Adding to Ship Detection Example

Mr. Bryan Bagnall
SPAWAR Systems Center, Pacific
Phone: 619-553-4061
Email: bryan.bagnall@navy.mil

Mr. Sparta Cheung
SPAWAR Systems Center, Pacific
Phone: 619-553-5927
Email: sparta.cheung@navy.mil

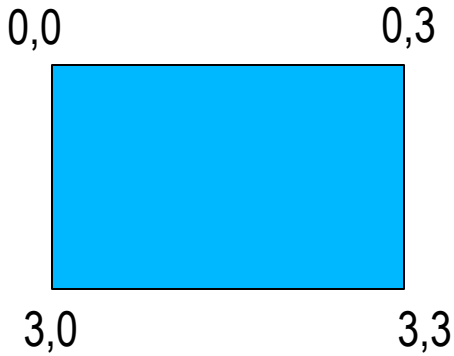
SPAWAR Systems Center PACIFIC

Overview of Talk

- Extracting the contours surrounding the blobs
- Using ossimGeoPolygon to store shapes
- Write contours to a Google Earth KML
- Writing the contours to an ESRI shapefile
- Masking out land using a shoreline shapefile
- Conclusions

Extracting the contours surrounding the blobs

- cvFindContours()
- /* contour retrieval mode */
- #define **CV_RETR_EXTERNAL 0**
- #define **CV_RETR_LIST 1**
- #define **CV_RETR_CCOMP 2**
- #define **CV_RETR_TREE 3**



- /* contour approximation method */
- #define **CV_CHAIN_CODE 0**
- #define **CV_CHAIN_APPROX_NONE 1**
- #define **CV_CHAIN_APPROX_SIMPLE 2**
- #define **CV_CHAIN_APPROX_TC89_L1 3**
- #define **CV_CHAIN_APPROX_TC89_KCOS 4**
- #define **CV_LINK_RUNS 5**

(0,0)(0,1)(0,2)(0,3)(1,3)(2,3)(3,3)(3,2)(3,1)(3,0)(2,0)(1,0)(0,0)

(0,0)(0,3)(3,3)(3,0)(0,0)

Extracting the contours surrounding the blobs

- int cvFindContours(CvArr* *image*, CvMemStorage* *storage*, CvSeq** *first_contour*, int *header_size=sizeof(CvContour)*, int *mode=CV_RETR_LIST*, int *method=CV_CHAIN_APPROX_SIMPLE*, CvPoint *offset=cvPoint(0*, 0)) Finds the contours in a binary image.
- Parameters: *image* – The source, an 8-bit single channel image. Non-zero pixels are treated as 1's, zero pixels remain 0's - the image is treated as binary. To get such a binary image from grayscale, one may use *Threshold*, *AdaptiveThreshold* or *Canny*. The function modifies the source image's content
- *storage* – Container of the retrieved contours
- *first_contour* – Output parameter, will contain the pointer to the first outer contour
- *header_size* – Size of the sequence header, if , and otherwise

Extracting the contours surrounding the blobs

- *mode* – Retrieval mode
  - **CV_RETR_EXTERNAL** - retrives only the extreme outer contours
  - **CV_RETR_LIST** - retrieves all of the contours and puts them in the list
  - **CV_RETR_CCOMP** - retrieves all of the contours and organizes them into a two-level hierarchy: on the top level are the external boundaries of the components, on the second level are the boundaries of the holes
  - **CV_RETR_TREE** - retrieves all of the contours and reconstructs the full hierarchy of nested contours
- *method* – Approximation method (for all the modes, except CV_LINK_RUNS, which uses built-in approximation)
  - **CV_CHAIN_CODE** - outputs contours in the Freeman chain code. All other methods output polygons (sequences of vertices)
  - **CV_CHAIN_APPROX_NONE** - translates all of the points from the chain code into points
  - **CV_CHAIN_APPROX_SIMPLE** - compresses horizontal, vertical, and diagonal segments and leaves only their end points
  - **CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS** - applies one of the flavors of the Teh-Chin chain approximation algorithm.
  - **CV_LINK_RUNS** - uses a completely different contour retrieval algorithm by linking horizontal segments of 1's. Only the CV_RETR_LIST retrieval mode can be used with this method.

Extracting the contours surrounding the blobs

- *offset* – Offset, by which every contour point is shifted. This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context
- The function retrieves contours from the binary image and returns the number of retrieved contours. The pointer first_contour is filled by the function. It will contain a pointer to the first outermost contour or NULL if no contours are detected (if the image is completely black). Other contours may be reached from first_contour using the h_next and v_next links. The sample in the *DrawContours* discussion shows how to use contours for connected component detection. Contours can be also used for shape analysis and object recognition - see squares.c in the OpenCV sample directory.

