

Семинар 3.2. Структуры данных и дополненная реальность :)

Разработал: Максимов А.Н.

Версия 1. 07/2017

Содержание

- Структуры данных
 - динамический массив
 - список
- Пространства имен, исключения
- OpenCv
- `cv::Mat`
- Выделение контрастных объектов при помощи ступенчатого преобразования
- Задача “кротовья нора”
- Пример полупрозрачные фигуры

Идея этого курса

Идея этого курса появилась, когда мой коллега прислал ссылку на ролик о дополненной реальности

<https://www.youtube.com/watch?v=kPMHcanq0xM>



Как это реализовать?

Очень упрощенно:

- Захватываем картинку из видеопотока
- Выделяем объект реального мира на картинке
- Рисуем на результирующем кадре наше изображение так, чтобы оно корректно располагалось относительно найденного объекта
- Отображаем на экран результирующий кадр

Как это реализовать?

Очень упрощенно:

- Захватываем картинку из видеопотока
- Выделяем объект реального мира на картинке
- Рисуем на результирующем кадре наше изображение так, чтобы оно корректно располагалось относительно найденного объекта
- Отображаем на экран результирующий кадр

Pnm формат

Простые форматы хранения изображений portable pixmap (иногда определяемые как PNM): цветных (PPM), полутоновых (PGM) и чёрно-белых (PBM)

Описание тут: https://en.wikipedia.org/wiki/Netpbm_format

Пример:

```
P2
# Shows the word "FEEP" (example from Netpbm man page on PGM)
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

OpenCV

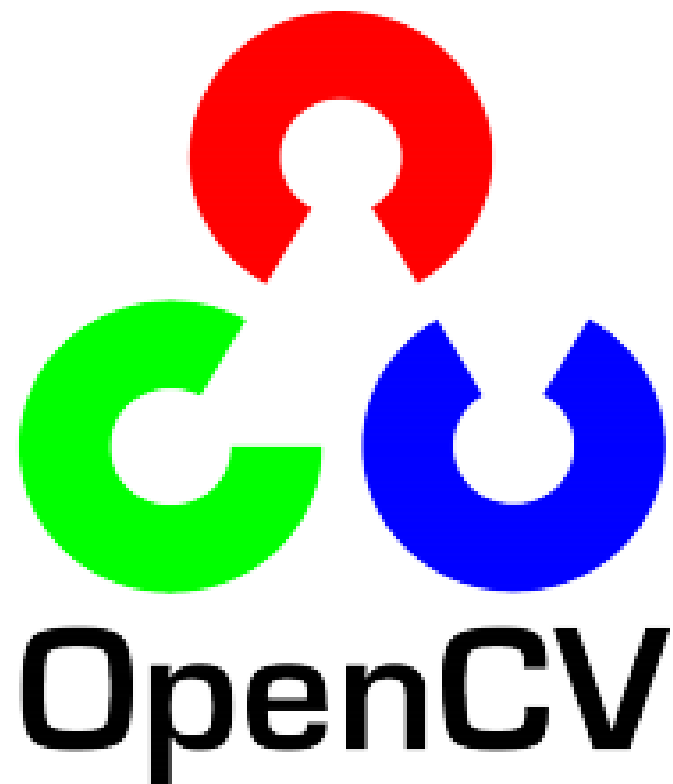
OpenCV - библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, и др. языков. Лицензии BSD.

Сайт: www.opencv.org

Текущая версия: 3.2 (2016-12-23)

Разбита на модули:

- Core – базовые структуры
- HighGUI - интерфейс, input/output
- ImgProc, Calib3D и др.



Модули OpenCV

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- core - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

- imgproc - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.

- video - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

- calib3d - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

- features2d - salient feature detectors, descriptors, and descriptor matchers.

- objdetect - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

- highgui - an easy-to-use interface to simple UI capabilities.

- videoio - an easy-to-use interface to video capturing and video codecs.

- gpu - GPU-accelerated algorithms from different OpenCV modules.

- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

OpenCV 3.1 – как поставить?

1. Visual Studio 2015

<https://lithiumdenis.wordpress.com/2016/01/28/windows-%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0-opencv-3-1-%D0%BD%D0%B0-vs-2015/>

2. Ubuntu Linux

<http://embedonix.com/articles/image-processing/installing-opencv-3-1-0-on-ubuntu/>

Изображение в OpenCV – `cv::Mat`

`Mat` – ключевая структура в OpenCV представляет матрицу или изображение.

Матричные операции с cv::Mat - сложение

