OpenCV -- 轮廓

虽然Canny之类的边缘检测算法可以根据像素间的差异检测出轮廓边界的像素，但是它并没有将轮廓作为一个整体。现在OpenCV提供了一个方便的函数来实现这一步功能，cvFindContours方法。这个方法是openCV 在创建动态对象时存取内存的技术。在处理轮廓的时候通常需要使用序列化。了解这些基本概念后，我们就可以讨论轮廓识别的具体细节。最后讨论一些实际应用。

内存

OpenCV使用内存存储器来统一管理各种动态对象的内存。内存存储器在底层被实现为一个有许多相同大小的内存块组成的双向链表，通过各种结构，OpenCV可以从内存春初期中快速的分配内存或者内存返回给内存存储器。OpenCV中基于内存存储器实现的函数，经常需要向内存存储器申请内存空间。内存存储器可以通过以下四个函数访问：

cvMemStorage* cvCreateMemStorage方法。

```
/** Creates new memory storage.
   block_size == 0 means that default,
   somewhat optimal size, is used (currently, it is 64K) */

CVAPI(CvMemStorage*)  cvCreateMemStorage( int block_size
CV_DEFAULT(0));

/** Creates a memory storage that will borrow memory blocks from
parent storage */
CVAPI(CvMemStorage*)  cvCreateChildMemStorage( CvMemStorage* parent
);

/** Releases memory storage. All the children of a parent must be
released before
   the parent. A child storage returns all the blocks to parent when
it is released */
CVAPI(void)  cvReleaseMemStorage( CvMemStorage** storage );

/** Clears memory storage. This is the only way(!!!) (besides
cvRestoreMemStoragePos)
   to reuse memory allocated for the storage - cvClearSeq,cvClearSet
...
   do not free any memory.
   A child storage returns all the blocks to the parent when it is
cleared */
CVAPI(void)  cvClearMemStorage( CvMemStorage* storage );
```

cvCreateMemStorage用于创建一个内存存储器，参数blocksize是对应存储器的内存块大小，如果设置为零，则采用默认的存储空间大小64KB，该函数返回一个新创建的内存存储指针。cvReleaseMemStorage是通过storage获取有效的指针，然后释放指针所映射的内存空间，释放的方式和CV其它的结构方式相类似。

cvClearMemStorage方法是用于清空内存所占用的存储器，注意，该函数仅有一种释放内存分配的内存方法。该函数和上面函数的区别在于，该函数是释放内存后返回给存储

器，而不是直接返回给系统。可以通过clear的方式，循环重复使用内存空间。

就像是malloc可以从堆中分配内存空间一样，OpenCV中的cvMemstorageAlloc也可以从一个内存存储器中分配空间，只需要向cvMemStorageAlloc获取存储空间大小，然后就会返回相应的存储空间。

序列

序列是内存存储器中可以存储的一种对象。序列的本质是某一种链表存储结构，序列在内存中的实现是一种双端列表，所以，序列可以被快速访问或者删除。

结构cvSeq的定义如下所示：

```
/**
   Read/Write sequence.
   Elements can be dynamically inserted to or deleted from the
sequence.
*/
#define
CV_SEQUENCE_FIELDS()                                                  \
    CV_TREE_NODE_FIELDS(CvSeq);
     \
    int       total;          /**< Total number of
elements.            */  \
    int       elem_size;      /**< Size of sequence element in
bytes.   */  \
    schar*    block_max;      /**< Maximal bound of the last
block.     */  \
    schar*    ptr;            /**< Current write
pointer.               */  \
    int       delta_elems;    /**< Grow seq this many at a
time.        */  \
    CvMemStorage* storage;    /**< Where the seq is
stored.             */  \
    CvSeqBlock* free_blocks;  /**< Free blocks
list.                    */  \
    CvSeqBlock* first;        /**< Pointer to the first sequence
block. */
typedef struct CvSeq
{
    CV_SEQUENCE_FIELDS()
}
CvSeq;
```

cvSeq结构体源码如上图所示。cvSeq可以通过cvCreateSeq方法来获取创建一个Seq对象。具体的函数定义如下所示：

```
/** Creates new empty sequence that will reside in the specified
storage */

CVAPI(CvSeq*)  cvCreateSeq( int seq_flags, size_t header_size,
                            size_t elem_size, CvMemStorage* storage
);
```

