Implementation of Algorithms for the Restoration of Retinal Images

David Samir Toro Hoyos

La Salle University
Faculty of Engineering
Automation Engineering
Bogotá
2016

Implementation of Algorithms for the Restoration of Retinal Images

David Samir Toro Hoyos

Degree work to obtain the title of engineer in automation

Supervised by
José Antonio Tumialan Borja
Automation engineering

La Salle University
Faculty of Engineering
Automation Engineering
Bogotá
2016

Thesis approval

_____
_____
_____
_____
_____

_____
Jury's signature

_____
Jury's signature

Acknowledgements

It is always difficult to thank enough to all the people that helped me in this daunting process. Of course, this piece of work would not be possible without the blessing of God so it is for his glory. Also, My work would not be realised without the worry and unconditional help of my family, my dear mother and father, Nayibe and Alonso, and brothers Ignacio, Jorge and Yuri. There were even kind people who offered their help when I needed it, Maria Santos and Lina Maria. I want to thank the family of Lina Maria and my brother Jorge for accompanying and encouraging me when my determination wavered and I needed strengths to continue my work.

Contents

List of Figures

List of Tables

List of Equations

List of Algorithms

List of Codes

Introduction

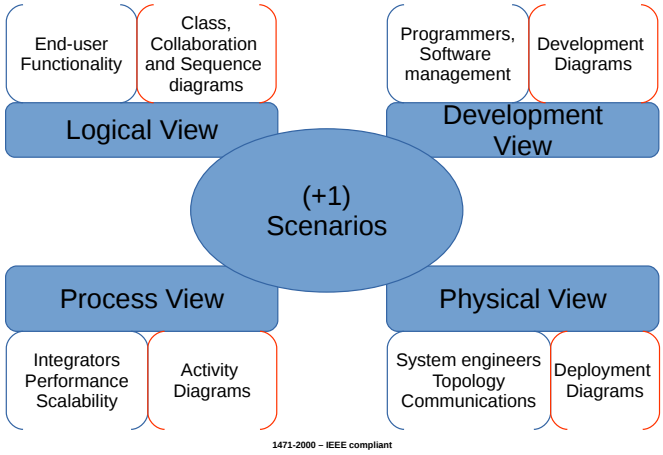
This document was conceived as a phase of a project at the AVARC research group of the Optometry Faculty at La Salle University. The phase consists in the implementation of an algorithm for the restoration of retinal images taken by mobile devices so that it can be further analysed and diagnosed to identify potential patients suffering diabetic retinopathy. It was taken into account the following objectives to ensure the implementation of the application:

1. **Develop algorithm(s) for image restoration:** In which a set of tools (algorithms) were developed for the realization of partial goals dividing the problem into several stages and processing each stage's result to get near the desired application.
2. **Quantitatively validate the application of the restoration algorithm(s):** Each algorithm was validated with its theory and in some cases, a comparison was carried out between its estimation and the desired output.
3. **Implement a restoration application for images:** All tools were organized and packed into one module according to their category with a wizard program inside to offer an automated solution to the restoration of retinal images.

In the world of computer science, some old methods remain robust and effective to solve their intended problems, so in this context some archaic algorithms were included in the main distribution of the program and were not redesigned, as a saying says "there is no need to reinvent the wheel" and thus these algorithms were wrapped into tools. Some algorithms are named in this document but not demonstrated either by its simplicity or because they are thoroughly documented and can be found easily in other sources. All the algorithms are documented and the main ones are tested by means of demonstrations, comparisons or using "unittest" (i.e. Unit testing framework) to ensure robustness and reliability of the code.

Generally the "4+1" View Model (see Figure 1) is widely used to explain complex systems or to convey program models (Phillippe Kruchten, 1995) not only because it complies to the IEEE standard (IEEE, 2000) but because of its simplicity. Despite that, to ensure a clear explanation and understanding of the algorithms this document uses mainly pseudo-codes with some concepts while leaving the code implementation out in the annexes for demonstration purposes. So the "4+1" View Model is partially used here showing in some cases the use classes, some diagrams and leaving out other models. As a plus, additional non-standard tests and demonstrations are given for the reader to confirm that each algorithm works and can be used in practical examples.

Figure 1. "4+1" View Model



Due to that it is not the main purpose of this study to show the programs' functionalities but just their algorithms they are explained using pseudo-codes and their implementations shown in the appendixes. The codes that explains themselves or are too short or depend heavily in the programming language definitions and functions a MWE is provided in their native language in the same section as their explanation instead of the appendixes.

Glossary

The concepts used in this document are a bit esoteric due to the vast majority of specialized procedures used in the algorithms but the intent of this is to make them as simple as possible to let the reader understand the document more easily. Below is a list of key words used throughout the document to encapsulate special concepts that may clarify and convey a better understanding of their meanings:

data persistence

In the context of a program, it consists in storing processed data in a "persistent" form to a non-volatile storage so that the next time the program is run it does not process the data again. This is useful for information that is frequently accessed but that is difficult to obtain e.g. Objects can be stored as byte streams (serialized) and then recreated back (de-serialized) when needed in a program.

De-serialization

It is the contrary process of serialization.

Serialization

It is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed.

Level

It represents a value that any pixel can take on an image i.e. there are usually 256 levels in an gray image and 2 levels in a binary image.

Channel

It refers to the color band in an image i.e. BGRA images have 4 channels compared to the single channel in GRAY images.

Histogram

Graphical representation (or its data equivalent) of the intensity distribution or the number of pixels in each level (intensity value) of a channel in an image.

Intensity

Pixel-level or the pixel's value.

BGR image

Refers to a RGB image with its channels reversed, thus channel 0 is for blue color conversely to Red in RGB images.

BGRA image

It is an RGB image that contains an additional A channel to support additional information like transparency for PNG formats.

Descriptor

Is the description of a feature in an image where elementary characteristics such us shape, color, texture, etc are taken to describe the feature.

Threshold

It is a type of image segmentation at which an image is partitioned into a foreground and background to isolate objects by converting from a gray-scale image to a binary image. Threshold can also refer to the value used to partition the image but for the sake of clarity it is fully used as threshold value.

Segmentation

It is the partitioning of an image into simpler and meaningful segments for the simplification of its analysis.

Mask

Perspective transform

Filter

hashable

hashing

Numpy Arrays

(SciPy, 2015b) In python one of the best ways to represent an array or matrix is using the numpy module. For mathematical matrices numpy provides a class called "matrix" and for arrays with element-wise operation support it provides the ndarray class. Throughout all the algorithms, arrays of the ndarray class are used extensively so it is important to understand its basic structure.

ndarray.ndim. The number of axes (dimensions or rank in python) of the array. Arrays of more than one dimension are like matrices but with a different behavior from their mathematical counterparts.

ndarray.size. The total number of elements of the array.

ndarray.shape. A tuple indicating the size of the array in each dimension. For a matrix with n rows and m columns the shape would be (n, m) therefore the length of the shape is the ndarray.ndim and the product of each element in it is ndarray.size.

ndarray.dtype. A python object describing the type of the elements in the array (i.e. all elements in an array must be of the same type). Some examples are numpy.int32 or numpy.int16 for for integers, and numpy.float32 or numpy.float64 for floats, with the numbers at the end of these types being the bytes in which they are structured or can allocate data in memory but be aware that some of them differ to this naming convention (more can be read in the numpy documentation).

Numpy arrays and matrices work a little different than the operations in MATLAB but with some relations than can be easily adapted from one way to the other. These relations can be better explained in (SciPy, 2015a).

To see the common terms used addressing python code see (Python Software Foundation, 2015).

Art State

Diabetes is a chronic disease that occurs when the pancreas does not produce enough insulin or the body cannot effectively use the insulin it produces to regulate blood's sugar causing severe complications over time. These complications affects especially the nerves and blood vessels causing macro and micro vascular changes leading to increases in the risk of heart disease and stroke (50% of people with diabetes die of cardiovascular disease), renal problems, neuropathy (nerve damage) and diabetic retinopathy (DR) (Faust, U, Ng, Ng, & Suri, 2010). Type 1 diabetes (T1D, previously known as insulin-dependent, juvenile or childhood-onset) is characterized by deficient insulin production in the pancreas requiring daily administration of insulin to the patient; currently its cause is not known and it is not preventable (this does not mean that some measurements should not be taken). Type 2 diabetes (T2D, formerly called non-insulin-dependent or adult-onset) is the most chronic worldwide which comprises 90% of people with diabetes around the world (WHO, 2016; WebMD, 2014).

The rate of diabetes is increasing, according to The World Health Organization (WHO) and International Diabetes Federation (IDF) in the year 2000 the number of people with diabetes were of 171 million and it was estimated to increase to 366 million by 2030 being by then the 7th leading cause of death (WHO & IDF, 2006), but just in 2014 the number of people with diabetes rose to 422 million and caused over 1.5 million deaths worldwide in 2012. This is probably going to continue to raise as it is consistent with the global prevalence of diabetes among adults (i.e. over 18 years of age) which has risen from 4.7% in 1980 to 8.5% in 2014. This only worsens in low and middle-income countries where it is estimated that more than 80% of diabetes deaths occur and diabetes prevalence has been rising more rapidly in the last years (WHO, 2016).

Many studies concord that people with diabetes are at risk of developing diabetic retinopathy (DR) ultimately leading to blindness as a result of long-term accumulated damage to the small blood vessels in the retina. Even though people with diabetes are 25 times more likely to develop blindness and DR is considered to have caused one percent of global blindness only one-half of the patients are aware of this disease. The most effective treatment for it can only be administered at the first stages of the disease thus regular DR screening is of paramount importance and recommended annually, but most developing countries lack the ability to fully record these DR cases.

To prevent it, many models has been considered to address this problem (Askew et al., 2012). Thanks to the resent years advancements in technology and the access to it have facilitated the acquisition of advanced appliances that allow fast and reliable DR screenings in which digital imaging technology plays a key role allowing to employ state-of-the-art processing techniques to automate the detection of abnormalities in retinal images (Faust et al., 2010).

One method in particular can take huge advantages from Telemedicine and digital imaging for diagnosing DR and that is the utilization of tele-ophthalmology by obtaining digital retinal images with specialized cameras (fundus cameras) and electronically transmitting them to an expert for their assessment and patient diagnosis. As a solution, it is an efficient alternative for early diagnostic and treatment of diabetic patients.

In addition, tele-ophthalmology involves the following benefits (Martínez Rubio, Moya Moya, Bellot Bernabé, & Belmonte Martínez, 2012):

- Unlike normal methods, providing healthcare to a high number of patients.
- Reducing waiting periods.
- Saving time in the diagnostic and treatment.
- Avoiding delays which can produce health problems for patients.
- Digital ocular fundus photographs are of low cost and of little discomfort for the patient.
- The application of telemedicine reduces the workload to experts letting them treat more outpatients efficiently.

Taking a particular subdivision from the ophthalmological screening methods are the non-mydriatic screening techniques. Numerous papers have demonstrated that it is one of the health interventions with the best cost-effectiveness ratio, high sensitivity and specificity. Non-mydriatic retinal cameras allow high-quality photographs through undilated pupils in digital format

offering benefits like facilitating medical compliance (i.e. patient correctly following medical advice), patients not requiring to be still at the assessment and images delivering a much broader field of view (Askew et al., 2012).

Specialized non-mydriatic fundus cameras can be cost-efficient in operational terms but exhibit a relative medium inversion cost, making them difficult to afford in some cases. Yet again, technology can offer solutions to further reduce costs while offering decent results that can be at pear with what would be obtained with professional fundus cameras and that is a mobile device (e.g. cellphones, tablets, etc) which is a ubiquitous device that offers many functionalities including the acquisition of digital images and though it is not cheap it is more likely that anyone have one than a non-mydriatic fundus camera, thus preventing the necessity to acquire one.

Screening of patients with adapted mobile devices can become a highly valid method for the detection and prevention of DR allowing early access to assessments while offering easiness of use and other benefits like availability.

Programming and Research Approach

Before we discuss the algorithms, it is important to understand the context in which they were conceived. Many programming languages were taken into consideration for the implementation of the algorithms but few sufficed the requirements of high performance while keeping the readability, portability, modularity and scientific basis that a language needs. Compiled languages like C++ offer high speeds but at the cost of lengthy codes and long developing times while their contra parts, interpreted languages like MATLAB and Python, sacrifice execution speed to offer short and fast code prototyping (i.e. while developing) because of their simple yet readable languages and even offering in some cases means to improve speed by compiling or translating to other languages like C or using them as back-ends (Cython.org, n.d.).

Table 1
Comparison of programming languages

Factor	**C++**	**MATLAB®**	**Python**
Interpreter type	compiled	just-in-time	Interpreted
Language abstraction	High-level	High-level	High-level
Standard	ISO standard	de-facto standard	de-facto standard
License	Open source	Licensed	Open source

Source: Author

Table 2
Comparison example of "for loops" in C and Python

C for loop

```c
#include <stdio.h>
#include <string.h>
int main() {
char s[5] = "Hello";
for(int i=0;i<strlen(s);i++){
    printf("%c", s[i]);
    return 0;
}
```

Python simulation of C for loop

```python
s = "Hello"
for i in range(len(s)):
    print s[i]
```

Pythonic for loop

```python
s = "Hello"
for letter in s:
    print letter
```

Source: Author

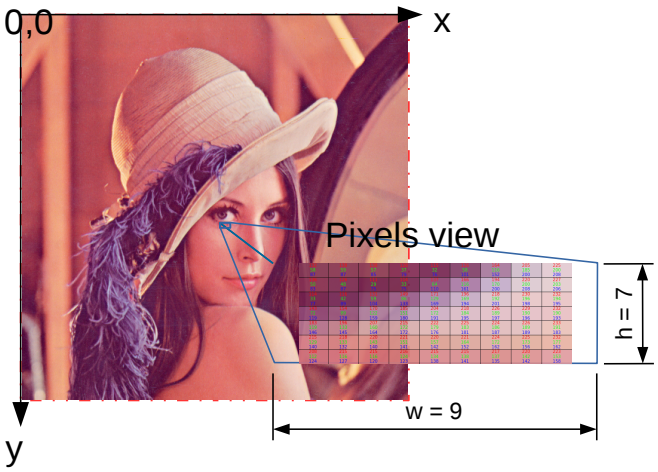
Resource managing:
memoization and data persistence
caching
file mapping
Programming strategy:
lazy evaluations
multi-processing
multi-threading

Images in openCV

An image can be represented as a matrix of width w and height h with its elements as pixels, where each pixel is a representation of a color in one point of a discrete plane (2D dimension). In the case of OpenCV and many other libraries for image manipulation, the use of Numpy arrays as the base for image representation is becoming the standard (Numpy is a fast and powerful library for array manipulation and one of the main modules for scientific development in python). A Numpy array is multidimensional container of items of the same type and size differing mainly to the mathematical matrix in which its operations are element-wise (i.e. corresponding index in each array).

The Image shape, hight and width (h, w) , then corresponds to a Numpy array with n rows and m columns (n, m) which in a Cartesian plane would be the (y, x) axes (Scipy, 2015).

Figure 2. Axes orientation of images



Algorithms design

There always emerge important questions in the development of every project like how should it be made? perhaps this peace of code is not the right one? or is it robust enough for every possible scenario? or is it the best solution to my problem?. Well, not all solutions are absolute and there could be better implementations but for the most of it I consider that experimentation is one of the best insurance methods that exists for scientific solutions. As I consider that without experimentation, a solution may not be well tested and it could not be pushed forward to the limits of their potential. Another good reason is that it opens room for creativity that could lead to unexpected inventions not yet discovered. So below is presented some of the experiments that led to the implementation of the main algorithm.

General algorithms

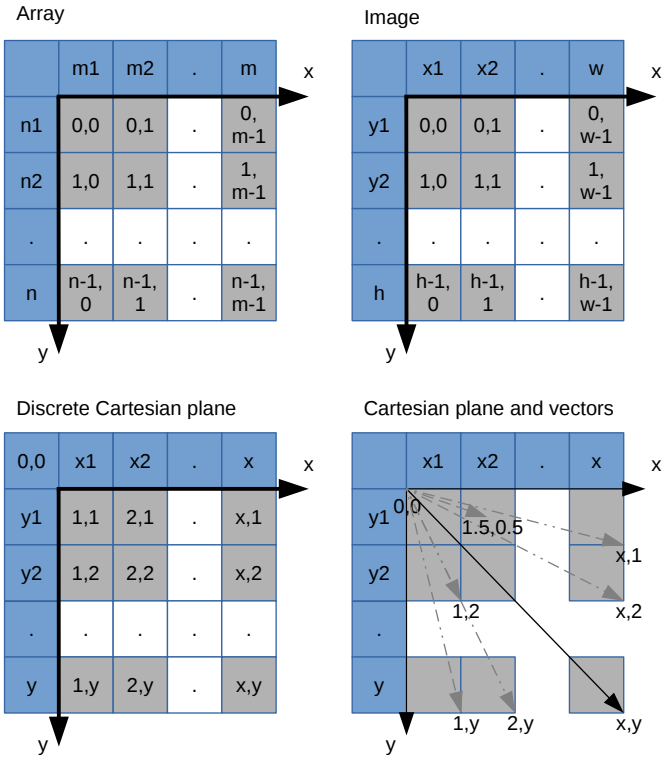
The area under a polygon (*poligonArea*). The area under a polygon can be calculated if the points of its vertices are known. This is described in (Darel Rex Finley, n.d.) which can be expressed by the general equation of the *poligonArea* as:

$$poligonArea = \frac{1}{2}\,|xsum - ysum|$$

where

$$ysum = x_0 y_n + \sum_{i=0}^{n-1} x_{i+1} y_i$$

Figure 3. Used representations



$$xsum = x_n y_0 + \sum_{i=0}^{n-1} x_i y_{i+1}$$

then replacing $xsum$ and $ysum$ we get

$$poligonArea = \frac{1}{2}\left|-x_0 y_n + x_n y_0 + \sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)\right| \tag{1}$$

This equation can calculate the area of any arbitrary polygon but with the limitation that the points that conform it must be in adequate order (i.e. if lines are traced passing at each point in order then these lines must not intersect each other). If for example we have the following points (see Figure 4):

$$points = \begin{bmatrix} -3 & -2 \\ -1 & 4 \\ 6 & 1 \end{bmatrix}$$

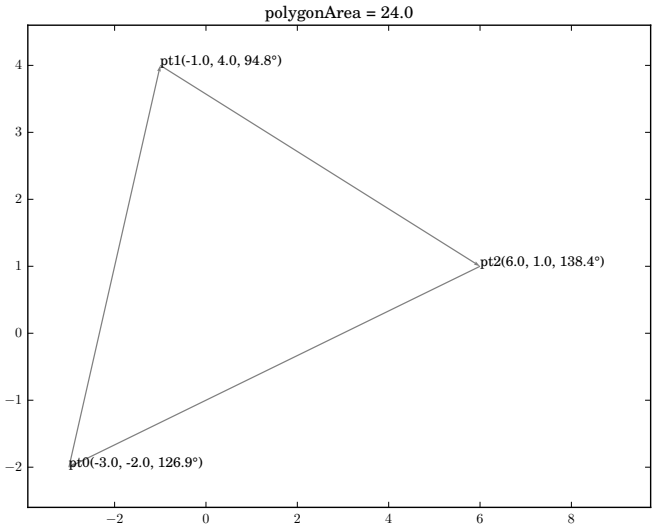
then we solve it as in Equation 1

$$\begin{aligned} ysum &= x_0 y_2 + x_1 y_0 + x_2 y_1 \\ &= (-3)\cdot 1 + (-1)\cdot(-2) + 6 \cdot 4 \\ &= 23 \\ xsum &= x_0 y_1 + x_1 y_2 + x_2 y_0 \\ &= (-3)\cdot 4 + (-1)\cdot 1 + 6 \cdot (-2) \\ &= -25 \end{aligned}$$

which is replaced in *poligonArea* giving:

$$poligonArea = Abs((-25) - 23)/2 = 24$$

Figure 4. polygonArea example

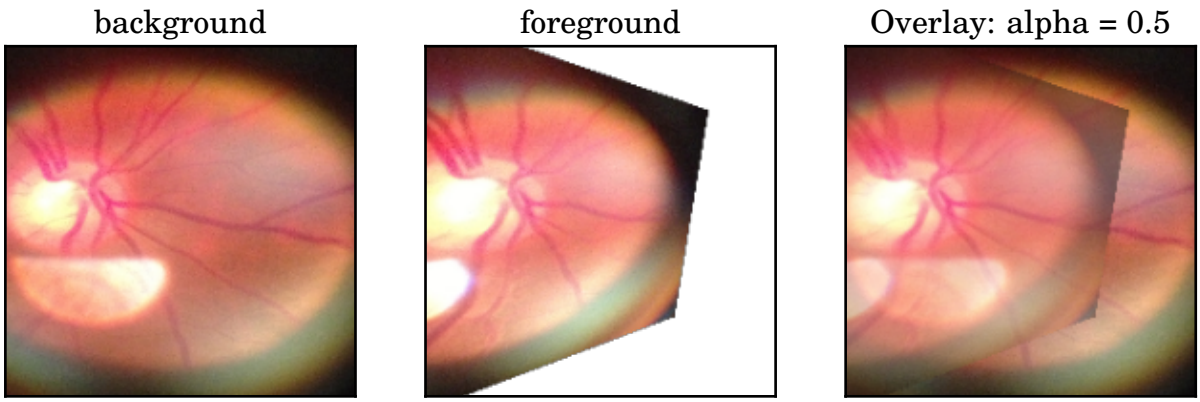


Overlay. Overlay foreground over background image with an *alpha* transparency.

$$Overlay_{i,j} = Fore_{i,j} \cdot alpha_{i,j} + Back_{i,j} \cdot (1 - alpha_{i,j}) \tag{2}$$

Figure 5 shows an overlay example with $alpha = 0.5$.

Figure 5. Overlay sample



Real image – Rendered image. There exists a real image I with shape (H, W) for height and width respectively which has a rendered image I_r with shape (H_r, W_r) of any view inside I (i.e. any part of I can be selected and cropped with a different resolution than I that is useful for zooming-in). To calculate the coordinate system in x, y of the real image I to the coordinate system x_r, y_r of the rendered image I_r where x_r, y_r is a point in the domain of I_r and I_r is a ROI contained between $x_2 - x_1$ and $y_2 - y_1$ in image I (i.e. I_r is a re-scaled ROI of I with a cropping box from the top-left point x_1, x_2 to the bottom-right point x_2, y_2) then the following equations apply:

To convert from I to I_r coordinates:

$$x_r(x),\ y_r(y) = W_r \cdot \frac{(x - x_1)}{(x_2 - x_1)},\ H_r \cdot \frac{(y - y_1)}{(y_2 - y_1)} \tag{3}$$

To convert from I_r to I coordinates:

$$x(x_r),\ y(y_r) = x_1 + x_r \cdot \frac{(x_2 - x_1)}{W_r},\ y_1 + y_r \cdot \frac{(y_2 - y_1)}{H_r} \tag{4}$$

where:

$$(H_r,\ W_r) = (y_{rmax} - y_{r0},\ x_{rmax} - x_{r0}) \begin{cases} x_{r0} < x_{rmax} & \in I_r \\ y_{r0} < y_{rmax} & \in I_r \\ x_{max} \geq x_2 > x_1 \geq x_0 & \in I \\ y_{max} \geq y_2 > y_1 \geq y_0 & \in I \end{cases}$$

Perspective transformation. These conventions are used when dealing with transformations matrices:

- When the transformations are with respect to the absolute Cartesian system it is referred as xyz and all transformation matrices (TM's) must be pre-multiplied at each step.
- When the transformations are relative with respect to the previous position it is referred as uvw and all TM's are post-multiplied at each step.

The perspective transform maps from x, y coordinates to u, v coordinates using transformation matrices. To transform any x, y point in an image a $3x3$ transformation matrix M is multiplied with the column vector $\begin{bmatrix} x & y & 1 \end{bmatrix}$ which yields another column vector $\begin{bmatrix} X' & Y' & c \end{bmatrix}$:

$$\begin{bmatrix} M_{0,0} & M_{0,1} & M_{0,2} \\ M_{1,0} & M_{1,1} & M_{1,2} \\ M_{2,0} & M_{2,1} & M_{2,2} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ c \end{bmatrix}$$

then the resulting u, v point where x, y is moved is calculated as:

$$u, v = \frac{x'}{c},\ \frac{y'}{c}$$

Domain transformations: scaling to original results. One of the techniques used to reduce processing times and standardize input images is to resize an original image I_o to a scaled image I_s which is used for all the algorithm's analyses and then applied to the original image. But there surges a pretty obvious problem, how to convert the result of I_s back to I_o? In the case of coordinates or points this can be solved using linear relations to convert from one domain to the other but if the result is a TM then additional transformation matrices are applied to adjust it for the original sample.

Suppose that a scaled transformation matrix (M_s) is found to convert the perspective from a scaled image A (A_s) to the perspective of a scaled image B (B_s). So M_s is used to convert A_s to the domain of B_s but the objective is to convert the original image A (A_o) to the domain of the original image B (B_o) with a original transformation matrix (M_o). Several steps are summarized to achieve this:

1. Use a scaling transformation A_{os} to transform the points from A_o to the domain of A_s.
2. Use M_s to transform the points of A_s to the domain of B_s.
3. Use a scaling transformation B_{so} to transform the points from B_s to the domain of B_o.

Now for the mathematical abstraction.

$$
\begin{aligned}
M_o &= B_{so} \cdot M_s \cdot A_{os} \\
&= \begin{bmatrix} Bx_{so} & 0 & 0 \\ 0 & By_{so} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_{s11} & M_{s12} & M_{s13} \\ M_{s21} & M_{s22} & M_{s23} \\ M_{s31} & M_{s32} & M_{s33} \end{bmatrix} \begin{bmatrix} Ax_{os} & 0 & 0 \\ 0 & Ay_{os} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} Ax_{os} \cdot Bx_{so} \cdot M_{s11} & Ay_{os} \cdot Bx_{so} \cdot M_{s12} & Bx_{so} \cdot M_{s13} \\ Ax_{os} \cdot By_{so} \cdot M_{s21} & Ay_{os} \cdot By_{so} \cdot M_{s22} & By_{so} \cdot M_{s23} \\ Ax_{os} \cdot M_{s31} & Ay_{os} \cdot M_{s32} & M_{s33} \end{bmatrix}
\end{aligned}
$$

where

$$
Ay_{os} = \frac{Ah_s}{Ah_o},\ Ax_{os} = \frac{Aw_s}{Aw_o},\ Bx_{so} = \frac{Bw_o}{Bw_s},\ By_{so} = \frac{Bh_o}{Bh_s}
$$

with (Aw_o, Ah_o) as the shape of A_o, (Aw_s, Ah_s) the shape of A_s, (Bw_o, Bh_o) the shape of B_o and (Bw_s, Bh_s) the shape of B_s. Then M_o can be rewritten as

$$
M_o = \begin{bmatrix} \left(\frac{Aw_s}{Aw_o}\right)\left(\frac{Bw_o}{Bw_s}\right) M_{s11} & \left(\frac{Ah_s}{Ah_o}\right)\left(\frac{Bw_o}{Bw_s}\right) M_{s12} & \left(\frac{Bw_o}{Bw_s}\right) M_{s13} \\ \left(\frac{Aw_s}{Aw_o}\right)\left(\frac{Bh_o}{Bh_s}\right) M_{s21} & \left(\frac{Ah_s}{Ah_o}\right)\left(\frac{Bh_o}{Bh_s}\right) M_{s22} & \left(\frac{Bh_o}{Bh_s}\right) M_{s23} \\ \left(\frac{Aw_s}{Aw_o}\right) M_{s31} & \left(\frac{Ah_s}{Ah_o}\right) M_{s32} & M_{s33} \end{bmatrix} \tag{5}
$$

Normalization. An array A is normalized to the range $[0, 1]$ with the simple function:

$$
N(A) = \frac{A - min(A)}{max(A)} \tag{6}
$$

where *min* and *max* are functions that return the minimum and maximum value of A respectively.

Load functions

In python there are several ways to load an image with functions from packages that provide different features making it easy to provide support to load images from different sources, formats and specifications. This capability was exploited by coding a way to seamlessly load images when provided the path to the source and determine the means to retrieve them.

Load from sockets. To load an image from a socket it was achieved using the the socket package from python by defining a server and a client with simple protocol to transfer pickled data.

Load from URLs. To load images from an Uniform Resource Locator the urllib family of modules from python can be used. In python there is urllib, urllib2 and urllib3 with some differences between python 2 and python 3. For the implementation it was used urlopen from urllib3 and urllib.request from python 2 and 3 respectably which returns a handle similar to a file object created by the built-in *open* object which opens files. These two API being similar facilitates more the implementation.

Load from files. It is achieved directly using the OpenCV function *cv2.imread* or images can be read directly using the build-in *open* function and converting the string to an array with np.fromstring and then decoding it with *cv2.imdecode* to obtain the image.

Load from memmaped files. This is one of the ways to save memory to a file and access that file directly from the disk instead of keeping it in the RAM. To create and retrieve a binary file in NumPy format (i.e. extension .npy) the save and load functions are provided in the *numpy.lib* module which creates a binary representation of a numpy array that can be simply loaded to memory, pickled (technically it is the pickled file itself different to the string representation used for other purposes e.g. store in databases as a string and not as a external file), transferred remotely, accessed with standard read and write modes and still behave like a normal numpy object which can be referenced or shared between processes (useful for multitasking).

Pre-processing, Filters and Enhancing Methods

Histogram equalization. Histogram equalization (HE) distributes the colors equally in an image based from its histogram, so if an image is bright most pixels will be confined to the highest levels or if an image is dark most pixels will be at the lowest levels and HE will distribute this levels to the levels that are lacking pixels improving the contrast of the image. That way images of the same scenery with different light conditions will be almost the same after image equalization or histograms confined to particular regions will be distributed to all regions (K & Mordvintsev, 2013).

To cover the full spectrum a transformation function or lookup table is needed to map from the input pixel of the specific regions to the output pixels of the full region. This is accomplished using the cumulative distribution function (CDF), in a sense HE helps standardize the CDF of an image when distributing all the colors equally throughout it. Code 1 is a MWE which demonstrates the histogram equalization. Additional explanations in (Gonzalez, Woods, & Eddins, 2004).