```
#include <iostream>

#include <opencv2/core/core.hpp>

using namespace std;
using namespace cv;

void addMatrixExample() {
    float a[2][2] = {{1, 2}, {3, 4}};
    float b[2][2] = {{5, 6}, {7, 8}};
    Mat A = Mat(2, 2, CV_32FC1, a);
    Mat B = Mat(2, 2, CV_32FC1, b);
    Mat C;

    C = A + B;

    cout << "C =" << endl << " " << C << endl << endl;
}

int main() {
    addMatrixExample();
}
```

Результат:

C =
[6, 8;
10, 12]

OpenCV namespace cv

```
#include <opencv/cv.h>

int main () {
    cv::Mat image ( cv :: Size (256 ,256) , CV_8UC3 ) ;
    for (int x = 0; x < image.size ().width , ++ x) {
        for (int y = 0; y < image.size().height ; ++ y) {
            image.at < cv :: Vec3b >(y ,x)[0] = x;           // Blue
            image.at < cv :: Vec3b >(y ,x)[1] = y;           // Green
            image.at < cv :: Vec3b >(y ,x)[2] = (x+y) /2;    // Red
        }
    }
    return 0;
}
```

Список (UML диаграмма)

Список:

Какой тип связи?

Dependency 

Aggregation 

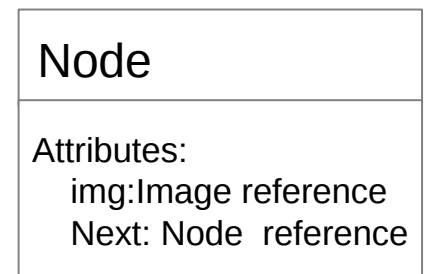
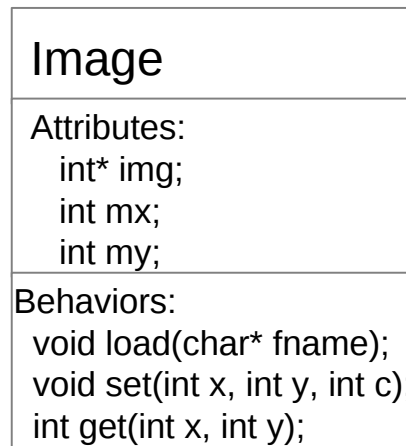
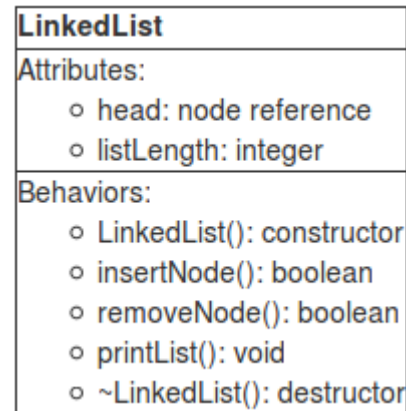
Inheritance 

Composition 

Association 

Directed Association 

Interface Type Implementation 



Пример из <http://pumpkinprogrammer.com/2014/06/13/c-tutorial-intro-to-linked-lists/>

Загрузка изображения из файла

```
vector<cv::Mat> stickers;  
cv::Mat image = cv::imread("image_board.jpg");
```

Выделение контрастных стикеров

