Information and Computing Sciences

Utrecht University

VSTEP

Master thesis project

Automatic conversion of scanned sea charts into 3D simulation models

Supervisors:
Remco Veltkamp
Ben Borrie
Pjotr van Schothorst

Author:
Andreea Barac

ICA3622479
Utrecht, 2012

I would like to thank my supervisor Remco Veltkamp who offered me a good supervision of my thesis and great support throughout the project. I would also like to thank VSTEP and to my co-supervisors, Ben Borrie and Pjotr van Schothorst, for their guidance.

My thanks also goes to the Romanian-American Foundation for their scholarship which allowed me to finish my master thesis. Last but not least, I would like to express my gratitude to my family and to my boyfriend, who patiently have been supporting me morally.

Abstract

Automatic nautical chart recognition and interpretation is a research topic that has been going on for many years. Nautical chart digitization has a variety of applications in navigation or in the development of navigational software, but also in educational applications like 3D training simulators which require a realistic representation of the seabed and its surroundings. This thesis presents a study on converting 2D scanned nautical chart images into 3D models. It is an exploration of some of the possibilities and problems occurring when designing and implementing such a system.

In order to obtain a 3D model, the scanned sea chart images have to be digitized. In digitizing a scanned sea chart, one of the major challenges is to properly separate and identify symbols on the map. The approached method first separates the background and the foreground pixels with a threshold-based segmentation method applied on the gray-scale image and then identifies individual objects in the image by searching for all connected components in the segmented binary image.

Another challenge is the classification of individual objects. The study brings a solution for the classification of different types of objects in a sea chart, focusing on the proper classification of spot soundings. Geometrical features like area, center of gravity, bounding box, density, orientation are used to build innovative decision rules that classify objects into several types of lines, characters or other symbols. The spot soundings are later recognized and interpolated to create a 3D surface of the maritime terrain. Tesseract OCR engine is used for character recognition. The spot soundings are interpolated using a method called Inverse Distance Weighting with Natural Neighbors. The interpolation method assumes that nearby points should have a greater influence than further away points. The nearby points are the vertices of the Delaunay triangle containing the interpolated point and are called natural neighbors.

The result of this research is a complete system that converts 2D scanned images into 3D simulation models. However, the performance of the algorithm is not 100% correct. Some issues remain and can be improved by further work.

Contents

Part I

Introduction

Chapter 1

Introduction

1.1 Project statement

VSTEP develops training simulators that are used at maritime schools, to let students practice ship handling and maneuvering in a safe and controlled environment. For schools, it is important to offer students local ports in the simulator, so they get a matching experience with the real on-board practice they will get after the simulator training. Creating realistic ports is however a lengthy and therefore expensive process.

Part of the port creation process may be automated using intelligent image interpretation algorithms, used on scanned sea charts. These elements could be automatically recognized and interpreted:

1. Coastlines
2. Seabed depth contours
3. Depth soundings, i.e. numbers on the chart that correspond with the water depth at low tide
4. Buoy data: identifier, shape, and light characteristics.

VSTEP's requirements were divided in two separate projects:

1. Creating an application in C++ that extracts spot soundings (numeric depth measurements) from a scanned sea chart, the resulting point cloud data should then be tessellated and then rendered to a 32-bit elevation GeoTIFF file. The resolution of the GeoTIFF should be user definable in terms of meters per pixel.
2. Creating an application in C++ that extracts depth contour lines from a scanned sea chart and then exports each contour line to a separate EPS file.

The conversion of a scanned sea chart into a 3D simulation model was further divided into several steps: segmentation, identification and classification of different symbols that appear on a scanned sea chart, digit recognition for identifying the spot soundings and interpolation of spot soundings. Because of time constraints, the current research brings a solution to the first part of the project, but some steps for the second part were also made. The following sections of this chapter present an overview of scanned sea charts and symbols, mentioning some of the features that characterize each symbol. The chapter ends with a resume of the previous work done in this field and an overview of the challenges that rise when designing a module for automatic conversion of scanned sea charts into 3D models.