Code 1: Histogram equalization

```
import cv2
import numpy as np
img = cv2.imread('image.jpg',0) # load the image
hist,levels = np.histogram(img.flatten(),256,[0,256]) # get histogram
cdf = hist.cumsum() # get CDF by accumulating histogram
cdf_m = np.ma.masked_equal(cdf,0) # mask to convert to range [0,1]
cdf_m = (cdf_m - cdf_m.min())*255/(cdf_m.max()-cdf_m.min()) # scale to range [0,255]
cdf = np.ma.filled(cdf_m,0).astype('uint8') # mask as a look-up table
img2 = cdf[img] # apply the mask to transform the inputs pixel to output pixels
```

OpenCV has a function to apply HE called `cv2.equalizeHist()` where its input is a grayscale image and the output is the equalized image.

A drawback is that it won't work well where there are large intensity variations, if for example a histogram is bimodal with the two peaks apart of each other (i.e. one peak at low levels and the other at high levels) or histogram is not confined to a particular region then these will be equalized causing lost of details in some cases due to saturation (where there is saturation lost of information is present too). To prevent it openCV provides a Contrast Limited Adaptive Histogram Equalization ($CLAHE$) function that is applied to sub-regions in the image. This approach does not consider the global contrast of the image as in HE preventing the problems that this implies, in contrast $CLAHE$ divides the image into small blocks called *tiles* of typically $8x8$ (the parameter is $tileGridSize$) to apply HE in each one and that way the histogram would be confined to small regions (solving the particular problems in HE) but this brings up another problem, if noise is present it will be amplified. To avoid this, a contrast limit is applied to any level above a specified contrast (the parameter is *clipLimit* with 40 as its default value) in a process where the pixels are clipped and distributed uniformly to other levels before HE is applied. Bilinear interpolation is applied to remove artefacts created at the borders of each tile after each individual equalization (K & Mordvintsev, 2013).

In general histogram equalization techniques are used in pre-processing operations to usually normalize the image patterns and lighting conditions (Médioni, 2005) but it has been observed that it intensifies noise (Antal & Hajdu, 2012) and for that noise attenuation is recommended before using a histogram equalization (Sopharak, Uyyanonvara, & Barman, 2013). For further optimizations using C++ and CUDA to process in the computer's GPU you can follow the advices in (Fierval, 2015).

Figure 6 shows a histogram equalization example.

Matrix decomposition to reduce levels. Matrix decomposition offers a different representation of the original image array and lets the information to be changed easily while affecting to all pixels instead to a single one. This can reduce the diversity of image levels to offer less colors than the original and reduce file storage but at an unwanted price, less details and added artefacts like edges that can prevent the image to be adequately computed.

Figure 7 shows two decomposition tests for a gray image with different patterns and intensities.

Figure 6. Histogram equalization

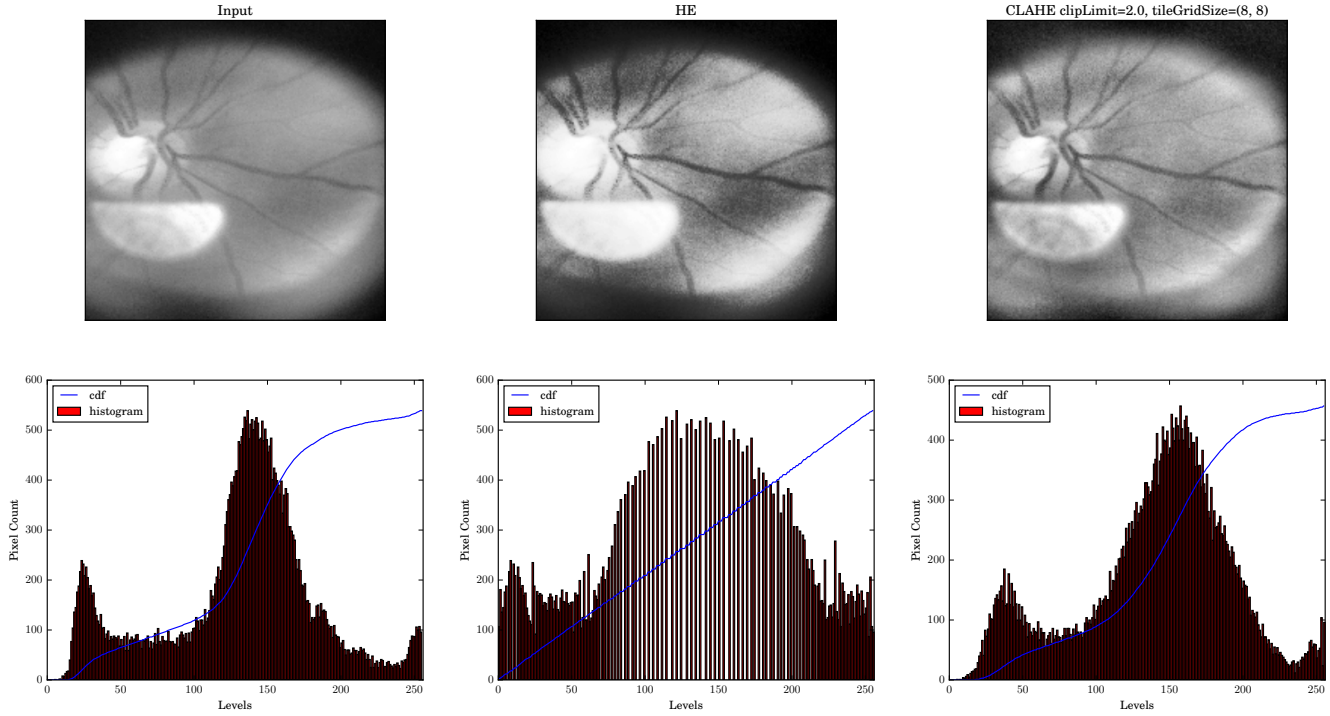
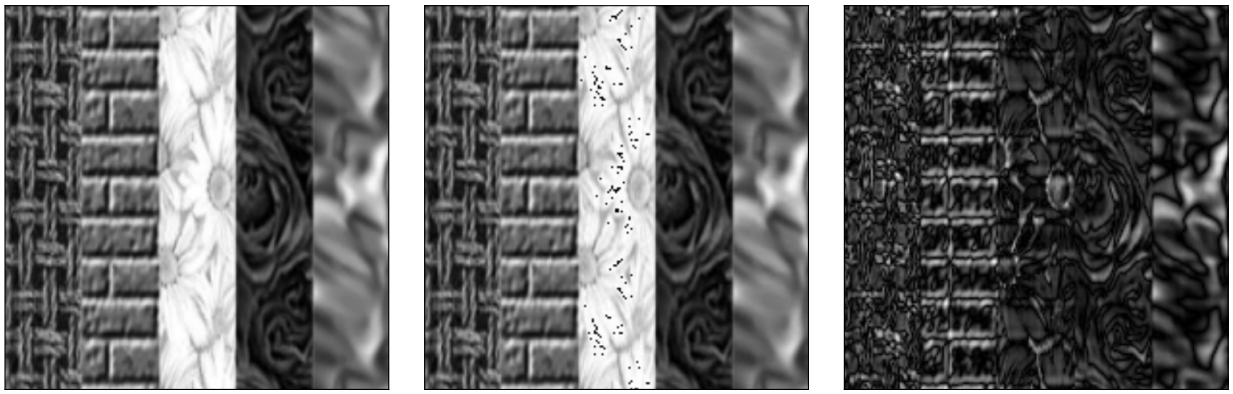


Figure 7. Decomposition tests



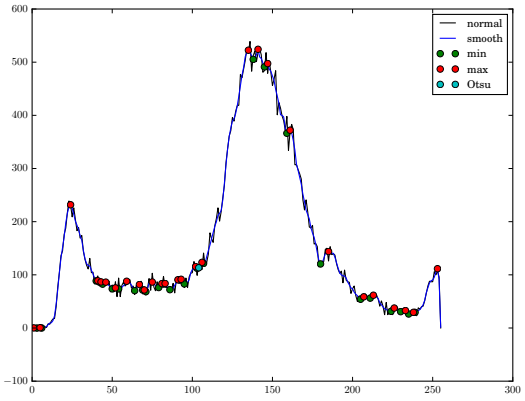
(a) Normal: s.shape = (200,) (b) Recomposed svd with s[50:]=0 (c) Recomposed svd with s[:1]=0

Smoothing with 1D-filters. These filters are used to smooth data which exhibit a noisy like behaviour but with trends to certain values. One of the applications is to smooth an image histogram to obtain its minima and maxima local values.

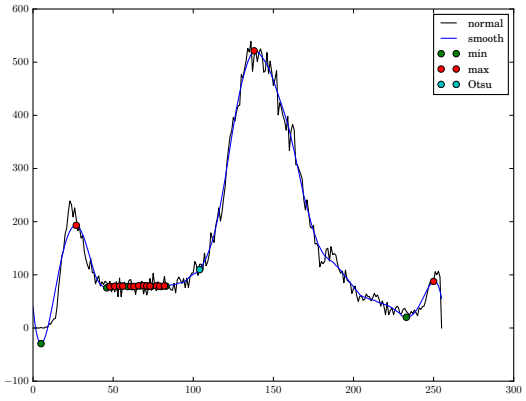
Gaussian Filter. It is one of the most widely used filters to reduce Gaussian noise in images but it produce a blur-like effect that eliminates details along with its application. Nonetheless, it is known that this can be a desirable effect used to simulate the loss of detail produced when an scenery is photographed from a farther viewpoint for applications where scale invariance is important (Rey Otero & Delbracio, 2014).

Bilateral Filter. The *bilateralFilter* can reduce unwanted noise very well while keeping edges fairly sharp. However, it is very slow compared to most filters so it is not recommended for real-time applications where the hardware is limited and can't cope with fast processing times giving retarded results. In the other hand, this filter does very well in applications where the time is not the most important factor for its functionality. The OpenCV package offers the function *bilateralFilter* (OpenCV, n.d.) which applies bilateral filtering to the input image as described in (CVonline, 2004) with the following arguments:

Figure 8. Savgol filter

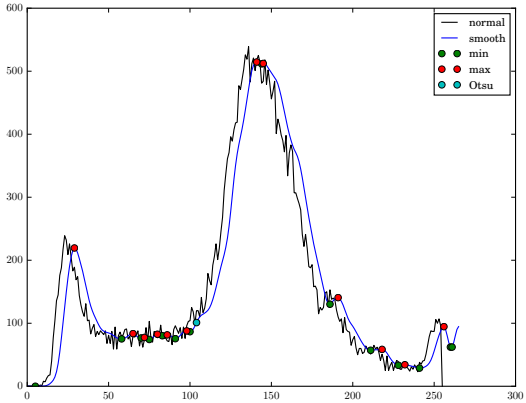


(a) Savgol with $window_{len}$ = 11 and polyorder = 5

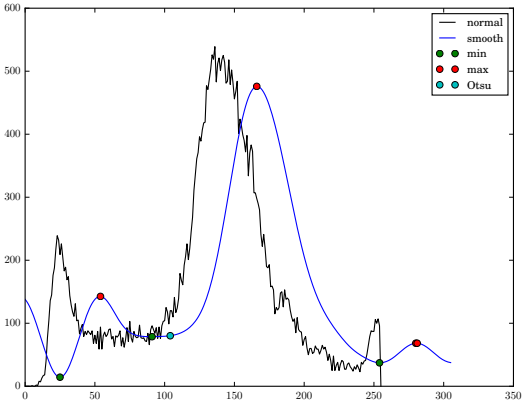


(b) Savgol with $window_{len}$ = 51 and polyorder = 5

Figure 9. Hanning filter



(a) Hanning with $window_{len}$ = 11



(b) Hanning with $window_{len}$ = 51

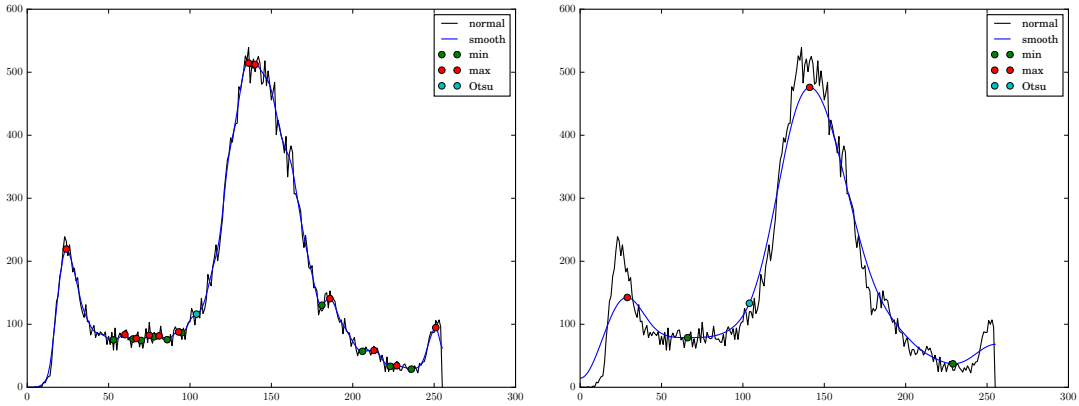
- *src*– Source 8-bit or floating-point, 1-channel or 3-channel image.
- *dst*– Destination image of the same size and type as *src*.
- *d*– Diameter of each pixel neighbourhood that is used during filtering. If it is non-positive, it is computed from *sigmaS pace*.
- *sigmaColor*– Filter sigma in the color space. A larger value of the parameter means that farther colors within the pixel neighbourhood (see *sigmaS pace*) will be mixed together, resulting in larger areas of semi-equal color.
- *sigmaS pace* – Filter sigma in the coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough (see *sigmaColor*). When $d > 0$, it specifies the neighbourhood size regardless of *sigmaS pace* . Otherwise, *d* is proportional to *sigmaS pace* .

Filter size. Large filters (d>5) are very slow, so it is recommended to use d=5 for real-time applications, and perhaps d=9 for off-line applications that need heavy noise filtering.

Sigma values. For simplicity, you can set the 2 sigma values to be the same. If they are small (<10), the filter will not have much effect, whereas if they are large (>150), they will have a very strong effect, making the image look "cartoonish".

NOTES: This filter is not recommended for in-place works (i.e. not for real time applications) but for off-line applications.

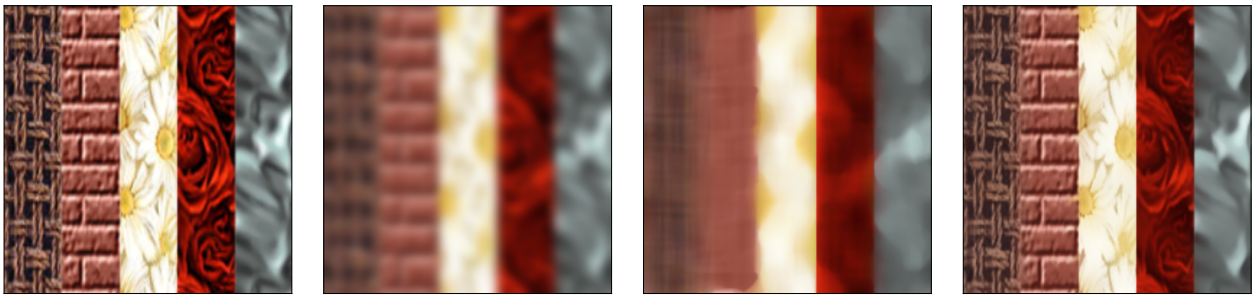
Figure 10. Hanning filter with shifted convolution



(a) Hanning with $window_{len}$ = 11

(b) Hanning with $window_{len}$ = 51

Figure 11. Spacial filters comparison



(a) Original

(b) Gauss: kernel = (21, 21)

(c) Median: ksize = 21

(d) Bilateral: d = 21, sigmaC = 82, sigmaS = 57

Figure 11 shows a comparison between the bilateral filter and the most common spacial filters, Gaussian and Mean filters.

Figure 12 shows the selection for the parameters in the Bilateral filter according to the image's shape.

Figure 13 shows the results of the Bilateral filter for an noisy image with different shapes with the parameters selected for the algorithm as shown in Figure 12.

SigmoidImageFilter. The sigmoid filter is based in the sigmoid function which is characterized by having an "S" shape with a dampening behavior when tending to $-\infty$ and an approximation to the value "1" when tending to ∞ (i.e. an adequate offset can be used to determine the threshold to damper or let through certain values). In particular this filter was inspired from one of the tools of Mevislab, the $SigmoidImageFilter$ which uses the Insight Segmentation and Registration Toolkit (ITK) described in (ITK, n.d.). The equation is as follows:

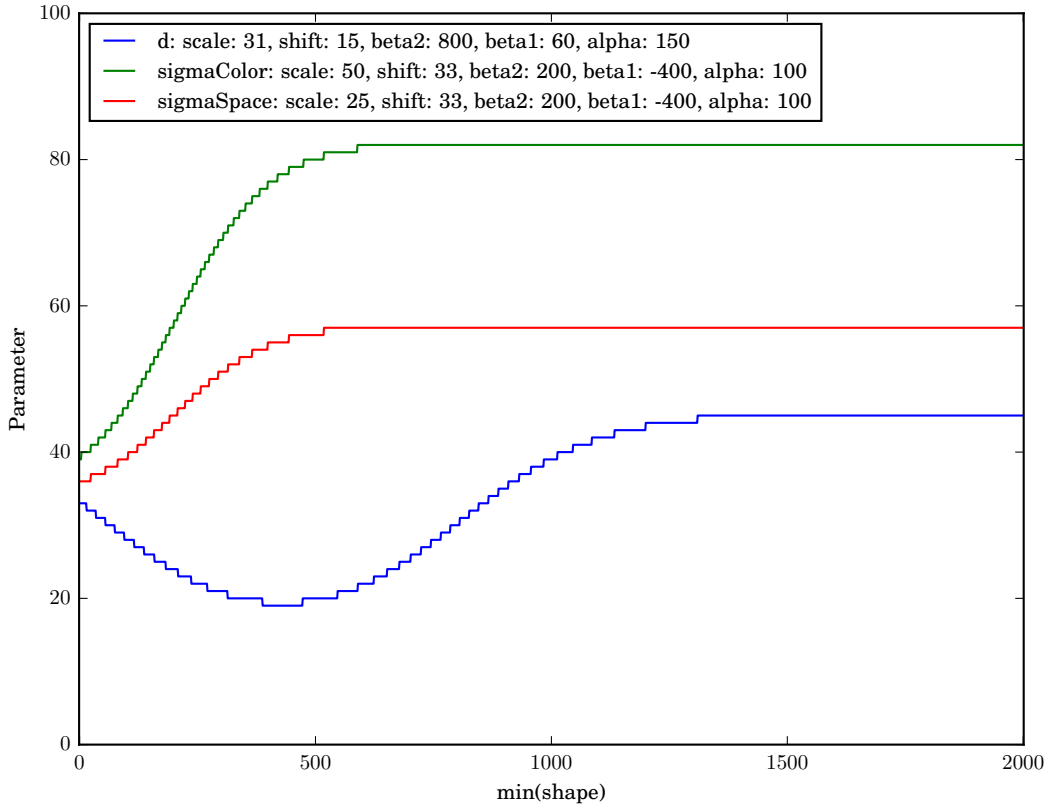
$$Sigmoid(x_i, \alpha, \beta, Max, Min) = (Max - Min)\left(\frac{1}{1 + e^{-(x_i - \beta)/\alpha}}\right) + Min \tag{7}$$

where Max is the maximum value at the output, Min is the minimum value at the output, β the offset and α a scaling factor.

The β parameter can be thought as the offset on the pixel value that you are trying to isolate i.e. if the object you are trying to segment is at a pixel intensity above 150, you would choose a β value that is around that value and the α parameter can be thought of as the scaling or variance of the sigmoid (CMISS, n.d.).

Normalized SigmoidImageFilter. The normalized sigmoid filter is extracted from the component in Equation 7 that computes any x value (for discrete it is x_i) with an offset β and a scaling factor α that its result is inside the range of $[0, 1]$. This is:

Figure 12. Selection of bilateral parameters acording to image shape



$$nS_i = normSigmoid(x_i, \alpha, \beta) = \frac{1}{1 + e^{(\beta - x_i)/\alpha}} \qquad \forall \quad x_i, \alpha, \beta \in \mathbb{R} \quad \wedge \quad 0 \leqslant nS_i \leqslant 1 \tag{8}$$

which stabilizes at the limits:

$$\begin{matrix} \lim\limits_{x \to -\infty} nS(x) = 0 \\ \lim\limits_{x \to \infty} nS(x) = 1 \end{matrix} \qquad \forall \quad \alpha > 0$$

This facilitates the calculation of α and β. For example, solving for α yields:

$$\alpha(\beta) = \frac{\beta - x}{\ln\left(\frac{1}{nS} - 1\right)}$$

And solving for β yields:

$$\beta(\alpha) = \alpha * \ln\left(\frac{1}{nS} - 1\right) + x$$

Also, Equation 7 can be expressed in therm of the Equation 8 as follows:

$$Sigmoid(x, \alpha, \beta, Max, Min) = (Max - Min) * normSigmoid(x, \alpha, \beta) + Min \tag{9}$$

which stabilizes at the limits:

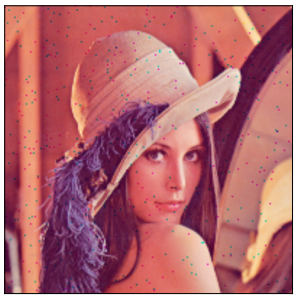
Figure 13. Bilateral filter exposed to different shapes

(a) Input shape = (50, 50)

(b) Filtered with d = 9, sigmaC = 42, sigmaS = 37

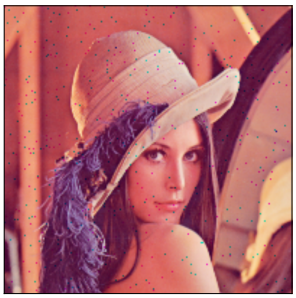
(c) Input shape = (100, 100)

(d) Filtered with d = 9, sigmaC = 46, sigmaS = 39

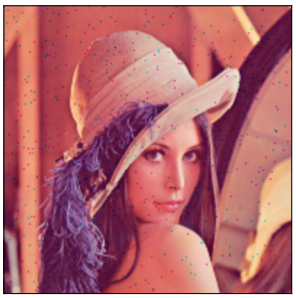


(e) Input shape = (600, 600)

(f) Filtered with d = 9, sigmaC = 82, sigmaS = 57



(g) Input shape = (1500, 1500)



(h) Filtered with d = 9, sigmaC = 82, sigmaS = 57

$$\begin{matrix} \lim_{x\to-\infty} Sigmoid(x) = Min \\ \lim_{x\to\infty} Sigmoid(x) = Max \end{matrix} \quad \forall \quad \alpha > 0$$

Custom filters using *normSigmoid*. Observing that the *normSigmoid* function is similar to a Butterworth filter it can be used to make custom filters as in signal processing allowing an image to be filtered (as in non-spacial or color filters) with common filters such as high-pass, low-pass, band-stop, band-pass or any custom filter created by the combination of the *normSigmoid* function and normalizations (see Equation 6).

The simplest filters can be made using a single *normSigmoid* function. These are the high-pass and low-pass filters created when α is positive and negative respectively. The others can be seen as a compound of these two.

$$lowpass(x,\alpha,\beta) = normSigmoid(x,\alpha,\beta) \quad \forall \quad \alpha < 0 \tag{10}$$

$$highpass(x,\alpha,\beta) = normSigmoid(x,\alpha,\beta) \quad \forall \quad \alpha > 0 \tag{11}$$

$$bandstop(x,\alpha,\beta) = lowpass(x,\alpha,\beta_1) - lowpass(x,\alpha,\beta_2) + 1 \quad \forall \quad \beta_1 > \beta_2 \tag{12}$$

$$bandpass(x,\alpha,\beta) = highpass(x,\alpha,\beta_1) - highpass(x,\alpha,\beta_2) \quad \forall \quad \beta_1 > \beta_2 \tag{13}$$

$$bandstop_{Inverted}(x,\alpha,\beta) = lowpass(x,\alpha,\beta_2) - lowpass(x,\alpha,\beta_1) - 1 \qquad \forall \quad \beta_1 > \beta_2 \tag{14}$$

$$bandpass_{Inverted}(x,\alpha,\beta) = highpass(x,\alpha,\beta_2) - highpass(x,\alpha,\beta_1) \qquad \forall \quad \beta_1 > \beta_2 \tag{15}$$

Figure 14 shows the response of each normalized filter and Figure 15 shows the comparison among them. Notice how all the filters converge if they share the same α and β parameters.

Figure 14. Common filters response using *normSigmoid*

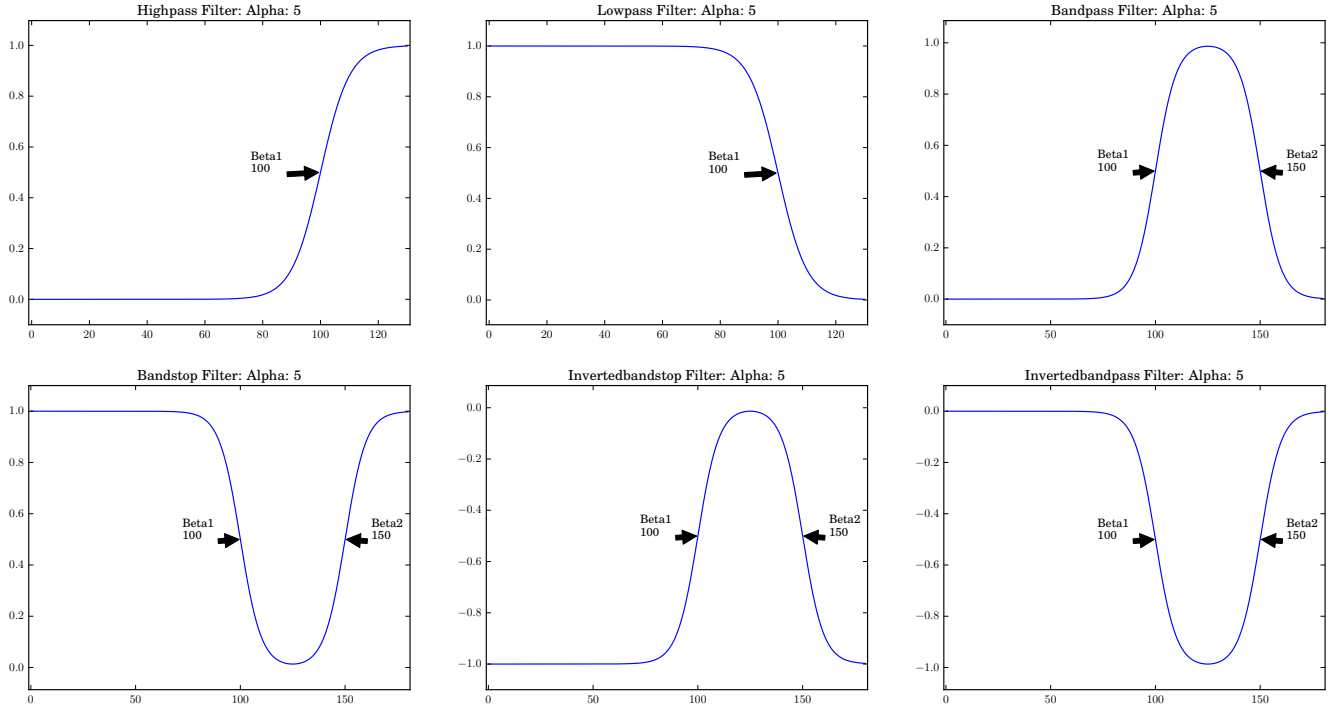
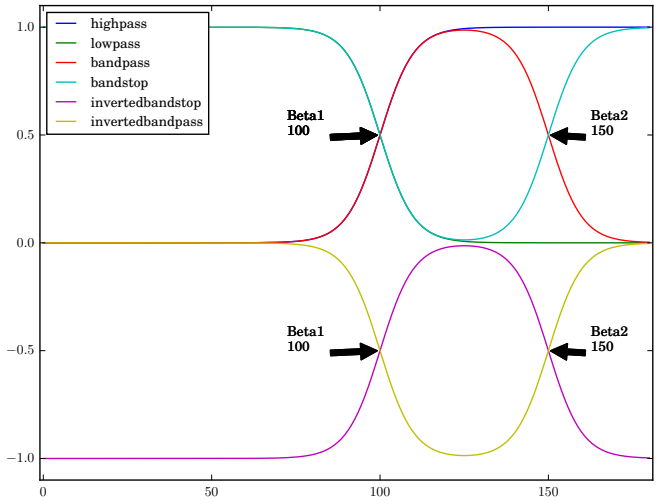
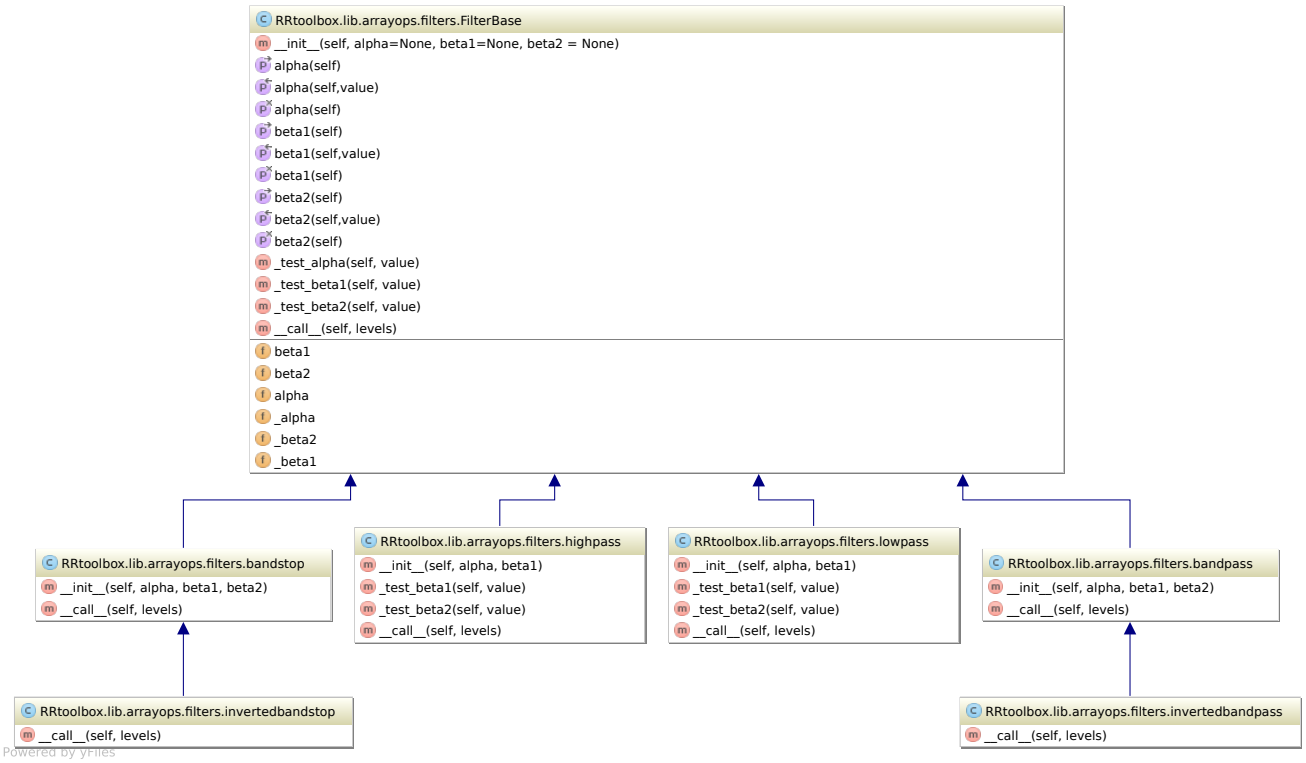


Figure 15. Common filters comparison



All the filters which range are between $[0, 1]$ are called normalized filters and can be derived from $normSigmoid$ or the $high-pass$ and $low-pass$ filters. This inheritance-like behaviour led to the normalized filters being implemented using classes. The class diagrams can be seen in Figure 16.