```
void recogniseStickersByThreshold(cv::Mat image, std::vector<cv::Mat> &stickers) {  
    cv::Mat image_hsv;  
    std::vector< std::vector<cv::Point> > contours;  
    cv::cvtColor(image, image_hsv, cv::COLOR_BGR2HSV ); // Преобразуем в hsv  
    cv::Mat tmp_img(image.size(), CV_8U);  
    // Выделение подходящих по цвету областей. Цвет задается константой :)  
    cv::inRange(image_hsv, cv::Scalar(key_light-2, key_sat-20, key_hue-35),  
cv::Scalar(key_light+2, key_sat+20, key_hue+35), tmp_img);  
    // "Замазать" огрехи в при выделении по цвету  
    cv::dilate(tmp_img, tmp_img, cv::Mat(), cv::Point(-1, -1), 3);  
    cv::erode(tmp_img, tmp_img, cv::Mat(), cv::Point(-1, -1), 1);  
    //Выделение непрерывных областей  
    cv::findContours(tmp_img, contours, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);  
    for (uint i = 0; i < contours.size(); i++) {        cv::Mat sticker;  
        //Для каждой области определяем ограничивающий прямоугольник  
        cv::Rect rect = cv::boundingRect(contours[i]);  
        image(rect).copyTo(sticker);  
        stickers.push_back(sticker); //Добавить к массиву распознанных стикеров  
    }  
}
```

Отображение на экран

Отображение на экран

В изображение можно рисовать, например, прямоугольник

```
cv::rectangle(image,rect,cv::Scalar(0,250,0),2);
```

Изображение можно масштабировать

```
cv::resize(image,image,cv::Size(800,600));
```

Изображение можно отображать на экран :)

```
cv::imshow("tmp",image);
```


Прозрачные выноски

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

void transparenRect(cv::Mat image,int x,int y,int w,int h,cv::Scalar clr) {
    cv::Mat roi = image(cv::Rect(x, y, w, h));
    cv::Mat color(roi.size(), CV_8UC3, clr);
    double alpha = 0.3;
    cv::addWeighted(color, alpha, roi, 1.0 - alpha , 0.0, roi);
}

int main( int argc, char** argv ) {
    cv::Mat image;
    std::vector<cv::Mat> stickers;
    image = cv::imread(argv[1]);
    transparenRect(image,100,100,300,500,cv::Scalar(255, 255, 255));
    cv::imshow("image",image);
    cv::waitKey(0);
}
```

Умные указатели и OpenCV

В OpenCV есть свой класс умных указателей с разделяемым владением:

```
template<typename T>  
struct cv::Ptr< T >
```

Ptr похож на boost::shared_ptr и std::shared_ptr из C++11.

Пример:

```
#include "cvstd.hpp"
```

```
...
```

```
Ptr<LSDDetector> bd = LSDDetector::createLSDDetector();
```

```
..
```

```
bd->detect( imageMat, lines, 2, 1, mask );
```

http://docs.opencv.org/3.1.0/d0/de7/structcv_1_1Ptr.html#details

Особенности cv::Ptr

- Разделяемость умных указателей реализована при помощи счетчика ссылок – при наличии перекрестных ссылок объект не удалится !!! (аналогично `std::shared_ptr`)
- Безопасно параллельно читать (но не писать) указатель Ptr из нескольких потоков. (аналогично для Mat)

Стиль и cv::Ptr

Инстансы алгоритмов должны создаваться при помощи cv::makePtr или статического фабричного метода, если он есть:

// **Хорошая практика**

```
Ptr<SomeAlgo> algo = makePtr<SomeAlgo>(...);
```

```
Ptr<SomeAlgo> algo = SomeAlgo::create(...);
```

// **Плохая практика (считается устаревшей)**

```
Ptr<SomeAlgo> algo = new SomeAlgo(...);
```

```
SomeAlgo * algo = new SomeAlgo(...);
```

```
SomeAlgo algo(...);
```

