Nota: Algunas de las imágenes que aparecen en esta presentación provienen del libro:
Visión por Computador: fundamentos y métodos.
Arturo de la Escalera Hueso. Prentice Hall.

Intelligent Systems Lab

Sistemas de Percepción

Visión por Computador

Arturo de la Escalera
José María Armingol
Fernando García
David Martín
Abdulla Al-Kaff

uc3m | Universidad Carlos III de Madrid

Intelligent Systems Lab

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

Intelligent Systems Lab

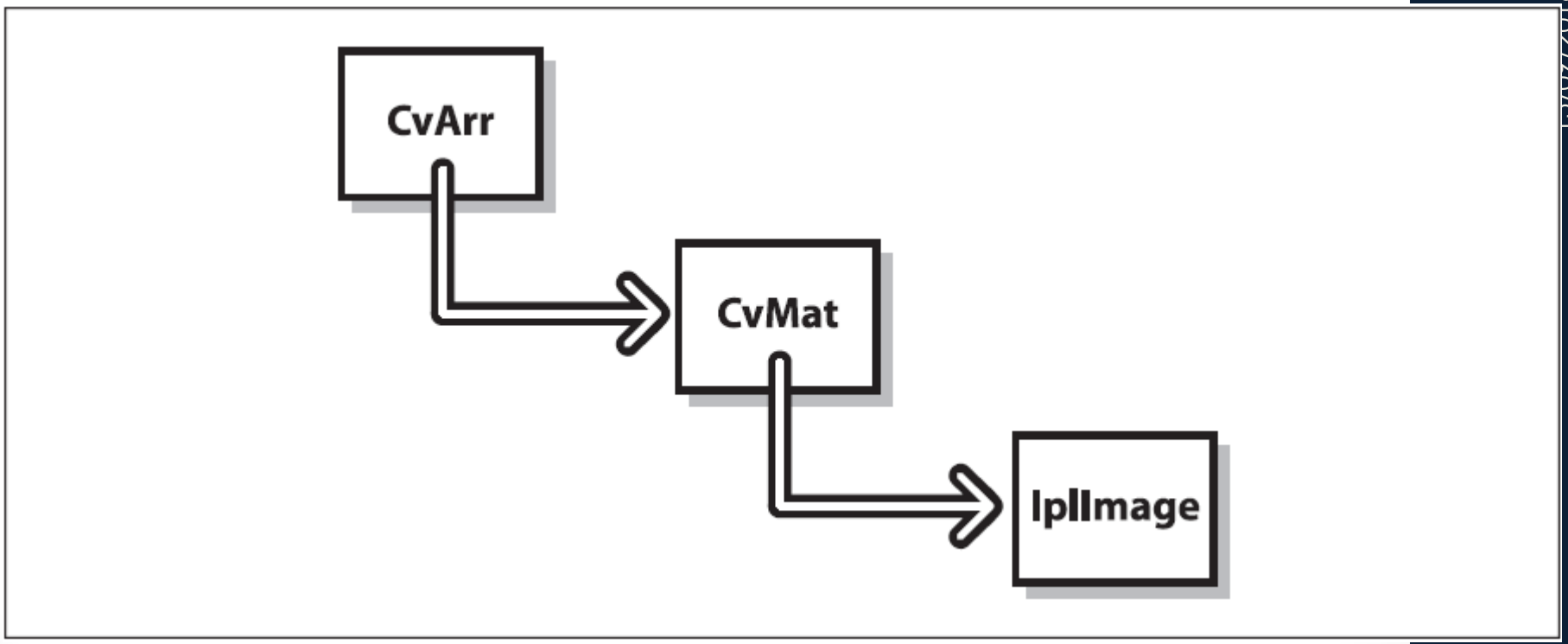
- Antiguas estructuras de OpenCV (lenguaje C):
  - OpenCV tiene distintos tipos de datos
    - Las especificaciones están en: cxtypes.h
  - CvPoint es la estructura más simpe
    - Dos enteros para definir un pixel en una imagen
  - CvSize, CvRect, CvScalar, ….

Table 3-1. Structures for points, size, rectangles, and scalar tuples

Structure	Contains	Represents
CvPoint	int x, y	Point in image
CvPoint2D32f	float x, y	Points in $\Re^2$
CvPoint3D32f	float x, y, z	Points in $\Re^3$
CvSize	int width, height	Size of image
CvRect	int x, y, width, height	Portion of image
CvScalar	double val[4]	RGBA value

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- Tipos de objetos para imágenes



IPLImage se mantiene por motivos de compatibilidad pero no debe usarse

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- cvMat y IplImage:

```
typedef struct CvMat {
    int type;
    int step;
    int* refcount;    // for internal use only
    union {
        uchar*  ptr;
        short*  s;
        int*    i;
        float*  fl;
        double* db;
    } data;
    union {
        int rows;
        int height;
    };
    union {
        int cols;
        int width;
    };
} CvMat;
```

```
typedef struct _IplImage {
  int                  nSize;
  int                  ID;
  int                  nChannels;
  int                  alphaChannel;
  int                  depth;
  char                 colorModel[4];
  char                 channelSeq[4];
  int                  dataOrder;
  int                  origin;
  int                  align;
  int                  width;
  int                  height;
  struct _IplROI*      roi;
  struct _IplImage*    maskROI;
  void*                imageId;
  struct _IplTileInfo* tileInfo;
  int                  imageSize;
  char*                imageData;
  int                  widthStep;
  int                  BorderMode[4];
  int                  BorderConst[4];
  char*                imageDataOrigin;
} IplImage;
```