Figure 16. Filters class diagrams



As the normalized filters suggest the color levels in the image I are saturated when passed through them:

$$filteredI_{saturated} = filter_{normalized}(I)$$

To filter the image I without saturation (only dampened values are saturated to zero) a simple element-wise matrix multiplication can be carried out as follows:

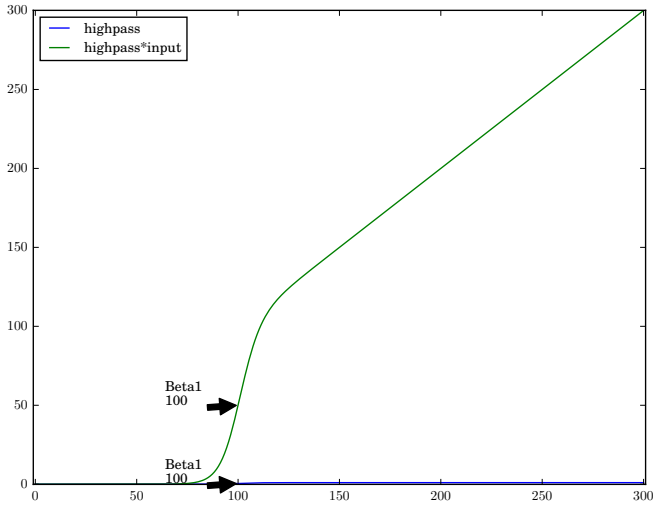
$$filteredI_{un-saturated} = mul(I, filter_{normalized}(I)) = I_{i,j} * filter_{normalized}(I_{i,j})$$

Where the $mul()$ function denotes element-wise multiplication or index by index operation which is different to matrix multiplication in mathematics.

Figure 17 shows an example of the normalized $high-pass$ filter customized to prevent saturation. These filters that are not comprised between 0 and 1 are referred as custom filters.

Sigmoid filtering and saturation.

Figure 17. Custom filtering without saturation



Segmentations

Several segmenting methods were taken into account but only threshold methods were developed for simplicity. The principal idea in segmenting retinal images is to over-thresh and further process them according to some general observations in the experimentation to obtain the final segmentation. Some methods were compared against an expert (i.e. what is considered to be a right threshold) and others to a well known automated threshold method, in this case the Otsu method. This with the intention of examining the algorithm robustness and reliability when applying to an unknown retinal image. Next subsections assume that a general threshold is applied to an image and an specific problem wants to be solved.

Convex hull with line cuts. This method separates an irregular object where two convexity defects are dominant, that is were the distances between the object and the convex hull are more pronounced. This is intended to be applied on objects with irregularities or protuberances where what is wanted are regular shapes like squares, circles and ellipses or objects with few defects.

One way to determine if an object is irregular is to compare its deviation from the convex hull by finding a ratio from their areas (James R. Bozeman & Matthew Pilling, 2015; Volodymyr Kindratenko, 2002). That is what the convexity ratio $r_{convexity}$ does:

$$r_{convexity} = \frac{Area_{object}}{Area_{hull}}$$

where $Area_{object}$ is the area of the object and $Area_{hull}$ is the area of its convex hull.

The pseudo code is as shown in algorithm 1.

Algorithm 1: split mask by defect lines

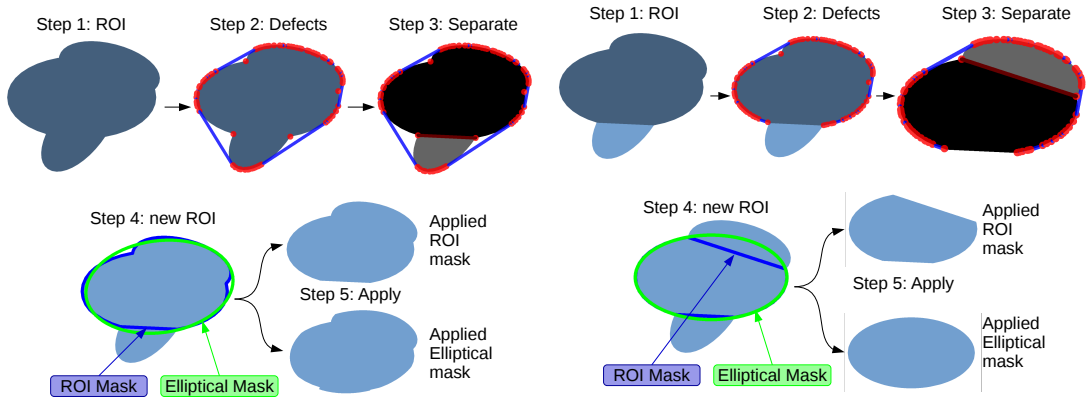
inputs : $bImage$ of object with protuberance or hump
$C_{rdesired}$ criteria to stop iteration
outputs : $bImage_{new}$ of object without protuberance or hump

$contours \leftarrow$ `findContours(`$bImage$`)// list of contours for each object;`
$cnt \leftarrow$ `biggestCnt(`$contours$`)// filter only the contours in order of the biggest object;`
$cnt_{area} \leftarrow$ `cv2.contourArea(`cnt`);`
while *True* **do**
  $hull \leftarrow$ `cv2.convexHull(`cnt`)// get contours of the convex hull;`
  $C_r \leftarrow cnt_{area}$ / `cv2.contourArea(`$hull$`)// calculate the convexity ratio;`
  if $C_r > C_{rdesired}$ **then**
    break;
  end
  $defects \leftarrow$ `cv2.convexityDefects(`cnt, $hull$`)//` $defects$ `contains the distances from the` cnt `and` $hull$ `contours;`
  `// to split` cnt `the two most biggest distances from` $defects$ `needs to be extracted`
  $distances \leftarrow$ `//get` $distances$ `from` $defects$;
  $twoMax \leftarrow$ `// get the two maximum values from` $distances$;
  $twoIndexes \leftarrow$ `// use` $twoMax$ `and` $defects$ `to get the equivalent indexes in` cnt ;
  `// get the minimum and maximum indexes to split` cnt
  $index_{left} \leftarrow$ `min(`$twoIndexes$`)// get the "from" index or left splitter;`
  $index_{right} \leftarrow$ `max(`$twoIndexes$`)// get the "to" index or right splitter;`
  `// side A`
  for $i \leftarrow index_{left}$ **to** $index_{right} - 1$ **do**
    $sideA_i \leftarrow cnt_i$;
  end
  `// side B`
  for $i \leftarrow 0$ **to** $index_{left} - 1$ **do**
    $sideB_i \leftarrow cnt_i$;
  end
  for $i = index_{right}$ **to** `len(`cnt`)`-1 **do**
    $sideB_i \leftarrow cnt_i$;
  end
  `// get biggest side and discard little side which is considered as protuberance or hump`
  $areaA \leftarrow$ `cv2.contourArea(`$sideA$`);`
  $areaB \leftarrow$ `cv2.contourArea(`$sideB$`);`
  if $areaA > areaB$ **then**
    $cnt \leftarrow sideA$;
    $cnt_{area} \leftarrow areaA$;
  else
    $cnt \leftarrow sideB$;
    $cnt_{area} \leftarrow areaB$;
  end
end
$bImage_{new} \leftarrow$ `//from` cnt `contours make a binary image;`
return $[bImage_{new}]$

This principle can be demonstrated using a model object as in Figure 18. The region of interest (ROI) is an irregular object with an obvious round main body and some humps sticking from it (it 1, step 1). These humps form defects with respect to the convex hull of the ROI (it 1, step 2) so the more pronounced the defect is the greater the distance from the ROI and the convex hull is. This way it can be detected where the biggest irregularity is by selecting the two most biggest distances from the ROI and the convex hull and finding their points in the binary image. After this, the main body can be easily segmented by cutting the ROI in two by these two points (it 1, step 3). When the ROI is divided in two it is the choice of user to select the new ROI

but for the sake of automation here and because what is wanted is easy enough as to know which object is bigger then a simple comparison is made to select the main body. As expected the main body is the object with biggest area (it 1, step3, colored in black) and the other (in gray) is a protuberance which is discarded. Once the new ROI is selected it can be used directly as a mask to apply it in the object (it 1, step 5, above), refined it to better match the object (it 1, step 5, below) or it can be further developed by applying the same algorithm in another iteration and sub-segment it until the desired ROI is reached (i.e. one way to stop the iterations is by using a goal with $r_{convexity}$ which approach to 1 the more it approach to its convex hull). Using the new ROI and iterating over it a second time though the same steps eliminates the second hump in the original ROI (it 2, step 4, ROI mask) producing an accumulative effect each time a new iteration is made. Likewise, as stated, the $r_{convexity}$ which was 0.945 in the iteration 1 and now is 0.987 in the iteration 2 is approaching to a ratio of 1 giving a good indication that the object has been successfully approached to its convex hull or a more regular shape. It can be seen that though the ROI mask (it 2, step 4) is edgy and unrefined it can be applied "as is" onto the original object to produce a segmentation without humps (it 2, step 5, above) but if an ellipse is fitted in the ROI mask it produces an Elliptical mask that can segment exactly the main body which is also of a well defined elliptical shape (it 2, step 5, below) inside the original ROI (it 1, step 1). Because the object was ideal the main elliptical body could be segmented precisely without problems producing staggering result for a simple method as it is.

Figure 18. Ideal threshold with defect lines



(a) Iteration 0: C_r = 0.945903192491

(b) Iteration 1: C_r = 0.987807853408

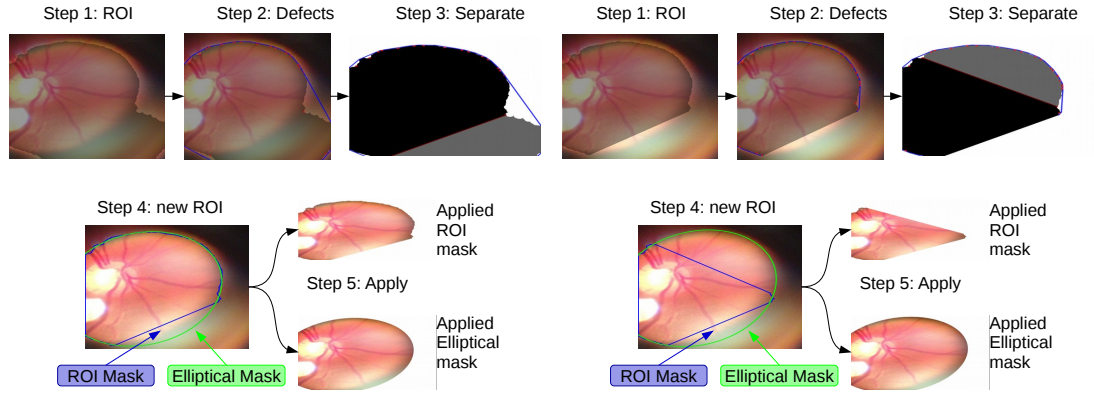
But in actual practice not all the objects are ideal, nonetheless this algorithm works for some of those cases too. Figure 19 Shows an example applying the convex hull with line cuts algorithm in a real case problem. Here the retinal image is over-threshed to give a full segmentation of the retinal area (It 1, step 1), but as it is seen this not only segmented the retinal area but a part affected by flares and noise produced by the bad focus of the camera plus the effects of the flash light. Because the retina is known to have a round shape it can be assumed that anything attached to it as that big protuberance is not part of it but something else. Applying the convex hull (It 1, step 2) and separating the objects by its defects (It 1, step 3) as in the ideal case previously explained gives as expected the main body of the object as shown in (It 1, step 3 and step 4), which a the end can be used to apply the original produced mask onto the over-threshed object to produce a under-threshed object with a better segmentation of the retinal area. But as it is known the retina is not formed by straight lines so a smooth version is produced by fitting an ellipse onto the ROI mask to produce the Elliptical mask which gives a more appealing result (it 1, step 5).

Polygon Test.
Watershed.
Cam and Mean Shift.
Unfinished Threshold methods: blobs, by vector deviation.
Alfa Vs. binary masks.
Mixed method Vs. Otsu threshold.

Object Recognition and Matching algorithms

Entropy.

Figure 19. Practical threshold with defect lines



(a) Iteration 0: $C_r = 0.958214053977$

(b) Iteration 1: $C_r = 0.987951675768$

Histogram comparison.
Histogram matching.
Feature detection.
SIFT.
ASIFT.
Matchers.
Homography.
Using rates and probabilities. By number of descriptors, rectangularity, regularity, inlier tests, entropy test

Post-processing, Masks and Lens Simulation

Circular lens.
Elliptical lens.

Optimization Methods

Process small images and convert results to apply in original images.
Weak references.
Cache.
mmaping.
Memoized functions.
Memoized structures.
Memory managers.

Prototyping

edger.
imtester.
Nodes.
Flowcharts.
Tests Frameworks.
Use of persistence to keep working on previous session data.
Configurations.
Paths.
Custom savings.

Custom caching support.

Implementation

The *RRtoolbox* package has many algorithms in development but it already can create some useful applications. This section presents some of them compiled in a complete software called *imrestore* which is an application to restore images in general but in this case is configured to restore retinal images.

The *imrestore* program was developed using a prototyping method which consisted in building step by step the objectives of the restoration tool. Each finished stage of the program was followed by a exhaustive procedure of testing and debugging techniques using profilers, debugging messages, inspection of variables at each line of code, visualization of the processed images, optimizations (both in CPU performance and memory usage) and any needed corrections which leaded to the repetition of the cycle until no further bugs were found. After a stable stage, only then new features were added consisting of additional options to customize the results as well as the coding of the proceeding steps in the program.

After the script comes the creation of the executable. The executable is a shell interface and a graphical interface can be added on top of it.

This algorithm is far behind with respect to precise segmentation of objects which lack robustness to noises like blurred areas, flares, low brightness, low contrast between actual object and background (there must be a clear distinction between background and foreground where the histogram is bimodal and not multi-modal for better results).

Everything related to the code is hosted in (David, 2016b) which contains the main source code package *RRtoolbox*, a development tool using sequential function charts called *RRtoolFC* (*FC* stands for Function Chart), documentation, tests, examples and the implemented *imrestore.py* script along with some supporting files. *RRtoolbox* documentation file is in (David, 2016a).

Code 2: imrestore code

```
#!/usr/local/bin/python
# -*- coding: utf-8 -*-
'''
imrestore (oriented to retinal images):

Restore images by merging and stitching techniques.

Optimization techniques:
    resize to smaller versions*

    memoization*:
        -persistence
        -serialization and de-serialization
        -caching

    multitasking*:
        -multiprocessing
        -multithreading

    lazy evaluations:
        -load on demand
        -use of weak references

    Memory mapped files*

STEPS:

    (1) Local features: Key-points and descriptors:
        -(1.1) SIFT, SURF, ORB, etc
        -ASIFT*

```

```
    (2) Select main or base image from set for merging:
        -Raw, Sorting, User input

    (3) Matching (spacial):
        -filter 0.7 below Hamming distance
        -key points classification

    (4) selection in matching set: (pre selection of good matches)
        (4.1) Best matches: for general purpose
        (4.2) Entropy: used when set is ensured to be of the same object
            (The program ensures that, if it is not the case).
        (4.3) Histogram comparison: use if set contains unwanted
            perspectives or images that do not correspond to image.
        (4.4) Custom function

    (5) Calculate Homography

    (6) Probability tests: (ensures that the matches images
    correspond to each other)

    (7) Merging
        (7.1) Histogram matching* (color)
        (7.2) Segmentation*
        (7.3) Alpha mask calculation*
        (7.4) Stitching and Merging

    (8) Overall filtering*:
        Bilateral filtering

    (9) Lens simulation for retinal photos*

* optional

Notes:

    Optimization techniques:

        Resize to smaller versions: process smaller versions of the
        inputs and convert the result back to the original versions.
        This reduces processing times, standardize the way data is
        processed (with fixed sizes), lets limited memory to be used,
        allows to apply in big images without breaking down algorithms
        that cannot do that.

        Memoization:
            Persistence: save data to disk for later use.

            serialization and de-serialization: (serialization, in
            python is refereed as pickling) convert live objects into
            a format that can be recorded; (de-serialization, in python
            referred as unpickling) it is used to restore serialized
            data into "live" objects again as if the object was created
            in program conserving its data from precious sessions.

            Caching: saves the result of a function depending or not in
            the inputs to compute data once and keep retrieving it from
            the cached values if asked.
```

```
    Multitasking:
        Multiprocessing: pass tasks to several processes using the
            computer's cores to achieve concurrency.
        Multithreading: pass tasks to threads to use "clock-slicing"
            of a processor to achieve "concurrency".

    Lazy  evaluations:
        load on demand: if data is from an external local file, it is
        loaded only when it is needed to be computed otherwise it is
        deleted from memory or cached in cases where it is extensively
        used. For remotes images (e.g. a server, URL) or in a inadequate
        format, it is downloaded and converted to a numpy format in a
        temporal local place.

        Use of weak references: in cases where the data is cached or
        has not been garbage collected, data is retrieved through
        weak references and if it is needed but has been garbage
        collected it is load again and assigned to the weak reference.

    Memory mapped files:
    Instantiate an object and keep it not in memory but in a file and
    access it directly there. Used when memory is limited or data is
    too big to fit in memory. Slow downs are negligible for read only
    mmaped files (i.e. "r") considering the gain in free memory, but
    it is a real drawback for write operations (i.e. "w","r+","w+").

Selection algorithms:
    Histogram comparison - used to quickly identify the images that
        most resemble a target
    Entropy - used to select the best enfoqued images of the same
        perspective of an object

Local features: Key-points and descriptors:
    ASIFT: used to add a layer of robustness onto other local
    feature methods to cover all affine transformations. ASIFT
    was conceived to complete the invariance to transformations
    offered by SIFT which simulates zoom invariance using gaussian
    blurring and normalizes rotation and translation. This by
    simulating a set of views from the initial image, varying the
    two camera axis orientations: latitude and longitude angles,
    hence its acronym Affine-SIFT. Whereas SIFT stands for Scale
    Invariant Feature Transform.

Matching (spacial):
    Calculate Homography: Used to find the transformation matrix
    to overlay a foreground onto a background image.

Filtering:
    Bilateral filtering: used to filter noise and make the image
    colors more uniform (in some cases more cartoonist-like)

Histogram matching (color): used to approximate the colors from the
    foreground to the background image.

Segmentation: detect and individualize the target objects (e.g. optic
    disk, flares) to further process them or prevent them to be altered.

Alfa mask calculation: It uses Alfa transparency obtained with sigmoid
```

```
        filters and binary masks from the segmentation to specify where an
        algorithm should have more effect or no effect at all
        (i.e. intensity driven).

    Stitching and Merging:
        This is an application point, where all previous algorithms are
        combined to stitch images so as to construct an scenery from the
        parts and merge them if overlapped or even take advantage of these
        to restore images by completing lacking information or enhancing
        poorly illuminated parts in the image. A drawback of this is that
        if not well processed and precise information is given or calculated
        the result could be if not equal worse than the initial images.

    Lens simulation for retinal photos: As its name implies, it is a
        post-processing method applied for better appeal of the image
        depending on the tastes of the user.
'''
from __future__ import division
# TODO install openCV 2.4.12 as described in http://stackoverflow.com/a/37283690/5288758
# to solve the error Process finished with exit code 139
# UPDATE: openCV 2.4.12 does not solve the error Process finished with exit code 139

__author__ = 'Davtoh'
# needed for installing executable
import six
import packaging
import packaging.specifiers
# program imports
import os
import cv2
import warnings
import numpy as np
from time import time
from RRtoolbox.tools.lens import simulateLens
from RRtoolbox.lib.config import MANAGER, FLOAT
from RRtoolbox.lib.image import hist_match
from RRtoolbox.lib.directory import getData, getPath, mkPath, increment_if_exits
from RRtoolbox.lib.cache import memoizedDict
from RRtoolbox.lib.image import loadFunc, imcoors
from RRtoolbox.lib.arrayops.mask import brightness, foreground, thresh_biggestCnt
from multiprocessing.pool import ThreadPool as Pool
from RRtoolbox.tools.selectors import hist_map, hist_comp, entropy
from RRtoolbox.tools.segmentation import retinal_mask
from RRtoolbox.lib.root import TimeCode, glob, lookinglob, profiler
from RRtoolbox.lib.descriptors import Feature, inlineRatio
from RRtoolbox.tools.segmentation import getBrightAlpha, bandpass, bandstop
from RRtoolbox.lib.plotter import matchExplorer, plotim, fastplt
from RRtoolbox.lib.arrayops.filters import getBilateralParameters
from RRtoolbox.lib.arrayops.convert import getSOpointRelation, dict2keyPoint
from RRtoolbox.lib.arrayops.basic import superpose, getTransformedCorners, transformPoint, \
    im2shapeFormat,normalize, getOtsuThresh, contours2mask, pad_to_fit_H, overlay

class VariableNotSettable(Exception):
    pass

class VariableNotDeletable(Exception):
    pass

