Computer Vision & Computer Graphics Term Project Reports

Arda Aytekin & Onur Varol

These projects consist of two applications one programmed in MATLAB for computer vision course and the other is programmed in C# using WPF and XNA technologies of the Microsoft for computer graphics course.

These two projects are combined for purpose. In computer vision course project which is called **Map Annotator** is built for segmentation of maps and exporting the objects of the map in XML format. The other project for computer graphics is called **Map Builder**. In this project sketch data is used to generate height map for the terrains of the scene and sketch information is also used to place objects to the scene. 3D scene is created using this sketch information and objects are placed on the coordinates corresponding to the sketches.

Map Annotator

Map annotator project is generated to subtract information from the image database. This information is objects in the images and terrains like forests and sea or contour lines. In this project tactical maps are gathered from the database of tactical decision game site Corpus Gazette.

In map annotator project images are segmented into different cluster which contains different parts of the maps. Normal clustering procedure is applied on image for given cluster number and random clusters. But in our problem we define the initial cluster mean and get insight about the clusters and the content among them.

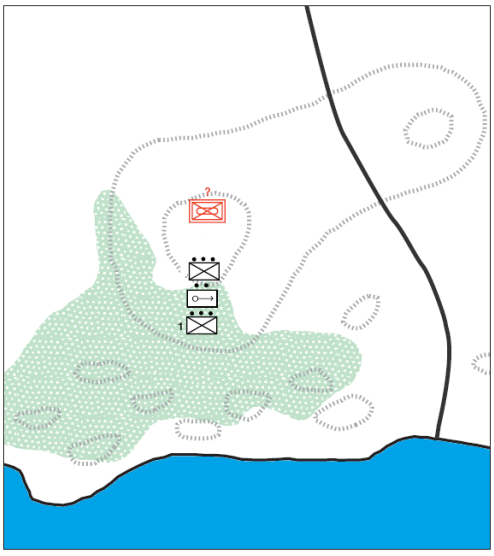

Figure 1: Sample test image

Clustering

In sample maps like shown in Figure 1, we can define the sea and forest and also the objects, contour lines and warnings can be seen. Using clustering algorithm we can extract these content into different clusters as shown in below.