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- OpenCV nuevas clases (C++):

The OpenCV C++ reference manual is here:
http://docs.opencv.org. Use ***Quick Search*** *to find descriptions of the particular functions and classes*

Key OpenCV Classes

Point_	Template 2D point class
Point3_	Template 3D point class
Size_	Template size (width, height) class
Vec	Template short vector class
Matx	Template small matrix class
Scalar	4-element vector
Rect	Rectangle
Range	Integer value range
Mat	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
SparseMat	Multi-dimensional sparse array
Ptr	Template smart pointer class

OpenCV Acceso a los píxeles de una imagen y obtenc

- **Mat**

OpenCV C++ n-dimensional dense array class

```
class CV_EXPORTS Mat
{
public:
    // ... a lot of methods ...
    ...

    /*! includes several bit-fields:
         - the magic signature
         - continuity flag
         - depth
         - number of channels
     */
    int flags;
    //! the array dimensionality, >= 2
    int dims;
    //! the number of rows and columns or (-1, -1) when the array has more than 2 dimensions
    int rows, cols;
    //! pointer to the data
    uchar* data;

    //! pointer to the reference counter;
    // when array points to user-allocated data, the pointer is NULL
    int* refcount;

    // other members
    ...
};
```

La clase Mat representa un arreglo numérico n dimensional, de un solo o multi canal. Puede usarse para almacenar vectores o matrices de valores reales o complejos e imágenes en nivel de gris o en color.

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- Mat métodos:

Mat::Mat	Mat::mul	Mat::operator CvMat
Mat::~Mat	Mat::cross	Mat::operator IplImage
Mat::operator =	Mat::dot	Mat::total
Mat::row	Mat::zeros	Mat::isContinuous
Mat::col	Mat::ones	Mat::elemSize
Mat::rowRange	Mat::eye	Mat::elemSize1
Mat::colRange	Mat::create	Mat::type
Mat::diag	Mat::addref	Mat::depth
Mat::clone	Mat::release	Mat::channels
Mat::copyTo	Mat::resize	Mat::step1
Mat::convertTo	Mat::reserve	Mat::size
Mat::assignTo	Mat::push_back	Mat::empty
Mat::setTo	Mat::pop_back	Mat::ptr
Mat::reshape	Mat::locateROI	Mat::at
Mat::t	Mat::adjustROI	Mat::begin
Mat::inv	Mat::operator()	Mat::end

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

Intelligent Systems Lab

class **Point_**

```
template<typename _Tp> class CV_EXPORTS Point_
{
public:
    typedef _Tp value_type;

    // various constructors
    Point_();
    Point_(_Tp _x, _Tp _y);
    Point_(const Point_& pt);
    Point_(const CvPoint& pt);
    Point_(const CvPoint2D32f& pt);
    Point_(const Size_<_Tp>& sz);
    Point_(const Vec<_Tp, 2>& v);

    Point_& operator = (const Point_& pt);
    //! conversion to another data type
    template<typename _Tp2> operator Point_<_Tp2>() const;

    //! conversion to the old-style C structures
    operator CvPoint() const;
    operator CvPoint2D32f() const;
    operator Vec<_Tp, 2>() const;

    //! dot product
    _Tp dot(const Point_& pt) const;
    //! dot product computed in double-precision arithmetics
    double ddot(const Point_& pt) const;
    //! cross-product
    double cross(const Point_& pt) const;
    //! checks whether the point is inside the specified rectangle
    bool inside(const Rect_<_Tp>& r) const;

    _Tp x, y; //< the point coordinates
};
```

```
pt1 = pt2 + pt3;
pt1 = pt2 - pt3;
pt1 = pt2 * a;
pt1 = a * pt2;
pt1 += pt2;
pt1 -= pt2;
pt1 *= a;
double value = norm(pt); // L2 norm
pt1 == pt2;
pt1 != pt2;
```

For your convenience, the following type aliases are defined:

```
typedef Point_<int> Point2i;
typedef Point2i Point;
typedef Point_<float> Point2f;
typedef Point_<double> Point2d;
```

Example:

```
Point2f a(0.3f, 0.f), b(0.f, 0.4f);
Point pt = (a + b)*10.f;
cout << pt.x << ", " << pt.y << endl;
```

```
Point pt;
pt.x = 10;
pt.y = 8;

Point pt = Point(10, 8);
```

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- Dibujar
  - Círculos:

```
void circle(
        InputOutputArray img,
        Point center,
        int radius,
        const Scalar& color,
        int thickness=1,
        intlineType=LINE_8,
        int shift=0 )
```

Scalar(blue, green, red);