```

```
206 def check_valid(fn):
207     """
208     checks that a file is valid for loading.
209     :param fn: filename
210     :return: True for valid, False for invalid.
211     """
212     test = os.path.isfile(fn)
213     if test and getData(fn)[-2].startswith("_"):
214         return False
215     return test
216 
217 class ImRestore(object):
218     """
219     Restore images by merging and stitching techniques.
220 
221     :param filenames: list of images or string to path which uses glob filter in path.
222             Loads image array from path, url, server, string
223             or directly from numpy array (supports databases)
224     :param debug: (0) flag to print messages and debug data.
225             0 -> do not print messages.
226             1 -> print normal messages.
227             2 -> print normal and debug messages.
228             3 -> print all messages and show main results.
229                 (consumes significantly more memory).
230             4 -> print all messages and show all stage results.
231                 (consumes significantly more memory).
232             5 -> print all messages, show all results and additional data.
233                 (consumes significantly more memory).
234     :param feature: (None) feature instance. It contains the configured
235             detector and matcher.
236     :param pool: (None) use pool Ex: 4 to use 4 CPUs.
237     :param cachePath: (None) saves memoization to specified path. This is
238             useful to save some computations and use them in next executions.
239             If True it creates the cache in current path.
240 
241             .. warning:: Cached data is not guaranteed to work between different
242                         configurations and this can lead to unexpected program
243                         behaviour. If a different configuration will be used it
244                         is recommended to clear the cache to recompute values.
245     :param clearCache: (0) clear cache flag.
246             * 0 do not clear.
247             * 1 re-compute data but other cache data is left intact.
248             * 2 All CachePath is cleared before use.
249             Notes: using cache can result in unexpected behaviour
250                 if some configurations does not match to the cached data.
251     :param loader: (None) custom loader function used to load images.
252             If None it loads the original images in color.
253     :param process_shape: (400,400) process shape, used to load pseudo images
254             to process features and then results are converted to the
255             original images. The smaller the image more memory and speed gain
256             If None it loads the original images to process the features but it
257             can incur to performance penalties if images are too big and RAM
258             memory is scarce.
259     :param load_shape: (None) custom shape used load images which are being merged.
260     :param baseImage: (None) First image to merge to.
261             * None -> takes first image from raw list.
262             * True -> selects image with most features.
263             * Image Name.
```

```
    :param selectMethod: (None) Method to sort images when matching. This
            way the merging order can be controlled.
            * (None) Best matches.
            * Histogram Comparison: Correlation, Chi-squared,
                Intersection, Hellinger or any method found in hist_map
            * Entropy.
            * custom function of the form: rating,fn <-- selectMethod(fns)
    :param distanceThresh: (0.75) filter matches by distance ratio.
    :param inlineThresh: (0.2) filter homography by inlineratio.
    :param rectangularityThresh: (0.5) filter homography by rectangularity.
    :param ransacReprojThreshold: (5.0) maximum allowed reprojection error
            to treat a point pair as an inlier.
    :param centric: (False) tries to attach as many images as possible to
            each matching. It is quicker since it does not have to process
            too many match computations.
    :param hist_match: (False) apply histogram matching to foreground
            image with merge image as template
    :param grow_scene: If True, allow the restored image to grow in shape if
            necessary at the merging process.
    :param expert: Path to an expert database. If provided it will use this data
            to generate the mask used when merging to the restored image.
    :param maskforeground:(False)
            * True, limit features area using foreground mask of input images.
                This mask is calculated to threshold a well defined object.
            * Callable, Custom function to produce the foreground image which
                receives the input gray image and must return the mask image
                where the keypoints will be processed.
    :param noisefunc: True to process noisy images or provide function.
    :param save: (False)
            * True, saves in path with name _restored_{base_image}
            * False, does not save
            * Image name used to save the restored image.
    :param overwrite: If True and the destine filename for saving already
            exists then it is replaced, else a new filename is generated
            with an index "{filename}_{index}.{extension}"
    """

    def __init__(self, filenames, **opts):
        self.profiler = opts.get("profiler",None)
        if self.profiler is None:
            self.profiler = profiler("ImRestore init")

        self.log_saved = [] # keeps track of last saved file.

        # for debug
        self.verbosity = opts.get("verbosity", 1)

        ################################## GET IMAGES ###################################
        if filenames is None or len(filenames)==0: # if images is empty use demonstration
            #test = MANAGER["TESTPATH"]
            #if self.verbosity: print "Looking in DEMO path {}".format(test)
            #fns = glob(test + "*",check=check_valid)
            raise Exception("List of filenames is Empty")
        elif isinstance(filenames, basestring):
            # if string assume it is a path
            if self.verbosity: print "Looking as {}".format(filenames)
            fns = glob(filenames,check=check_valid)
        elif not isinstance(filenames, basestring) and \
```

```
                    len(filenames) == 1 and "*" in filenames[0]:
                filenames = filenames[0] # get string
                if self.verbosity: print "Looking as {}".format(filenames)
                fns = glob(filenames,check=check_valid)
            else: # iterator containing data
                fns = filenames # list file names

            # check images
            if not len(fns)>1:
                raise Exception("list of images must be "
                                "greater than 1, got {}".format(len(fns)))

            self.filenames = fns

            # for multiprocessing
            self.pool = opts.get("pool",None)
            if self.pool is not None: # convert pool count to pool class
                NO_CPU = cv2.getNumberOfCPUs()
                if self.pool <= NO_CPU:
                    self.pool = Pool(processes = self.pool)
                else:
                    raise Exception("pool of {} exceeds the "
                                    "number of processors {}".format(self.pool,NO_CPU))
            # for features
            self.feature = opts.get("feature",None)
            # init detector and matcher to compute descriptors
            if self.feature is None:
                self.feature = Feature(pool=self.pool, debug=self.verbosity)
                self.feature.config(name='a-sift-flann')
            else:
                self.feature.pool = self.pool
                self.feature.debug = self.verbosity

            # select method to order images to feed in superposition
            self.selectMethod = opts.get("selectMethod",None)
            best_match_list = ("bestmatches", "best matches")
            entropy_list = ("entropy",)
            if callable(self.selectMethod):
                self._orderValue = 3
            elif self.selectMethod in hist_map:
                self._orderValue = 2
            elif self.selectMethod in entropy_list:
                self._orderValue = 1
            elif self.selectMethod in best_match_list or self.selectMethod is None:
                self._orderValue = 0
            else:
                raise Exception("selectMethod {} not recognized".format(self.selectMethod))

            # distance threshold to filter best matches
            self.distanceThresh = opts.get("distanceThresh",0.75) # filter ratio

            # threshold for inlineRatio
            self.inlineThresh = opts.get("inlineThresh",0.2) # filter ratio
            # ensures adequate value [0,1]
            assert self.inlineThresh<=1 and self.inlineThresh>=0

            # threshold for rectangularity
            self.rectangularityThresh = opts.get("rectangularityThresh",0.5) # filter ratio
```

```
        # ensures adequate value [0,1]
        assert self.rectangularityThresh<=1 and self.rectangularityThresh>=0

        # threshold to for RANSAC reprojection
        self.ransacReprojThreshold = opts.get("ransacReprojThreshold",5.0)

        self.centric = opts.get("centric",False) # tries to attach as many images as possible
        # it is not memory efficient to compute descriptors from big images
        self.process_shape = opts.get("process_shape", (400, 400)) # use processing shape
        self.load_shape = opts.get("load_shape", None) # shape to load images for merging
        self.minKps = 3 # minimum len of key-points to find Homography
        self.histMatch = opts.get("hist_match",False)
        self.denoise=opts.get("denoise", None)

        ############################## OPTIMIZATION MEMOIZEDIC ############################
        self.cachePath = opts.get("cachePath",None)
        if self.cachePath is not None:
            if self.cachePath is True:
                self.cachePath = os.path.abspath(".") # MANAGER["TEMPPATH"]
            if self.cachePath == "{temp}":
                self.cachePath = self.cachePath.format(temp=MANAGER["TEMPPATH"])
            self.feature_dic = memoizedDict(os.path.join(self.cachePath,"descriptors"))
            if self.verbosity: print "Cache path is in {}".format(self.feature_dic._path)
            self.clearCache = opts.get("clearCache",0)
            if self.clearCache==2:
                self.feature_dic.clear()
                if self.verbosity: print "Cache path cleared"
        else:
            self.feature_dic = {}

        self.expert = opts.get("expert",None)
        if self.expert is not None:
            self.expert = memoizedDict(self.expert) # convert path

        # to select base image ahead of any process
        baseImage = opts.get("baseImage",None)
        if isinstance(baseImage,basestring):
            base_old = baseImage
            try: # tries user input
                if baseImage not in fns:
                    base, path, name, ext = getData(baseImage)
                    if not path: # if name is incomplete look for it
                        base, path, _,_ = getData(fns[0])
                    baseImage = lookinglob(baseImage, "".join((base, path)))
                    # selected image must be in fns
                    if baseImage is None:
                        raise IndexError
            except IndexError: # tries to find image based in user input
                # generate informative error for the user
                raise Exception("{} or {} is not in image list"
                                "\n A pattern is {}".format(
                                base_old,baseImage,fns[0]))

        self.baseImage = baseImage

        if self.verbosity: print "No. images {}...".format(len(fns))

        # make loader
```

```
        self.loader = opts.get("loader",None) # BGR loader
        if self.loader is None: self.loader = loadFunc(1)
        self._loader_cache = None # keeps last image reference
        self._loader_params = None # keeps last track of last loading options to reload

        self.save = opts.get("save",False)
        self.grow_scene = opts.get("grow_scene",True)
        self.maskforeground = opts.get("maskforeground",False)
        self.overwrite = opts.get("overwrite",False)

        # processing variables
        self._feature_list = None

    @property
    def denoise(self):
        return self._noisefunc
    @denoise.setter
    def denoise(self, value):
        if value is False:
            value = None
        if value is True:
            value = "mild"
        if value in ("mild","heavy","normal",None) or callable(value):
            self._noisefunc = value
        else:
            raise Exception("denoise '{}' not recognised".format(value))
    @denoise.deleter
    def denoise(self):
        del self._noisefunc

    @property
    def feature_list(self):
        if self._feature_list is None:
            return self.compute_keypoints()
        return self._feature_list
    @feature_list.setter
    def feature_list(self,value):
        raise VariableNotSettable("feature_list is not settable")
    @feature_list.deleter
    def feature_list(self):
        self._feature_list = None

    def loadImage(self, path=None, shape=None):
        """
        load image from source

        :param path: filename, url, .npy, server, image in string
        :param shape: shape to convert image
        :return: BGR image
        """
        params = (path, shape)
        if self._loader_cache is None or params != self._loader_params:
            # load new image and cache it
            img = self.loader(path) # load image
            if shape is not None:
                img = cv2.resize(img,shape)
            self._loader_cache = img # this keeps a reference
            self._loader_params = params
```

```
            return img
        else: # return cached image
            return self._loader_cache

    def compute_keypoints(self):
        """
        computes key-points from file names.

        :return: self.feature_list
        """
        ################### Local features: Key-points and descriptors #################
        fns = self.filenames
        def getMask(img):
            """
            helper function to get mask
            :param img: gray image
            :return:
            """
            if callable(self.maskforeground):
                mask = self.maskforeground(img)
            if self.maskforeground is True:
                mask = foreground(img)

            if self.verbosity > 4:
                fastplt(overlay(img.copy(),mask*255,alpha=mask*0.5),block=True,
                        title="{} mask to detect features".format(getData(path)[-2]))
            return mask

        with TimeCode("Computing features...\n", profiler=self.profiler,
                      profile_point=("Computing features",),
                      endmsg="Computed feature time was {time}\n",
                      enableMsg=self.verbosity) as timerK:

            self._feature_list = [] # list of key points and descriptors
            for index,path in enumerate(fns):
                img = self.loadImage(path, self.load_shape)
                lshape = img.shape[:2]
                try:
                    point = profiler(msg=path,tag="cached")
                    if self.cachePath is None or self.clearCache==1 \
                            and path in self.feature_dic:
                        raise KeyError # clears entry from cache
                    kps, desc, pshape = self.feature_dic[path] # thread safe
                    if pshape is None:
                        raise ValueError
                except (KeyError, ValueError) as e: # not memorized
                    point = profiler(msg=path,tag="processed")
                    if self.verbosity: print "Processing features for {}...".format(path)
                    if lshape != self.process_shape:
                        img = cv2.resize(img, self.process_shape)
                    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                    # get features
                    kps, desc = self.feature.detectAndCompute(img, getMask(img))
                    pshape = img.shape[:2] # get process shape
                    # to memoize
                    self.feature_dic[path] = kps, desc, pshape

                # re-scale keypoints to original image
```

```
554                 if lshape != pshape:
555                     # this necessarily does not produce the same result
556                     """
557                     # METHOD 1: using Transformation Matrix
558                     H = getSOpointRelation(process_shape, lshape, True)
559                     for kp in kps:
560                         kp["pt"]=tuple(cv2.perspectiveTransform(
561                             np.array([[kp["pt"]]]), H).reshape(-1, 2)[0])
562                     """
563                     # METHOD 2:
564                     rx,ry = getSOpointRelation(pshape, lshape)
565                     for kp in kps:
566                         x,y = kp["pt"]
567                         kp["pt"] = x*rx,y*ry
568                         kp["path"] = path
569                 else:
570                     for kp in kps: # be very carful, this should not appear in self.feature_dic
571                         kp["path"] = path # add paths to key-points
572 
573                 # number of key-points, index, path, key-points, descriptors
574                 self._feature_list.append((len(kps),index,path,kps,desc))
575                 if self.verbosity: print "\rFeatures {}/{}...".format(index + 1, len(fns)),
576                 # for profiling individual processing times
577                 if self.profiler is not None: self.profiler._close_point(point)
578 
579         return self._feature_list
580 
581     def restore(self):
582         """
583         Restore using file names (self.file_names) with
584         base image (self.baseImage) and configurations.
585 
586         :return: self.restored
587         """
588 
589         fns = self.filenames # in this process fns should not be changed
590         ############################ Pre-selection from a set ##############################
591         baseImage = self.baseImage # baseImage option should not be update
592         # initialization and base image selection
593         if baseImage is None: # select first image as baseImage
594            _,_,baseImage,kps_base,desc_base = self.feature_list[0]
595         elif isinstance(baseImage,basestring):
596             kps_base,desc_base = self.feature_dic[baseImage]
597         elif baseImage is True: # sort images
598             self.feature_list.sort(reverse=True) # descendant: from bigger to least
599             # select first for most probable
600             _,_,baseImage,kps_base,desc_base = self.feature_list[0]
601         else:
602             raise Exception("baseImage must be None, True or String")
603 
604         if self.verbosity: print "baseImage is", baseImage
605         self.used = [baseImage] # select first image path
606         # load first image for merged image
607         self.restored = self.loadImage(baseImage, self.load_shape)
608         self.failed = [] # registry for failed images
609 
610         ########################## Order set initialization ############################
611         if self._orderValue: # obtain comparison with structure (value, path)
```

```
            if self._orderValue == 1: # entropy
                comparison = zip(*entropy(fns, loadfunc=loadFunc(1, self.process_shape),
                                          invert=False)[:2])
                if self.verbosity: print "Configured to sort by entropy..."
            elif self._orderValue == 2: # histogram comparison
                comparison = hist_comp(fns, loadfunc=loadFunc(1, self.process_shape),
                                       method=self.selectMethod)
                if self.verbosity:
                    print "Configured to sort by {}...".format(self.selectMethod)
            elif self._orderValue == 3:
                comparison = self.selectMethod(fns)
                if self.verbosity: print "Configured to sort by Custom Function..."
            else:
                raise Exception("DEBUG: orderValue {} does "
                    "not correspond to {}".format(self._orderValue,self.selectMethod))
        elif self.verbosity: print "Configured to sort by best matches"

        with TimeCode("Restoring ...\n",profiler=self.profiler,
                      profile_point=("Restoring",),
                      endmsg= "Restoring overall time was {time}\n",
                      enableMsg= self.verbosity) as timerR:

            while True:
                with TimeCode("Matching ...\n",profiler=self.profiler,
                              profile_point=("Matching",),
                              endmsg= "Matching overall time was {time}\n",
                              enableMsg= self.verbosity) as timerM:
                    ################### remaining keypoints to match ####################
                    # initialize key-point and descriptor base list
                    kps_remain,desc_remain = [],[]
                    for _,_,path,kps,desc in self.feature_list:
                        # append only those which are not in the base image
                        if path not in self.used:
                            kps_remain.extend(kps)
                            desc_remain.extend(desc)

                    if not kps_remain: # if there is not image remaining to stitch break
                        if self.verbosity: print "All images used"
                        break

                    desc_remain = np.array(desc_remain) # convert descriptors to array

                    ############################ Matching ################################
                    # select only those with good distance (hamming, L1, L2)
                    raw_matches = self.feature.matcher.knnMatch(desc_remain,
                                    trainDescriptors = desc_base, k = 2) #2
                    # If path=2, it will draw two match-lines for each key-point.
                    classified = {}
                    for m in raw_matches:
                        # filter by Hamming, L1 or L2 distance
                        if m[0].distance < m[1].distance * self.distanceThresh:
                            m = m[0]
                            kp1 = kps_remain[m.queryIdx]  # keypoint in query image
                            kp2 = kps_base[m.trainIdx]  # keypoint in train image

                            key = kp1["path"] # ensured that key is not in used
                            if key in classified:
                                classified[key].append((kp1,kp2))
```

```
                else:
                    classified[key] = [(kp1,kp2)]

        ######################### Order set ###############################
        # use only those in classified of histogram or entropy comparison
        if self._orderValue:
            ordered = [(val,path) for val,path
                       in comparison if path in classified]
        else: # order with best matches
            ordered = sorted([(len(kps),path)
                        for path,kps in classified.items()],reverse=True)


    with TimeCode("Merging ...\n",profiler=self.profiler,
                  profile_point=("Merging",),
                  endmsg= "Merging overall time was {time}\n",
                  enableMsg= self.verbosity) as timerH:

        # feed key-points in order according to order set
        for rank, path in ordered:
            point = profiler(msg=path) # profiling point
            ######################### Calculate Homography ##################
            mkp1,mkp2 = zip(*classified[path]) # probably good matches
            if len(mkp1)>self.minKps and len(mkp2)>self.minKps:

                # get only key-points
                p1 = np.float32([kp["pt"] for kp in mkp1])
                p2 = np.float32([kp["pt"] for kp in mkp2])
                if self.verbosity > 4:
                    print 'Calculating Homography for {}...'.format(path)

                # Calculate homography of fore over back
                H, status = cv2.findHomography(p1, p2,
                            cv2.RANSAC, self.ransacReprojThreshold)
            else: # not sufficient key-points
                if self.verbosity > 1:
                    print 'Not enough key-points for {}...'.format(path)
                H = None

            # test that there is homography
            if H is not None: # first test
                # load fore image
                fore = self.loadImage(path, self.load_shape)
                h,w = fore.shape[:2] # image shape

                # get corners of fore projection over back
                projection = getTransformedCorners((h,w),H)
                c = imcoors(projection) # class to calculate statistical data
                lines, inlines = len(status), np.sum(status)

                # ratio to determine how good fore is in back
                inlineratio = inlineRatio(inlines,lines)

                Test = inlineratio>self.inlineThresh \
                        and c.rotatedRectangularity>self.rectangularityThresh

                text = "inlines/lines: {}/{}={}, " \
                       "rectangularity: {}, test: {}".format(
```

```
                            inlines, lines, inlineratio, c.rotatedRectangularity,
                            ("failed","succeeded")[Test])

                        if self.verbosity>1: print text

                        if self.verbosity > 3: # show matches
                            matchExplorer("Match " + text, fore,
                                          self.restored, classified[path], status, H)

                        ######################## probability test ####################
                        if Test: # second test

                            if self.verbosity>1: print "Test succeeded..."
                            while path in self.failed: # clean path in fail registry
                                try: # race-conditions safe
                                    self.failed.remove(path)
                                except ValueError:
                                    pass

                            ################### merging and stitching ################
                            self.merge(path, H)

                            # used for profiling
                            if self.profiler is not None:
                                self.profiler._close_point(point)

                            if not self.centric:
                                break
                        else:
                            self.failed.append(path)
                    else:
                        self.failed.append(path)

            # if all classified have failed then end
            if set(classified.keys()) == set(self.failed):
                if self.verbosity:
                    print "Restoration finished, these images do not fit: "
                    for index in classified.keys():
                        print index
                break

        with TimeCode("Post-processing ...\n",profiler=self.profiler,
                      profile_point=("Post-processing",),
                      endmsg= "Post-processing overall time was {time}\n",
                      enableMsg= self.verbosity) as timerP:
            processed = self.post_process_restoration(self.restored)
            if processed is not None:
                self.restored = processed

        # profiling post-processing
        self.time_postprocessing = timerP.time_end

        ################################### Save image ###################################
        if self.save:
            self.saveImage()

        return self.restored # return merged image
```

```
786     def saveImage(self, path = None, overwrite = None):
787         """
788         save restored image in path.
789 
790         :param path: filename, string to format or path to save image.
791                 if path is not a string it would be replaced with the string
792                 "{path}restored_{name}{ext}" to format with the formatting
793                 "{path}", "{name}" and "{ext}" from the baseImage variable.
794         :param overwrite: If True and the destine filename for saving already
795             exists then it is replaced, else a new filename is generated
796             with an index "{filename}_{index}.{extension}"
797         :return: saved path, status (True for success and False for fail)
798         """
799         if path is None:
800             path = self.save
801         if overwrite is None:
802             overwrite = self.overwrite
803 
804         bbase, bpath, bname, bext = getData(self.used[0])
805         if isinstance(path,basestring):
806             # format path if user has specified so
807             data = getData(self.save.format(path="".join((bbase, bpath)),
808                                             name=bname, ext=bext))
809             # complete any data lacking in path
810             for i,(n,b) in enumerate(zip(data,(bbase, bpath, bname, bext))):
811                 if not n: data[i] = b
812         else:
813             data = bbase,bpath,"_restored_",bname,bext
814         # joint parts to get string
815         fn = "".join(data)
816         mkPath(getPath(fn))
817 
818         if not overwrite:
819             fn = increment_if_exits(fn)
820 
821         if cv2.imwrite(fn,self.restored):
822             if self.verbosity: print "Saved: {}".format(fn)
823             self.log_saved.append(fn)
824             return fn, True
825         else:
826             if self.verbosity: print "{} could not be saved".format(fn)
827             return fn, False
828 
829     def merge(self, path, H, shape = None):
830         """
831         merge image to main restored image.
832 
833         :param path: file name to load image
834         :param H: Transformation matrix of image in path over restored image.
835         :param shape: custom shape to load image in path
836         :return: self.restored
837         """
838         if shape is None:
839             shape = self.load_shape
840 
841         fore = self.loadImage(path,shape) # load fore image
842 
843         if self.histMatch: # apply histogram matching
```

```
            fore = hist_match(fore, self.restored)

        if self.verbosity > 1: print "Merging..."

        # pre process alpha mask
        alpha = self.pre_process_fore_Mask(self.restored,fore,H)

        # process expert alpha mask if alpha was not provided by the user
        if alpha is None and self.expert is not None:

            if not hasattr(self,"_restored_mask"):

                bname = "".join(getData(self.used[-1])[-2:])
                try:
                    bdata = self.expert[bname]
                except KeyError:
                    raise KeyError("{} is not in self.expert".format(bname))

                bsh = bdata["shape"]
                bm_retina = contours2mask(bdata["coors_retina"],bsh)
                bm_otic_disc = contours2mask(bdata["coors_optic_disc"],bsh)
                bm_defects = contours2mask(bdata["coors_defects"],bsh)

                self._restored_mask = np.logical_and(np.logical_or(np.logical_not(bm_retina),
                                            bm_defects), np.logical_not(bm_otic_disc))

            fname = "".join(getData(path)[-2:])
            try:
                fdata = self.expert[fname]
            except KeyError:
                raise KeyError("{} is not in self.expert".format(fname))

            fsh = fdata["shape"]
            fm_retina = contours2mask(fdata["coors_retina"],fsh)
            #fm_otic_disc = contours2mask(fdata["coors_optic_disc"],fsh)
            fm_defects = contours2mask(fdata["coors_defects"],fsh)

            fmask = np.logical_and(fm_retina,np.logical_not(fm_defects))

            self._restored_mask = maskm = np.logical_and(self._restored_mask,fmask)

            h, w = self.restored.shape[:2]
            alpha = cv2.warpPerspective(maskm.copy().astype(np.uint8), H, (w, h))

        ################################## SUPERPOSE ###################################

        # fore on top of back
        alpha_shape = fore.shape[:2]
        if self.grow_scene: # this makes the images bigger if possible
            # fore(x,y)*H = fore(u,v) -> fore(u,v) + back(u,v)
            ((left,top),(right,bottom)) = pad_to_fit_H(fore.shape, self.restored.shape, H)
             # moved transformation matrix with pad
            H_back = FLOAT([[1,0,left],[0,1,top],[0,0,1]]) # in back
            H_fore = H_back.dot(H) # in fore
            # need: top_left, bottom_left, top_right,bottom_right
            h2,w2 = self.restored.shape[:2]
            w,h = int(left + right + w2),int(top + bottom + h2)
            # this memory inefficient, image is copied to prevent cross-references
```

```
            self.restored = cv2.warpPerspective(self.restored.copy(), H_back, (w, h))
            fore = cv2.warpPerspective(fore.copy(), H_fore, (w, h))
        else: # this keeps back shape
            H_fore = H
            H_back = np.eye(3)
            h, w = self.restored.shape[:2]
            fore = cv2.warpPerspective(fore.copy(), H_fore, (w, h))

        if alpha is None: # if no pre-processing function for alpha implemented
            alpha = self.post_process_fore_Mask(self.restored,fore)

        if alpha is None: # create valid mask for stitching
            alpha = cv2.warpPerspective(np.ones(alpha_shape), H_fore, (w, h))

        if self.verbosity > 3: # show merging result
            fastplt(alpha, title="alpha mask from {}".format(path),block=True)

        self.restored = overlay(self.restored, fore, alpha) # overlay fore on top of back

        if H_fore is None: # H is not modified use itself
            H_fore = H

        if self.verbosity > 4: # show merging result
            plotim("Last added from {}".format(path), self.restored).show()

        ######################## update base features ######################
        # make projection to test key-points inside it
        if self.verbosity > 1: print "Updating key-points..."
        # fore projection in restored image
        projection = getTransformedCorners(fore.shape[:2],H_fore)
        # update key-points
        newkps, newdesc = [], []
        for _,_,p,kps,desc in self.feature_list:
            # append all points in the base image and update their position
            if p in self.used: # transform points in back
                for kp,dsc in zip(kps,desc): # kps,desc
                    pt = kp["pt"] # get point
                    if H_back is not None: # update point
                        pt = tuple(transformPoint(pt,H_back))
                        kp["pt"] = pt
                    # include only those points outside foreground
                    if cv2.pointPolygonTest(projection, pt, False) == -1:
                        newkps.append(kp)
                        newdesc.append(dsc)
            elif p == path: # transform points in fore
                # include only those points inside foreground
                for kp,dsc in zip(kps,desc): # kps,desc
                    kp["pt"] = tuple(transformPoint(kp["pt"],H_fore))
                    newkps.append(kp)
                    newdesc.append(dsc)
        # update kps_base and desc_base
        kps_base = newkps
        desc_base = np.array(newdesc)

        if self.verbosity > 4: # show keypints in merging
            plotim("merged Key-points", # draw key-points in image
                   cv2.drawKeypoints(
                       im2shapeFormat(self.restored,self.restored.shape[:2]+(3,)),
```

```
                                [dict2keyPoint(index) for index in kps_base],
                                flags=4, color=(0,0,255))).show()
            if self.verbosity: print "This image has been merged: {}...".format(path)
            self.used.append(path) # update used

            return self.restored

        def post_process_restoration(self, image):
            """
            Post-process a merged retinal image.

            :param image: retinal image
            :return: filtered and with simulated lens
            """
            if callable(self.denoise):
                return self.denoise(image)
            # detect how much noise to process and convert it to beta parameters
            if self.denoise is not None:
                # slower but interactive for heavy noise
                # filter using parameters and bilateral filter
                params = getBilateralParameters(image.shape, self.denoise)
                return cv2.bilateralFilter(image, *params)

        def post_process_fore_Mask(self, back, fore):
            pass

        def pre_process_fore_Mask(self, back, fore, H):
            pass

    class RetinalRestore(ImRestore):
        """
        Restore retinal images by merging and stitching techniques. These parameters are
        added to :class:`ImRestore`:

        :param lens: flag to determine if lens are applied. True
            to simulate lens, False to not apply lens.
        :param enclose: flag to enclose and return only retinal area.
            True to return ROI, false to leave image "as is".
        """
        def __init__(self, filenames, **opts):
            super(RetinalRestore,self).__init__(filenames, **opts)
            self.maskforeground = opts.get("maskforeground",lambda img: retinal_mask(img,True))
            self.denoise=opts.get("denoise", True)
            self.lens = opts.get("lens",False)
            self.enclose = opts.get("enclose",False)

        #__init__.__doc__ = ImRestore.__init__.__doc__+__init__.__doc__

        def post_process_fore_Mask(self, back, fore):
            """
            Function evaluated just after superposition and before overlay.

            :param back: background image
            :param fore: foreground image
            :return: alpha mask
            """

            def _mask(back,fore):
```

```
            """
            get bright alpha mask (using histogram method)

            :param back: BGR background image
            :param fore: BGR foreground image
            :return: alpha mask
            """
            # TODO, not working for every retinal scenario
            foregray = brightness(fore)
            # get window with Otsu to prevent expansion
            thresh,w = cv2.threshold(foregray,0,1,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
            return getBrightAlpha(brightness(back).astype(float),
                                  foregray.astype(float), window=w)

        def get_beta_params(P):
            """
            Automatically find parameters for bright alpha masks.

            :param P: gray image
            :return: beta1,beta2
            """
            # process histogram for uint8 gray image
            hist, bins = np.histogram(P.flatten(), 256)

            # get Otsu thresh as beta2
            beta2 = bins[getOtsuThresh(hist)]
            return np.min(P), beta2 # beta1, beta2

        def mask(back, fore):
            """
            Get bright alpha mask (using Otsu method)

            :param back: BGR background image
            :param fore: BGR foreground image
            :return: alpha mask
            """
            # find retinal area and its alpha
            mask_back, alpha_back = retinal_mask(back,biggest=True,addalpha=True)
            mask_fore, alpha_fore = retinal_mask(fore,biggest=True,addalpha=True)

            # convert uint8 to float
            backgray = brightness(back).astype(float)
            foregray = brightness(fore).astype(float)

            # scale from 0-1 to 0-255
            backm = alpha_back*255
            forem = alpha_fore*255

            # get alpha masks fro background and foreground
            backmask = bandstop(3, *get_beta_params(backm[mask_back.astype(bool)]))(backgray)
            foremask = bandpass(3, *get_beta_params(forem[mask_fore.astype(bool)]))(foregray)

            # merge masks
            alphamask = normalize(foremask * backmask * (backm/255.))
            return alphamask


        pshape = (400,400) # process shape
```

```
        # rescaling of the image to process mask
        if pshape is not None:
            oshape = back.shape[:2]
            back = cv2.resize(back,pshape)
            fore = cv2.resize(fore,pshape)

        # get alpha mask
        alphamask = mask(back,fore)

        # rescaling mask to original shape
        if pshape is not None:
            alphamask = cv2.resize(alphamask,oshape[::-1])

        return alphamask

    def post_process_restoration(self, image):
        """
        Post-process a merged retinal image.

        :param image: retinal image
        :return: filtered and with simulated lens
        """
        image_ = super(RetinalRestore,self).post_process_restoration(image)
        if image_ is not None:
            image = image_

        # simulation of lens
        if self.lens:
            # call function to overlay lens on image
            try:
                image = simulateLens(image)
            except Exception as e:
                warnings.warn("simulate lens failed with error {}".format(e))

        # crop retinal area only
        if self.enclose:
            # convert to gray
            gray = brightness(image)
            # get object mask
            mask = foreground(gray)
            # get contour of biggest area
            cnt = thresh_biggestCnt(mask)
            # get enclosure box
            x,y,w,h = cv2.boundingRect(cnt)
            # crop image
            if len(image.shape)>2:
                image = image[x:x+w,y:y+h,:]
            else:
                image = image[x:x+w,y:y+h]

        return image

def feature_creator(string):
    """
    Converts a string to a feature object.

    :param string: any supported feature detector in openCV. the format is
            "[a-]<sift|surf|orb>[-flann]" (str) Ex: "a-sift-flann" where
```

```
            "a-" or "-flann" are optional.
    :return: feature object
    """
    return Feature().config(string)

def tuple_creator(string):
    """
    Process string to get tuple.

    :param string: string parameters with "," (colon) as separator
            Ex: param1,param2,...,paramN
    :return: tuple
    """
    tp = []
    func = string_interpreter()
    for i in string.split(","):
        try:
            tp.append(func(i))
        except:
            tp.append(i)
    return tuple(tp)

def loader_creator(string):
    """
    creates an image loader.

    :param string: flag, x size, y size. Ex 1: "0,100,100" loads gray images of shape
            (100,100) in gray scale. Ex 2: "1" loads images in BGR color and with
            original shapes. Ex 3: "0,200,None" loads gray images of shape (200,None)
            where None is calculated to keep image ratio.
    :return: loader
    """
    params = tuple_creator(string)
    try:
        flag = params[0]
    except:
        flag=1
    try:
        x = params[1]
    except:
        x=None
    try:
        y = params[2]
    except:
        y=None
    return loadFunc(flag,(x,y))

def denoise_creator(string):
    """
    creates an function to de-noise images using bilateral filter.

    :param string: d, sigmaColor, sigmaSpace. Ex: 27,75,75 creates the
            filter to de-noise images.
    :return: denoiser
    """
    d, sigmaColor, sigmaSpace = tuple_creator(string)
    def denoiser(image):
        return cv2.bilateralFilter(image, d, sigmaColor, sigmaSpace)
```

```
    return denoiser

def string_interpreter(empty=None, commahandler=None, handle=None):
    """
    create a string interpreter
    :param empty: (None) variable to handle empty strings
    :param commahandler: (tuple_creator) function to handle comma separated strings
    :return: interpreter function
    """
    def interprete_string(string):
        if string == "":
            return empty
        if "," in string:
            if commahandler is None:
                return tuple_creator(string)
            else:
                return commahandler(string)
        if string.lower() == "none":
            return None
        if string.lower() == "true":
            return True
        if string.lower() == "false":
            return False
        if handle is None:
            try:
                return int(string)
            except:
                return string
        else:
            return handle(string)
    interprete_string.__doc__="""
        Interpret strings.

        :param string: string to interpret.
        :return: interpreted string. If empty string (i.e. '') it returns {}.
                If 'None' returns None. If 'True' returns True. If 'False' returns False.
                If comma separated it applies {} else applies {}.
        """.format(empty, commahandler, handle)
    return interprete_string

class NameSpace(object):
    """
    used to store variables
    """

def shell(args=None, namespace=None):
    """
    Shell to run in terminal the imrestore program

    :param args: (None) list of arguments. If None it captures the
            arguments in sys.
    :param namespace: (None) namespace to place variables. If None
            it creates a namespace.
    :return: namespace
    """
    if namespace is None:
        namespace = NameSpace()
    import argparse
```

```
    parser = argparse.ArgumentParser(formatter_class=argparse.RawDescriptionHelpFormatter,
                                     description="Restore images by merging and stitching "
                                                 "techniques.",
                                     epilog=__doc__ +
                                            "\nContributions and bug reports are appreciated."
                                            "\nauthor: David Toro"
                                            "\ne-mail: davsamirtor@gmail.com"
                                            "\nproject: https://github.com/davtoh/RRtools")
    parser.add_argument('filenames', nargs='*',
                        help='List of images or path to images. Glob operations can be '
                             'achieved using the wildcard sign "*". '
                             'It can load image from files, urls, servers, strings'
                             'or directly from numpy arrays (supports databases)'
                             'Because the shell process wildcards before it gets '
                             'to the parser it creates a list of filtered files in '
                             'the path. Use quotes in shell to prevent this behaviour '
                             'an let the restorer do it instead e.g. "/path/to/images/*.jpg". '
                             'if "*" is used then folders and filenames that start with an '
                             'underscore "_" are ignored by the restorer')
    parser.add_argument('-v','--verbosity',type=int,default=1,
                        help="""(0) flag to print messages and debug data.
                                0 -> do not print messages.
                                1 -> print normal messages.
                                2 -> print normal and debug messages.
                                3 -> print all messages and show main results.
                                    (consumes significantly more memory).
                                4 -> print all messages and show all results.
                                    (consumes significantly more memory).
                                5 -> print all messages, show all results and additional data.
                                    (consumes significantly more memory).""")
    parser.add_argument('-f','--feature', type=string_interpreter(commahandler=feature_creator),
                        help='Configure detector and matcher')
    parser.add_argument('-u','--pool', action='store', type=int,
                        help='Use pool Ex: 4 to use 4 CPUs')
    parser.add_argument('-c','--cachePath',default=None,
                        help="""
                           saves memoization to specified path. This is useful to save
                           some computations and use them in next executions.
                           Cached data is not guaranteed to work between different
                           configurations and this can lead to unexpected program
                           behaviour. If a different configuration will be used it
                           is recommended to clear the cache to recompute values.
                           If True it creates the cache in current path.
                           """)
    parser.add_argument('-e','--clearCache', type=int, default=0,
                        help='clear cache flag.'
                            '* 0 do not clear.'
                            '* 1 re-compute data but other cache data is left intact.'
                            '* 2 All CachePath is cleared before use.'
                            'Notes: using cache can result in unexpected behaviour '
                            'if some configurations does not match to the cached data.')
    parser.add_argument('--loader', type=string_interpreter(commahandler=loader_creator),
                        nargs='?', help='Custom loader function used to load images. '
                            'By default or if --loader flag is empty it loads the '
                            'original images in color. The format is "--loader colorflag, '
                            'x, y" where colorflag is -1,0,1 for BGRA, gray and BGR images '
                            'and the load shape are represented by x and y. '
                            'Ex 1: "0,100,100" loads gray images of shape (100,100) in '
```

```
                        'gray scale. Ex 2: "1" loads images in BGR color and with '
                        'original shapes. Ex 3: "0,200,None" loads gray images of shape '
                        '(200,None) where None is calculated to keep image ratio.')
    parser.add_argument('-p','--process_shape', default=(400,400), type=string_interpreter(),
                        nargs='?', help='Process shape used to convert to pseudo images '
                        'to process features and then convert to the '
                        'original images. The smaller the image more memory and speed '
                        'gain. By default process_shape is 400,400'
                        'If the -p flag is empty it loads the original '
                        'images to process the features but it can incur to performance'
                        ' penalties if images are too big and RAM memory is scarce')
    parser.add_argument('-l','--load_shape', default=None, type=string_interpreter(),
                        nargs='?', help='shape used to load images which are beeing merged.')
    parser.add_argument('-b','--baseImage', default=True, type=string_interpreter(), nargs='?',
                        help='Specify image''s name to use from path as first image to merge '
                        'in the empty restored image. By default it selects the image '
                        'with most features. If the -b flag is empty it selects the '
                        'first image in filenames as base image')
    parser.add_argument('-m','--selectMethod',
                        help='Method to sort images when matching. This '
                        'way the merging order can be controlled.'
                        '* (None) Best matches'
                        '* Histogram Comparison: Correlation, Chi-squared,'
                        'Intersection, Hellinger or any method found in hist_map'
                        '* Entropy'
                        '* custom function of the form: rating,fn <-- selectMethod(fns)')
    parser.add_argument('-d','--distanceThresh', type = float, default=0.75,
                        help='Filter matches by distance ratio')
    parser.add_argument('-i','--inlineThresh', type = float, default=0.2,
                        help='Filter homography by inlineratio')
    parser.add_argument('-r','--rectangularityThresh', type = float, default=0.5,
                        help='Filter homography by rectangularity')
    parser.add_argument('-j','--ransacReprojThreshold', type = float, default=10.0,
                        help='Maximum allowed reprojection error '
                            'to treat a point pair as an inlier')
    parser.add_argument('-n','--centric', action='store_true',
                        help='Tries to attach as many images as possible to '
                            'each matching. It is quicker since it does not have to process '
                            'too many match computations')
    parser.add_argument('-t','--hist_match', action='store_true',
                        help='Apply histogram matching to foreground '
                            'image with merge image as template')
    parser.add_argument('-s','--save', default=True, type = string_interpreter(False), nargs='?',
                        help='Customize image name used to save the restored image.'
                            'By default it saves in path with name "_restored_{base_image}".'
                            'if the -s flag is specified empty it does not save. Formatting '
                            'is supported so for example the default name can be achived as '
                            '"-s {path}_restored_{name}{ext}"')
    parser.add_argument('-o','--overwrite', action='store_true',
                        help = 'If True and the destine filename for saving already'
                            'exists then it is replaced, else a new filename is generated'
                            'with an index "{filename}_{index}.{extension}"')
    parser.add_argument('-g','--grow_scene', action='store_true',
                        help='Flag to allow image to grow the scene so that that the final '
                            'image can be larger than the base image')
    parser.add_argument('-y','--denoise', default=None,
                        type=string_interpreter(False,commahandler=denoise_creator),
                        help="Flag to process noisy images. Use mild, normal, heavy or "
```

```
                        "provide parameters for a bilateral filter as "
                        "'--denoise d,sigmaColor,sigmaSpace' as for example "
                        "'--denoise 27,75,75'. By default it is None which can be "
                        "activated according to the restorer, if an empty flag is "
                        "provided as '--denoise' it deactivates de-noising images.")
    parser.add_argument('-a','--lens', action='store_true',
                        help='Flag to apply lens to retinal area. Else do not apply lens')
    parser.add_argument('-k','--enclose', action='store_true',
                        help='Flag to enclose and return only retinal area. '
                            'Else leaves image "as is"')
    parser.add_argument('-z','--restorer',choices = ['RetinalRestore','ImRestore'],
                        default='RetinalRestore',
                        help='imrestore is for images in general but it can be parametrized. '
                            'By default it has the profile "retinalRestore" for retinal '
                            'images but its general behaviour can be restorerd by '
                            'changing it to "imrestore"')
    parser.add_argument('-x','--expert', default=None,help='path to the expert variables')
    parser.add_argument('-q','--console', action='store_true',
                        help='Enter interactive mode to let user execute commands in console')
    parser.add_argument('-w','--debug', action='store_true',
                        help='Enter debug mode to let programmers find bugs')
    parser.add_argument('--onlykeys', action='store_true',
                        help='Only compute keypoints. This is useful when --cachePath is '
                            'used and the user wants to have the keypoints cached beforehand')

    # parse sys and get argument variables
    args = vars(parser.parse_args(args=args, namespace=namespace))

    if args['debug']:
        print "debug ON."
        import pdb; pdb.set_trace()

    # this is needed because the shell process wildcards before it gets to argparse
    # creating a list in the path thus it must be filtered. Use quotes in shell
    # to prevent this behaviour
    if len(args['filenames'])>1:
        args['filenames'] = [p for p in args['filenames'] if check_valid(p) or "*" in p]
    else:
        args['filenames'] = args['filenames'][0]

    # print parsed arguments
    if args['verbosity']>1:
        print "Parsed Arguments\n",args

    # use configuration
    r = args.pop("restorer")
    if r == 'RetinalRestore':
        if args['verbosity']: print "Configured for retinal restoration..."
        self = RetinalRestore(**args)
    elif r == 'ImRestore':
        if args['verbosity']: print "Configured for general restoration..."
        self = ImRestore(**args)
    else:
        raise Exception("no restoration class called {}".format(r))

    if namespace is not None:
        # update namespace from early stages so it can have access to the restorer
        namespace.restorer = self
```

```
    if args['console']:
        print "interactive ON."
        print "restoring instance is 'namespace.restorer' or 'self'"
        print "Ex: type 'self.restore()' to proceed with restoration."
        import code; code.interact(local=locals())
    elif args['onlykeys']:
        self.compute_keypoints()
    else:
        # start restoration
        self.restore()

    return namespace # return given namespace

if __name__ == "__main__":
    shell() # run the shell
    # TODO visualizator for alpha mask
    # TODO implement standard deviation in bright areas to detect optic disk
```

