### Key OpenCV Classes

- **Point, Point3**
- **Size**
- **Vec**
- **Scalar**
- **Rect**
- **Range**
- **Mat**
- **Mat3D**
- **SparseMat**
- **Ptr**

### Matrix Basics

- **Create a matrix**
  - Mat image(240, 320, CV_8UC3);
  - [Re]allocate a pre-declared matrix
  - image.create(480, 640, CV_8UC3);
- **Create a matrix initialized with a constant**
  - Mat A33(3, 3, CV_32F, Scalar(5));
  - Mat B33(3, 3, CV_32F); B33 = Scalar(5);
  - Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
- **Create a matrix initialized with specified values**
  - double a = CV_PI/3;
  - Mat A22 = Mat_<float>(2, 2) <<
    cos(a), -sin(a),
    sin(a), cos(a);
  - float B22data[] = {cos(a),-sin(a),
    sin(a), cos(a)};
  - Mat B22 = Mat(2, 2, CV_32F, B22data.clone());

- **Initialize a random matrix**
  - randn(image, Scalar(0), CV_8UC3);
  - randn(image, Scalar(256));
  - randn(image, Scalar1(128), Scalar1(10));

- **Convert matrix to/from other structures**
  - (without copying the data)
  - Mat image_alias = image;
  - float* Idata=new float[480*640*3];
  - Mat I(480, 640, CV_32FC3, Idata);
  - vector<Point> iptvec(10);
  - Mat IP(iptvec); // I - 10x1 CV_32SC2 matrix
  - CvMat oldC0 = cvCreateImage(cvSize(320, 240), I16);
  - Mat newC = cvArrToMat(oldC0);
  - IplImage oldC1 = newC;
  - CvMat oldC2 = newC;
  - ... (with copying the data)

### Matrix Manipulations: Copying, Shuffling, Part Access

- **Copy matrix to another one**
  - src.copyTo(dst)
  - src.convertTo(dst, type, scale, shift)
  - src.clone()

- **Make deep copy of a matrix**
  - m.clone()
  - m.reshape(nch, nrows)
  - m.reshape(nrows, nch)

- **Copy matrix to another one**
  - src.clone()

- **Create a multi-dimensional sparse array**
  - SparseMat

- **Create a multi-dimensional dense array**
  - Mat

- **Multiply**
  - multiply(a2, ra1, a2, 1./255);

- **Add**
  - add(a1, a2, rgba);

- **Subtract**
  - subtract(Scalar::all(255), a1, ra1);

### Simple Matrix Operations

- **Mat dyImage(image.size(), image.type());**
  - for(int y = 1; y < image.rows-1; y++) {
    - Vec3b* prevRow = image.ptr<Vec3b>(y-1);
    - Vec3b* nextRow = image.ptr<Vec3b>(y+1);
    - for(int x = 0; x < image.cols; x++){
      - dyImage.at<Vec3b>(y,x)[c] = saturate_cast<uchar>(
        nextRow[x][c] - prevRow[x][c];
    }
  }

### Matrices and Vectors

- **Scalar**
- **Vec**
  - Vec3b
  - Vec3f
  - Vec3l

### Filter2D

- **filter2D(image, image, depth(), Mat(Mat<float>(3,3),
  - ((-1, -1, -1, 1, 0, -1), Point1(1,1), 128));**
Geometrical Transformations

resize() Resize image
getRectSubPix() Extract an image patch
warpAffine() Warp image affirmly
warpPerspective() Warp image perspectivey
remap() Generic image warping
convertMaps() Optimize maps for a faster remap() execution

Example. Decimate image by factor of $\sqrt{2}$:
Mat dst; resize(src, dst, Size(), 1./sqrt(2), 1./sqrt(2));

Various Image Transformations
cvtColor() Convert image from one color space to another
threshold() Convert grayscale image to binary image
adaptiveThreshold() Using a fixed or a variable threshold
floodFill() Find a connected component using region growing algorithm
integral() Compute integral image
distanceTransform() Build distance map or discrete Voronoi diagram for a binary image.
watershed() Marker-based image segmentation algorithms. See the samples watershed.cpp and grabcut.cpp.

Histograms
calcHist() Compute image(s) histogram
calcBackProject() Back-project the histogram
equalizeHist() Normalize image brightness and contrast
compareHist() Compare two histograms

Example. Compute Hue-Saturation histogram of an image:
Mat hsv; MatND tempH;
cvtColor(hsv, 1, planes, Mat());
calcHist(&hsv, 1, planes, Mat(), tempH, 2, hsize, 0);

Contours
See contours.cpp and squares.c samples on what are the contours and how to use them.

Data I/O
XML/YAML storages are collections (possibly nested) of scalar values, structures and heterogeneous lists.

Writing data to YAML (or XML)
// Type of the file is determined from the content
FileStorage fs("test.yml", FileStorage::READ);
int i1 = (int)fs["i1"]; double r1 = (double)fs["r1"];
string str1 = (string)fs["str1"];
Mat M; fs["mtx"] >> M;
FileNode tl = fs["mylist"];
Mat M; fs["mtx"] >> M;
FileNode tl = fs["mylist"];
Scalar values, structures and heterogeneous lists can be written to the file storages using << operator.

Reading the data back
// Type of the file is determined from the content
FileStorage fs("test.yml", FileStorage::READ);
int i1 = (int)fs["i1"]; double r1 = (double)fs["r1"];
string str1 = (string)fs["str1"];
Mat M; fs["mtx"] >> M;
FileNode tl = fs["mylist"];
The functions can read/write images in the following formats:
BMP (.bmp), JPEG (.jpg, .jpeg), TIFF (.tif, .tiff), PNG (.png), PBM/PGM/PPM (.p?m), Sun Raster (.sr), JPEG 2000 (.jpg).
Every format supports 8-bit, 1- or 3-channel images. Some formats (PNG, JPEG 2000) support 16 bits per channel.

Reading video from a file or from a camera
VideoCapture cap;
if(cap.open(string(argv[1]))) else cap.open(0);
Mat frame; namedWindow("video", 1);
for(;;) {
cap >> frame; if(!frame.data) break;
imshow("video", frame);
if(waitKey(30) == 0) break;
}

Camera Calibration, Pose Estimation and Depth Estimation
calibrateCamera() Calibrate camera from several views of a pattern calibration.
findChessboardCorners() Find feature points on the checkerboard calibration pattern.
solvePnP() Solve the pose from the known projections of its feature points.
stereoCalibrate() Compute stereo camera.
stereoRectify() Compute the rectification transforms for a calibrated stereo camera.
initUndistortRectifyMap() Compute rectification map (for initUndistortRectifyMap())
StereoBM, StereoSGBM Compute disparity maps for each stereo camera head.
reprojectImageTo3D() Convert disparity map to 3D point cloud.
findHomography() Find best-fit perspective transformation between two 2D point sets.
To calibrate a camera, you can use calibration.cpp or stereo_calib.cpp samples. To get the disparity maps and the point clouds, use stereo_match.cpp sample.

Object Detection
matchTemplate() Compute proximity map for given template.
CascadeClassifier Viola’s Cascade of Boosted classifiers using Haar or LBP features. Suits for detecting faces, facial features and some other objects without diverse textures. See facedetect.cpp
HOGDescriptor N. Dalal’s object detector using Histogram-of-Oriented-Gradients (HOG) features. Suits for detecting people, cars and other objects with well-defined silhouettes. See peopledetect.cpp