Introduction to OpenCV

Serban Porumbescu, Ph.D.

Hidden Elephant

LIMBIC

Overview

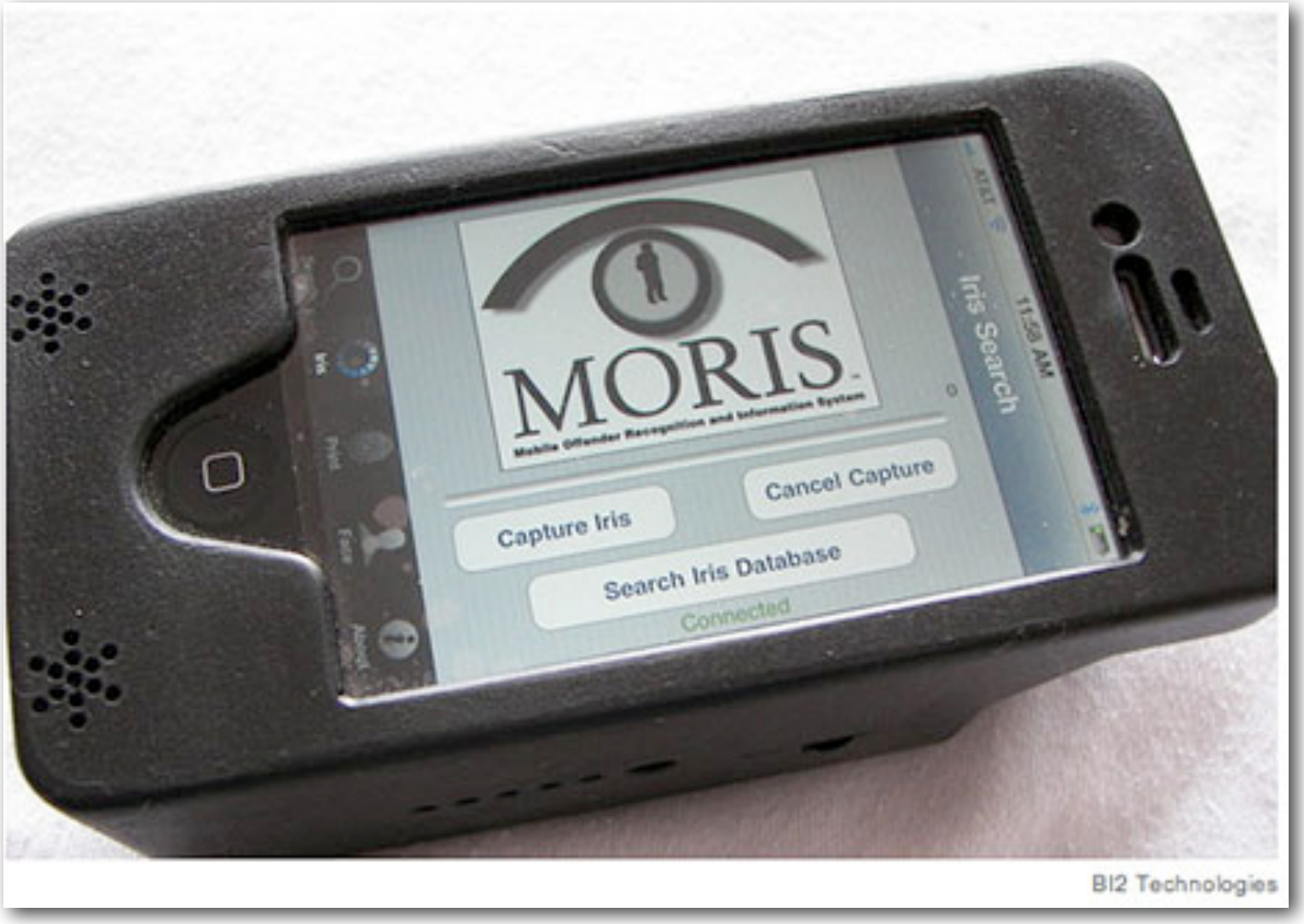
- What is OpenCV?
- Nuts & Bolts (building, data structures)
- Image Manipulations (color space conversion)
- Template Matching

What is OpenCV

- Library for realtime Computer Vision
- Computer Vision is about making machines "see"