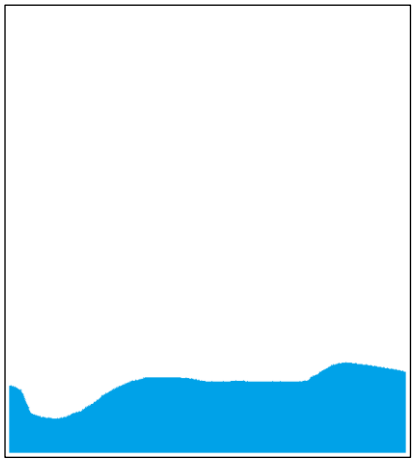

Figure 2: Sea

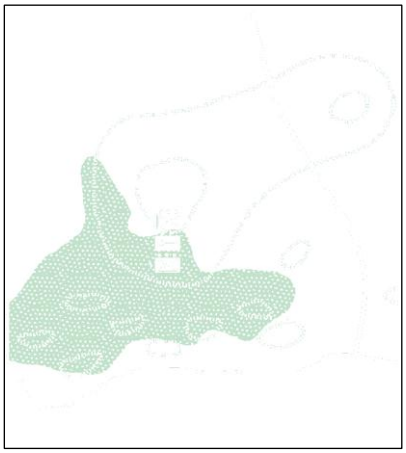

Figure 3: Forest terrain

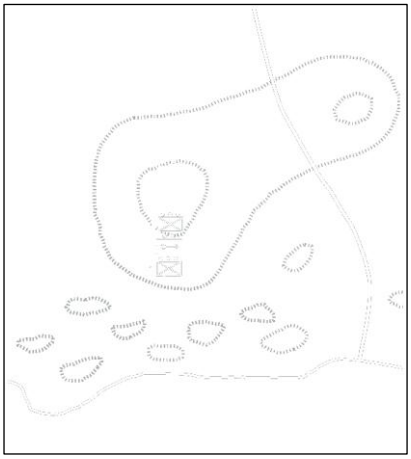

Figure 4: Contour lines

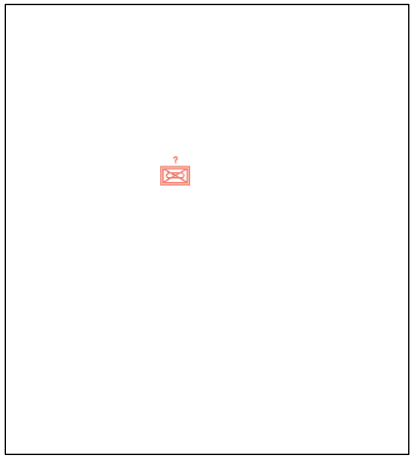

Figure 5: Warning pbjects

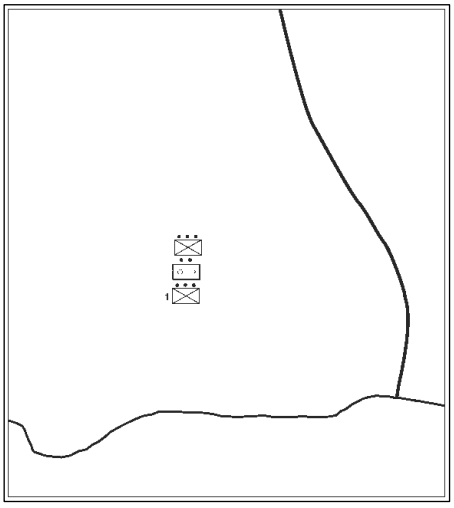

Figure 6: Objects

As we seen in result of our clustering algorithm which is the implementation of the K-Means algorithm, clusters are generated for different contents.

Black channel of the images like Figure 6 is used to extract objects for further operations. For this purpose Windowed Hough Transform is used (1).

The object detection task of the project is carried out using the so-called “Rectangle Detection based on a Windowed Hough Transform” method [1] on the “black” channel obtained from “K-means Clustering” process.

The problem here, in this project, is to detect certain rectangular images representing military vehicles. These rectangles may have arbitrary dimensions and/or orientations. If a “Generalized Hough Transform” were to be used to detect a generic rectangle, there would be 5 parameters in the accumulator array to reflect the freedoms in the center coordinates (2 variables; x, y), length, width and the orientation. If, on the other hand, other algorithms such as those the authors of the paper [1] mention about are to be used, all the dimensions of the rectangles in the image must be known prior to using the algorithm.

Since the rectangular shapes in the maps may have arbitrary dimensions and/or rotations in our case, the proposed algorithm is implemented and used during the project. This algorithm is based on the Hough Transform to detect line segments and it uses the geometric properties of a rectangle consisting of 4 line segments.

To apply the algorithm correctly, each and every pixel of the image is scanned and a “donut-like” shaped window is used to cover some neighborhood of the related pixel. This “donut-like” neighborhood is selected with respect to the minimum and maximum line segment lengths to be “detected”. In other words, a minimum diameter is used to disregard line segments smaller than a threshold, while the maximum diameter is used to ignore the lines having exaggerated lengths. And the inner region (i.e. the “donut” region) is scanned to form the Hough Image of the neighborhood. An example is given in Figure 7 to reflect this concept:

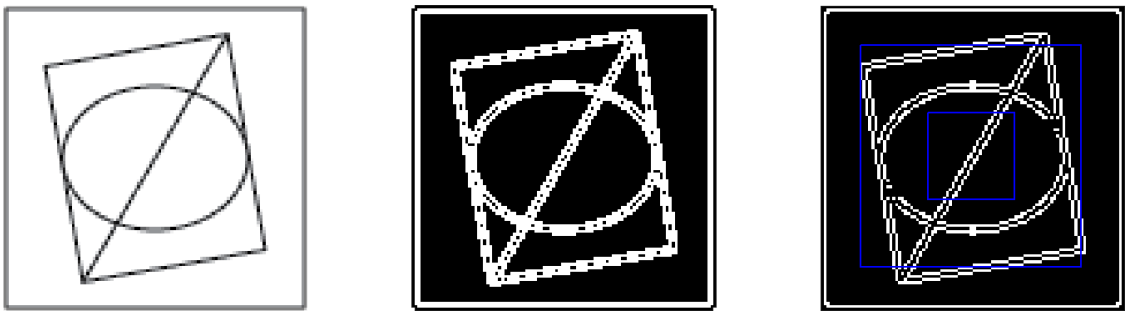

Figure 7: Example of a Donut-Region

The above figure shows the image on the left hand side, and its “Canny enhanced” edge map in the middle. This 10° rotated rectangle image with an elliptic shape and a main diagonal inside is created synthetically to measure the performance of the proposed algorithm. The right hand side of Figure 7shows the rectangular region located on the image center. The region between the smaller and the

greater blue rectangles is scanned and the following Hough Image is obtained given on Figure 8**Error! Reference source not found.**:

Figure 8: Hough Image of the Test Region

The left hand side Hough Image is obtained by constructing a 2D accumulator array of discretized θ and ρ variables. These Hough Images are of size ${}^{4}/_{3}\ D_{max}\ \times {}^{4}/_{3}\ D_{max}$, where the D_{max} is the greater length of the window rectangles. The middle image of Figure 8**Error! Reference source not found.** shows the enhanced Hough Image obtained by utilizing a "Butterfly Evaluator" to reject noise apparent in the image file. Finally the local maxima of the enhanced Hough Image are obtained and shown in the right hand side part of the above figure.