Extracting the contours surrounding the blobs

```
CvMemStorage* storage = cvCreateMemStorage(0);
CvSeq* contours = NULL;
cvFindContours(blobMask, storage, &contours, sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_NONE);
int contourCount = 0;

for(CvSeq* currentContour = contours; currentContour!=NULL; currentContour=currentContour->h_next)
{
contourCount++;
vector<ossimGpt> contour;
for(int contourPosition = 0; contourPosition < currentContour->total; ++contourPosition)
{
CvPoint* p = CV_GET_SEQ_ELEM(CvPoint, currentContour, contourPosition);
contourPoint = ossimIpt(p->x-offset, p->y-offset) + iUpperLeftTilePoint;
ossimDpt temp_dpt = ossimDpt(contourPoint.x, contourPoint.y);
ossimGpt temp_gpt;
geom->localToWorld(temp_dpt, temp_gpt);
contour.push_back(temp_gpt);
//writer.writePlacemark("contour points","",temp_gpt.lat, temp_gpt.lon, 0);
}
polygons.push_back(contour);
}
currentBlobNumber++;
cvReleaseMemStorage(&storage);
}
```

Using ossimGeoPolygon to store shapes

- ossimGeoPolygon
  - OSSIM data structure for holding polygons in world coordinates (latitude/longitude)
  - An ossimGeoPolygon is just a vector<ossimGpt>
- ossimGpt
  - OSSIM data structure for holding world coordinates (latitude/longitude)
  - ossimImageGeometry::localToWorld takes an image coordinate (ossimDpt) and returns a world coordinate (ossimGpt)

SPAWAR Systems Center PACIFIC

Using ossimGeoPolygon to store shapes

- We can add a private variable to our shipDetectionFilter
- vector<ossimGeoPolygon> polygons;
- Then when we extract the coordinates from the blobs we can store them in this variable
- Some important steps
  - vector<ossimGpt> contour;
  - geom->localToWorld(temp_dpt, temp_gpt);
  - contour.push_back(temp_gpt);
  - polygons.push_back(contour);

Writing the contours to a Google Earth KML

- Now that we have the coordinates from the blobs extracted using cvFindContours() and stored using a ossimGeoPolygon in our shipDetectionFilter class, we can now write them to files
- A good place to add this function is either in the destructor, or as a method that we are able to call after the chain has finished executing
  - We want to make sure that we call these functions only once at the end of writing to avoid re-writing the files repeatedly as would be the case if we put this code in the getTile() function

Writing the contours to a Google Earth KML

We will add a Placemark for each contour that we extract

```
void kmlWriter::writePolygon(string name, string description, ossimGeoPolygon& coordinates, int extrude, int tessellate, string altitude){
_stream << "<Placemark>" << endl;
_stream << "<name>" << name << "</name>" << endl;
_stream << "<description>" << description << "</description>" << endl;
_stream << "<Polygon>" << endl;
_stream << "<extrude>" << extrude << "</extrude>" << endl;
_stream << "<tessellate>" << tessellate << "</tessellate>" << endl;
_stream << "<altitudeMode>" << altitude << "</altitudeMode>" << endl;
_stream << "<outerBoundaryIs>" << endl;
_stream << "<LinearRing>" << endl;
_stream << "<coordinates>" << endl;
for(int i = 0; i< coordinates.size(); i++){
_stream << ""<< setprecision(15)  << coordinates[i].lon << "," << coordinates[i].lat << "," << coordinates[i].hgt << endl;
}
_stream << "</coordinates>" << endl;
_stream << "</LinearRing>" << endl;
_stream << "</outerBoundaryIs>" << endl;
_stream << "</Polygon>" << endl;
_stream << "</Placemark>" << endl;
}
```

Writing the contours to an ESRI Shapefile

- http://shapelib.maptools.org/
- .SHP File API
  - http://shapelib.maptools.org/shp_api.html

Writing the contours to an ESRI Shapefile

- **Shape Types**
- Shapes have types associated with them. The following is a list of the different shapetypes supported by Shapefiles. At this time all shapes in a Shapefile must be of the same type (with the exception of NULL shapes).
- #define SHPT_NULL 0
- 2D Shape Types (pre ArcView 3.x):
- #define SHPT_POINT 1 Points
- #define SHPT_ARC 3 Arcs (Polylines, possible in parts)
- #define SHPT_POLYGON 5 Polygons (possible in parts)
- #define SHPT_MULTIPOINT 8 MultiPoint (related points)

Writing the contours to an ESRI Shapefile

- 3D Shape Types (may include "measure" values for vertices):
- #define SHPT_POINTZ 11
- #define SHPT_ARCZ 13
- #define SHPT_POLYGONZ 15
- #define SHPT_MULTIPOINTZ 18
- 2D + Measure Types:
- #define SHPT_POINTM 21
- #define SHPT_ARCM 23
- #define SHPT_POLYGONM 25
- #define SHPT_MULTIPOINTM 28 Complex (TIN-like) with Z, and Measure:
- #define SHPT_MULTIPATCH 31

Writing the contours to an ESRI Shapefile

- **SHPObject**
- An individual shape is represented by the SHPObject structure. SHPObject's created with SHPCreateObject(), SHPCreateSimpleObject(), or SHPReadObject() should be disposed of with SHPDestroyObject().