Some Computer Vision Examples

http://www.youtube.com/watch?v=VSMoLqG018E

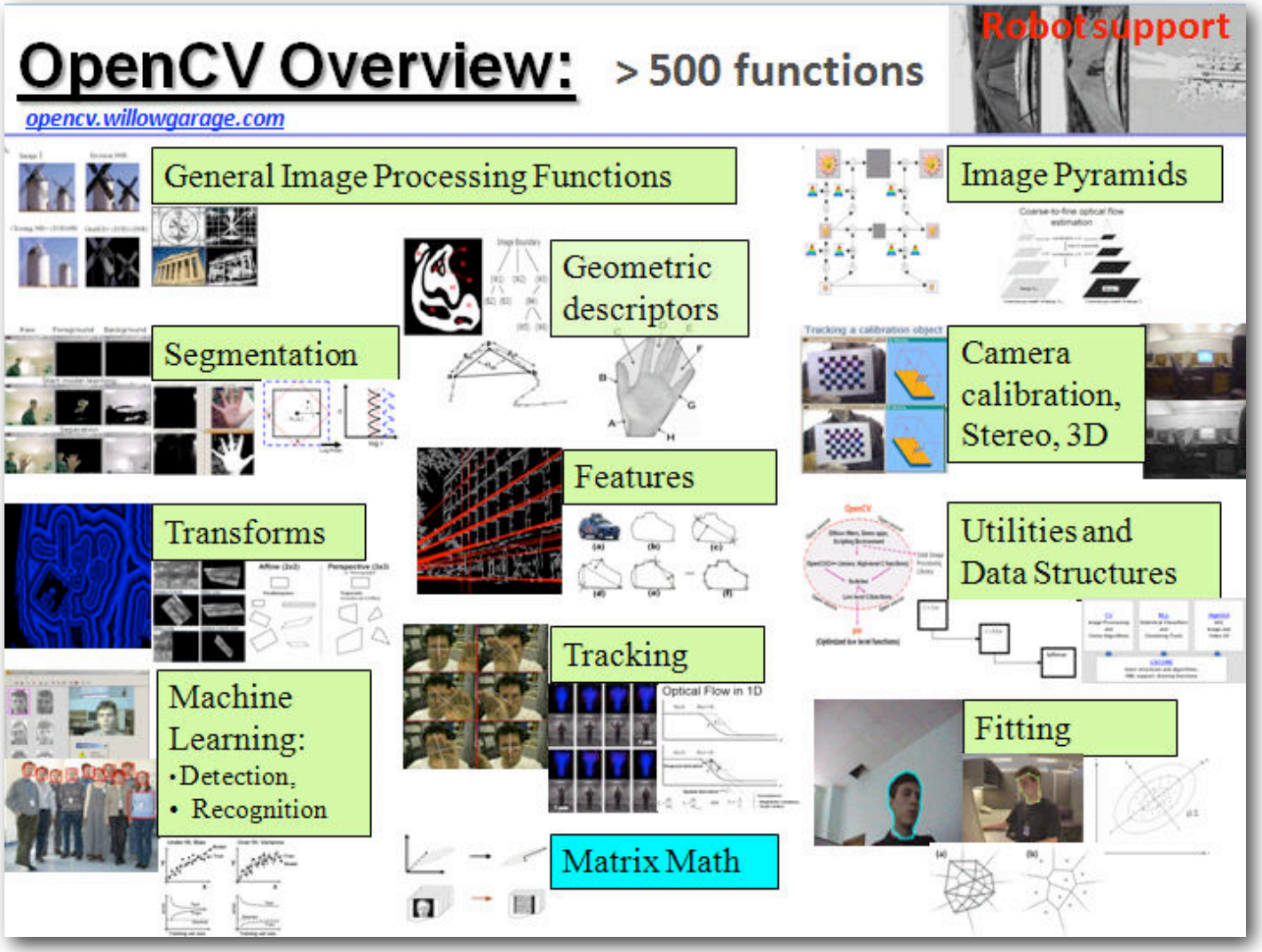


http://www.tuaw.com/2011/07/13/police-adopting-iphone-based-facial-recognition-device/

Get your own PR2

http://www.willowgarage.com/pages/pr2/overview

What is OpenCV



Building OpenCV

Grab Some Tools

- www.macports.org/install.php
- $ sudo port install cmake

Build OpenCV

- $ git clone git://github.com/niw/iphone_opencv_test.git
- niw.at/articles/2009/03/14/using-opencv-on-iphone/en

opencv_cmake.sh

```
if [ -z "$SDK_VERSION" ]; then
  SDK_VERSION="4.3"
fi

if [ -z "$IPHONEOS_VERSION_MIN" ]; then
  IPHONEOS_VERSION_MIN="3.0"
fi
```