1.2 Scanned sea charts

A sea chart or a nautical chart is a graphic representation of a maritime area and the adjacent coastal regions. Usually, it shows depths of water, natural features of the seabed, details of the coastline, information on tides and currents and human-made structures such as harbors, buildings and bridges. With its help, the navigator plots courses, ascertains positions and views the relationship of the ship with the surrounding area. The sea chart assists the navigator in avoiding dangers and arriving safely at his destination.

Nautical charts may take the form of charts printed on paper or electronic navigational charts. Conventional maritime charts are printed on large sheets of paper at a variety of scales. A scanned nautical chart represents the digital scanned image of a paper chart. The quality of a scanned chart depends on the scanner resolution and on the quality of the original printed chart (for example, an old printed chart will be blurred and faded). An example of a scanned sea chart is displayed in Figure 1.1.

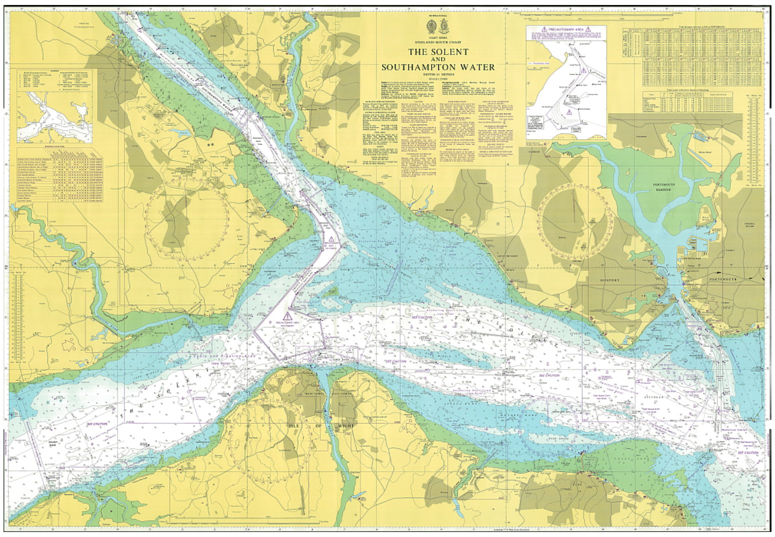
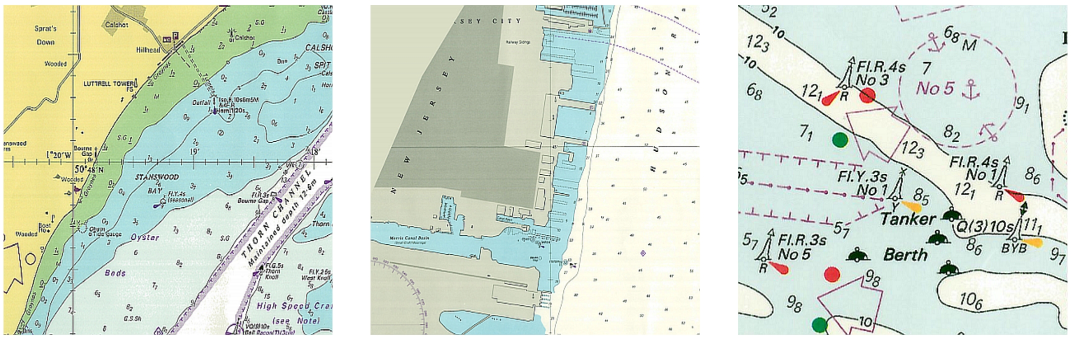


Figure 1.1. The Solent and Southampton Water

Most charts use color to emphasize various features and to facilitate chart reading and interpretation, as can be seen in Figure 1.2:

- land areas are shown in *gray or a yellowish* color;
- water areas are shown in *white*, except in shallow regions, which are displayed in *blue*;
- submerged areas which at times uncover at some tidal stages are shown in *green*;

- *purple* is used for many purposes on charts;
- buoys and other aids are appropriately colored in *red, green, white, yellow*, while lighted buoys of any color have a purple disc over a dot;
- *black* is used for most symbols and printed information.