CvSeq提供很多种形式来组合，可以设置特定的大小区域，便于存储数据。flag的属性定义如下所示：

```
#define CV_SEQ_ELTYPE_BITS           12
#define CV_SEQ_ELTYPE_MASK           ((1 << CV_SEQ_ELTYPE_BITS) - 1)
#define CV_SEQ_ELTYPE_POINT          CV_32SC2  /**< (x,y) */
#define CV_SEQ_ELTYPE_CODE           CV_8UC1   /**< freeman code:
0..7 */
#define CV_SEQ_ELTYPE_GENERIC        0
#define CV_SEQ_ELTYPE_PTR            CV_USRTYPE1
#define CV_SEQ_ELTYPE_PPOINT         CV_SEQ_ELTYPE_PTR  /**< &(x,y)
*/
#define CV_SEQ_ELTYPE_INDEX          CV_32SC1  /**< #(x,y) */
#define CV_SEQ_ELTYPE_GRAPH_EDGE     0  /**< &next_o, &next_d,
&vtx_o, &vtx_d */
#define CV_SEQ_ELTYPE_GRAPH_VERTEX   0  /**< first_edge, &(x,y) */
#define CV_SEQ_ELTYPE_TRIAN_ATR      0  /**< vertex of the binary
tree   */
#define CV_SEQ_ELTYPE_CONNECTED_COMP 0  /**< connected component  */
#define CV_SEQ_ELTYPE_POINT3D        CV_32FC3  /**< (x,y,z)  */
#define CV_SEQ_KIND_BITS        2
#define CV_SEQ_KIND_MASK        (((1 << CV_SEQ_KIND_BITS) - 1)
<<CV_SEQ_ELTYPE_BITS)
/** types of sequences */
#define CV_SEQ_KIND_GENERIC     (0 << CV_SEQ_ELTYPE_BITS)
#define CV_SEQ_KIND_CURVE       (1 << CV_SEQ_ELTYPE_BITS)
#define CV_SEQ_KIND_BIN_TREE    (2 << CV_SEQ_ELTYPE_BITS)
/** types of sparse sequences (sets) */
#define CV_SEQ_KIND_GRAPH       (1 << CV_SEQ_ELTYPE_BITS)
#define CV_SEQ_KIND_SUBDIV2D    (2 << CV_SEQ_ELTYPE_BITS)
#define CV_SEQ_FLAG_SHIFT       (CV_SEQ_KIND_BITS +
CV_SEQ_ELTYPE_BITS)
/** flags for curves */
#define CV_SEQ_FLAG_CLOSED     (1 << CV_SEQ_FLAG_SHIFT)
#define CV_SEQ_FLAG_SIMPLE     (0 << CV_SEQ_FLAG_SHIFT)
#define CV_SEQ_FLAG_CONVEX     (0 << CV_SEQ_FLAG_SHIFT)
#define CV_SEQ_FLAG_HOLE       (2 << CV_SEQ_FLAG_SHIFT)
```