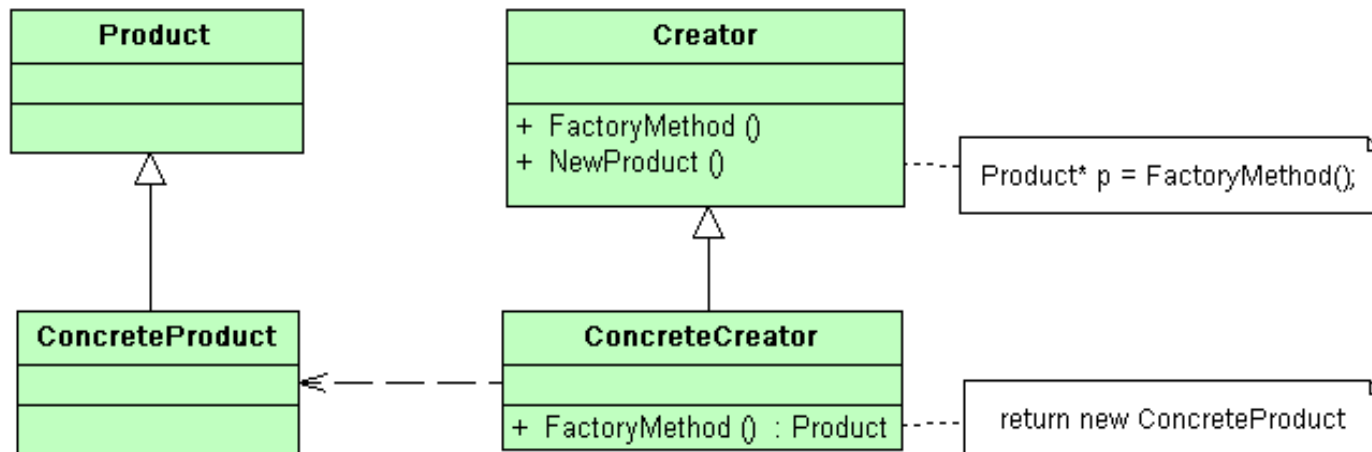
```
Ptr<SomeAlgo> algo = Algorithm::create<SomeAlgo>("name");
```

См. http://docs.opencv.org/trunk/db/dfa/tutorial_transition_guide.html

Про шаблон фабричный метод

Определяет интерфейс для создания объекта, но оставляет подклассам решение о том, какой класс инстанцировать. Фабричный метод позволяет классу делегировать создание подклассов. Используется, когда:

- классу заранее неизвестно, объекты каких подклассов ему нужно создавать.
- класс спроектирован так, чтобы объекты, которые он создаёт, специфицировались подклассами.
- класс делегирует свои обязанности одному из нескольких вспомогательных подклассов, и планируется локализовать знание о том, какой класс принимает эти обязанности на себя



Особенности cv::Ptr

- Разделяемость умных указателей реализована при помощи счетчика ссылок – при наличии перекрестных ссылок объект не удалится !!! (аналогично `std::shared_ptr`)
- Безопасно параллельно читать (но не писать) указатель Ptr из нескольких потоков. (аналогично для Mat)

Гистограммы

Расчет гистограмм для набора массивов

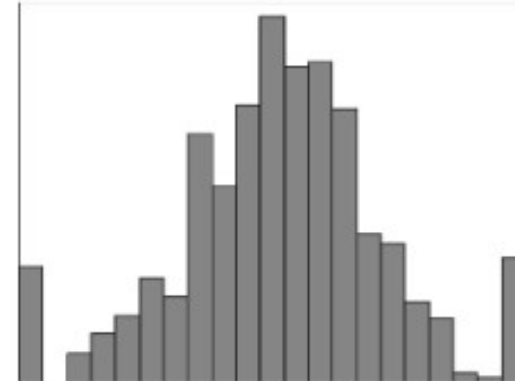
```
void calcHist(const Mat* images, int nimages, const int*  
    channels, InputArray mask, OutputArray hist, int dims,  
    const int* histSize, const float** ranges, bool uniform=true,  
    bool accumulate=false )
```