(a) Yellowish land, water, submerged areas, purple symbols

(b) Grayish land, water, submerged areas

(c) Buoys, purple and black symbols

Figure 1.2. Different colors used in nautical charts

Nautical charts are labeled with navigational and depth information, such as channels, anchorages, geographic names, fixed aids to navigation (e.g. lighthouses), bottom characteristics, depths, underwater hazards and obstructions, pilotage areas, dangerous wrecks. Other information, such as the meaning of symbols and abbreviations, special notes of caution, units of measurement may be found on the chart. The next section gives an overview on the symbols present in a sea chart and their features.

1.3 Symbols in sea chart images

The chart uses symbols to provide information about the nature and position of features useful to navigators. Figure 3.1 shows some examples of chart symbols. For a more detailed review of the symbols on nautical charts refer to [29].

1. **Soundings**
   Depths which have been measured are indicated by numbers shown on the chart. Usually they are black, may be either vertical or slanting (depending on the font face) and are written with different font sizes and faces. Decimal digits are written as subscripts, without a decimal point. Soundings above sea level are indicated using a horizontal line underneath them.
2. **Depth of contour lines**
   Contour lines are lines connecting points of equal depth; the depth is usually indicated on a chart with numbers which have a grayish color and are aligned with the contour line.
3. **Letters, words**
   They can represent geographic names or abbreviations, but also information about the chart. They are usually black or purple, may have a variety of orientations, font sizes and font faces even inside the same chart.

4. **Lines**
   Usually, straight lines represent a straight course of water or tidal information, while the curved lines represent contour lines. There are several types of lines that may be present on a sea chart:

   - *Vertical straight lines* - they can be either black or purple, continuous or dotted. Usually, they are used to delimit an area with specific properties: restricted areas, international boundaries and national limits, etc.
   - *Horizontal straight lines* - just like the vertical straight lines, they can be either black or purple, continuous or dotted and are used to delimit the same type of areas.
   - *Oblique straight lines* - they can be black or purple, continuous or dotted. They can symbolize natural or cultural features, ports, etc.
   - *Curved lines* - they may be black or gray and may have different sizes; curved lines are mostly used to represent natural features, like coastlines, shorelines, rivers, lakes, contour lines etc.

5. **Other symbols**
   A lot of symbols are used in nautical charts to provide more information to the navigator. They can represent aids of navigation, like lighthouses and buoys, vegetation, cultural features etc.. They may be black, purple, yellow, red, green, have different sizes and different shapes (dots, circles, polygons, stars etc.). Refer to [9] for more information about chart symbols.

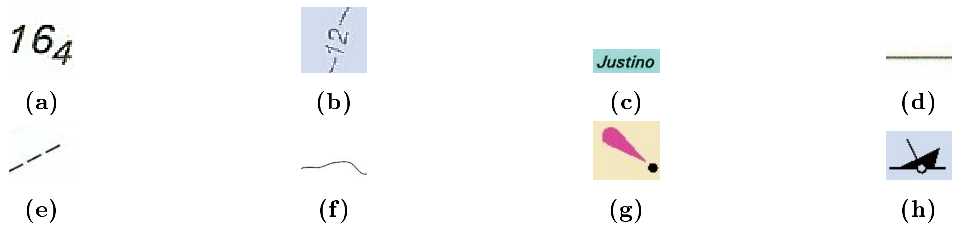


Figure 1.3. Symbols in nautical charts: a) soundings; b) depth of a contour line; c) string of characters; d) horizontal line; e) oblique lines; f) curved line; g, h) examples of other symbols.

1.4 Previous work

Research on automatic map recognition has been going on for many years resulting in a huge amount of publications. Early reports already introduced the first steps of this automatic procedure: digitization of the original paper chart using a scanner and thresholding. The following steps are different, depending on the method approached by the authors.