Executable

There are several options to create executables and installers from python sources ("deployment - Python Wiki," n.d.). From them Pyinstaller suffices for all the needs to create them. It is a python package so it must be installed after python, these procedures are found in subsubsection A.1.3 and further reading in (Cortesi, Bajo, Caban, & McMillan, n.d.). To create one-file executable from a script the following command format is used:

```
$ pyinstaller -p <path to sources> -n <name of program> -F <path to script> --version-file=<path to
 ↪  version file>
```

The -p option is to add a path to search for imports, without this the executable would not have all the features. -n Name to assign to the bundled app and spec file. -F Create a one-file bundled executable.

Lets create *imrestore* executable then, so open the Terminal where *RRtoolbox* folder and *imrestore.py* are (i.e. like in the repository) then type the following command to create the executable:

```
$ pyinstaller -p ./ -n imrestore -F ./imrestore.py/ --version-file=version
```

This will create the folders *build* where temporal files are compiled to be used in the creation of the executable and *dist* where the executable is placed once pyinstaller finishes building it. Under windows 7 the msvcr100.dll is needed so it is advised to install Microsoft Visual C++ 2010 before runing pyinstaller to let it find the necesary files. Notice that the executable is only compatible for computers with the same operating system as the one used to create it. Once the executable is created the folloing command format can be used to run it:

Code 3

```
$ ./imrestore <path to folder with images> -s <save as filename> -c <cache file>
```

To test *imrestore* once pyinstaller finishes creating the executable just enter the command:

```
$ ./dist/imrestore tests/im1*
```

or if the terminal path is in *dist* folder:
For further reading the capabilities of *imrestore* and configurations just type in:

```
$ ./imrestore ../tests/im1*
```

```
$ ./imrestore -h
```

or its equivalent

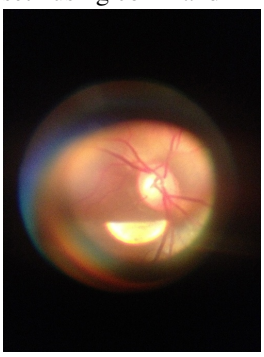
```
$ ./imrestore --help
```

Results

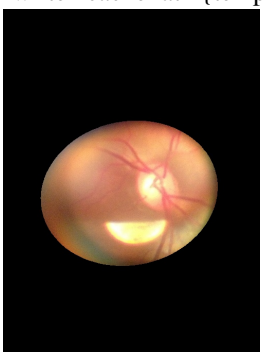
Using the basic execution of *imrestore* as in Code 3 but with some additional options

```
imrestore results/set1/*.* -lens -overwrite -cachePath {temp}
```

Figure 20. Result for set1 using command 'imrestore results/set1/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_0176



(b) Restored image

Table 3
Profiling for set1 using command 'imrestore results/set1/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.3975 secs
- 'Computing features' -> 1.6288 secs
  - (cached) 'IMG_0173' -> 0.1453 secs
  - (cached) 'IMG_0176' -> 0.1657 secs
  - (cached) 'IMG_0179' -> 0.1637 secs
  - (cached) 'IMG_0180' -> 0.0776 secs
  - (cached) 'IMG_0182' -> 0.4056 secs
  - (cached) 'IMG_0184' -> 0.2854 secs
  - (cached) 'IMG_0185' -> 0.0543 secs
- 'Restoring' -> 0.4533 secs
  - 'Matching' -> 0.0577 secs
  - 'Merging' -> 0.3954 secs
- 'Post-processing' -> 0.3154 secs

'ImRestore init' -> 9.3586 secs
- 'Computing features' -> 8.2685 secs
  - (processed) 'IMG_0173' -> 1.1794 secs
  - (processed) 'IMG_0176' -> 1.1576 secs
  - (processed) 'IMG_0179' -> 1.1716 secs
  - (processed) 'IMG_0180' -> 1.1678 secs
  - (processed) 'IMG_0182' -> 1.1883 secs
  - (processed) 'IMG_0184' -> 1.1594 secs
  - (processed) 'IMG_0185' -> 1.2064 secs
- 'Restoring' -> 0.7995 secs
  - 'Matching' -> 0.0648 secs
  - 'Merging' -> 0.7344 secs
    - 'IMG_0173' -> 0.6915 secs
- 'Post-processing' -> 0.2906 secs

Figure 21. Result for set2 using command 'imrestore results/set2/*.* –lens –overwrite –cachePath {temp}'

(a) IMG_0409 (b) IMG_0408 (c) Restored image

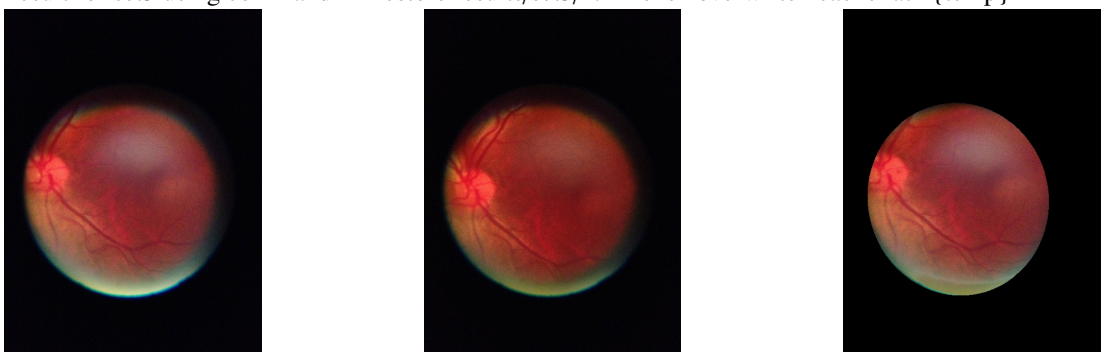
Table 4

Profiling for set2 using command 'imrestore results/set2/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 3.0191 secs
- 'Computing features' -> 1.3202 secs
  - (cached) 'IMG_0405' -> 0.1566 secs
  - (cached) 'IMG_0406' -> 0.1207 secs
  - (cached) 'IMG_0407' -> 0.1049 secs
  - (cached) 'IMG_0408' -> 0.1619 secs
  - (cached) 'IMG_0409' -> 0.1667 secs
  - (cached) 'IMG_0410' -> 0.1187 secs
- 'Restoring' -> 1.2985 secs
  - 'Matching' -> 0.0323 secs
  - 'Merging' -> 1.2658 secs
    - 'IMG_0408' -> 1.2351 secs
- 'Post-processing' -> 0.4005 secs

'ImRestore init' -> 10.0473 secs
- 'Computing features' -> 9.0070 secs
  - (processed) 'IMG_0405' -> 1.4506 secs
  - (processed) 'IMG_0406' -> 1.3965 secs
  - (processed) 'IMG_0407' -> 1.5363 secs
  - (processed) 'IMG_0408' -> 1.5071 secs
  - (processed) 'IMG_0409' -> 1.5712 secs
  - (processed) 'IMG_0410' -> 1.5027 secs
- 'Restoring' -> 0.8119 secs
  - 'Matching' -> 0.0268 secs
  - 'Merging' -> 0.7849 secs
    - 'IMG_0408' -> 0.7620 secs
- 'Post-processing' -> 0.2283 secs

Figure 22. Result for set3 using command 'imrestore results/set3/*.* –lens –overwrite –cachePath {temp}'



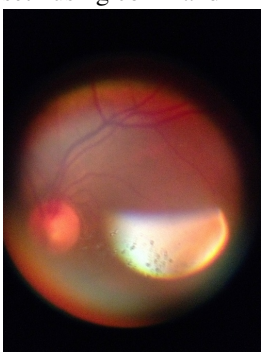
(a) IMG_0431 (b) IMG_0429 (c) Restored image

Table 5
Profiling for set3 using command 'imrestore results/set3/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 4.1253 secs
- 'Computing features' -> 2.2530 secs
  - (cached) 'IMG_0428' -> 0.0924 secs
  - (cached) 'IMG_0429' -> 0.3340 secs
  - (cached) 'IMG_0430' -> 0.2959 secs
  - (cached) 'IMG_0431' -> 0.1206 secs
  - (cached) 'IMG_0432' -> 0.2006 secs
  - (cached) 'IMG_0433' -> 0.3176 secs
  - (cached) 'IMG_0434' -> 0.3411 secs
- 'Restoring' -> 1.4384 secs
  - 'Matching' -> 0.0798 secs
  - 'Merging' -> 1.3583 secs
    - 'IMG_0429' -> 1.1816 secs
- 'Post-processing' -> 0.4340 secs

'ImRestore init' -> 13.8436 secs
- 'Computing features' -> 12.4398 secs
  - (processed) 'IMG_0428' -> 1.7227 secs
  - (processed) 'IMG_0429' -> 1.7231 secs
  - (processed) 'IMG_0430' -> 1.8142 secs
  - (processed) 'IMG_0431' -> 1.8121 secs
  - (processed) 'IMG_0432' -> 1.7423 secs
  - (processed) 'IMG_0433' -> 1.7625 secs
  - (processed) 'IMG_0434' -> 1.8109 secs
- 'Restoring' -> 1.1208 secs
  - 'Matching' -> 0.0937 secs
  - 'Merging' -> 1.0268 secs
    - 'IMG_0429' -> 0.9621 secs
- 'Post-processing' -> 0.2829 secs

Figure 23. Result for set4 using command 'imrestore results/set4/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_0411

(b) Restored image

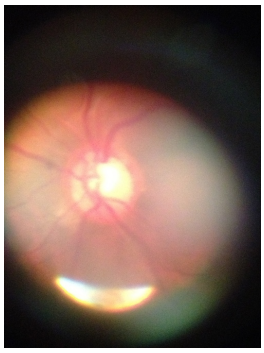
Table 6

Profiling for set4 using command 'imrestore results/set4/.* –lens –overwrite –cachePath {temp}'*

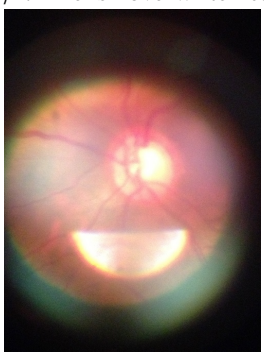
'ImRestore init' -> 3.6004 secs	'ImRestore init' -> 21.3726 secs
'Computing features' -> 2.9687 secs	'Computing features' -> 20.9466 secs
(cached) 'IMG_0411' -> 0.3201 secs	(processed) 'IMG_0411' -> 1.8449 secs
(cached) 'IMG_0412' -> 0.1031 secs	(processed) 'IMG_0412' -> 1.7487 secs
(cached) 'IMG_0413' -> 0.1520 secs	(processed) 'IMG_0413' -> 1.7904 secs
(cached) 'IMG_0418' -> 0.1898 secs	(processed) 'IMG_0418' -> 1.7764 secs
(cached) 'IMG_0419' -> 0.3758 secs	(processed) 'IMG_0419' -> 1.7444 secs
(cached) 'IMG_0420' -> 0.0851 secs	(processed) 'IMG_0420' -> 1.6846 secs
(cached) 'IMG_0421' -> 0.1947 secs	(processed) 'IMG_0421' -> 1.6818 secs
(cached) 'IMG_0423' -> 0.1643 secs	(processed) 'IMG_0423' -> 1.7175 secs
(cached) 'IMG_0424' -> 0.1584 secs	(processed) 'IMG_0424' -> 1.7362 secs
(cached) 'IMG_0425' -> 0.1458 secs	(processed) 'IMG_0425' -> 1.7171 secs
(cached) 'IMG_0426' -> 0.0426 secs	(processed) 'IMG_0426' -> 1.7084 secs
(cached) 'IMG_0427' -> 0.1792 secs	(processed) 'IMG_0427' -> 1.7076 secs
'Restoring' -> 0.2444 secs	'Restoring' -> 0.1185 secs
'Matching' -> 0.0256 secs	'Matching' -> 0.0360 secs
'Merging' -> 0.2186 secs	'Merging' -> 0.0822 secs
'Post-processing' -> 0.3874 secs	'Post-processing' -> 0.3075 secs

Figure 24. Result for set5 using command 'imrestore results/set5/*.* –lens –overwrite –cachePath {temp}'

(a) IMG_0253

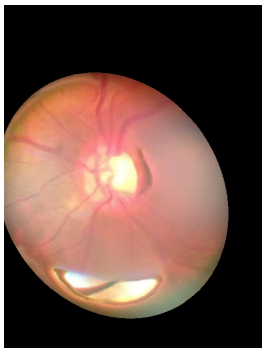


(b) IMG_0254



(c) IMG_0250

(d) IMG_0252



(e) Restored image

Table 7

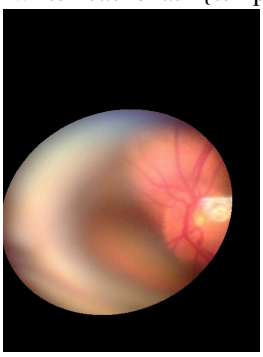
Profiling for set5 using command 'imrestore results/set5/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 4.3732 secs
- 'Computing features' -> 1.2738 secs
  - (cached) 'IMG_0249' -> 0.1184 secs
  - (cached) 'IMG_0250' -> 0.0628 secs
  - (cached) 'IMG_0251' -> 0.0890 secs
  - (cached) 'IMG_0252' -> 0.2341 secs
  - (cached) 'IMG_0253' -> 0.0847 secs
  - (cached) 'IMG_0254' -> 0.0694 secs
  - (cached) 'IMG_0256' -> 0.1919 secs
- 'Restoring' -> 2.6938 secs
  - 'Matching' -> 0.0966 secs
  - 'Merging' -> 2.5967 secs
    - 'IMG_0254' -> 0.7968 secs
    - 'IMG_0250' -> 0.7303 secs
    - 'IMG_0252' -> 1.0237 secs
- 'Post-processing' -> 0.4056 secs

'ImRestore init' -> 16.1532 secs
- 'Computing features' -> 12.6336 secs
  - (processed) 'IMG_0249' -> 1.8673 secs
  - (processed) 'IMG_0250' -> 1.8877 secs
  - (processed) 'IMG_0251' -> 1.7574 secs
  - (processed) 'IMG_0252' -> 1.8123 secs
  - (processed) 'IMG_0253' -> 1.7521 secs
  - (processed) 'IMG_0254' -> 1.7058 secs
  - (processed) 'IMG_0256' -> 1.7952 secs
- 'Restoring' -> 3.2721 secs
  - 'Matching' -> 0.1403 secs
  - 'Merging' -> 3.1312 secs
    - 'IMG_0254' -> 1.1349 secs
    - 'IMG_0250' -> 1.0850 secs
    - 'IMG_0252' -> 0.8909 secs
- 'Post-processing' -> 0.2474 secs

Figure 25. Result for set6 using command 'imrestore results/set6/*.* –lens –overwrite –cachePath {temp}'

(a) IMG_0257



(b) Restored image

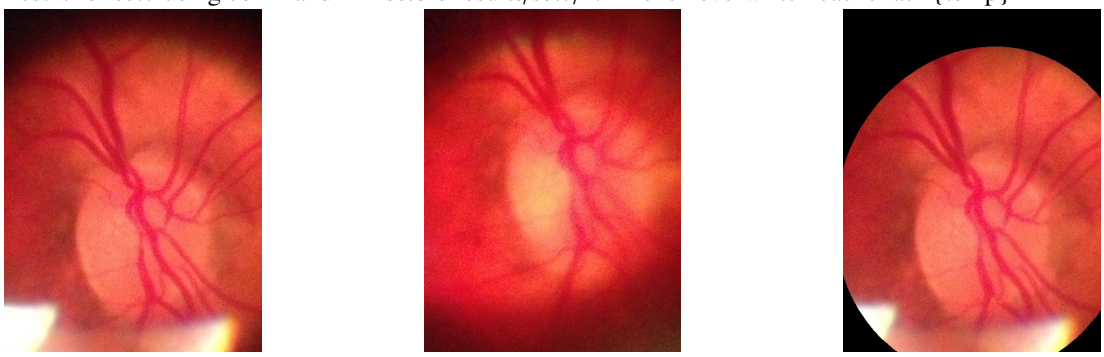
Table 8

Profiling for set6 using command 'imrestore results/set6/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.0236 secs
- 'Computing features' -> 1.6326 secs
  - (cached) 'IMG_0257' -> 0.1407 secs
  - (cached) 'IMG_0258' -> 0.2139 secs
  - (cached) 'IMG_0259' -> 0.1097 secs
  - (cached) 'IMG_0260' -> 0.0918 secs
  - (cached) 'IMG_0262' -> 0.3063 secs
  - (cached) 'IMG_0263' -> 0.0989 secs
  - (cached) 'IMG_0264' -> 0.0922 secs
- 'Restoring' -> 0.0456 secs
  - 'Matching' -> 0.0184 secs
  - 'Merging' -> 0.0270 secs
- 'Post-processing' -> 0.3454 secs

'ImRestore init' -> 13.1851 secs
- 'Computing features' -> 12.8655 secs
  - (processed) 'IMG_0257' -> 1.8973 secs
  - (processed) 'IMG_0258' -> 1.9041 secs
  - (processed) 'IMG_0259' -> 1.8280 secs
  - (processed) 'IMG_0260' -> 1.8462 secs
  - (processed) 'IMG_0262' -> 1.7901 secs
  - (processed) 'IMG_0263' -> 1.7197 secs
  - (processed) 'IMG_0264' -> 1.8139 secs
- 'Restoring' -> 0.0447 secs
  - 'Matching' -> 0.0191 secs
  - 'Merging' -> 0.0254 secs
- 'Post-processing' -> 0.2750 secs

Figure 26. Result for set7 using command 'imrestore results/set7/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_0270 (b) IMG_0271 (c) Restored image

Table 9
Profiling for set7 using command 'imrestore results/set7/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.6436 secs
- 'Computing features' -> 0.8550 secs
  - (cached) 'IMG_0267' -> 0.1104 secs
  - (cached) 'IMG_0269' -> 0.0461 secs
  - (cached) 'IMG_0270' -> 0.2348 secs
  - (cached) 'IMG_0271' -> 0.2121 secs
  - (cached) 'IMG_0273' -> 0.0413 secs
- 'Restoring' -> 1.4218 secs
  - 'Matching' -> 0.1126 secs
  - 'Merging' -> 1.3089 secs
    - 'IMG_0271' -> 1.2883 secs
- 'Post-processing' -> 0.3669 secs

'ImRestore init' -> 11.0449 secs
- 'Computing features' -> 9.9199 secs
  - (processed) 'IMG_0267' -> 1.9438 secs
  - (processed) 'IMG_0269' -> 2.3519 secs
  - (processed) 'IMG_0270' -> 1.9637 secs
  - (processed) 'IMG_0271' -> 1.8002 secs
  - (processed) 'IMG_0273' -> 1.8200 secs
- 'Restoring' -> 0.8630 secs
  - 'Matching' -> 0.0626 secs
  - 'Merging' -> 0.8000 secs
    - 'IMG_0271' -> 0.7781 secs
- 'Post-processing' -> 0.2620 secs

Figure 27. Result for set8 using command 'imrestore results/set8/*.* –lens –overwrite –cachePath {temp}'

(a) IMG_0306

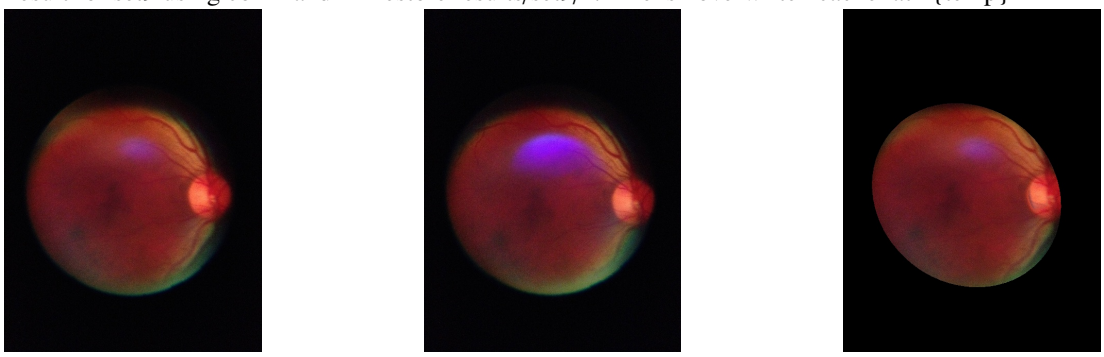
(b) Restored image

Table 10

Profiling for set8 using command 'imrestore results/set8/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.7088 secs	'ImRestore init' -> 13.2039 secs
'Computing features' -> 1.8329 secs	'Computing features' -> 12.8682 secs
(cached) 'IMG_0306' -> 0.2198 secs	(processed) 'IMG_0306' -> 1.9878 secs
(cached) 'IMG_0307' -> 0.2404 secs	(processed) 'IMG_0307' -> 1.8109 secs
(cached) 'IMG_0308' -> 0.1309 secs	(processed) 'IMG_0308' -> 1.8146 secs
(cached) 'IMG_0309' -> 0.1688 secs	(processed) 'IMG_0309' -> 1.8870 secs
(cached) 'IMG_0310' -> 0.3589 secs	(processed) 'IMG_0310' -> 1.7256 secs
(cached) 'IMG_0311' -> 0.0760 secs	(processed) 'IMG_0311' -> 1.8106 secs
(cached) 'IMG_0357' -> 0.0705 secs	(processed) 'IMG_0357' -> 1.7765 secs
'Restoring' -> 0.4236 secs	'Restoring' -> 0.0587 secs
'Matching' -> 0.0206 secs	'Matching' -> 0.0240 secs
'Merging' -> 0.4028 secs	'Merging' -> 0.0345 secs
'Post-processing' -> 0.4524 secs	'Post-processing' -> 0.2769 secs

Figure 28. Result for set9 using command 'imrestore results/set9/*.* –lens –overwrite –cachePath {temp}'



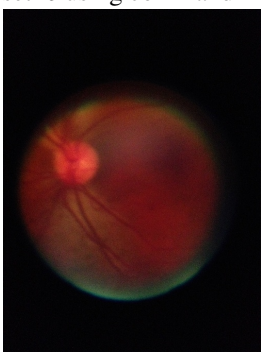
(a) IMG_0358 (b) IMG_0359 (c) Restored image

Table 11
Profiling for set9 using command 'imrestore results/set9/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.3111 secs
- 'Computing features' -> 0.6344 secs
  - (cached) 'IMG_0358' -> 0.1688 secs
  - (cached) 'IMG_0359' -> 0.1155 secs
  - (cached) 'IMG_0360' -> 0.0353 secs
- 'Restoring' -> 1.3800 secs
  - 'Matching' -> 0.0237 secs
  - 'Merging' -> 1.3560 secs
    - 'IMG_0359' -> 1.3484 secs
- 'Post-processing' -> 0.2967 secs