Гистограммы

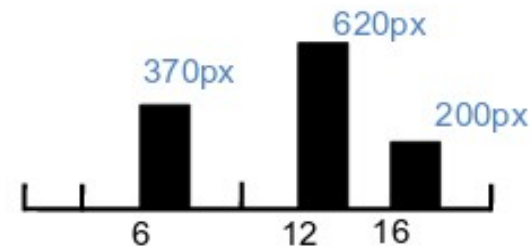
Для полутонового изображения гистограмма обладает следующими свойствами:

- это массив чисел
- каждый элемент массива содержит число точек с цветом, который попадает в соответствующий диапазон

Гистограммы



- Gray Levels: $\{0, 1, \dots, 255\}$
- Number of Bins: 32
- Number of intensity levels for each bin: $256 / 32 = 8$



<http://www.micc.unifi.it/lisanti/downloads/OpenCV-PPM-Histogram.pdf>

Гистограмма

Расчет гистограмм для набора массивов

```
void calcHist(const Mat* images, int nimages, const int*  
    channels, InputArray mask, OutputArray hist, int dims,  
    const int* histSize, const float** ranges, bool uniform=true,  
    bool accumulate=false )
```

Пример

```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/imgproc/imgproc.hpp"
...
int main(int, char**) {
    Mat gray=imread("image.jpg",0);
    namedWindow( "Gray", 1 );   imshow( "Gray", gray );
    int histSize = 256;   // bin size
    float range[] = { 0, 255 };
    const float *ranges[] = { range };
    MatND hist;
    calcHist( &gray, 1, 0, Mat(), hist, 1, &histSize, ranges, true, false ); // Calculate histogram
    double total;
    total = gray.rows * gray.cols;
    for( int h = 0; h < histSize; h++ )      { // Show the calculated histogram in command window
        float binVal = hist.at<float>(h);
        cout<<" "<<binVal;
    }
```

Пример

```
// Plot the histogram
```

```
int hist_w = 512; int hist_h = 400;
```

```
int bin_w = cvRound( (double) hist_w/histSize );
```

```
Mat histImage( hist_h, hist_w, CV_8UC1, Scalar( 0,0,0) );
```

```
normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat() );
```

```
for( int i = 1; i < histSize; i++ ) {
```

```
    line( histImage, Point( bin_w*(i-1), hist_h - cvRound(hist.at<float>(i-1)) ) ,
```

```
        Point( bin_w*(i), hist_h - cvRound(hist.at<float>(i)) ) ,
```

```
        Scalar( 255, 0, 0), 2, 8, 0 );
```

```
}
```

```
namedWindow( "Result", 1 );  imshow( "Result", histImage );
```

```
waitKey(0);
```

```
return 0;
```

```
}
```

Сравнение гистограмм

`double compareHist(const SparseMat& H1, const SparseMat& H2, int method)`

Параметры:

H1 – First compared histogram.

H2 – Second compared histogram of the same size as H1 .

method – Метод сравнения:

CV_COMP_CORREL Correlation

CV_COMP_CHISQR Chi-Square

CV_COMP_CHISQR_ALT Alternative Chi-Square

CV_COMP_INTERSECT Intersection

CV_COMP_BHATTACHARYYA Bhattacharyya distance

CV_COMP_HELLINGER Synonym for CV_COMP_BHATTACHARYYA

CV_COMP_KL_DIV Kullback-Leibler divergence

<http://docs.opencv.org/3.0-beta/modules/imgproc/doc/histograms.html#comparehist>

Методы сравнения гистограмм

CV_COMP_CORREL - корреляционный метод

$$d_{\text{correl}}(H_1, H_2) = \frac{\sum_i H_1'(i) \cdot H_2'(i)}{\sqrt{\sum_i H_1'^2(i) \cdot H_2'^2(i)}} \quad (14.1)$$

где, N равняется числу столбцов гистограммы.