Data Structures and Primitive Types

Primitive Types

```
typedef struct CvPoint
{
    int x;
    int y;
}
CvPoint;
```

Primitive Types

```
typedef struct CvPoint2D32f
{
    float x;
    float y;
}
CvPoint2D32f;
```

Primitive Types

```
typedef struct CvPoint3D32f
{
    float x;
    float y;
    float z;
}
CvPoint3D32f;
```

Primitive Types

```
typedef struct
{
    int width;
    int height;
}
CvSize;
```

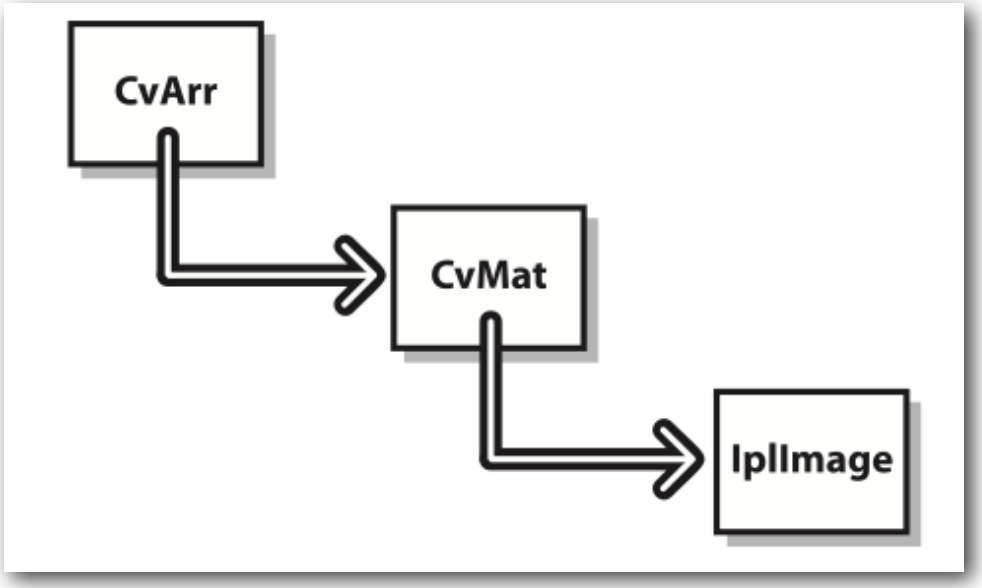
Primitive Types

```
typedef struct CvScalar
{
    double val[4];
}
CvScalar;
```

Primitive Types

```
typedef struct CvRect
{
    int x;
    int y;
    int width;
    int height;
}
CvRect;
```

Data Structures



```
typedef struct CvMat
{
    int type;
    int step;

    /* for internal use only */
    int* refcount;
    int hdr_refcount;

    union
    {
        uchar* ptr;
        short* s;
        int* i;
        float* fl;
        double* db;
    } data;

#ifdef __cplusplus
    union
    {
        int rows;
        int height;
    };

    union
    {
        int cols;
        int width;
    };
#else
    int rows;
    int cols;
#endif

}
CvMat;
```

```
typedef struct CvMat
{
    int type;
    int step;

    /* for internal use only */
    int* refcount;
    int hdr_refcount;

    union
    {
        uchar* ptr;
        short* s;
        int* i;
        float* fl;
        double* db;
    } data;

#ifdef __cplusplus
    union
    {
        int rows;
        int height;
    };

    union
    {
        int cols;
        int width;
    };
#else
    int rows;
    int cols;
#endif

}
CvMat;
```

```
#define CV_8U   0
#define CV_8S   1
#define CV_16U  2
#define CV_16S  3
#define CV_32S  4
#define CV_32F  5
#define CV_64F  6
#define CV_USRTYPE1 7
```

```
typedef struct CvMat
{
    int type;
    int step;

    /* for internal use only */
    int* refcount;
    int hdr_refcount;

    union
    {
        uchar* ptr;
        short* s;
        int* i;
        float* fl;
        double* db;
    } data;

#ifdef __cplusplus
    union
    {
        int rows;
        int height;
    };

    union
    {
        int cols;
        int width;
    };
#else
    int rows;
    int cols;
#endif

}
CvMat;
```

```
typedef struct CvMat
{
    int type;
    int step;

    /* for internal use only */
    int* refcount;
    int hdr_refcount;

    union
    {
        uchar* ptr;
        short* s;
        int* i;
        float* fl;
        double* db;
    } data;

#ifdef __cplusplus
    union
    {
        int rows;
        int height;
    };

    union
    {
        int cols;
        int width;
    };
#else
    int rows;
    int cols;
#endif

}
CvMat;
```

```
typedef struct CvMat
{
    int type;
    int step;

    /* for internal use only */
    int* refcount;
    int hdr_refcount;

    union
    {
        uchar* ptr;
        short* s;
        int* i;
        float* fl;
        double* db;
    } data;

#ifdef __cplusplus
    union
    {
        int rows;
        int height;
    };

    union
    {
        int cols;
        int width;
    };
#else
    int rows;
    int cols;
#endif

}
CvMat;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