These local maxima represent the peaks (i.e. the lines) existent in the donut region located on the center of the image. The following geometric properties are used to select the proper pairs of those peaks to detect a, if any, rectangle in the test region [1]:

- The (four) peaks appear in pairs: the first one is formed by peaks H_1 and H_2, at $\theta = \alpha_1$; the second one is formed by peaks H_3 and H_4, at $\theta = \alpha_0$. Two of these peaks can be seen in the right hand side image clearly, appearing in the top-middle and bottom-middle parts of the image.
- Two peaks belonging to the same pair are symmetric with respect to the θ axis, i.e., $\rho_1 + \rho_2 = 0$ and $\rho_3 + \rho_4 = 0$.
- The two pairs are separated by $\Delta\theta = 90°$ in the θ axis, i.e., $|\alpha_1 - \alpha_0| = 90°$.
- The heights of two peaks within the same pair are exactly the same, and represent the length of the respective line segments, i.e., $C(\rho_1, \theta_1) = C(\rho_2, \theta_2) = b$ and $C(\rho_3, \theta_3) = C(\rho_4, \theta_4) = a$.
- The vertical distances (ρ axis) between peaks within each pair are exactly the sides of the rectangle, i.e., $\rho_1 - \rho_2 = a$ and $\rho_3 - \rho_4 = b$.

Discretization steps for ρ and θ are selected as in the paper [1], and the threshold values are chosen more strict than that of the paper to search for the rectangle in the given example of Figure 7**Error! Reference source not found.**. With angular thresholds of $2°$ and a normalized length threshold of 0.4, using the $D_{min} = 30\ px, D_{max} = 75\ px$ rectangular window, the $10°$ rotated rectangle of height $73\ px$ and width $61\ px$ is detected to be a rectangle of,

- height $74.25\ px$,
- width $62.25\ px$, and,
- an orientation angle of $9°$

As can be seen, the algorithm works perfectly to detect the rectangular object created synthetically.

One final problem of this algorithm is that it finds more than one rectangles when the neighboring pixels of the center is scanned with the same rectangular region/window. To select the proper one among these rectangles (i.e., to remove the duplicated rectangles), each and every rectangle is given an error measure constructed by using their distance and angular errors (i.e., their deviation from the thresholds) and the rectangle having the least error measure is selected to be the one we are searching for.

The algorithm used in this manner is given in the Appendix, and implemented to detect the rectangular objects existent in the "black" channel of the image obtained from the "K-means clustering" process.

Classification

In classification step clipped images are converted into feature vectors. We use IDM features for the feature extraction in this problem. IDM is a rotation and scale invariant feature which is used for sketch based applications.

IDM features are generated by scaling image in size of 12x12 and rotated for some distinct angle values. These images are used to generate feature vectors. These feature vectors are then classified using the SVM.

Figure 9: Training set

Classification is applied for 4 class dataset in each class there is one training image. Objects are the synthetic in images, so classification is applied these objects end up with an high classification rates. But objects clipped from the images using the windowed Hough transform may contain some noise and these images may classify incorrectly.

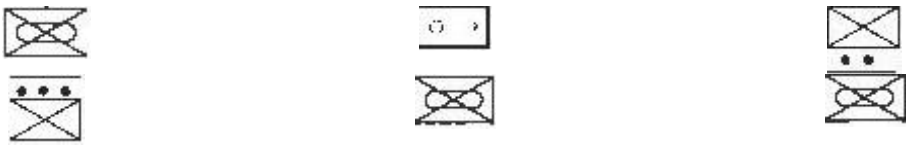

Figure 10: Some test images

XML Exporter

Classification results and also other components like contour lines and terrain are planned to export as an XML file for the use of Map builder. Current version of our project is capable of the exporting object coordinates and the classification results.