-1 para rellenar el círculo

Optional

```
Point center(img.rows / 2, img.cols / 2);
int radius = 25;
circle (img, center, radius, Scalar(0, 255, 0));
```

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- Dibujar
  - Líneas:

```
void line(
        InputOutputArray img,
        Point pt1,
        Point pt2,
        const Scalar& color,
        int thickness = 1,
        int lineType = LINE_8,
        int shift=0 )
```

Scalar(blue, green, red);

Optional

```
Point start(img.rows / 2, img.cols / 2);
Point end (0, 0);
line (img, start, end, Scalar(0, 255, 0));
```

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- Dibujar
  - Rectángulo:

```
void rectangle(
        InputOutputArray img,
        Point pt1,
        Point pt2,
        const Scalar& color,
        int thickness = 1,
        int lineType = LINE_8,
        int shift=0 )
```

Scalar(blue, green, red);

-1 para rellenar el rectángulo

Optional

```
Point start(img.rows / 2, img.cols / 2);
Point end (img.rows , img.cols );
rectangle (img, start, end, Scalar(0, 255, 0));
```

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

Intelligent Systems Lab

- Ejemplo 01: Dibujar

L02_drawings.cpp

(Global Scope)

```cpp
// L02_drawings.cpp: Draw different shapes in the image
//
#include "opencv\cv.hpp"
#include <io

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
{
    // Object instantiation
    Mat img_original, img_modified;
    // Load image from disk
    img_original = imread("lena.jpg");

    // Copy images-> 2 ways to copy:
    //    img_original.clone()
    //    img_original.copyTo(img_modified)

    //img_modified = img_original.clone();
    img_original.copyTo(img_modified);

    if (!img_original.data){
        cout << "error loading image" << endl;
        return 1;
    }

    // Center of the image
    Point center(img_modified.rows / 2, img_modified.cols / 2);
    int radius = 25;

    // Draw a circle
    circle(img_modified, center, radius, Scalar(0, 255, 0));
```

L02_drawings.cpp

(Global Scope)

```cpp

    // Draw a line
    line(img_modified, center, Point(0,0), Scalar(255, 0, 0));

    // Draw a rectangle
    rectangle(img_modified, center, Point(img_modified.rows,
              img_modified.cols), Scalar(0, 0, 255));

    // Create window canvas to show image
    namedWindow("original", CV_WINDOW_AUTOSIZE);
    namedWindow("modified", CV_WINDOW_AUTOSIZE);

    // Show images
    imshow("original", img_original);
    imshow("modified", img_modified);

    // Function for show the image in ms.
    // 0 means wait until keyboard is pressed
    waitKey(0);

    // Free memory
    destroyWindow("original");
    destroyWindow("modified");
    // End of the program
    return 0;
}
```

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

- Ejemplo 02:Obtener el histograma:
  - Cargar la imagen (si es en color, separar los tres canales)
  - Configurar el histograma para cada canal
  - Calcular el histograma
  - Dibujar cada histograma
  - Liberar la memoria

```
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 )
```

Optional

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

Intelligent Systems Lab

L02_histogram.cpp*

(Global Scope) — main(int, char

```cpp
// L02_histogram.cpp: Create histogram of one image
//
#include "opencv\cv.hpp"
#include <iostream>

using namespace cv;
using namespace s

int main(int, char**)
{
    // Object instantiation
    Mat img_original;

    // Load image from disk
    img_original = imread("mandril.jpg", 0);
    if (!img_original.data){
        cout << "error loading image" << endl;
        return 1;
    }

    // Create window canvas to show image
    namedWindow("original", CV_WINDOW_AUTOSIZE);
    namedWindow("histogram", CV_WINDOW_AUTOSIZE);

    // Initialize parameters
    int histSize = 256;    // bin size
    float range[] = { 0, 255 };
    const float *ranges[] = { range };

    // Calculate histogram
    Mat hist;
    calcHist(&img_original, 1, 0, Mat(), hist, 1, &histSize, ranges, true, false);

```

OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

L02_histogram.cpp

(Global Scope) | main(int, char **)

```
33
34      // Show the calculated histogram in command window
35      double total;
36      total = img_original.rows * img_original.cols;
37      for (int h = 0; h < histSize; h++)
38      {
39          float binVal = hist.at<float>(h);
40          cout << " " << binVal;
41      }
42
43      // Plot the histogram
44      int hist_w = 512; int hist_h = 400;
45      int bin_w = cvRound((double)hist_w / histSize);
46
47      Mat histImage(hist_h, hist_w, CV_8UC1, Scalar(0, 0, 0));
48
49      // In order to fit in window sized 400x500 must be normalized:
50      // max value of same color pixel = 2740 --> 500
51      normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
52
53      for (int i = 1; i < histSize; i++)
54      {
55          line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(hist.at<float>(i - 1))),
56              Point(bin_w*(i), hist_h - cvRound(hist.at<float>(i))),
57              Scalar(255, 0, 0), 2, 8, 0);
58      }
59
60      imshow("original", img_original);
61      imshow("histogram", histImage);
62
63      waitKey(0);
64
65      destroyAllWindows();
66      return 0;
67  }
```