```
typedef struct _IplImage
{
    int  nSize;             /* sizeof(IplImage) */
    int  ID;                /* version (=0)*/
    int  nChannels;         /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  alphaChannel;      /* Ignored by OpenCV */
    int  depth;             /* Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16S,
                               IPL_DEPTH_32S, IPL_DEPTH_32F and IPL_DEPTH_64F are supported.  */
    char colorModel[4];     /* Ignored by OpenCV */
    char channelSeq[4];     /* ditto */
    int  dataOrder;         /* 0 - interleaved color channels, 1 - separate color channels.
                               cvCreateImage can only create interleaved images */
    int  origin;            /* 0 - top-left origin,
                               1 - bottom-left origin (Windows bitmaps style).  */
    int  align;             /* Alignment of image rows (4 or 8).
                               OpenCV ignores it and uses widthStep instead.    */
    int  width;             /* Image width in pixels.                           */
    int  height;            /* Image height in pixels.                          */
    struct _IplROI *roi;    /* Image ROI. If NULL, the whole image is selected. */
    struct _IplImage *maskROI;      /* Must be NULL. */
    void  *imageId;                 /* "           " */
    struct _IplTileInfo *tileInfo;  /* "           " */
    int  imageSize;         /* Image data size in bytes
                               (==image->height*image->widthStep
                               in case of interleaved data)*/
    char *imageData;        /* Pointer to aligned image data.         */
    int  widthStep;         /* Size of aligned image row in bytes.    */
    int  BorderMode[4];     /* Ignored by OpenCV.                     */
    int  BorderConst[4];    /* Ditto.                                 */
    char *imageDataOrigin;  /* Pointer to very origin of image data
                               (not necessarily aligned) -
                               needed for correct deallocation */
}
IplImage;
```

Function	Description
cvAbs	Absolute value of all elements in an array
cvAbsDiff	Absolute value of differences between two arrays
cvAbsDiffS	Absolute value of difference between an array and a scalar
cvAdd	Elementwise addition of two arrays
cvAddS	Elementwise addition of an array and a scalar
cvAddWeighted	Elementwise weighted addition of two arrays (alpha blending)
cvAvg	Average value of all elements in an array
cvAvgSdv	Absolute value and standard deviation of all elements in an array
cvCalcCovarMatrix	Compute covariance of a set of *n*-dimensional vectors
cvCmp	Apply selected comparison operator to all elements in two arrays
cvCmpS	Apply selected comparison operator to an array relative to a scalar
cvConvertScale	Convert array type with optional rescaling of the value
cvConvertScaleAbs	Convert array type after absolute value with optional rescaling
cvCopy	Copy elements of one array to another
cvCountNonZero	Count nonzero elements in an array
cvCrossProduct	Compute cross product of two three-dimensional vectors
cvCvtColor	Convert channels of an array from one color space to another
cvDet	Compute determinant of a square matrix
cvDiv	Elementwise division of one array by another
cvDotProduct	Compute dot product of two vectors
cvEigenVV	Compute eigenvalues and eigenvectors of a square matrix
cvFlip	Flip an array about a selected axis
cvGEMM	Generalized matrix multiplication
cvGetCol	Copy elements from column slice of an array
cvGetCols	Copy elements from multiple adjacent columns of an array

Function	Description
cvMahalonobis	Compute Mahalonobis distance between two vectors
cvMax	Elementwise max operation on two arrays
cvMaxS	Elementwise max operation between an array and a scalar
cvMerge	Merge several single-channel images into one multichannel image
cvMin	Elementwise min operation on two arrays
cvMinS	Elementwise min operation between an array and a scalar
cvMinMaxLoc	Find minimum and maximum values in an array
cvMul	Elementwise multiplication of two arrays
cvNot	Bitwise inversion of every element of an array
cvNorm	Compute normalized correlations between two arrays
cvNormalize	Normalize elements in an array to some value
cvOr	Elementwise bit-level OR of two arrays
cvOrS	Elementwise bit-level OR of an array and a scalar
cvReduce	Reduce a two-dimensional array to a vector by a given operation
cvRepeat	Tile the contents of one array into another
cvSet	Set all elements of an array to a given value
cvSetZero	Set all elements of an array to 0
cvSetIdentity	Set all elements of an array to 1 for the diagonal and 0 otherwise
cvSolve	Solve a system of linear equations
cvSplit	Split a multichannel array into multiple single-channel arrays
cvSub	Elementwise subtraction of one array from another
cvSubS	Elementwise subtraction of a scalar from an array
cvSubRS	Elementwise subtraction of an array from a scalar
cvSum	Sum all elements of an array
cvSVD	Compute singular value decomposition of a two-dimensional array
cvSVBkSb	Compute singular value back-substitution

UIImage <==> OpenCV