Writing the contours to an ESRI Shapefile

- typedef struct {
- int nSHPType; Shape Type (SHPT_* - see list above)
- int nShapeId; Shape Number (-1 is unknown/unassigned)
- int nParts; # of Parts (0 implies single part with no info)
- int *panPartStart; Start Vertex of part
- int *panPartType; Part Type (SHPP_RING if not SHPT_MULTIPATCH)
- int nVertices; Vertex list
- double *padfX;
- double *padfY;
- double *padfZ; (all zero if not provided)
- double *padfM; (all zero if not provided)
- double dfXMin; Bounds in X, Y, Z and M dimensions
- double dfYMin; double dfZMin; double dfMMin; double dfXMax; double dfYMax; double dfZMax; double dfMMax; } SHPObject;

SPAWAR Systems Center PACIFIC

Writing the contours to an ESRI Shapefile

```
SHPObject * SHPCreateObject(
int nSHPType,
int iShape,
int nParts,
int * panPartStart,
int * panPartType,
int nVertices,
double *padfX, double * padfY, double *padfZ,
double *padfM );
```

Writing the contours to an ESRI Shapefile

```
void shipDetectionFilter::writeShpFile(){
ossimFilename shapefileName("detections.shp");
SHPHandle hSHP = SHPCreate(shapefileName.c_str(),SHPT_POLYGON);

SHPObject *psObject;
if(hSHP == NULL){
cout << "Shapefile could not be created" << endl;
}

DBFHandle hDBF;
hDBF = DBFCreate(shapefileName.c_str());

//Add all fields that we wish to access from DBF
DBFAddField(hDBF, "Area", FTString, 30, 0);

for(int i = 0; i<polygons.size(); i++){
int nVertices = polygons[i].size();

int* panParts = new int[nVertices+1];
double* padfX = new double[nVertices+1];
double* padfY = new double[nVertices+1];
```

Writing the contours to an ESRI Shapefile

```
  for(int j=0; j<nVertices; j++){
padfX[j] = polygons[i][j].lon;
padfY[j] = polygons[i][j].lat;
//padfZ[j] = polygons[i][j].hgt;
}
//the last must be equal to the first
padfX[nVertices] = polygons[i][0].lon;
padfY[nVertices] = polygons[i][0].lat;

psObject = SHPCreateObject(SHPT_POLYGON, -1, 1, panParts, NULL, nVertices + 1,
padfX, padfY, NULL, NULL);
SHPWriteObject(hSHP, -1, psObject );
DBFWriteStringAttribute(hDBF, i, 0, ossimString::toString(polygons[i].area()));
SHPDestroyObject(psObject );
delete [] panParts;
delete [] padfX;
delete [] padfY;
}

SHPClose(hSHP );
DBFClose( hDBF );
}
```

Masking out land using a shoreline shapefile

- You may have discovered that there are MANY false alarms for the ship detector coming from land
- To combat this, we can mask out the land before processing
- OSSIM has a class which makes this easy for us
- ossimMaskFilter
  - We can add this to the image chain before our shipDetectionFilter to allow our filter to only see parts of the image with water
  - Can work with a shapefile input such as the world vector shoreline

Masking out land using a shoreline shapefile

```
//mask land
ossimRefPtr<ossimImageHandler> inputShp =
ossimImageHandlerRegistry::instance()->open(inputShpName);
cout << "Creating mask filter..." << endl;
ossimRefPtr<ossimMaskFilter> maskFlt = new ossimMaskFilter();
if ( inputShp.valid() ){
if ( inputShp->getClassName() == "ossimOgrGdalTileSource" ){
ossimViewInterface* shpView = PTR_CAST(ossimViewInterface, inputShp.get());
if (shpView){
// Test masking image handler and shape file.
// Set the shape reader's view to that of the image's.
shpView->setView(ih->getImageGeometry().get());
// Turn fill on...
ossimRefPtr<ossimProperty> fillProp =
new ossimStringProperty(ossimString("fill_flag"),
ossimString("1"));
inputShp->setProperty(fillProp);

//ossimRefPtr<ossimMaskFilter> maskFlt = new ossimMaskFilter();
maskFlt->setMaskType(ossimMaskFilter::OSSIM_MASK_TYPE_INVERT);
maskFlt->connectMyInputTo(0, ih.get());
maskFlt->setMaskSource(inputShp.get());
maskFlt->initialize();
}
}
}
```