'ImRestore init' -> 7.1786 secs
- 'Computing features' -> 5.7595 secs
  - (processed) 'IMG_0358' -> 1.8553 secs
  - (processed) 'IMG_0359' -> 2.0050 secs
  - (processed) 'IMG_0360' -> 1.8769 secs
- 'Restoring' -> 1.1000 secs
  - 'Matching' -> 0.0215 secs
  - 'Merging' -> 1.0782 secs
    - 'IMG_0359' -> 1.0670 secs
- 'Post-processing' -> 0.3191 secs

Figure 29. Result for set10 using command 'imrestore results/set10/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_0369

(b) Restored image

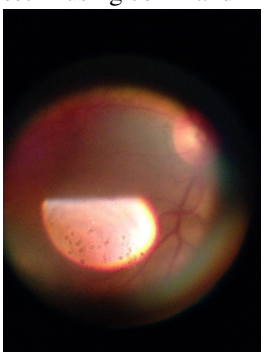
Table 12

Profiling for set10 using command 'imrestore results/set10/.* –lens –overwrite –cachePath {temp}'*

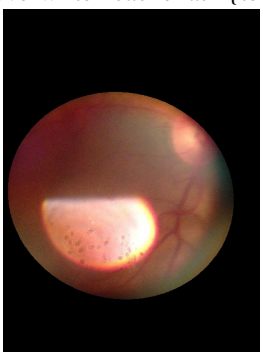
'ImRestore init' -> 2.8325 secs
- 'Computing features' -> 2.0102 secs
  - (cached) 'IMG_0361' -> 0.1797 secs
  - (cached) 'IMG_0362' -> 0.1535 secs
  - (cached) 'IMG_0363' -> 0.0478 secs
  - (cached) 'IMG_0364' -> 0.0840 secs
  - (cached) 'IMG_0365' -> 0.2969 secs
  - (cached) 'IMG_0366' -> 0.1463 secs
  - (cached) 'IMG_0367' -> 0.0336 secs
  - (cached) 'IMG_0368' -> 0.1962 secs
  - (cached) 'IMG_0369' -> 0.1155 secs
  - (cached) 'IMG_0370' -> 0.2277 secs
- 'Restoring' -> 0.5046 secs
  - 'Matching' -> 0.0233 secs
  - 'Merging' -> 0.4811 secs
- 'Post-processing' -> 0.3177 secs

'ImRestore init' -> 15.0763 secs
- 'Computing features' -> 13.8494 secs
  - (processed) 'IMG_0361' -> 1.3640 secs
  - (processed) 'IMG_0362' -> 1.4216 secs
  - (processed) 'IMG_0363' -> 1.3109 secs
  - (processed) 'IMG_0364' -> 1.5012 secs
  - (processed) 'IMG_0365' -> 1.4416 secs
  - (processed) 'IMG_0366' -> 1.3494 secs
  - (processed) 'IMG_0367' -> 1.3219 secs
  - (processed) 'IMG_0368' -> 1.3395 secs
  - (processed) 'IMG_0369' -> 1.3588 secs
  - (processed) 'IMG_0370' -> 1.3836 secs
- 'Restoring' -> 0.9844 secs
  - 'Matching' -> 0.0474 secs
  - 'Merging' -> 0.9367 secs
    - 'IMG_0368' -> 0.8363 secs
- 'Post-processing' -> 0.2424 secs

Figure 30. Result for set11 using command 'imrestore results/set11/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_0379



(b) Restored image

Table 13

Profiling for set11 using command 'imrestore results/set11/.* –lens –overwrite –cachePath {temp}'*

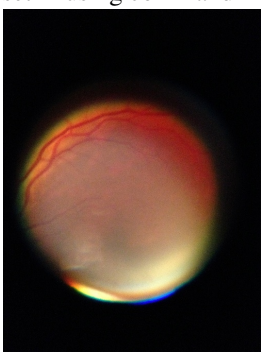
'ImRestore init' -> 1.6799 secs

- 'Computing features' -> 1.2339 secs
  - (cached) 'IMG_0373' -> 0.1339 secs
  - (cached) 'IMG_0374' -> 0.0601 secs
  - (cached) 'IMG_0375' -> 0.1447 secs
  - (cached) 'IMG_0377' -> 0.1687 secs
  - (cached) 'IMG_0379' -> 0.1378 secs
  - (cached) 'IMG_0380' -> 0.1247 secs
- 'Restoring' -> 0.0539 secs
  - 'Matching' -> 0.0213 secs
  - 'Merging' -> 0.0324 secs
- 'Post-processing' -> 0.3921 secs

'ImRestore init' -> 8.6642 secs

- 'Computing features' -> 8.3305 secs
  - (processed) 'IMG_0373' -> 1.3712 secs
  - (processed) 'IMG_0374' -> 1.4154 secs
  - (processed) 'IMG_0375' -> 1.3672 secs
  - (processed) 'IMG_0377' -> 1.3205 secs
  - (processed) 'IMG_0379' -> 1.4250 secs
  - (processed) 'IMG_0380' -> 1.3941 secs
- 'Restoring' -> 0.0537 secs
  - 'Matching' -> 0.0252 secs
  - 'Merging' -> 0.0283 secs
- 'Post-processing' -> 0.2800 secs

Figure 31. Result for set12 using command 'imrestore results/set12/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_0382

(b) Restored image

Table 14

Profiling for set12 using command 'imrestore results/set12/.* –lens –overwrite –cachePath {temp}'*

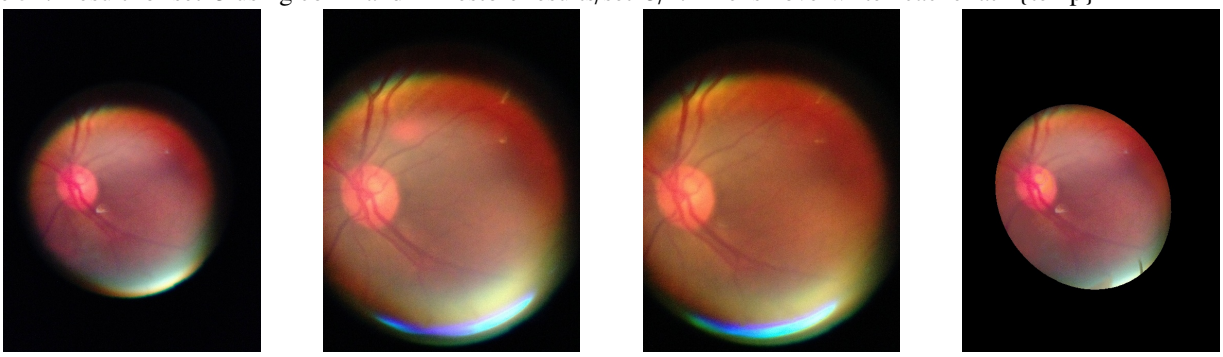
'ImRestore init' -> 1.4818 secs

- 'Computing features' -> 0.8779 secs
  - (cached) 'IMG_0381' -> 0.1663 secs
  - (cached) 'IMG_0382' -> 0.1818 secs
  - (cached) 'IMG_0383' -> 0.0320 secs
  - (cached) 'IMG_0384' -> 0.3067 secs
- 'Restoring' -> 0.2298 secs
  - 'Matching' -> 0.0351 secs
  - 'Merging' -> 0.1944 secs
- 'Post-processing' -> 0.3742 secs

'ImRestore init' -> 6.2643 secs

- 'Computing features' -> 5.9383 secs
  - (processed) 'IMG_0381' -> 1.4969 secs
  - (processed) 'IMG_0382' -> 1.4921 secs
  - (processed) 'IMG_0383' -> 1.4574 secs
  - (processed) 'IMG_0384' -> 1.4683 secs
- 'Restoring' -> 0.0538 secs
  - 'Matching' -> 0.0121 secs
  - 'Merging' -> 0.0415 secs
- 'Post-processing' -> 0.2722 secs

Figure 32. Result for set13 using command 'imrestore results/set13/*.* –lens –overwrite –cachePath {temp}'



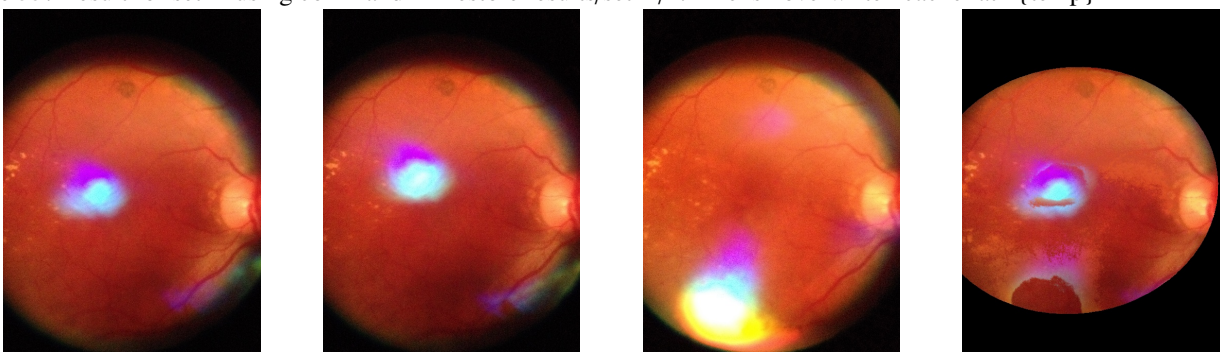
(a) IMG_0386 (b) IMG_0387 (c) IMG_0388 (d) Restored image

Table 15
Profiling for set13 using command 'imrestore results/set13/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 3.5901 secs
- 'Computing features' -> 0.8233 secs
  - (cached) 'IMG_0386' -> 0.0633 secs
  - (cached) 'IMG_0387' -> 0.3907 secs
  - (cached) 'IMG_0388' -> 0.0345 secs
  - (cached) 'IMG_0389' -> 0.0503 secs
  - (cached) 'IMG_0390' -> 0.0670 secs
- 'Restoring' -> 2.5577 secs
  - 'Matching' -> 0.0732 secs
  - 'Merging' -> 2.4841 secs
    - 'IMG_0387' -> 1.4720 secs
    - 'IMG_0388' -> 0.9413 secs
- 'Post-processing' -> 0.2091 secs

'ImRestore init' -> 9.4853 secs
- 'Computing features' -> 7.3447 secs
  - (processed) 'IMG_0386' -> 1.4506 secs
  - (processed) 'IMG_0387' -> 1.4599 secs
  - (processed) 'IMG_0388' -> 1.4659 secs
  - (processed) 'IMG_0389' -> 1.4829 secs
  - (processed) 'IMG_0390' -> 1.4523 secs
- 'Restoring' -> 1.8204 secs
  - 'Matching' -> 0.0452 secs
  - 'Merging' -> 1.7748 secs
    - 'IMG_0388' -> 0.9052 secs
    - 'IMG_0387' -> 0.8526 secs
- 'Post-processing' -> 0.3202 secs

Figure 33. Result for set14 using command 'imrestore results/set14/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_0402 (b) IMG_0403 (c) IMG_0404 (d) Restored image

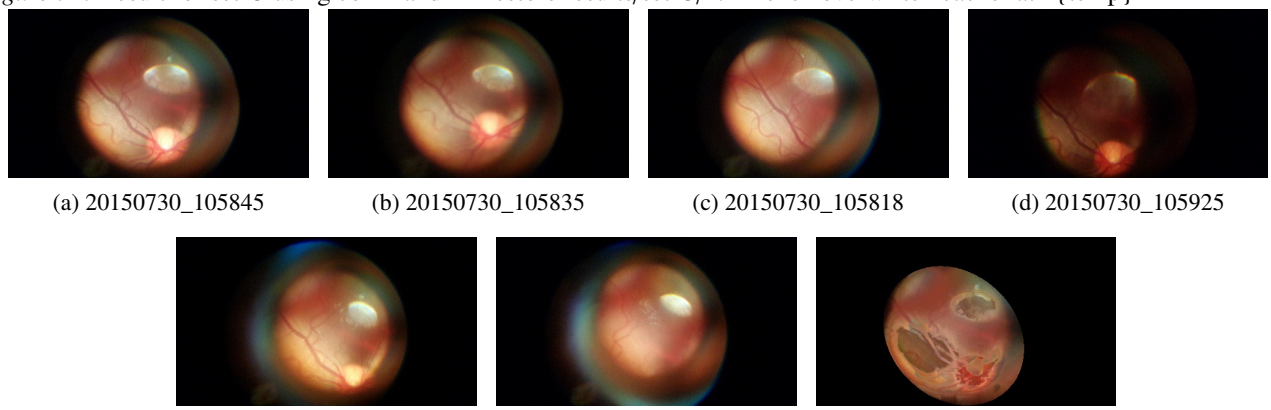
Table 16

Profiling for set14 using command 'imrestore results/set14/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 3.5072 secs
- 'Computing features' -> 1.5996 secs
  - (cached) 'IMG_0401' -> 0.5073 secs
  - (cached) 'IMG_0402' -> 0.1567 secs
  - (cached) 'IMG_0403' -> 0.2047 secs
  - (cached) 'IMG_0404' -> 0.3344 secs
- 'Restoring' -> 1.5741 secs
  - 'Matching' -> 0.0403 secs
  - 'Merging' -> 1.5334 secs
    - 'IMG_0403' -> 0.8398 secs
    - 'IMG_0404' -> 0.6823 secs
- 'Post-processing' -> 0.3335 secs

'ImRestore init' -> 8.1383 secs
- 'Computing features' -> 6.2119 secs
  - (processed) 'IMG_0401' -> 1.5540 secs
  - (processed) 'IMG_0402' -> 1.5758 secs
  - (processed) 'IMG_0403' -> 1.5386 secs
  - (processed) 'IMG_0404' -> 1.5134 secs
- 'Restoring' -> 1.6803 secs
  - 'Matching' -> 0.0479 secs
  - 'Merging' -> 1.6320 secs
    - 'IMG_0403' -> 0.8226 secs
    - 'IMG_0404' -> 0.8028 secs
- 'Post-processing' -> 0.2462 secs

Figure 34. Result for set15 using command 'imrestore results/set15/*.* –lens –overwrite –cachePath {temp}'



(a) 20150730_105845 (b) 20150730_105835 (c) 20150730_105818 (d) 20150730_105925

(e) 20150730_105822 (f) 20150730_105830 (g) Restored image

Table 17

Profiling for set15 using command 'imrestore results/set15/.* –lens –overwrite –cachePath {temp}'*

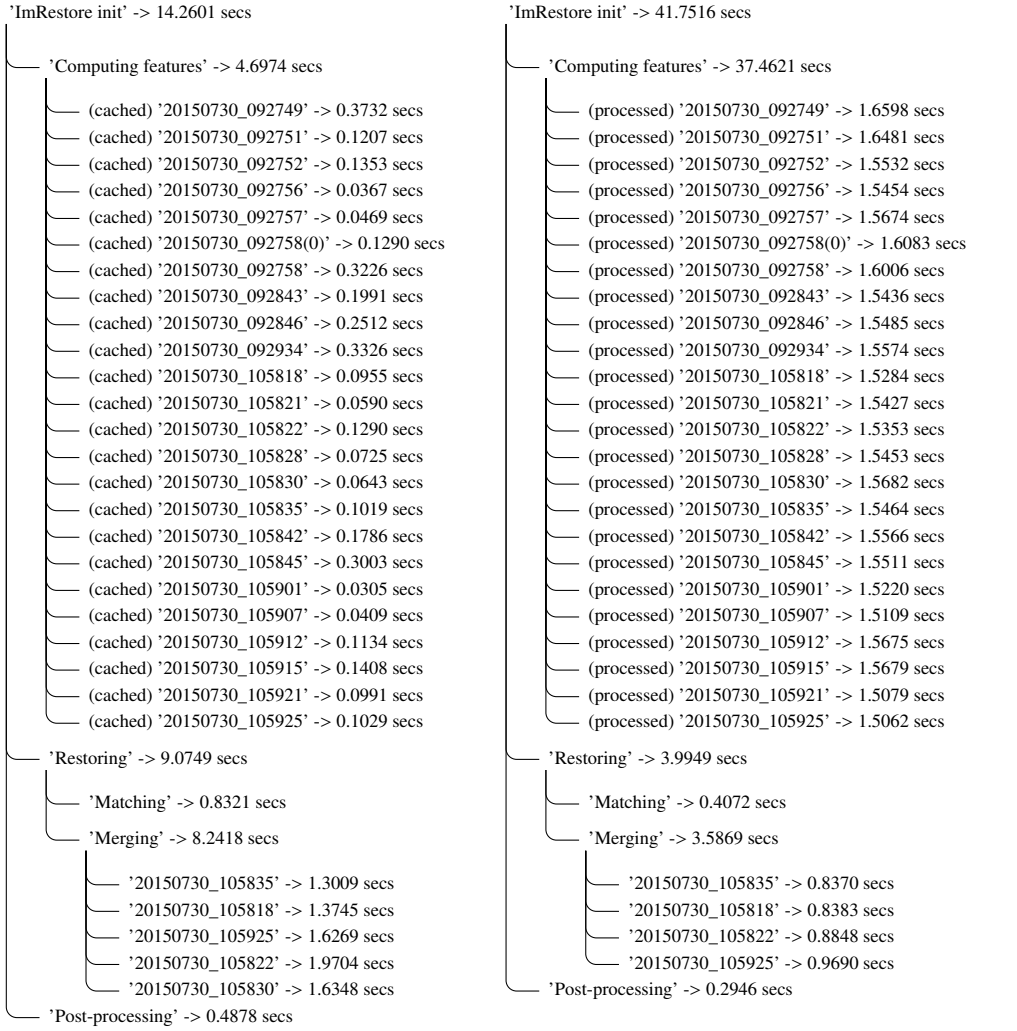
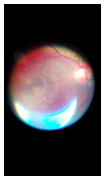
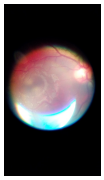


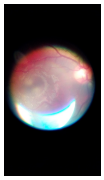
Figure 35. Result for set16 using command 'imrestore results/set16/*.* –lens –overwrite –cachePath {temp}'



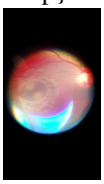
(a) IMG_20150730_115646_3



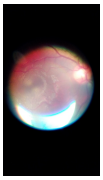
(b) IMG_20150730_115646_2



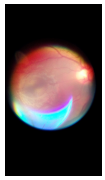
(c) IMG_20150730_115646_1



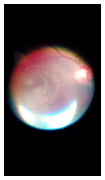
(d) IMG_20150730_115646_4



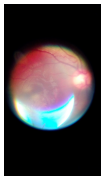
(e) IMG_20150730_115646



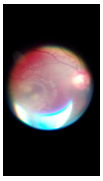
(f) IMG_20150730_115646_5



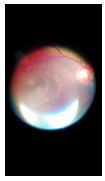
(g) IMG_20150730_115645_4



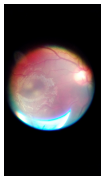
(h) IMG_20150730_115651_2



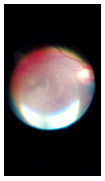
(i) IMG_20150730_115651_1



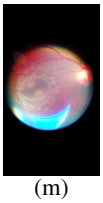
(j) IMG_20150730_115645_3



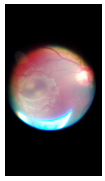
(k) IMG_20150730_115647_4



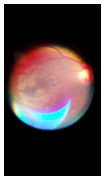
(l) IMG_20150730_115645_2



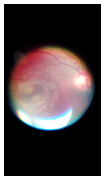
(m) IMG_20150730_115647_2



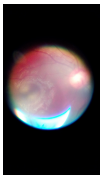
(n) IMG_20150730_115647_3



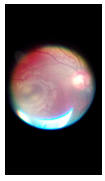
(o) IMG_20150730_115647_1



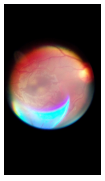
(p) IMG_20150730_115651



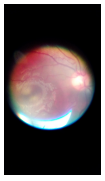
(q) IMG_20150730_115647_5



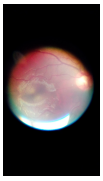
(r) IMG_20150730_115648



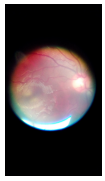
(s) IMG_20150730_115647



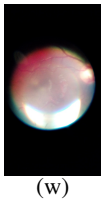
(t) IMG_20150730_115648_1



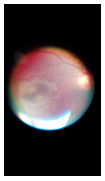
(u) IMG_20150730_115650_3



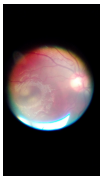
(v) IMG_20150730_115648_3



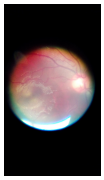
(w) IMG_20150730_115645_1



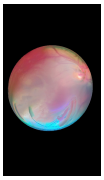
(x) IMG_20150730_115650_4



(y) IMG_20150730_115648_2



(z) IMG_20150730_115648_4



(aa) Restored image

Table 18
Profiling for set16 using command 'imrestore results/set16/.* –lens –overwrite –cachePath {temp}'*

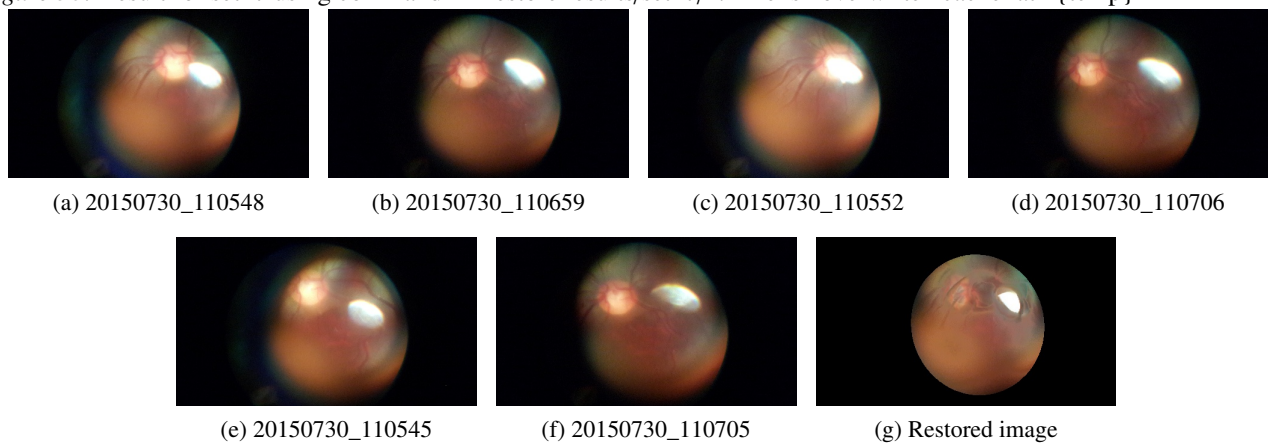
'ImRestore init' -> 20.7811 secs

- 'Computing features' -> 0.4400 secs
  - (cached) 'IMG_20150730_115646_2' -> 0.0131 secs
  - (cached) 'IMG_20150730_115649_2' -> 0.0084 secs
  - (cached) 'IMG_20150730_115642' -> 0.0075 secs
  - (cached) 'IMG_20150730_115644' -> 0.0078 secs
  - (cached) 'IMG_20150730_115644_1' -> 0.0074 secs
  - (cached) 'IMG_20150730_115644_2' -> 0.0079 secs
  - (cached) 'IMG_20150730_115644_3' -> 0.0066 secs
  - (cached) 'IMG_20150730_115645' -> 0.0062 secs
  - (cached) 'IMG_20150730_115645_1' -> 0.0052 secs
  - (cached) 'IMG_20150730_115645_2' -> 0.0066 secs
  - (cached) 'IMG_20150730_115645_3' -> 0.0067 secs
  - (cached) 'IMG_20150730_115645_4' -> 0.0063 secs
  - (cached) 'IMG_20150730_115646' -> 0.0071 secs
  - (cached) 'IMG_20150730_115646_1' -> 0.0101 secs
  - (cached) 'IMG_20150730_115646_3' -> 0.0085 secs
  - (cached) 'IMG_20150730_115646_4' -> 0.0067 secs
  - (cached) 'IMG_20150730_115646_5' -> 0.0066 secs
  - (cached) 'IMG_20150730_115647' -> 0.0064 secs
  - (cached) 'IMG_20150730_115647_1' -> 0.0071 secs
  - (cached) 'IMG_20150730_115647_2' -> 0.0062 secs
  - (cached) 'IMG_20150730_115647_3' -> 0.0058 secs
  - (cached) 'IMG_20150730_115647_4' -> 0.0066 secs
  - (cached) 'IMG_20150730_115647_5' -> 0.0059 secs
  - (cached) 'IMG_20150730_115648' -> 0.0051 secs
  - (cached) 'IMG_20150730_115648_1' -> 0.0044 secs
  - (cached) 'IMG_20150730_115648_2' -> 0.0040 secs
  - (cached) 'IMG_20150730_115648_3' -> 0.0035 secs
  - (cached) 'IMG_20150730_115648_4' -> 0.0035 secs
  - (cached) 'IMG_20150730_115649' -> 0.0033 secs
  - (cached) 'IMG_20150730_115649_1' -> 0.0031 secs
  - (cached) 'IMG_20150730_115649_3' -> 0.0031 secs
  - (cached) 'IMG_20150730_115649_4' -> 0.0031 secs
  - (cached) 'IMG_20150730_115649_5' -> 0.0032 secs
  - (cached) 'IMG_20150730_115650' -> 0.0034 secs
  - (cached) 'IMG_20150730_115650_1' -> 0.0033 secs
  - (cached) 'IMG_20150730_115650_2' -> 0.0040 secs
  - (cached) 'IMG_20150730_115650_3' -> 0.0034 secs
  - (cached) 'IMG_20150730_115650_4' -> 0.0034 secs
  - (cached) 'IMG_20150730_115651' -> 0.0036 secs
  - (cached) 'IMG_20150730_115651_1' -> 0.0040 secs
  - (cached) 'IMG_20150730_115651_2' -> 0.0069 secs
- 'Restoring' -> 20.1291 secs
  - 'Matching' -> 1.7202 secs
  - 'Merging' -> 18.4061 secs
    - 'IMG_20150730_115646_2' -> 0.7252 secs
    - 'IMG_20150730_115646_1' -> 0.6706 secs
    - 'IMG_20150730_115646_4' -> 0.6998 secs
    - 'IMG_20150730_115646' -> 0.7035 secs
    - 'IMG_20150730_115646_5' -> 0.6991 secs
    - 'IMG_20150730_115645_4' -> 0.6875 secs
    - 'IMG_20150730_115651_2' -> 0.6928 secs
    - 'IMG_20150730_115651_1' -> 0.7131 secs
    - 'IMG_20150730_115645_3' -> 0.7580 secs
    - 'IMG_20150730_115647_4' -> 0.6940 secs
    - 'IMG_20150730_115645_2' -> 0.7704 secs
    - 'IMG_20150730_115647_2' -> 0.7019 secs
    - 'IMG_20150730_115647_3' -> 0.7846 secs
    - 'IMG_20150730_115647_1' -> 0.7229 secs
    - 'IMG_20150730_115651' -> 0.7341 secs
    - 'IMG_20150730_115647_5' -> 0.7275 secs
    - 'IMG_20150730_115648' -> 0.7614 secs
    - 'IMG_20150730_115647' -> 0.7846 secs
    - 'IMG_20150730_115648_1' -> 0.7230 secs
    - 'IMG_20150730_115650_3' -> 0.7360 secs

'ImRestore init' -> 68.6732 secs

- 'Computing features' -> 48.6980 secs
  - (processed) 'IMG_20150730_115646_2' -> 1.0816 secs
  - (processed) 'IMG_20150730_115649_2' -> 1.0938 secs
  - (processed) 'IMG_20150730_115642' -> 1.0859 secs
  - (processed) 'IMG_20150730_115644' -> 1.0716 secs
  - (processed) 'IMG_20150730_115644_1' -> 1.0886 secs
  - (processed) 'IMG_20150730_115644_2' -> 1.0932 secs
  - (processed) 'IMG_20150730_115644_3' -> 1.0860 secs
  - (processed) 'IMG_20150730_115645' -> 1.0952 secs
  - (processed) 'IMG_20150730_115645_1' -> 1.1337 secs
  - (processed) 'IMG_20150730_115645_2' -> 1.1613 secs
  - (processed) 'IMG_20150730_115645_3' -> 1.4286 secs
  - (processed) 'IMG_20150730_115645_4' -> 1.2307 secs
  - (processed) 'IMG_20150730_115646' -> 1.1399 secs
  - (processed) 'IMG_20150730_115646_1' -> 1.1334 secs
  - (processed) 'IMG_20150730_115646_3' -> 1.1463 secs
  - (processed) 'IMG_20150730_115646_4' -> 1.1531 secs
  - (processed) 'IMG_20150730_115646_5' -> 1.1442 secs
  - (processed) 'IMG_20150730_115647' -> 1.1610 secs
  - (processed) 'IMG_20150730_115647_1' -> 1.1855 secs
  - (processed) 'IMG_20150730_115647_2' -> 1.1758 secs
  - (processed) 'IMG_20150730_115647_3' -> 1.1664 secs
  - (processed) 'IMG_20150730_115647_4' -> 1.2338 secs
  - (processed) 'IMG_20150730_115647_5' -> 1.2973 secs
  - (processed) 'IMG_20150730_115648' -> 1.2115 secs
  - (processed) 'IMG_20150730_115648_1' -> 1.2218 secs
  - (processed) 'IMG_20150730_115648_2' -> 1.1857 secs
  - (processed) 'IMG_20150730_115648_3' -> 1.2016 secs
  - (processed) 'IMG_20150730_115648_4' -> 1.2230 secs
  - (processed) 'IMG_20150730_115649' -> 1.2107 secs
  - (processed) 'IMG_20150730_115649_1' -> 1.1945 secs
  - (processed) 'IMG_20150730_115649_3' -> 1.2033 secs
  - (processed) 'IMG_20150730_115649_4' -> 1.2000 secs
  - (processed) 'IMG_20150730_115649_5' -> 1.2388 secs
  - (processed) 'IMG_20150730_115650' -> 1.2344 secs
  - (processed) 'IMG_20150730_115650_1' -> 1.2056 secs
  - (processed) 'IMG_20150730_115650_2' -> 1.2216 secs
  - (processed) 'IMG_20150730_115650_3' -> 1.2213 secs
  - (processed) 'IMG_20150730_115650_4' -> 1.2362 secs
  - (processed) 'IMG_20150730_115651' -> 1.2203 secs
  - (processed) 'IMG_20150730_115651_1' -> 1.2344 secs
  - (processed) 'IMG_20150730_115651_2' -> 1.2461 secs
- 'Restoring' -> 19.7787 secs
  - 'Matching' -> 1.7206 secs
  - 'Merging' -> 18.0554 secs
    - 'IMG_20150730_115646_2' -> 0.8245 secs
    - 'IMG_20150730_115646_1' -> 0.7067 secs
    - 'IMG_20150730_115646_4' -> 0.7005 secs
    - 'IMG_20150730_115646' -> 0.6869 secs
    - 'IMG_20150730_115646_5' -> 0.7066 secs
    - 'IMG_20150730_115645_4' -> 0.6875 secs
    - 'IMG_20150730_115651_2' -> 0.7005 secs
    - 'IMG_20150730_115651_1' -> 0.7230 secs
    - 'IMG_20150730_115645_3' -> 0.7226 secs
    - 'IMG_20150730_115647_4' -> 0.7168 secs
    - 'IMG_20150730_115645_2' -> 0.7108 secs
    - 'IMG_20150730_115647_2' -> 0.7199 secs
    - 'IMG_20150730_115647_3' -> 0.7312 secs
    - 'IMG_20150730_115647_1' -> 0.6718 secs
    - 'IMG_20150730_115651' -> 0.7073 secs
    - 'IMG_20150730_115647_5' -> 0.6876 secs
    - 'IMG_20150730_115648' -> 0.6863 secs
    - 'IMG_20150730_115647' -> 0.6920 secs
    - 'IMG_20150730_115648_1' -> 0.7031 secs
    - 'IMG_20150730_115650_3' -> 0.7765 secs