```objc
#import <Foundation/Foundation.h>
#import <opencv/cv.h>

@interface OpenCVUIKitUtils : NSObject {

}

// OpenCV to UIImage
+ (IplImage *)CreateIplImageFromUIImage:(UIImage *)image;

// UIImage to OpenCV
+ (UIImage *)UIImageFromIplImage:(IplImage *)image;

@end
```

```
+ (IplImage *)CreateIplImageFromUIImage:(UIImage *)image
{
  CGImageRef imageRef = image.CGImage;

  CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();

  IplImage *iplimage = cvCreateImage(cvSize(image.size.width,image.size.height),
                                     IPL_DEPTH_8U,
                                     4);

  CGContextRef contextRef = CGBitmapContextCreate(iplimage->imageData,
                                                  iplimage->width,
                                                  iplimage->height,
                                                  iplimage->depth,
                                                  iplimage->widthStep,
                                                  colorSpace,
                                                  kCGImageAlphaPremultipliedLast |
                                                   kCGBitmapByteOrderDefault);

  CGContextDrawImage(contextRef, CGRectMake(0, 0, image.size.width,
                                            image.size.height), imageRef);
  CGContextRelease(contextRef);
  CGColorSpaceRelease(colorSpace);

  IplImage *ret = cvCreateImage(cvGetSize(iplimage), IPL_DEPTH_8U, 3);
  cvCvtColor(iplimage, ret, CV_RGBA2BGR);
  cvReleaseImage(&iplimage);

  return ret;
}
```

```
+ (IplImage *)CreateIplImageFromUIImage:(UIImage *)image
{
  CGImageRef imageRef = image.CGImage;

  CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();

  IplImage *iplimage = cvCreateImage(cvSize(image.size.width,image.size.height),
                                     IPL_DEPTH_8U,
                                     4);

  CGContextRef contextRef = CGBitmapContextCreate(iplimage->imageData,
                                                  iplimage->width,
                                                  iplimage->height,
                                                  iplimage->depth,
                                                  iplimage->widthStep,
                                                  colorSpace,
                                                  kCGImageAlphaPremultipliedLast |
                                                   kCGBitmapByteOrderDefault);

  CGContextDrawImage(contextRef, CGRectMake(0, 0, image.size.width,
                                            image.size.height), imageRef);
  CGContextRelease(contextRef);
  CGColorSpaceRelease(colorSpace);

