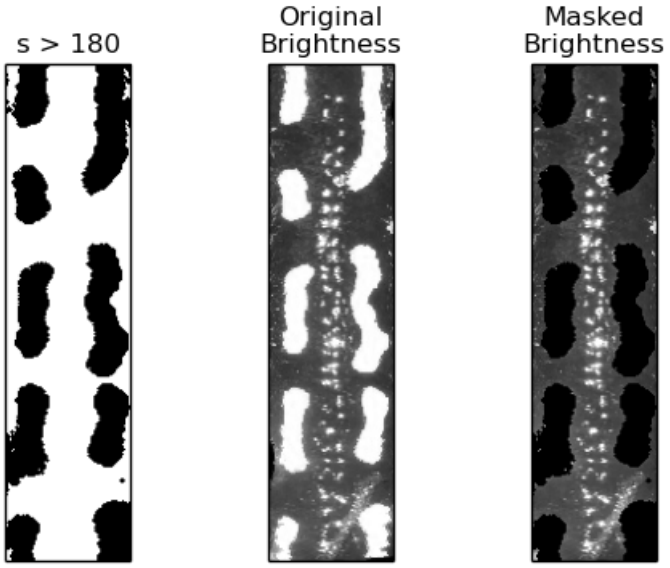
The image is loaded and converted into the HSV color space. The HSV color space describes an image by its hue (H), saturation (S) and brightness (V) component.

```
image_in = cv2.cvtColor(cv2.imread("glare1.jpg"), cv2.COLOR_BGR2RGB); # Load the glared image
h, s, v = cv2.split(cv2.cvtColor(image_in, cv2.COLOR_RGB2HSV)) # split into HSV components

subplot(1,4,1); _imshow(image_in, "original")
subplot(1,4,2); _imshow(h, "Hue")
subplot(1,4,3); _imshow(s, "Saturation")
subplot(1,4,4); _imshow(v, "Brightness")
```



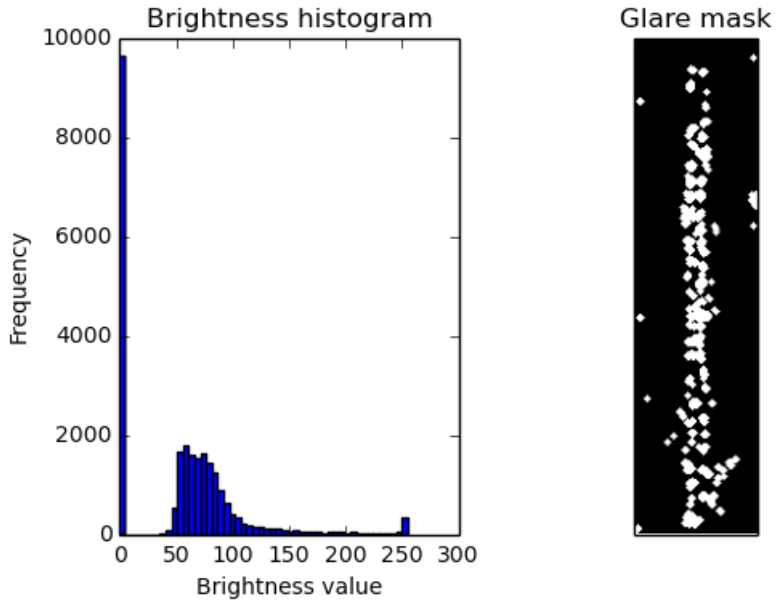
The yellow areas of the pattern show a strong color saturation whereas the center areas are not very saturated. This can be used to derive a general rule for pixels that are subject to specular reflections: An image can only contain glare, when its color is not saturated and it has a high brightness. Since light reflections are white, any pixel containing glare must have no saturation (since white has no color or saturation). Accordingly, we can first filter out the areas that have a strong saturation.

```
nonSat = s < 180 # Find all pixels that are not very saturated

# Slightly decrease the area of the non-satuared pixels by a erosion operation.
disk = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
nonSat = cv2.erode(nonSat.astype(np.uint8), disk)

# Set all brightness values, where the pixels are still saturated to 0.
v2 = v.copy()
v2[nonSat == 0] = 0;

subplot(1,3,1); _imshow(nonSat, "s > 180")
subplot(1,3,2); _imshow(v, "Original\nBrightness")
subplot(1,3,3); _imshow(v2, "Masked\nBrightness")
```



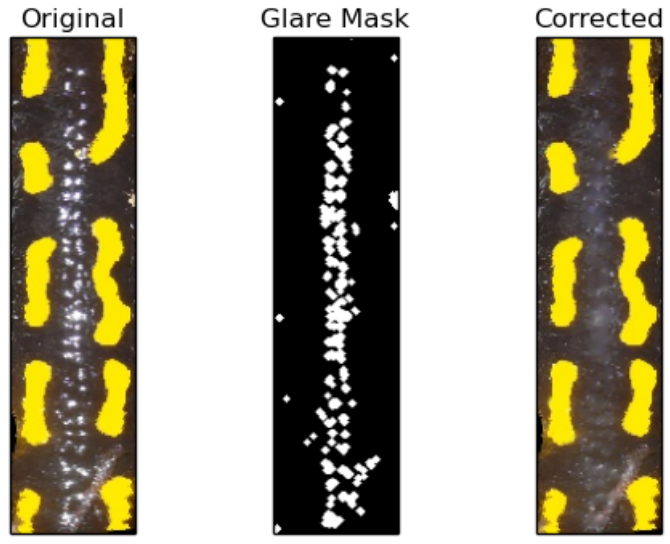
Looking at the masked brightness figure, it is obvious that now all bright pixels belong to the glare region. We can therefore create a histogram of the color values and see, where the peak for the glared pictures appears. In the figure below, the glared pixels are really saturated, so have a brightness of 255. We choose a threshold of 200 to filter out the glared pixels:

```
subplot(1,2,1); hist(v2.flatten(), bins=50);  # draw the histogram of pixel brightnesses
xlabel("Brightness value");
ylabel("Frequency");
title("Brightness histogram")

glare = v2 > 200;    # filter out very bright pixels.
# Slightly increase the area for each pixel
glare = cv2.dilate(glare.astype(np.uint8), disk);
glare = cv2.dilate(glare.astype(np.uint8), disk);
subplot(1,2,2); _imshow(glare);
title("Glare mask");
```



This is the finally obtained glare mask. The glared pixels can now be interpolated with an inpainting operation. This operation fills the masked pixels with the values that stem from the adjacent non-masked pixels. The corrected image is a good approximation of the original image, when no glare would be present.

```
corrected = cv2.inpaint(image_in, glare, 5, cv2.INPAINT_NS)
subplot(1,3,1); _imshow(image_in, "Original")
subplot(1,3,2); _imshow(glare, "Glare Mask")
subplot(1,3,3); _imshow(corrected, "Corrected");
```



Di, 12 Jul 2016 - Maximilian Matthe