Figure 36. Result for set17 using command 'imrestore results/set17/*.* –lens –overwrite –cachePath {temp}'



(a) 20150730_110548 (b) 20150730_110659 (c) 20150730_110552 (d) 20150730_110706

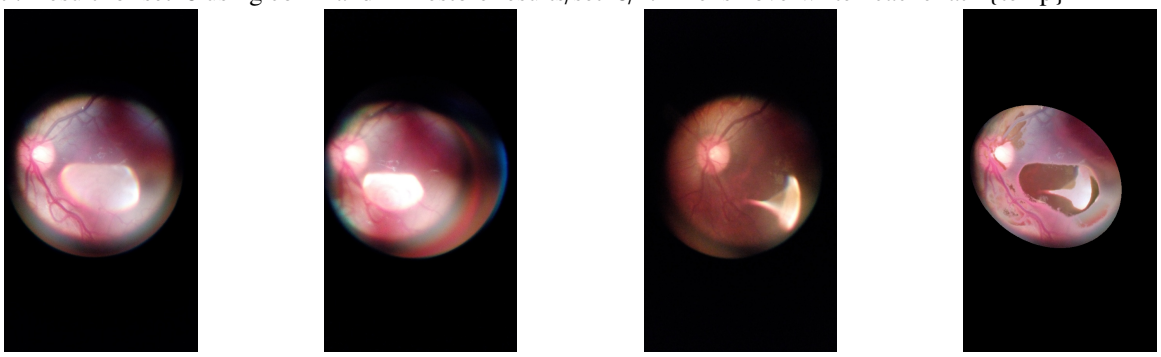
(e) 20150730_110545 (f) 20150730_110705 (g) Restored image

Table 19

Profiling for set17 using command 'imrestore results/set17/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 10.2073 secs	'ImRestore init' -> 22.1322 secs
'Computing features' -> 3.2556 secs	'Computing features' -> 18.4215 secs
(cached) '20150730_110405' -> 0.2753 secs	(processed) '20150730_110405' -> 1.0917 secs
(cached) '20150730_110530' -> 0.2416 secs	(processed) '20150730_110530' -> 0.9893 secs
(cached) '20150730_110539' -> 0.1702 secs	(processed) '20150730_110539' -> 1.0032 secs
(cached) '20150730_110545' -> 0.2880 secs	(processed) '20150730_110545' -> 1.0170 secs
(cached) '20150730_110548' -> 0.1267 secs	(processed) '20150730_110548' -> 1.0630 secs
(cached) '20150730_110552' -> 0.1074 secs	(processed) '20150730_110552' -> 1.0861 secs
(cached) '20150730_110645' -> 0.3342 secs	(processed) '20150730_110645' -> 1.0000 secs
(cached) '20150730_110648' -> 0.0737 secs	(processed) '20150730_110648' -> 1.0169 secs
(cached) '20150730_110650' -> 0.2950 secs	(processed) '20150730_110650' -> 1.0271 secs
(cached) '20150730_110656' -> 0.0602 secs	(processed) '20150730_110656' -> 1.0786 secs
(cached) '20150730_110657' -> 0.0787 secs	(processed) '20150730_110657' -> 1.0917 secs
(cached) '20150730_110659' -> 0.0505 secs	(processed) '20150730_110659' -> 1.1108 secs
(cached) '20150730_110705' -> 0.1878 secs	(processed) '20150730_110705' -> 1.1421 secs
(cached) '20150730_110706' -> 0.1304 secs	(processed) '20150730_110706' -> 1.0867 secs
(cached) '20150730_110711' -> 0.0349 secs	(processed) '20150730_110711' -> 1.0659 secs
(cached) '20150730_110714' -> 0.0295 secs	(processed) '20150730_110714' -> 1.0691 secs
(cached) '20150730_110717' -> 0.0322 secs	(processed) '20150730_110717' -> 1.0891 secs
'Restoring' -> 6.5630 secs	'Restoring' -> 3.4437 secs
'Matching' -> 0.3204 secs	'Matching' -> 0.2256 secs
'Merging' -> 6.2417 secs	'Merging' -> 3.2174 secs
'20150730_110659' -> 1.1539 secs	'20150730_110659' -> 0.6375 secs
'20150730_110552' -> 1.0402 secs	'20150730_110552' -> 0.6176 secs
'20150730_110706' -> 1.0032 secs	'20150730_110706' -> 0.6334 secs
'20150730_110545' -> 1.2199 secs	'20150730_110545' -> 0.6290 secs
'20150730_110705' -> 1.4824 secs	'20150730_110705' -> 0.6339 secs
'Post-processing' -> 0.3887 secs	'Post-processing' -> 0.2670 secs

Figure 37. Result for set18 using command 'imrestore results/set18/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_20150730_105626 (b) IMG_20150730_105748 (c) IMG_20150730_134235 (d) Restored image

Table 20

Profiling for set18 using command 'imrestore results/set18/.* –lens –overwrite –cachePath {temp}'*

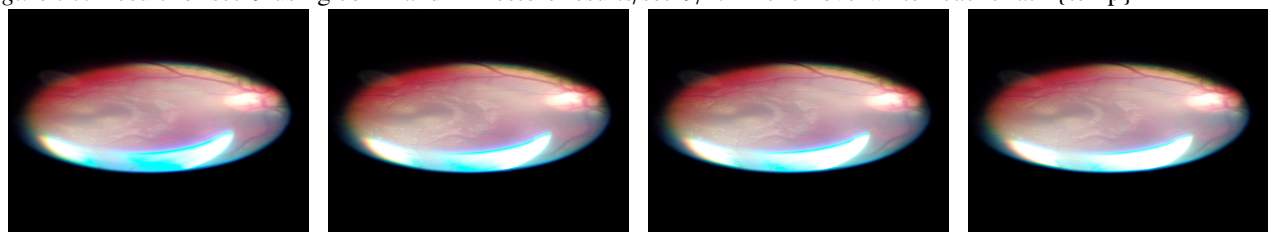
'ImRestore init' -> 8.3028 secs

- 'Computing features' -> 4.7222 secs
  - (cached) 'IMG_20150730_105626' -> 0.5616 secs
  - (cached) 'IMG_20150730_105632' -> 0.5313 secs
  - (cached) 'IMG_20150730_105655' -> 0.1401 secs
  - (cached) 'IMG_20150730_105748' -> 0.0758 secs
  - (cached) 'IMG_20150730_105753' -> 0.1064 secs
  - (cached) 'IMG_20150730_105800' -> 0.4234 secs
  - (cached) 'IMG_20150730_115251' -> 0.1562 secs
  - (cached) 'IMG_20150730_115254' -> 0.0867 secs
  - (cached) 'IMG_20150730_115301' -> 0.0733 secs
  - (cached) 'IMG_20150730_131444' -> 0.0939 secs
  - (cached) 'IMG_20150730_131454' -> 0.0638 secs
  - (cached) 'IMG_20150730_134219' -> 0.3296 secs
  - (cached) 'IMG_20150730_134225' -> 0.0978 secs
  - (cached) 'IMG_20150730_134231' -> 0.2208 secs
  - (cached) 'IMG_20150730_134235' -> 0.1166 secs
  - (cached) 'IMG_20150730_134254' -> 0.0686 secs
  - (cached) 'IMG_20150730_134300' -> 0.2575 secs
- 'Restoring' -> 3.1377 secs
  - 'Matching' -> 0.2373 secs
  - 'Merging' -> 2.8999 secs
    - 'IMG_20150730_105748' -> 1.0887 secs
    - 'IMG_20150730_134235' -> 1.1020 secs
- 'Post-processing' -> 0.4430 secs

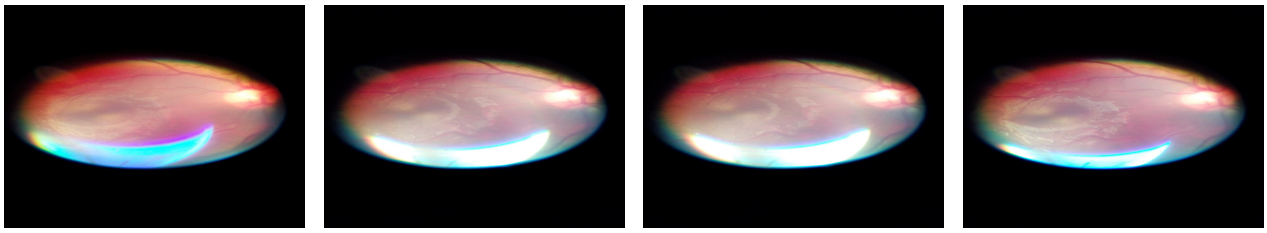
'ImRestore init' -> 21.4424 secs

- 'Computing features' -> 19.4687 secs
  - (processed) 'IMG_20150730_105626' -> 1.1089 secs
  - (processed) 'IMG_20150730_105632' -> 1.0642 secs
  - (processed) 'IMG_20150730_105655' -> 1.1255 secs
  - (processed) 'IMG_20150730_105748' -> 1.0984 secs
  - (processed) 'IMG_20150730_105753' -> 1.1192 secs
  - (processed) 'IMG_20150730_105800' -> 1.1131 secs
  - (processed) 'IMG_20150730_115251' -> 1.1260 secs
  - (processed) 'IMG_20150730_115254' -> 1.1425 secs
  - (processed) 'IMG_20150730_115301' -> 1.1593 secs
  - (processed) 'IMG_20150730_131444' -> 1.1446 secs
  - (processed) 'IMG_20150730_131454' -> 1.1339 secs
  - (processed) 'IMG_20150730_134219' -> 1.1527 secs
  - (processed) 'IMG_20150730_134225' -> 1.2122 secs
  - (processed) 'IMG_20150730_134231' -> 1.1806 secs
  - (processed) 'IMG_20150730_134235' -> 1.1658 secs
  - (processed) 'IMG_20150730_134254' -> 1.1610 secs
  - (processed) 'IMG_20150730_134300' -> 1.1723 secs
- 'Restoring' -> 1.7526 secs
  - 'Matching' -> 0.1983 secs
  - 'Merging' -> 1.5538 secs
    - 'IMG_20150730_105748' -> 0.6903 secs
    - 'IMG_20150730_134235' -> 0.7111 secs
- 'Post-processing' -> 0.2211 secs

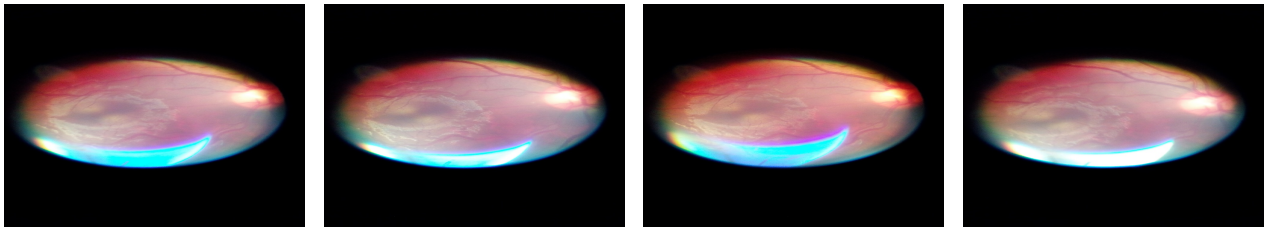
Figure 38. Result for set19 using command 'imrestore results/set19/*.* –lens –overwrite –cachePath {temp}'



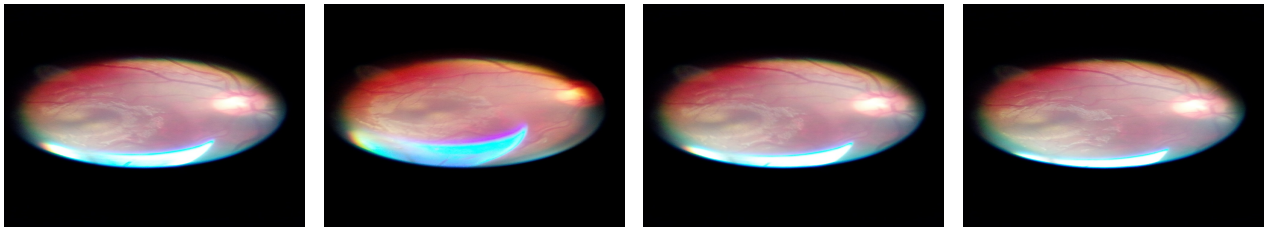
(a) IMG_20150730_115646_3 (b) IMG_20150730_115646_2 (c) IMG_20150730_115646_1 (d) IMG_20150730_115646



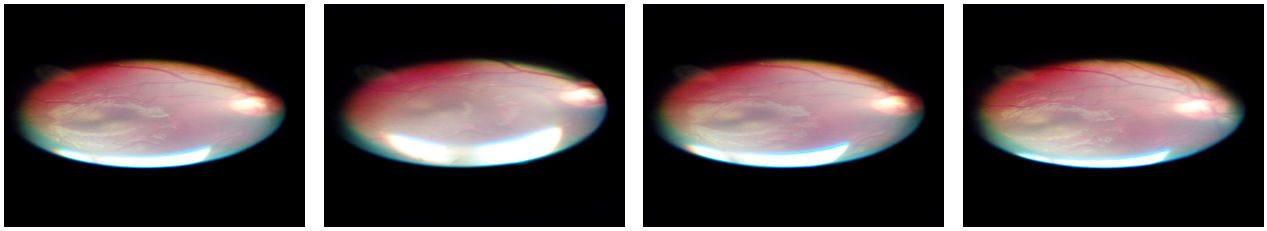
(e) IMG_20150730_115646_5 (f) IMG_20150730_115645_4 (g) IMG_20150730_115645_3 (h) IMG_20150730_115647_4



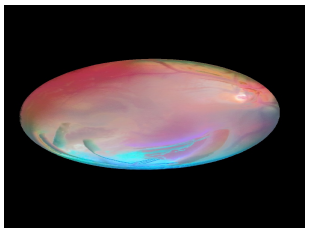
(i) IMG_20150730_115647_2 (j) IMG_20150730_115647_3 (k) IMG_20150730_115647_1 (l) IMG_20150730_115651



(m) IMG_20150730_115648 (n) IMG_20150730_115647 (o) IMG_20150730_115648_1 (p) IMG_20150730_115648_3



(q) IMG_20150730_115650_3 (r) IMG_20150730_115645_1 (s) IMG_20150730_115650_4 (t) IMG_20150730_115648_4



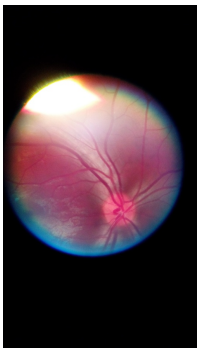
(u) Restored image

Table 21
Profiling for set19 using command 'imrestore results/set19/.* –lens –overwrite –cachePath {temp}'*

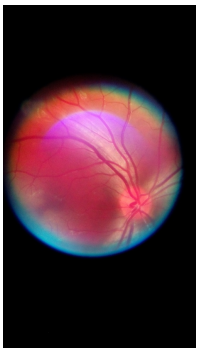
- 'ImRestore init' -> 34.6989 secs
  - 'Computing features' -> 5.4379 secs
    - (cached) 'IMG_20150730_115647_1' -> 0.3089 secs
    - (cached) 'IMG_20150730_115642' -> 0.2403 secs
    - (cached) 'IMG_20150730_115644' -> 0.0498 secs
    - (cached) 'IMG_20150730_115644_1' -> 0.0915 secs
    - (cached) 'IMG_20150730_115644_2' -> 0.1338 secs
    - (cached) 'IMG_20150730_115644_3' -> 0.0597 secs
    - (cached) 'IMG_20150730_115645_1' -> 0.1268 secs
    - (cached) 'IMG_20150730_115645_3' -> 0.1252 secs
    - (cached) 'IMG_20150730_115645_4' -> 0.0593 secs
    - (cached) 'IMG_20150730_115646' -> 0.0700 secs
    - (cached) 'IMG_20150730_115646_1' -> 0.1731 secs
    - (cached) 'IMG_20150730_115646_2' -> 0.2175 secs
    - (cached) 'IMG_20150730_115646_3' -> 0.1336 secs
    - (cached) 'IMG_20150730_115646_5' -> 0.1090 secs
    - (cached) 'IMG_20150730_115647' -> 0.1860 secs
    - (cached) 'IMG_20150730_115647_2' -> 0.2197 secs
    - (cached) 'IMG_20150730_115647_3' -> 0.3308 secs
    - (cached) 'IMG_20150730_115647_4' -> 0.2551 secs
    - (cached) 'IMG_20150730_115648' -> 0.1362 secs
    - (cached) 'IMG_20150730_115648_1' -> 0.0927 secs
    - (cached) 'IMG_20150730_115648_2' -> 0.1640 secs
    - (cached) 'IMG_20150730_115648_3' -> 0.0181 secs
    - (cached) 'IMG_20150730_115648_4' -> 0.0947 secs
    - (cached) 'IMG_20150730_115649' -> 0.1233 secs
    - (cached) 'IMG_20150730_115649_1' -> 0.0245 secs
    - (cached) 'IMG_20150730_115649_2' -> 0.0285 secs
    - (cached) 'IMG_20150730_115649_4' -> 0.0464 secs
    - (cached) 'IMG_20150730_115650' -> 0.0198 secs
    - (cached) 'IMG_20150730_115650_3' -> 0.0438 secs
    - (cached) 'IMG_20150730_115650_4' -> 0.0492 secs
    - (cached) 'IMG_20150730_115651' -> 0.1005 secs
  - 'Restoring' -> 29.0284 secs
    - 'Matching' -> 1.7487 secs
    - 'Merging' -> 27.2763 secs
      - 'IMG_20150730_115646_2' -> 0.9506 secs
      - 'IMG_20150730_115646_1' -> 1.2903 secs
      - 'IMG_20150730_115646' -> 1.4719 secs
      - 'IMG_20150730_115646_5' -> 1.3498 secs
      - 'IMG_20150730_115645_4' -> 1.6471 secs
      - 'IMG_20150730_115645_3' -> 1.4249 secs
      - 'IMG_20150730_115647_4' -> 1.2952 secs
      - 'IMG_20150730_115647_2' -> 1.1961 secs
      - 'IMG_20150730_115647_3' -> 0.9967 secs
      - 'IMG_20150730_115647_1' -> 1.2609 secs
      - 'IMG_20150730_115651' -> 1.4530 secs
      - 'IMG_20150730_115648' -> 1.5387 secs
      - 'IMG_20150730_115647' -> 1.6602 secs
      - 'IMG_20150730_115648_1' -> 1.7500 secs
      - 'IMG_20150730_115648_3' -> 1.7439 secs
      - 'IMG_20150730_115650_3' -> 1.7528 secs
      - 'IMG_20150730_115645_1' -> 1.7576 secs
      - 'IMG_20150730_115650_4' -> 1.2932 secs
      - 'IMG_20150730_115648_4' -> 1.0025 secs
  - 'Post-processing' -> 0.2325 secs

- 'ImRestore init' -> 60.5456 secs
  - 'Computing features' -> 42.9186 secs
    - (processed) 'IMG_20150730_115647_1' -> 1.3225 secs
    - (processed) 'IMG_20150730_115642' -> 1.2921 secs
    - (processed) 'IMG_20150730_115644' -> 1.2912 secs
    - (processed) 'IMG_20150730_115644_1' -> 1.3234 secs
    - (processed) 'IMG_20150730_115644_2' -> 1.3165 secs
    - (processed) 'IMG_20150730_115644_3' -> 1.3775 secs
    - (processed) 'IMG_20150730_115645_1' -> 1.3199 secs
    - (processed) 'IMG_20150730_115645_3' -> 1.3305 secs
    - (processed) 'IMG_20150730_115645_4' -> 1.3880 secs
    - (processed) 'IMG_20150730_115646' -> 1.3216 secs
    - (processed) 'IMG_20150730_115646_1' -> 1.3255 secs
    - (processed) 'IMG_20150730_115646_2' -> 1.3432 secs
    - (processed) 'IMG_20150730_115646_3' -> 1.3724 secs
    - (processed) 'IMG_20150730_115646_5' -> 1.3488 secs
    - (processed) 'IMG_20150730_115647' -> 1.3940 secs
    - (processed) 'IMG_20150730_115647_2' -> 1.3935 secs
    - (processed) 'IMG_20150730_115647_3' -> 1.3864 secs
    - (processed) 'IMG_20150730_115647_4' -> 1.4172 secs
    - (processed) 'IMG_20150730_115648' -> 1.4173 secs
    - (processed) 'IMG_20150730_115648_1' -> 1.3990 secs
    - (processed) 'IMG_20150730_115648_2' -> 1.4315 secs
    - (processed) 'IMG_20150730_115648_3' -> 1.4193 secs
    - (processed) 'IMG_20150730_115648_4' -> 1.3970 secs
    - (processed) 'IMG_20150730_115649' -> 1.3939 secs
    - (processed) 'IMG_20150730_115649_1' -> 1.4014 secs
    - (processed) 'IMG_20150730_115649_2' -> 1.4128 secs
    - (processed) 'IMG_20150730_115649_4' -> 1.4715 secs
    - (processed) 'IMG_20150730_115650' -> 1.4092 secs
    - (processed) 'IMG_20150730_115650_3' -> 1.4488 secs
    - (processed) 'IMG_20150730_115650_4' -> 1.4529 secs
    - (processed) 'IMG_20150730_115651' -> 1.4281 secs
  - 'Restoring' -> 17.3907 secs
    - 'Matching' -> 1.1260 secs
    - 'Merging' -> 16.2619 secs
      - 'IMG_20150730_115646_2' -> 0.7408 secs
      - 'IMG_20150730_115646_1' -> 0.7740 secs
      - 'IMG_20150730_115646' -> 0.8301 secs
      - 'IMG_20150730_115646_5' -> 0.8209 secs
      - 'IMG_20150730_115645_4' -> 0.8168 secs
      - 'IMG_20150730_115645_3' -> 0.8326 secs
      - 'IMG_20150730_115647_4' -> 0.8695 secs
      - 'IMG_20150730_115647_2' -> 0.7697 secs
      - 'IMG_20150730_115647_3' -> 0.7746 secs
      - 'IMG_20150730_115647_1' -> 0.7760 secs
      - 'IMG_20150730_115651' -> 0.7935 secs
      - 'IMG_20150730_115648' -> 0.7729 secs
      - 'IMG_20150730_115647' -> 0.7952 secs
      - 'IMG_20150730_115650_3' -> 0.7746 secs
      - 'IMG_20150730_115648_2' -> 0.7966 secs
      - 'IMG_20150730_115648_3' -> 0.7744 secs
      - 'IMG_20150730_115648_1' -> 0.8100 secs
      - 'IMG_20150730_115645_1' -> 0.7947 secs
      - 'IMG_20150730_115648_4' -> 0.8743 secs
      - 'IMG_20150730_115650_4' -> 0.9046 secs
  - 'Post-processing' -> 0.2364 secs

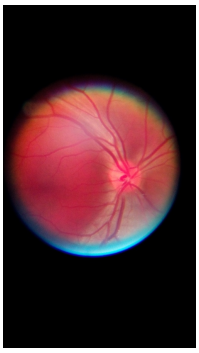
Figure 39. Result for set20 using command 'imrestore results/set20/*.* –lens –overwrite –cachePath {temp}'



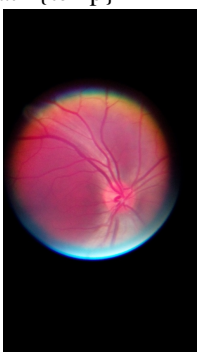
(a) IMG_20150730_130654



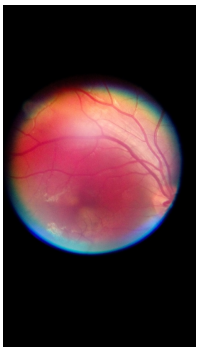
(b) IMG_20150730_130659



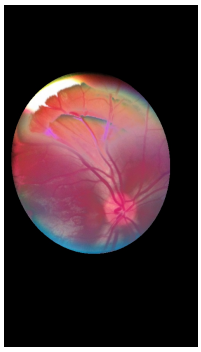
(c) IMG_20150730_130723



(d) IMG_20150730_130713



(e) IMG_20150730_130706



(f) Restored image

Table 22
Profiling for set20 using command 'imrestore results/set20/.* –lens –overwrite –cachePath {temp}'*

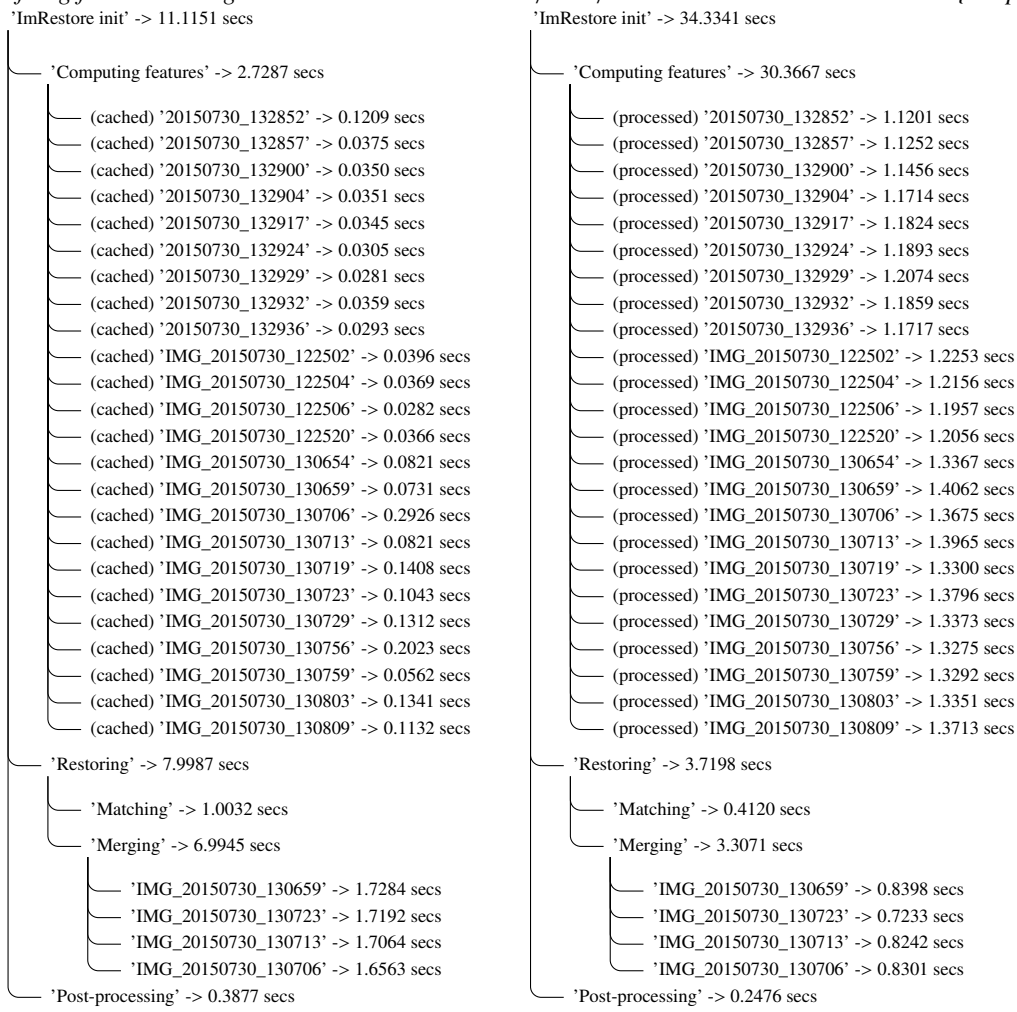
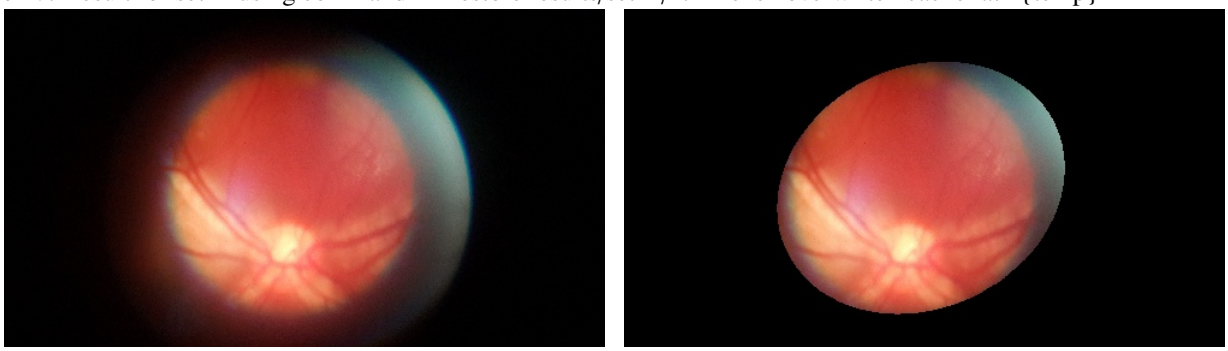