  IplImage *ret = cvCreateImage(cvGetSize(iplimage), IPL_DEPTH_8U, 3);
  cvCvtColor(iplimage, ret, CV_RGBA2BGR);
  cvReleaseImage(&iplimage);

  return ret;
}
```

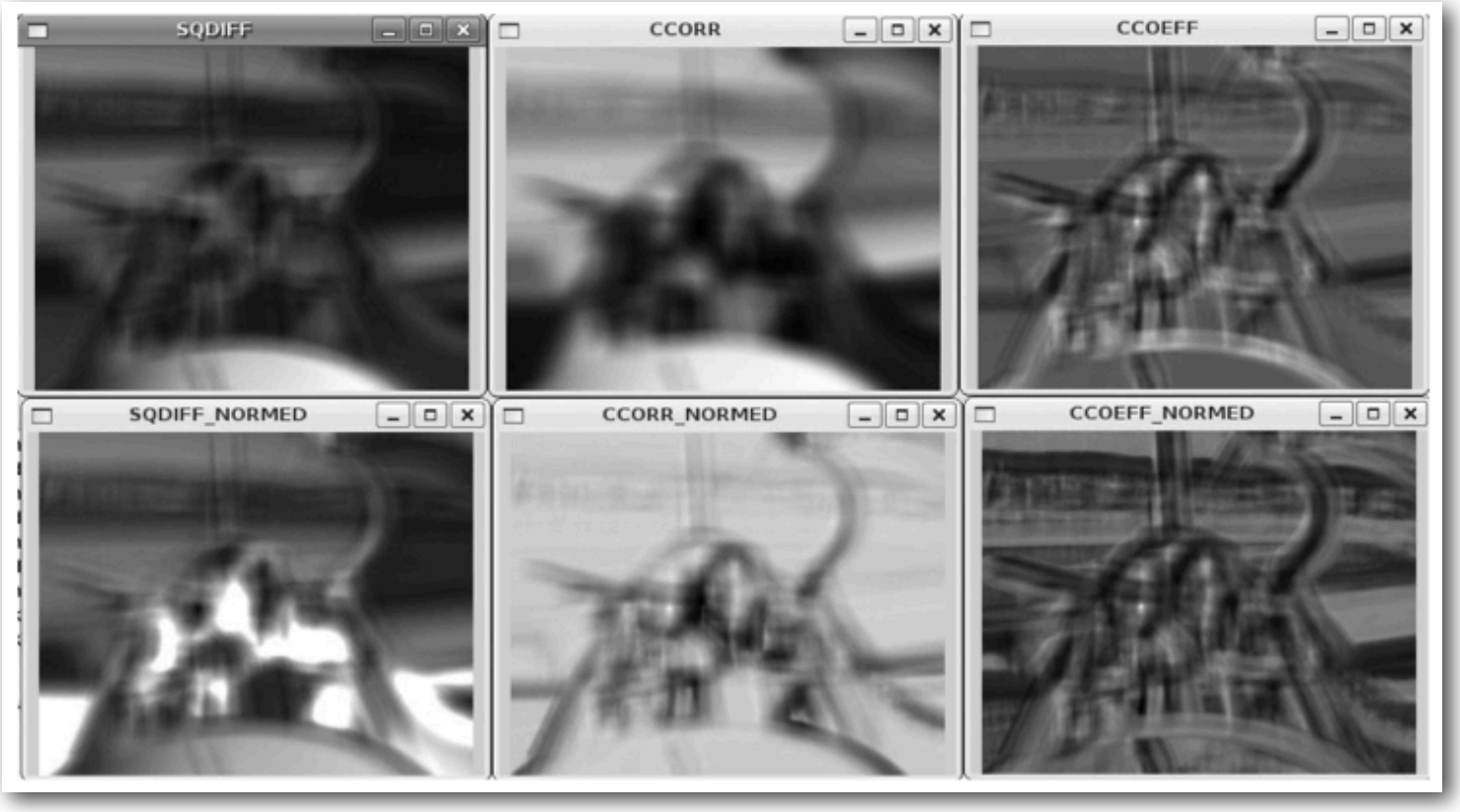
```
+ (IplImage *)CreateIplImageFromUIImage:(UIImage *)image
{
  CGImageRef imageRef = image.CGImage;

  CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();

  IplImage *iplimage = cvCreateImage(cvSize(image.size.width,image.size.height),
                                     IPL_DEPTH_8U,
                                     4);

  CGContextRef contextRef = CGBitmapContextCreate(iplimage->imageData,
                                                  iplimage->width,
                                                  iplimage->height,
                                                  iplimage->depth,
                                                  iplimage->widthStep,
                                                  colorSpace,
                                                  kCGImageAlphaPremultipliedLast |
                                                   kCGBitmapByteOrderDefault);

  CGContextDrawImage(contextRef, CGRectMake(0, 0, image.size.width,
                                            image.size.height), imageRef);
  CGContextRelease(contextRef);
  CGColorSpaceRelease(colorSpace);