Sample XML file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<MapAnnotation id="02-Jun-2011 16:15:07" xml_tb_version="3.1">
  <objects>
    <boundingBox x1="100" y1="50" x2="100" y2="50">
    </boundingBox>
    <class index="1">
    </class>
  </objects>
  <objects>
    <boundingBox x1="110" y1="55" x2="110" y2="55">
    </boundingBox>
    <class index="7">
    </class>
  </objects>
  <isohips>
    <point x="100" y="50">
    </point>
    <point x="105" y="55">
    </point>
    <point x="110" y="60">
    </point>
    <point x="115" y="65">
    </point>
    <point x="1100" y="50">
    </point>
    <point x="1105" y="55">
    </point>
    <point x="1110" y="60">
    </point>
    <point x="1115" y="65">
    </point>
  </isohips>
  <isohips>
    <point x="100" y="50">
    </point>
    <point x="105" y="55">
    </point>
    <point x="110" y="60">
    </point>
    <point x="115" y="65">
    </point>
  </isohips>
</MapAnnotation>
```

Map Builder

In this project sketch based application for the 3D military map generation is made. Military units are recognizes and terrains and objects are created in 3D maps. Map builder application is for 3D realization of the given 2D image or sketch data. In this project different technologies are combined together. XNA framework is used for creating 3D platform and graphic applications are created using Microsoft Game Studio. Sketch based application is also created as a Microsoft WPF (Windows Presentation Foundation) application and InkCanvas is used for sketch operations.

Microsoft game development platform XNA is the main part of the projects and also WPF is used for the sketch data interaction with the users. These two technologies are not working together normally, but Arcane.XNA is supply library which modify the XNA project to be able to work in WPF form controls.

InkCanvas

InkCanvas is a WPF control for sketch interactions with users. It is used in our application to get sketches which is used to determine objects and the contour lines. Sytlus points are also collected in this application for future use like generating height map and the sketch recognition for object classification.

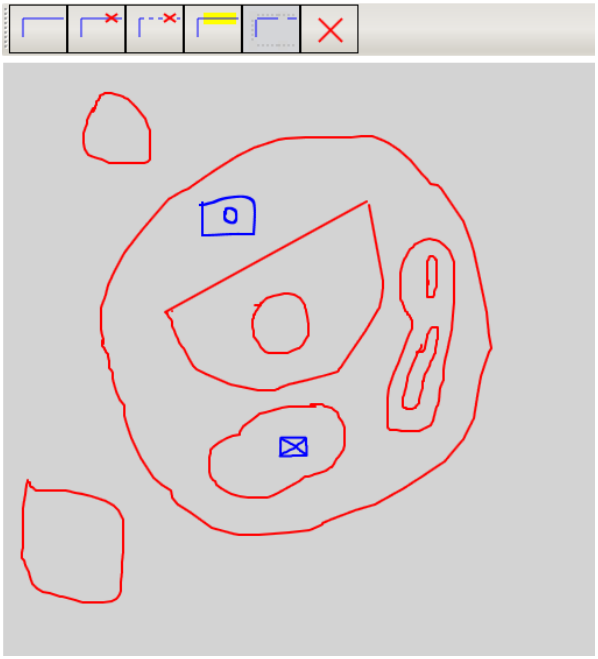

Figure 11: InkCanvas control

Terrain generation

In this application 3D world is created using the height map which is represented as a bitmap files and pixel values are also correspond to the heights of the locations. These height maps are created

as a step functions between the area of the contour lines and each after each line pixel values are increases.

Figure 12: Height map

Height maps are also filtered with Gaussian kernel to smooth the surface of the bitmap file and indirectly the surface of the terrain.

This height map then compiled with the XNA model builder and created terrain is saved as an xnb file. One disadvantage of the XNA is height maps are not allowed to change dynamically. XNA builds each model before compile the project to optimize the performance of the pipeline.

3D world

XNA can create 3D representation of the world and then objects are applied on the scene in real time. Object coordinates are determined using the image coordinates and the height map values.

Figure 13: Game scene

Object models are gathered from the Google 3D warehouse and converted into fbx file format which is used by XNA. But conversion of the files increase file size very much and model files took more memory size after conversation. That's why the models are used in limited variety. Also sketch recognition is planned to contain 20 different classes of data. This data set is called COAD (Course and action diagrams). However implementation of the IDM features is not working properly.

Discussion

- K-means (color) clustering worked well with predefined cluster means
    - Reduced image sizes may improve the overall clustering performance
- Windowed Hough Transform uses few parameters than that of a Generalized Hough Transform for a generic rectangle
    - A priori knowledge about the locations of the rectangles may be used to further improve the performance – hard in our case
- IDM features are sufficient for this specific type of problem
    - Bounding boxes of the detected rectangles do NOT contain some useful information, which leads to misdetections
- XNA platform does NOT support real-time model modifications
    - Another platform may be chosen, if at all possible
- Contour lines are not continuous
    - Morphological operations can be implemented to close the general shape
- Water and tree information needs to be sent to the program

References

1. Jung, C.R., Schramm, R., “Rectangle Detection based on a Windowed Hough Transform”, 2004.

2. OU YAN G T. , DAV I S R.: A visual approach to sketched symbol recognition. In Proc. International Joint Conferences on Artificial Intelligence (2009)