Figure 40. Result for set21 using command 'imrestore results/set21/*.* –lens –overwrite –cachePath {temp}'



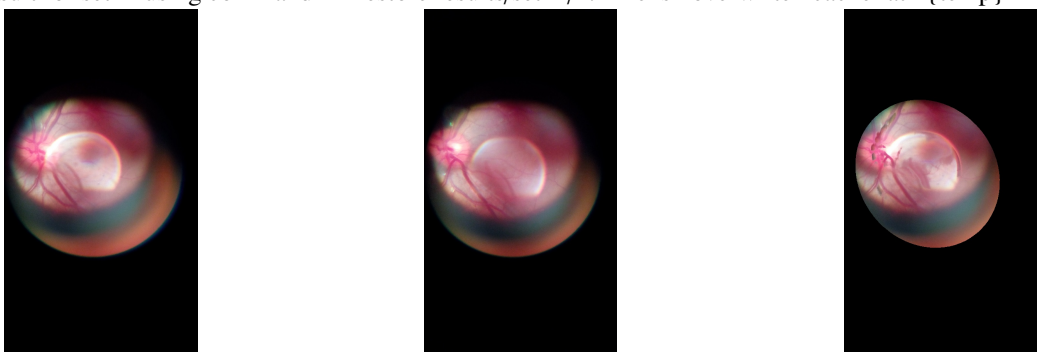
(a) 20150730_131816 (b) Restored image

Table 23
Profiling for set21 using command 'imrestore results/set21/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.5120 secs
- 'Computing features' -> 1.9658 secs
  - (cached) '20150730_131816' -> 0.1886 secs
  - (cached) '20150730_131823' -> 0.1233 secs
  - (cached) '20150730_131827' -> 0.0368 secs
  - (cached) '20150730_131834' -> 0.2139 secs
  - (cached) '20150730_131836' -> 0.0332 secs
  - (cached) '20150730_131839' -> 0.3446 secs
  - (cached) '20150730_131844' -> 0.0652 secs
  - (cached) '20150730_131913' -> 0.1980 secs
  - (cached) '20150730_131915' -> 0.1077 secs
  - (cached) '20150730_131916' -> 0.1804 secs
- 'Restoring' -> 0.0829 secs
  - 'Matching' -> 0.0164 secs
  - 'Merging' -> 0.0663 secs
- 'Post-processing' -> 0.4633 secs

'ImRestore init' -> 15.2817 secs
- 'Computing features' -> 14.9645 secs
  - (processed) '20150730_131816' -> 1.4477 secs
  - (processed) '20150730_131823' -> 1.4114 secs
  - (processed) '20150730_131827' -> 1.3893 secs
  - (processed) '20150730_131834' -> 1.5559 secs
  - (processed) '20150730_131836' -> 1.5134 secs
  - (processed) '20150730_131839' -> 1.4927 secs
  - (processed) '20150730_131844' -> 1.5271 secs
  - (processed) '20150730_131913' -> 1.4812 secs
  - (processed) '20150730_131915' -> 1.5830 secs
  - (processed) '20150730_131916' -> 1.5330 secs
- 'Restoring' -> 0.0415 secs
  - 'Matching' -> 0.0162 secs
  - 'Merging' -> 0.0252 secs
- 'Post-processing' -> 0.2756 secs

Figure 41. Result for set22 using command 'imrestore results/set22/*.* –lens –overwrite –cachePath {temp}'



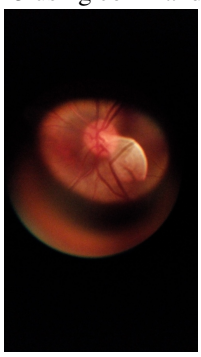
(a) IMG_20150730_121756 (b) IMG_20150730_121801 (c) Restored image

Table 24

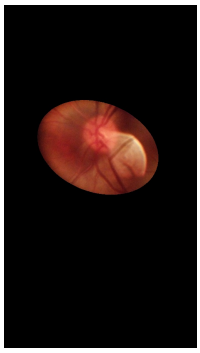
Profiling for set22 using command 'imrestore results/set22/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 3.5485 secs	'ImRestore init' -> 11.3040 secs
'Computing features' -> 1.1545 secs	'Computing features' -> 10.0899 secs
(cached) 'IMG_20150730_121750' -> 0.1755 secs	(processed) 'IMG_20150730_121750' -> 1.7666 secs
(cached) 'IMG_20150730_121756' -> 0.1219 secs	(processed) 'IMG_20150730_121756' -> 1.7344 secs
(cached) 'IMG_20150730_121801' -> 0.1463 secs	(processed) 'IMG_20150730_121801' -> 1.7166 secs
(cached) 'IMG_20150730_121804' -> 0.0424 secs	(processed) 'IMG_20150730_121804' -> 1.5761 secs
(cached) 'IMG_20150730_121804_1' -> 0.1547 secs	(processed) 'IMG_20150730_121804_1' -> 1.6669 secs
(cached) 'IMG_20150730_121804_2' -> 0.0518 secs	(processed) 'IMG_20150730_121804_2' -> 1.5886 secs
'Restoring' -> 1.9051 secs	'Restoring' -> 0.9617 secs
'Matching' -> 0.1368 secs	'Matching' -> 0.0899 secs
'Merging' -> 1.7679 secs	'Merging' -> 0.8715 secs
'IMG_20150730_121801' -> 1.5780 secs	'IMG_20150730_121801' -> 0.8273 secs
'Post-processing' -> 0.4889 secs	'Post-processing' -> 0.2525 secs

Figure 42. Result for set23 using command 'imrestore results/set23/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_20150730_122200



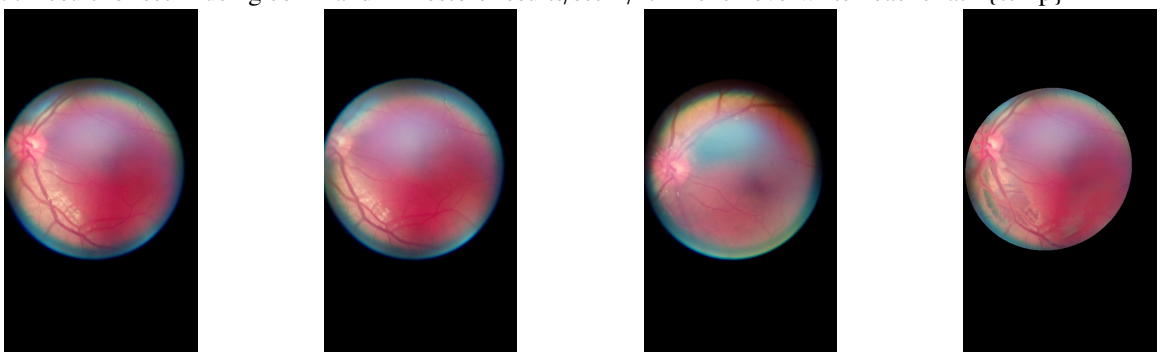
(b) Restored image

Table 25

Profiling for set23 using command 'imrestore results/set23/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.9900 secs	'ImRestore init' -> 15.4115 secs
'Computing features' -> 2.0860 secs	'Computing features' -> 15.0188 secs
(cached) 'IMG_20150730_122140' -> 0.1654 secs	(processed) 'IMG_20150730_122140' -> 1.6928 secs
(cached) 'IMG_20150730_122154' -> 0.1608 secs	(processed) 'IMG_20150730_122154' -> 1.9413 secs
(cached) 'IMG_20150730_122200' -> 0.2576 secs	(processed) 'IMG_20150730_122200' -> 1.7696 secs
(cached) 'IMG_20150730_122216' -> 0.1510 secs	(processed) 'IMG_20150730_122216' -> 1.5820 secs
(cached) 'IMG_20150730_122225' -> 0.0641 secs	(processed) 'IMG_20150730_122225' -> 1.6442 secs
(cached) 'IMG_20150730_122227' -> 0.2141 secs	(processed) 'IMG_20150730_122227' -> 1.6105 secs
(cached) 'IMG_20150730_122229' -> 0.1084 secs	(processed) 'IMG_20150730_122229' -> 1.6106 secs
(cached) 'IMG_20150730_122235' -> 0.1704 secs	(processed) 'IMG_20150730_122235' -> 1.5659 secs
(cached) 'IMG_20150730_122237' -> 0.0422 secs	(processed) 'IMG_20150730_122237' -> 1.5361 secs
'Restoring' -> 0.5488 secs	'Restoring' -> 0.1205 secs
'Matching' -> 0.0404 secs	'Matching' -> 0.0558 secs
'Merging' -> 0.5082 secs	'Merging' -> 0.0645 secs
'Post-processing' -> 0.3553 secs	'Post-processing' -> 0.2722 secs

Figure 43. Result for set24 using command 'imrestore results/set24/*.* –lens –overwrite –cachePath {temp}'



(a) IMG_20150730_125216 (b) IMG_20150730_125212 (c) IMG_20150730_125228 (d) Restored image

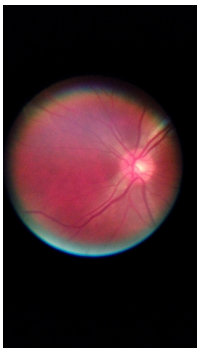
Table 26

Profiling for set24 using command 'imrestore results/set24/.* –lens –overwrite –cachePath {temp}'*

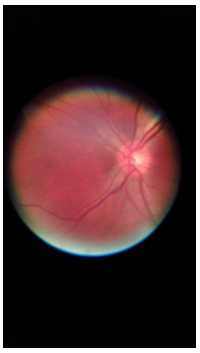
'ImRestore init' -> 4.9750 secs
- 'Computing features' -> 2.1716 secs
  - (cached) 'IMG_20150730_125212' -> 0.2864 secs
  - (cached) 'IMG_20150730_125216' -> 1.1185 secs
  - (cached) 'IMG_20150730_125222' -> 0.1188 secs
  - (cached) 'IMG_20150730_125228' -> 0.0471 secs
- 'Restoring' -> 2.4085 secs
  - 'Matching' -> 0.0534 secs
  - 'Merging' -> 2.3547 secs
    - 'IMG_20150730_125212' -> 0.9259 secs
    - 'IMG_20150730_125228' -> 1.4046 secs
- 'Post-processing' -> 0.3949 secs

'ImRestore init' -> 8.3052 secs
- 'Computing features' -> 7.0466 secs
  - (processed) 'IMG_20150730_125212' -> 1.7509 secs
  - (processed) 'IMG_20150730_125216' -> 1.7781 secs
  - (processed) 'IMG_20150730_125222' -> 1.7383 secs
  - (processed) 'IMG_20150730_125228' -> 1.7527 secs
- 'Restoring' -> 1.0156 secs
  - 'Matching' -> 0.0418 secs
  - 'Merging' -> 0.9734 secs
    - 'IMG_20150730_125212' -> 0.9560 secs
- 'Post-processing' -> 0.2430 secs

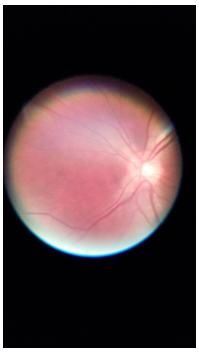
Figure 44. Result for set25 using command 'imrestore results/set25/*.* –lens –overwrite –cachePath {temp}'



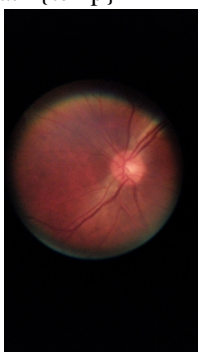
(a) IMG_20150730_125349



(b) IMG_20150730_125354



(c) IMG_20150730_125344



(d) IMG_20150730_125402

(e) Restored image

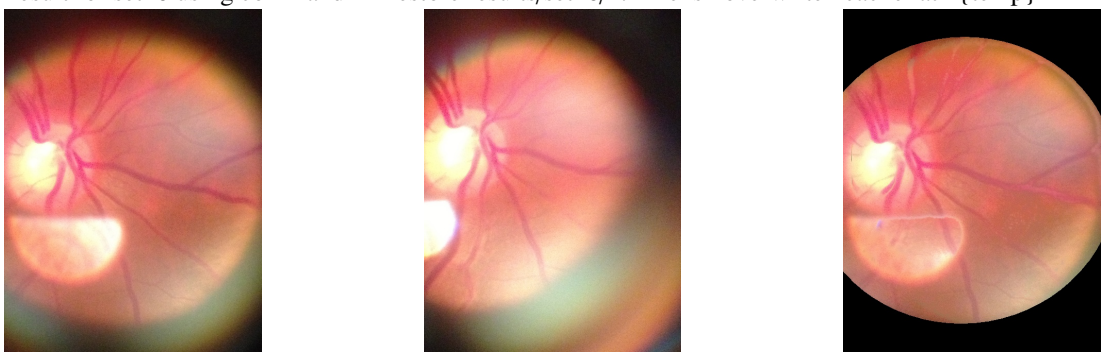
Table 27

Profiling for set25 using command 'imrestore results/set25/.* –lens –overwrite –cachePath {temp}'*

- 'ImRestore init' -> 5.7722 secs
  - 'Computing features' -> 1.8874 secs
    - (cached) 'IMG_20150730_125339' -> 0.2127 secs
    - (cached) 'IMG_20150730_125344' -> 0.3839 secs
    - (cached) 'IMG_20150730_125349' -> 0.2096 secs
    - (cached) 'IMG_20150730_125354' -> 0.1148 secs
    - (cached) 'IMG_20150730_125358' -> 0.1122 secs
    - (cached) 'IMG_20150730_125402' -> 0.0845 secs
    - (cached) 'IMG_20150730_125408' -> 0.1056 secs
    - (cached) 'IMG_20150730_125411' -> 0.2205 secs
  - 'Restoring' -> 3.4196 secs
    - 'Matching' -> 0.1055 secs
    - 'Merging' -> 3.3136 secs
      - 'IMG_20150730_125354' -> 1.0118 secs
      - 'IMG_20150730_125344' -> 1.0137 secs
      - 'IMG_20150730_125402' -> 0.8530 secs
  - 'Post-processing' -> 0.4652 secs

- 'ImRestore init' -> 29.8481 secs
  - 'Computing features' -> 22.9239 secs
    - (processed) 'IMG_20150730_125339' -> 3.2190 secs
    - (processed) 'IMG_20150730_125344' -> 2.6912 secs
    - (processed) 'IMG_20150730_125349' -> 2.9007 secs
    - (processed) 'IMG_20150730_125354' -> 2.6710 secs
    - (processed) 'IMG_20150730_125358' -> 2.9841 secs
    - (processed) 'IMG_20150730_125402' -> 2.8605 secs
    - (processed) 'IMG_20150730_125408' -> 2.6375 secs
    - (processed) 'IMG_20150730_125411' -> 2.8093 secs
  - 'Restoring' -> 6.1433 secs
    - 'Matching' -> 0.1722 secs
    - 'Merging' -> 5.9704 secs
      - 'IMG_20150730_125354' -> 1.7722 secs
      - 'IMG_20150730_125344' -> 1.8453 secs
      - 'IMG_20150730_125402' -> 2.2211 secs
  - 'Post-processing' -> 0.7809 secs

Figure 45. Result for set26 using command 'imrestore results/set26/*.* –lens –overwrite –cachePath {temp}'



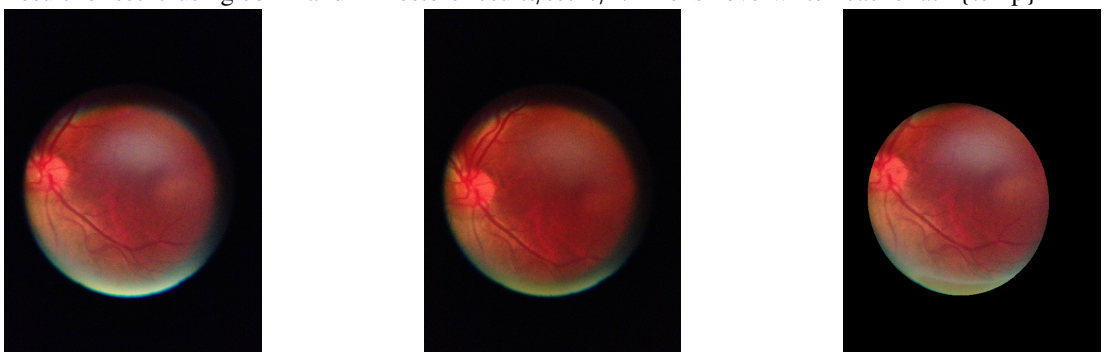
(a) im1_1 (b) im1_2 (c) Restored image

Table 28

Profiling for set26 using command 'imrestore results/set26/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 3.5687 secs	'ImRestore init' -> 12.3033 secs
'Computing features' -> 1.6030 secs	'Computing features' -> 11.1193 secs
(cached) 'im1_1' -> 0.1539 secs	(processed) 'im1_1' -> 1.9545 secs
(cached) 'im1_2' -> 0.1679 secs	(processed) 'im1_2' -> 1.8902 secs
(cached) 'IMG_0243' -> 0.1319 secs	(processed) 'IMG_0243' -> 2.1546 secs
(cached) 'IMG_0244' -> 0.1629 secs	(processed) 'IMG_0244' -> 1.6863 secs
(cached) 'IMG_0247' -> 0.0901 secs	(processed) 'IMG_0247' -> 1.7245 secs
(cached) 'IMG_0248' -> 0.2915 secs	(processed) 'IMG_0248' -> 1.6601 secs
'Restoring' -> 1.5432 secs	'Restoring' -> 0.9438 secs
'Matching' -> 0.0881 secs	'Matching' -> 0.0666 secs
'Merging' -> 1.4548 secs	'Merging' -> 0.8768 secs
'im1_2' -> 1.2725 secs	'im1_2' -> 0.8522 secs
'Post-processing' -> 0.4226 secs	'Post-processing' -> 0.2402 secs

Figure 46. Result for set27 using command 'imrestore results/set27/*.* –lens –overwrite –cachePath {temp}'



(a) im5_3 (b) im5_1 (c) Restored image

Table 29
Profiling for set27 using command 'imrestore results/set27/.* –lens –overwrite –cachePath {temp}'*

'ImRestore init' -> 2.6286 secs
- 'Computing features' -> 0.3586 secs
  - (cached) 'im5_1' -> 0.0688 secs
  - (cached) 'im5_2' -> 0.0404 secs
  - (cached) 'im5_3' -> 0.1711 secs
- 'Restoring' -> 1.9799 secs
  - 'Matching' -> 0.0266 secs
  - 'Merging' -> 1.9530 secs
    - 'im5_1' -> 1.2304 secs
- 'Post-processing' -> 0.2901 secs

'ImRestore init' -> 7.5800 secs
- 'Computing features' -> 5.2576 secs
  - (processed) 'im5_1' -> 1.7468 secs
  - (processed) 'im5_2' -> 1.7661 secs
  - (processed) 'im5_3' -> 1.7243 secs
- 'Restoring' -> 2.0602 secs
  - 'Matching' -> 0.0306 secs
  - 'Merging' -> 2.0292 secs
    - 'im5_1' -> 1.0750 secs
    - 'im5_2' -> 0.9542 secs
- 'Post-processing' -> 0.2623 secs

Conclusions

Recommendations

Future Work

References

Antal, B. & Hajdu, A. (2012, January). "Improving microaneurysm detection using an optimally selected subset of candidate extractors and preprocessing methods." *Pattern Recognition*, *45*(1), 264–270. doi:10.1016/j.patcog.2011.06.010

Askew, D. A., Crossland, L., Ware, R. S., Begg, S., Cranstoun, P., Mitchell, P., & Jackson, C. L. (2012, September). "Diabetic retinopathy screening and monitoring of early stage disease in general practice: Design and methods." *Contemporary Clinical Trials*, *33*(5), 969–975. doi:10.1016/j.cct.2012.04.011

CMISS. (n.d.). Sigmoid filter — Continuum Mechanics, Image analysis, Signal processing and System Identification. Retrieved April 20, 2016, from http://www.cmiss.org/cmgui/wiki/SigmoidFilter

Cortesi, D., Bajo, G., Caban, W., & McMillan, G. (n.d.). PyInstaller Manual — PyInstaller 3.2 documentation. Retrieved July 25, 2016, from http://pythonhosted.org/PyInstaller/

CVonline. (2004). Bilateral Filtering. Retrieved April 1, 2016, from http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html

Cython.org. (n.d.). Cython: C-Extensions for Python. Retrieved March 9, 2016, from http://cython.org/

Darel Rex Finley. (n.d.). Area of a polygon algorithm - Math Open Reference. Retrieved from http://www.mathopenref.com/coordpolygonarea2.html

David, T. (2016a). RRtoolbox documentation. Retrieved from https://github.com/davtoh/RRtools/blob/master/documentation/_build/latex/RRtoolbox.pdf

David, T. (2016b). RRtools - Retinal Restoration Tools. Retrieved July 25, 2016, from https://github.com/davtoh/RRtools%20https://github.com/davtoh/RRtools/blob/master/documentation/_build/latex/RRtoolbox.pdf

Faust, O., U, R. A., Ng, E. Y. K., Ng, K.-H., & Suri, J. S. (2010, April). "Algorithms for the Automated Detection of Diabetic Retinopathy Using Digital Fundus Images: A Review." *Journal of Medical Systems*, *36*(1), 145–157. doi:10.1007/s10916-010-9454-7

Fierval. (2015). Fast Image Pre-processing with OpenCV 2.4, C++, CUDA: Memory, CLAHE. Retrieved June 3, 2016, from http://funcvis.org/blog/?p=54

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2004). Digital Image Processing Using Matlab - Gonzalez Woods & Eddins.pdf. doi:10.1117/1.3115362

IEEE. (2000). IEEE SA - 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. Retrieved March 31, 2016, from https://standards.ieee.org/findstds/standard/1471-2000.html

ITK. (n.d.). ITK SigmoidImageFilter. Retrieved April 20, 2016, from http://www.itk.org/Doxygen/html/classitk_1_1SigmoidImageFilter.html

James R. Bozeman & Matthew Pilling. (2015). The Convexity Ratio and Applications. Retrieved April 25, 2016, from http://lyndonstate.edu/wp-content/uploads/2015/02/The-Convexity-Ratio-and-Applications.pdf

K, A. R. & Mordvintsev, A. (2013). Histograms - 2: Histogram Equalization. Retrieved June 7, 2016, from http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histograms/py_histogram_equalization/py_histogram_equalization.html

Martínez Rubio, M., Moya Moya, M., Bellot Bernabé, A., & Belmonte Martínez, J. (2012, December). "Diabetic retinopathy screening and teleophthalmology." *Archivos de la Sociedad Española de Oftalmología*, *87*(12), 392–5. doi:10.1016/j.oftal.2012.04.004

Médioni, G. (2005). *Emerging Topics in Computer Version*. Prentice Hall Computer. Retrieved from https://books.google.com.co/books?id=qV90QgAACAAJ

OpenCV. (n.d.). Image Filtering — OpenCV 2.4.12.0 documentation. Retrieved April 1, 2016, from http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#bilateralfilter

deployment - Python Wiki. (n.d.). Retrieved July 25, 2016, from https://wiki.python.org/moin/deployment

Phillippe Kruchten. (1995, November). "Architectural Blueprints—The "4+1" View Model of Software Architecture." *IEEE Software*, *12*(6), 540–555. doi:10.1145/216591.216611

Python Software Foundation. (2015). Glossary — Python 2.7.11 documentation. Retrieved February 16, 2016, from https://docs.python.org/2/glossary.html

Rey Otero, I. & Delbracio, M. (2014, December). "Anatomy of the SIFT Method." *Image Processing On Line*, *4*, 370–396. doi:10.5201/ipol.2014.82

SciPy. (2015a). Numpy for Matlab users — NumPy v1.11.dev0 Manual. Retrieved March 20, 2016, from https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html

SciPy. (2015b). Quickstart tutorial — NumPy v1.11.dev0 Manual. Retrieved March 20, 2016, from https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

Scipy. (2015). The N-dimensional array (ndarray) — NumPy v1.10 Manual. Retrieved March 20, 2016, from http://docs.scipy.org/doc/numpy/reference/arrays.ndarray.html

Sopharak, A., Uyyanonvara, B., & Barman, S. (2013, July). "Simple hybrid method for fine microaneurysm detection from non-dilated diabetic retinopathy retinal images." *Computerized Medical Imaging and Graphics*. Retinal Image Analysis, *37*(5–6), 394–402. doi:10.1016/j.compmedimag.2013.05.005

Volodymyr Kindratenko. (2002). CVonline: Convexity ratio (Regions). Retrieved April 25, 2016, from http://homepages.inf.ed.ac.uk/cgi/rbf/CVONLINE/entries.pl?TAG156

WebMD. (2014). Type 1 Diabetes - Prevention. Retrieved from http://www.webmd.com/diabetes/tc/type-1-diabetes-prevention

WHO. (2016). Diabetes: Fact sheet. Retrieved February 16, 2016, from http://www.who.int/mediacentre/factsheets/fs312/en/

WHO & IDF. (2006). Definition and diagnosis of diabetes mellitus and intermediate hyperglycaemia. doi:ISBN9241594934

Appendix A RRtoolbox: Python Implementation

RRtoolbox is a package created while realizing the project algorithms for their testing, prototyping and demonstration. Some of the algorithm implementations are explained here, how to set them up, their class diagrams and other program-related topics that are not in the main document.

A.1 Procedures to set-up RRtoolbox

The algorithm was implemented in python using some of its build-in functions and others that are provided via modules. If the OS is Linux there is a high probability that python 2.7 is already installed with the distribution but if that is not the case it can be installed via the package manager. Here it will be assumed that the Linux distribution is based in Debian and each time something is typed in the console it will be referenced with a "$" sign.

A.1.1 Install Python.

Python is a programming language.

For Linux. In Linux it is pretty straightforward:

```
$ sudo apt-get install python2.7
```

For Windows.

- Go to www.python.org
- Under Downloads select Windows
- Select the latest release for Python 2
- Download according to the Windows version and computer architecture
- Once downloaded run the installer and follow the Setup steps.
- Once installed register python executable path in a Windows variable.

Register a path in a windows variable (window 7).

- Go to Control Panel (menu)
- System and security (icon)
- System (icon)
- Advanced system settings (left menu)
- Environment variables (button)
- Under the path variable append the python variable
- Click Ok button
- Now under the command Prompt the user should be able to execute a python command.

To append a variable to window path.

- Double click in Path
- Once the "Edit System Variable" Dialog is opened
- Under "Variable value" box: at the end of all its variables insert the following

  ```
  ; C:\Python27
  ```

- Click OK button

A.1.2 Install pip. Pip is a python module used as a package manager to install more packages.

For Linux. In Linux it is pretty straightforward:

```
$ sudo apt-get install python-pip
```

For Windows.

- Go to https://pip.pypa.io/en/stable/installing/
- Download the get-pip.py
- Run the script

  ```
  $ python get-pip.py
  ```

A.1.3 Install packages to Python. To install numpy, matplotlib, dill and joblib in both Linux and Windows:

```
$ pip install numpy
$ pip install matplotlib
$ pip install dill
$ pip install joblib
$ pip install pyinstaller
```

Numpy and matplotlib pacakges also can be obtained installing Scipy which comes with them and other scientific modules.

```
$ pip install scipy
```

Or follow the instructions at https://www.scipy.org/install.html. To install OpenCV 2.4.13 (or newer versions from the 2.4 release):

Code 4: Install OpenCV

```
# https://help.ubuntu.com/community/OpenCV
# http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html
version="$(wget -q -O - http://sourceforge.net/projects/opencvlibrary/files/opencv-unix | egrep -m1 -o
 ↪  '\"[2](\.[0-9]+)+' | cut -c2-)"
echo "Preparing to install OpenCV" $version
# pre-requisites
mkdir OpenCV
cd OpenCV
echo "Removing any pre-installed ffmpeg and x264"
sudo apt-get -qq remove ffmpeg x264 libx264-dev
echo "Installing Dependenices"
sudo apt-get -qq install libopencv-dev build-essential checkinstall cmake pkg-config yasm libjpeg-dev
 ↪  libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine2-dev
 ↪  libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev python-dev python-numpy libtbb-dev
 ↪  libqt4-dev libgtk2.0-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev
 ↪  libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils ffmpeg cmake qt5-default checkinstall
# downloading
if [ -f "OpenCV-$version.zip" ]
then
    echo "using OpenCV-$version.zip already in folder."
else
    echo "Downloading OpenCV" $version
    wget -O OpenCV-$version.zip
     ↪  http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/$version/opencv-"$version".zip/download
fi
# installing
echo "Installing OpenCV" $version
unzip OpenCV-$version.zip
cd opencv-$version
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D
 ↪  BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
 ↪  BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..
make -j2
sudo checkinstall
sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
sudo ldconfig
echo "OpenCV" $version "ready to be used"
```

In Linux.

- create an empty file "install_opencv.sh" and place the code Code 4
- run in the terminal

```
$ chmod +x install_opencv.sh
$ ./install_opencv.sh
```

Note: the script install_opencv.sh will download and install the latest OpenCV release of 2.4 version. It builds the release from source which takes a considerate amount of time and computer power so an stable power connection and closing all unrelated programs is recommended.

In Windows. Download the installer at https://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.13/opencv-2.4.13.exe/download and install.

1 Annexes