  IplImage *ret = cvCreateImage(cvGetSize(iplimage), IPL_DEPTH_8U, 3);
  cvCvtColor(iplimage, ret, CV_RGBA2BGR);
  cvReleaseImage(&iplimage);

  return ret;
}
```

```
+ (IplImage *)CreateIplImageFromUIImage:(UIImage *)image
{
  CGImageRef imageRef = image.CGImage;

  CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();

  IplImage *iplimage = cvCreateImage(cvSize(image.size.width,image.size.height),
                                     IPL_DEPTH_8U,
                                     4);

  CGContextRef contextRef = CGBitmapContextCreate(iplimage->imageData,
                                                  iplimage->width,
                                                  iplimage->height,
                                                  iplimage->depth,
                                                  iplimage->widthStep,
                                                  colorSpace,
                                                  kCGImageAlphaPremultipliedLast |
                                                   kCGBitmapByteOrderDefault);

  CGContextDrawImage(contextRef, CGRectMake(0, 0, image.size.width,
                                            image.size.height), imageRef);
  CGContextRelease(contextRef);
  CGColorSpaceRelease(colorSpace);

  IplImage *ret = cvCreateImage(cvGetSize(iplimage), IPL_DEPTH_8U, 3);
  cvCvtColor(iplimage, ret, CV_RGBA2BGR);
  cvReleaseImage(&iplimage);

  return ret;
}
```

```
+ (IplImage *)CreateIplImageFromUIImage:(UIImage *)image
{
  CGImageRef imageRef = image.CGImage;

  CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();

  IplImage *iplimage = cvCreateImage(cvSize(image.size.width,image.size.height),
                                     IPL_DEPTH_8U,
                                     4);

  CGContextRef contextRef = CGBitmapContextCreate(iplimage->imageData,
                                                  iplimage->width,
                                                  iplimage->height,
                                                  iplimage->depth,
                                                  iplimage->widthStep,
                                                  colorSpace,
                                                  kCGImageAlphaPremultipliedLast |
                                                   kCGBitmapByteOrderDefault);

  CGContextDrawImage(contextRef, CGRectMake(0, 0, image.size.width,
                                            image.size.height), imageRef);
  CGContextRelease(contextRef);
  CGColorSpaceRelease(colorSpace);

  IplImage *ret = cvCreateImage(cvGetSize(iplimage), IPL_DEPTH_8U, 3);
  cvCvtColor(iplimage, ret, CV_RGBA2BGR);
  cvReleaseImage(&iplimage);

  return ret;
}
```

```
+ (UIImage *)UIImageFromIplImage:(IplImage *)image
{
  NSLog(@"IplImage (%d, %d) %d bits by %d channels, %d bytes/row %s",
        image->width, image->height, image->depth, image->nChannels,
        image->widthStep, image->channelSeq);

  CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
  NSData *data = [NSData dataWithBytes:image->imageData length:image->imageSize];
  CGDataProviderRef provider = CGDataProviderCreateWithCFData((CFDataRef)data);

  CGImageRef imageRef = CGImageCreate(image->width, image->height,
                                      image->depth,
                                      image->depth * image->nChannels,
                                      image->widthStep,
                                      colorSpace,
                                      kCGImageAlphaNone|kCGBitmapByteOrderDefault,
                                      provider,
                                      NULL,
                                      false,
                                      kCGRenderingIntentDefault);

  UIImage *ret = [UIImage imageWithCGImage:imageRef];
  CGImageRelease(imageRef);
  CGDataProviderRelease(provider);
  CGColorSpaceRelease(colorSpace);