Возвращаемое значение лежит в интервале $[-1, 1]$, 1 - максимальное соответствие, -1 - максимальное несоответствие, 0 - нет никакой корреляции.

CV_COMP_CHISQR - хи-квадрат

$$d_{\text{chi-square}}(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)} \quad (14.2)$$

Возвращаемое значение лежит в интервале $[0, \infty)$. 0 - максимальное соответствие, а крайнее несоответствие зависит от количества элементов гистограммы.

<http://blog.vidikon.com/?cat=1&paged=7>

Методы сравнения гистограмм

CV_COMP_INTERSECT - пересечение

$$d_{\text{intersection}}(H_1, H_2) = \sum_i \min(H_1(i), H_2(i)) \quad (14.3)$$

Если гистограммы нормализованы к 1, то совершенное соответствие гистограмм это 1, а совершенное несоответствие - 0.

CV_COMP_BHATTACHARYYA - расстояние Бхатачария

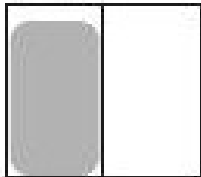
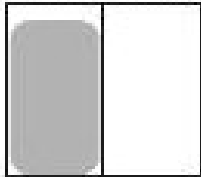
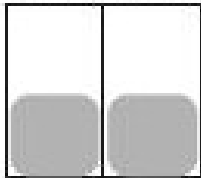
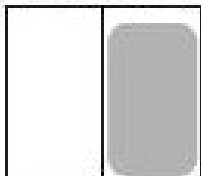
$$d_{\text{Bhattacharyya}}(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_i H_1(i) \cdot \sum_i H_2(i)}}} \quad (14.4)$$

Совершенное соответствие - это 0, несоответствие - это 1.

Методы сравнения гистограмм

Histograms:

Matching measures:

Model:	Correlation:	Chi square:	Intersection	Bhattacharyya:	EMD:
					
Exact match:					
	1.0	0.0	1.0	0.0	0.0
Half match:					
	0.7	0.67	0.5	0.55	0.5
Total mis-match:					
	-1.0	2.0	0.0	1.0	1.0

Прочеть.

1. <https://eetingcpp.com/index.php/br/items/an-overview-on-smart-pointers.html>

Вычитание фона

- describes basic functions that enable building statistical model of background for its further subtraction.
- Background statistics functions:
 - ✓ Average
 - ✓ Standard deviation
 - ✓ Running average

$$\mu_{ij}^t = \alpha \cdot I_{ij}^t + (1 - \alpha) \cdot \mu_{ij}^{t-1}, 0 \leq \alpha \leq 1$$



<http://ce.sharif.edu/courses/86-87/1/ce823/resources/root/OpenCv/OpenCV%20Tutorial.pdf>

Пример:

http://docs.opencv.org/3.2.0/d1/dc5/tutorial_background_subtraction.html

Задания

Task1_hist. Написать программу для расчета гистограмм для набора изображений, которые находятся в папке