可以依据不同的type来分配不同的存储空间属性。存储不同的数据。CV也提供了许多方法来处理这些序列，比如删除序列，出栈等方法。

```
/** Creates sequence header for array.
   After that all the operations on sequences that do not alter the
content
   can be applied to the resultant sequence */
CVAPI(CvSeq*) cvMakeSeqHeaderForArray( int seq_type, int header_size,
                                       int elem_size, void* elements,
int total,
                                       CvSeq* seq, CvSeqBlock* block
);
```

```
/** Extracts sequence slice (with or without copying sequence
elements) */
CVAPI(CvSeq*) cvSeqSlice( const CvSeq* seq, CvSlice slice,
                         CvMemStorage* storage CV_DEFAULT(NULL),
                         int copy_data CV_DEFAULT(0));

CV_INLINE CvSeq* cvCloneSeq( const CvSeq* seq, CvMemStorage* storage
CV_DEFAULT(NULL))
{
    return cvSeqSlice( seq, CV_WHOLE_SEQ, storage, 1 );
}
/** Removes sequence slice */
CVAPI(void)  cvSeqRemoveSlice( CvSeq* seq, CvSlice slice );

/** Inserts a sequence or array into another sequence */
CVAPI(void)  cvSeqInsertSlice( CvSeq* seq, int before_index, const
CvArr* from_arr );

/** a < b ? -1 : a > b ? 1 : 0 */
typedef int (CV_CDECL* CvCmpFunc)(const void* a, const void* b, void*
userdata );

/** Sorts sequence in-place given element comparison function */
CVAPI(void) cvSeqSort( CvSeq* seq, CvCmpFunc func, void* userdata
CV_DEFAULT(NULL) );

/** Finds element in a [sorted] sequence */
CVAPI(schar*) cvSeqSearch( CvSeq* seq, const void* elem, CvCmpFunc
func,
                           int is_sorted, int* elem_idx,
                           void* userdata CV_DEFAULT(NULL) );
/** Reverses order of sequence elements in-place */
CVAPI(void) cvSeqInvert( CvSeq* seq );

/** Splits sequence into one or more equivalence classes using the
specified criteria */
CVAPI(int)  cvSeqPartition( const CvSeq* seq, CvMemStorage* storage,
                            CvSeq** labels, CvCmpFunc is_equal, void*
userdata );
/************ Internal sequence functions ************/
CVAPI(void)  cvChangeSeqBlock( void* reader, int direction );

CVAPI(void)  cvCreateSeqBlock( CvSeqWriter* writer );

/** Adds several new elements to the end of sequence */
CVAPI(void)  cvSeqPushMulti( CvSeq* seq, const void* elements,
                             int count, int in_front CV_DEFAULT(0) );
/** Removes several elements from the end of sequence and optionally
saves them */
CVAPI(void)  cvSeqPopMulti( CvSeq* seq, void* elements,
                            int count, int in_front CV_DEFAULT(0) );
```

```
/** Inserts a new element in the middle of sequence.
   cvSeqInsert(seq,0,elem) == cvSeqPushFront(seq,elem) */
CVAPI(schar*)  cvSeqInsert( CvSeq* seq, int before_index,
                            const void* element CV_DEFAULT(NULL));
/** Removes specified sequence element */
CVAPI(void)  cvSeqRemove( CvSeq* seq, int index );

/** Removes all the elements from the sequence. The freed memory
   can be reused later only by the same sequence unless
cvClearMemStorage
   or cvRestoreMemStoragePos is called */
CVAPI(void)  cvClearSeq( CvSeq* seq );

/** Retrieves pointer to specified sequence element.
   Negative indices are supported and mean counting from the end
   (e.g -1 means the last sequence element) */
CVAPI(schar*)  cvGetSeqElem( const CvSeq* seq, int index );

/** Calculates index of the specified sequence element.
   Returns -1 if element does not belong to the sequence */
CVAPI(int)  cvSeqElemIdx( const CvSeq* seq, const void* element,
                         CvSeqBlock** block CV_DEFAULT(NULL) );
/** Initializes sequence writer. The new elements will be added to
the end of sequence */
CVAPI(void)  cvStartAppendToSeq( CvSeq* seq, CvSeqWriter* writer );

/** Combination of cvCreateSeq and cvStartAppendToSeq */
CVAPI(void)  cvStartWriteSeq( int seq_flags, int header_size,
                              int elem_size, CvMemStorage* storage,
                              CvSeqWriter* writer );
/** Closes sequence writer, updates sequence header and returns
pointer
   to the resultant sequence
   (which may be useful if the sequence was created using
cvStartWriteSeq))
*/
CVAPI(CvSeq*)  cvEndWriteSeq( CvSeqWriter* writer );

/** Updates sequence header. May be useful to get access to some of
previously
   written elements via cvGetSeqElem or sequence reader */
CVAPI(void)   cvFlushSeqWriter( CvSeqWriter* writer );

/** Initializes sequence reader.
   The sequence can be read in forward or backward direction */
CVAPI(void) cvStartReadSeq( const CvSeq* seq, CvSeqReader* reader,
                           int reverse CV_DEFAULT(0) );
```

从API层可以看到，提供了存储、读取、计算、增删改除等方法。还是比较强大的库。其次还有序列数组转换方法等。

轮廓

接下去说下查找轮库，轮廓实质上是一系列点所组成的结构，也就是一条条曲线，在OpenCV中一般用序列结构来存储这些轮廓信息。openCV提供了CVFindContours方法来查找这些点，实质上是从一副二值化图片中，查找相应的点存储到序列中。cvFindContours方法定义如下：

```
/**********************************************************************
********************\
*                              Contours
retrieving                                          *
\**********************************************************************
********************/
/** @brief Retrieves outer and optionally inner boundaries of white
(non-zero) connected
   components in the black (zero) background
@see cv::findContours, cvStartFindContours, cvFindNextContour,
cvSubstituteContour, cvEndFindContours
*/
CVAPI(int)  cvFindContours( CvArr* image, CvMemStorage* storage,
CvSeq** first_contour,
                            int header_size
CV_DEFAULT(sizeof(CvContour)),
                            int mode CV_DEFAULT(CV_RETR_LIST),
                            int method
CV_DEFAULT(CV_CHAIN_APPROX_SIMPLE),
                            CvPoint offset CV_DEFAULT(cvPoint(0,0)));
```

cvFindContours方法的输入必须是一副单通道的灰度图像，也就是8位图。