  return ret;
}
```

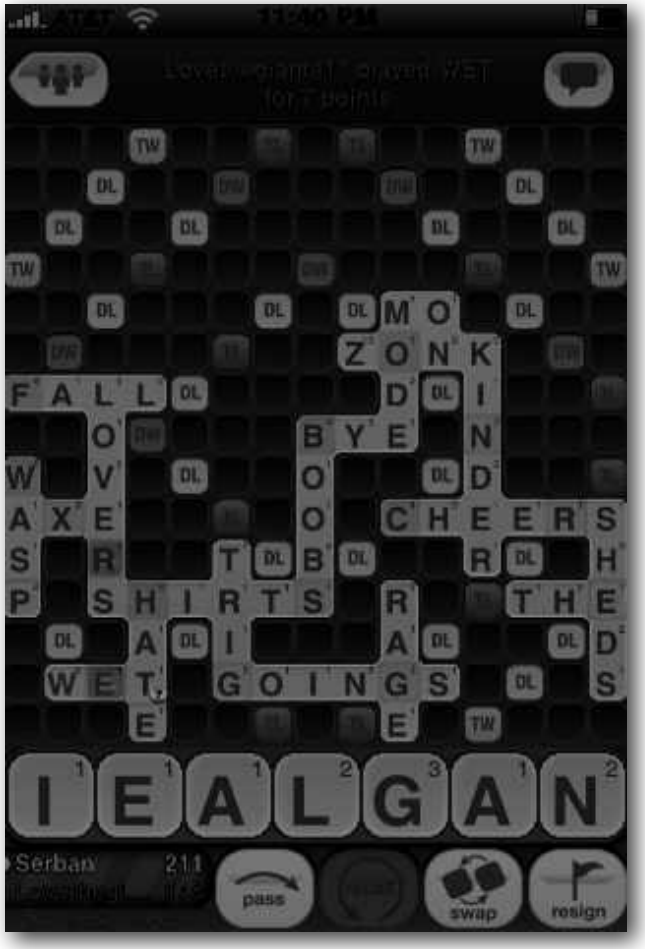
Template Matching



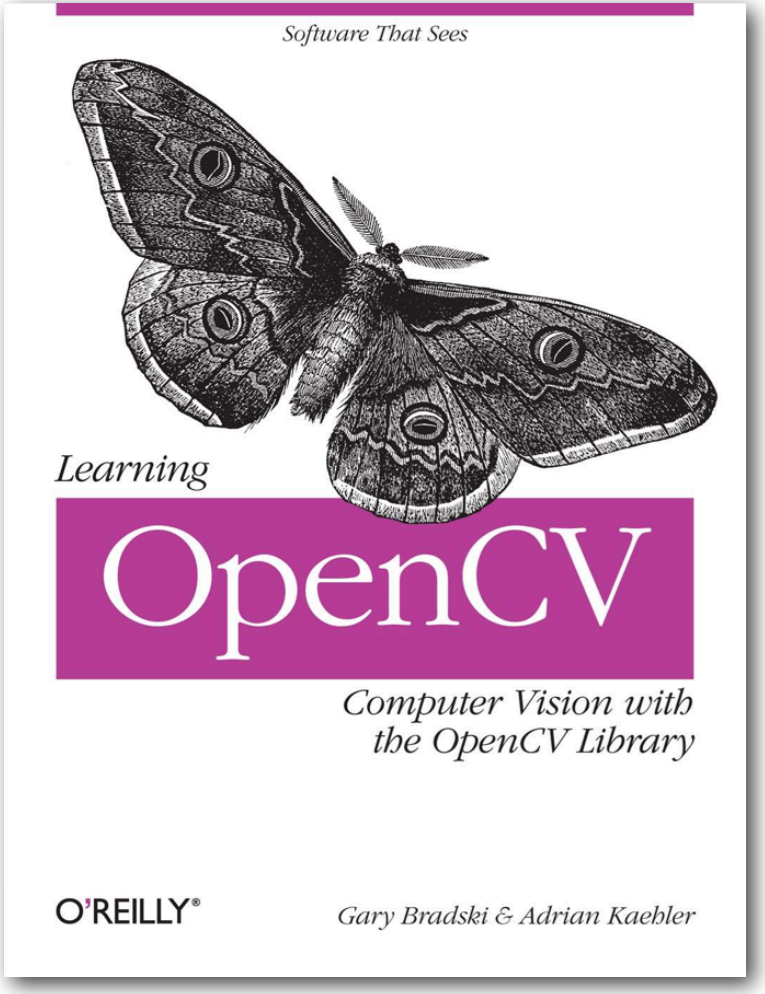

AT&T
7:48 PM
Serban played SUE for 9 points
Lovethegi... 225
Serban 198
17 letters remain
play
recall
swap
resign


Lovethegiants11 played WET
for 7 points
Lovethegi... 148
26 letters remain
pass
recall
swap
resign


Serban 211
pass
swap
resign


Serban 211
pass
swap
resign

Resources



opencv.willowgarage.com/wiki/

@cvpapers

@opencvlibrary

niw.at/articles/2009/03/14/using-opencv-on-iphone/en

Thanks! Questions?

@serban

serban@hiddenelephant.com