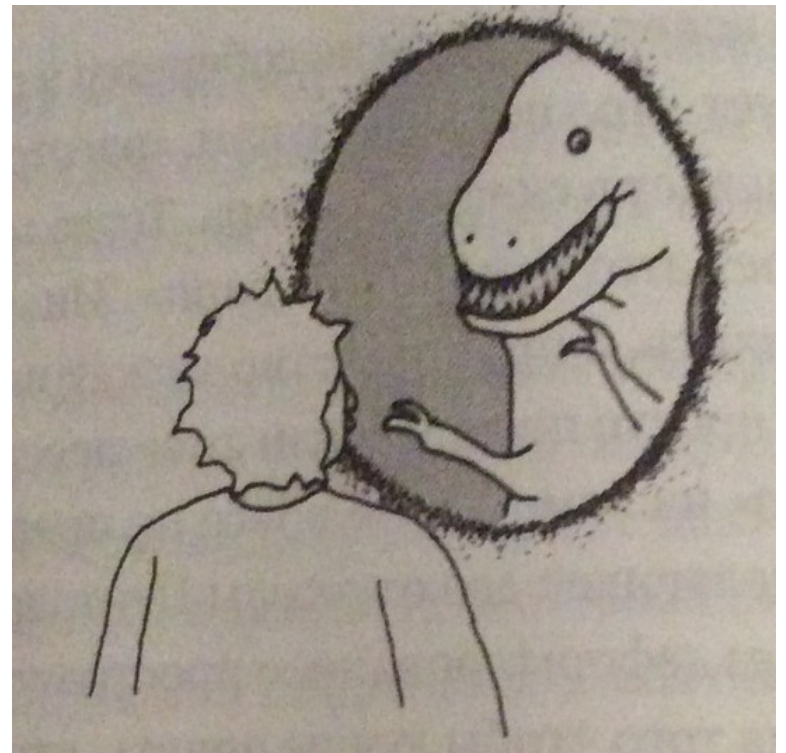
- Написать функктор, позволяющий упорядочить набор изображений по степени похожести (сравниваем на основе гистограмм) на образец при помощи функции `std::sort`

<https://varrunr.wordpress.com/2012/07/30/passing-extra-parameters-to-sort/>
)

- написать функцию сохранения гистограмм для изображений в папке в файл
- реализовать функцию сравнения изображения по гистограмме с набором гистограмм, которые считаны из файла

Реализация “Кротовой норы”

Физики представляют
“кротовую нору” так :)



Задания

Task2. Реализовать программу “кротовая нора”

- читает изображение с доской и стикером;
- читает изображение “динозавра” (или группу изображений динозавра dino1.jpg ... dinoN.jpg помещает в односвязный список)
- находит розовый стикер на доске
- отображает динозавра “привязанного” к точке розового стикера (раз в секунду отображает следующее фото динозавра)

Порядок выполнения

1. Необходимо создать репозиторий на github и прислать ссылку на репозиторий на rt.practic@dev.rtsoft.ru
2. Для каждой задачи необходимо создавать свой каталог, например, task1
3. Код задачи должен собираться. Желательно, чтобы добавить Makefile

Литература по C++

1. Бьерн Страуструп. Язык программирования C++.
2. Тут есть несколько неплохих туториалов. <http://www.cprogramming.com/tutorial.html>
3. Introduction to Programming Concepts in C++ (хорошие базовые лекции по C++ на английском)

<http://staffwww.fullcoll.edu/brippe/csci123/lectures.aspx>

4. <http://rsc-team.ru/index.pl?rzd=2&group=lection&ind=22>
5. Г. Саттер Решение сложных задач на C++.
6. Связанный список <http://pumpkinprogrammer.com/2014/06/13/c-tutorial-intro-to-linked-lists/>

Литература по OpenCV

1. http://www.cs.uml.edu/~xinwenfu/Classes/91.204.201/Slides/91.204.201_02_OpenCVCore.ppt
2. Преобразование mat в массив
<http://stackoverflow.com/questions/26681713/convert-mat-to-array-vector-in-opencv>
3. Поиск красных кругов.
<https://www.solarianprogrammer.com/2015/05/08/detect-red-circles-image-using-opencv/>
4. Обзорная лекция по OpenCV <http://image.ing.bth.se/ipl-bth/siamak.khatibi/AIPBTH10LP2/lectures/Lec-10.pdf>

Литература по C++

1. Бьерн Страуструп. Язык программирования C++.
2. Тут есть несколько неплохих туториалов. <http://www.cprogramming.com/tutorial.html>
3. Introduction to Programming Concepts in C++ (хорошие базовые лекции по C++ на английском)

<http://staffwww.fullcoll.edu/brippe/csci123/lectures.aspx>

4. <http://rsc-team.ru/index.pl?rzd=2&group=lection&ind=22>
5. Г. Саттер Решение сложных задач на C++.
6. Связанный список <http://pumpkinprogrammer.com/2014/06/13/c-tutorial-intro-to-linked-lists/>