1.4.1 Features for character recognition in scanned sea charts

In 1996, Trier et al. [33] published a survey of feature extraction methods for character recognition. Their prime interest was to recognize hand-printed digits in hydrographic maps. Trier pointed the selection of a feature extraction method as being the most important part in achieving a high recognition performance and discussed different methods in terms of invariance properties, reconstructability and expected distortions and variability of the characters. They also discussed methods for different representation of the characters, such as solid binary characters, character contours, skeletons or gray level subimages of each individual character. When dealing with a gray-scale image, template matching, unitary image transforms, zoning, geometric moment invariants or Zernike moments may be used for character recognition. They obtained the best results by using Zernike moments, which can be made invariant w.r.t. rotation and illumination and mentioned that at least 8-11 moments are needed for a complete description of a character. Projection histograms may also be used for binary images, but Trier concluded that the Zernike moments gave the best performance in terms of recognition accuracy. For binary contours, they discussed methods that use contour profiles, spline curve approximations, elliptic Fourier descriptors or other Fourier descriptors and reported that the elliptic Fourier descriptors have the best performance; these descriptors can be made independent of the starting point, invariant w.r.t rotation or size. After calculating the features of the characters, they propose training a neural network classifier as the last step of the recognition process. The work of Trier et al. inspired a lot of researchers that worked on character recognition.

1.4.2 Automatic interpretation of maps

Some authors focused on the automatic extraction of contour lines from topographic maps. A topographic map essentially consists of feature lines and area features; different features are printed with different colors. Brown is used to depict contour lines, which are smooth, continuous curves that sometimes overlap. The algorithms for contour lines extraction usually have three steps: segmentation by thresholding, contour thinning and reconstruction of broken contour lines. Ghircoias and Brad [24] designed a semi-automatic algorithm that used K-Means[1] to segment the scanned map. They used as initial centers a set of colors selected by the user and removed the resulting noise on the basis of a minimum distance criteria. The reconstruction of the broken contours was based on the concept of medial axis of a curve[2], approximated by the vertices of the Voronoi diagram. Arrighi and Soille [16] developed a method based on morphological transforms. They extracted the contour lines by selecting all the pixels with red hue and removed the noise with the hit-or-miss transform[3]. For reconnection, they extracted the endpoints of the lines with the hit-or-miss transform, skeletonized the lines and used a combination of distance and direction criteria. After the reconstruction, the user could insert contour line elevations to generate a digital elevation model. The same segmentation method was used by Xin et al. [36]. However, they tracked the contour

[1] Refer to chapter 2 for an overview of the K-Means algorithm.

[2] The medial axis of a curve represents the locus of the center of the circles that are tangent to the curve in more than two points.

[3] The hit-or-miss transform is a general binary morphological operation that is used to look for particular patterns of foreground and background pixels in the image; it takes as input a binary image and a structuring element that may contain both foreground and background pixels and produces another binary image as output.

lines with an active contour model algorithm[1] and used a Generalized Gradient Vector Flow algorithm[2] to reconnect contour lines. Frischknecht et al. [23] used a knowledge-based template matching algorithm for automatic interpretation of topographic maps. Here, segmentation is used as a preselection technique to find regions of high, low or no interest. Individual segments are found with a run-length-encoded method. Features as area, perimeter, compactness, orientation, center of gravity, height, width are computed for each segment to enlarge the knowledge-basis.

Because these methods are focused on finding and reconstructing contour lines in maps, they are not suited for the current project, in which all symbols have to be identified and all the soundings recognized. Furthermore, nautical charts display more information and symbols with different colors than topographic maps. Thus, they cannot be used to automatically interpret a nautical charts.

Ablameyko et al. [13] developed a method for an automatic/interactive interpretation of color maps. After the binarization of the color image using RGB and HSV color spaces, they vectorized the image by thinning the objects, extracting the contours and calculating geometric features and relations between objects. They also proposed a method for automatic identification of dashed lines based on spatial distances and the minimal and maximal width and length of the objects. In the end, they created a module for interactive image interpretation. Some parts of Ablameyko's method are similar to the proposed algorithm, however their method is not an automatic process for image interpretation since it requires user intervention.

1.4.3 Automatic character recognition in maps

Other authors focused on the automatic recognition of characters in maps. Eikvil et al. [20] first extracted the lines in the map and then identified objects in the image by extracting the remaining foreground pixels. They extracted features based on the Fourier expansion of the contour of the symbols, which can be made invariant to scale, shift and rotation, classified the objects using Bayes and grouped them based on their location in the image. Chiang et al. [17] tried to recognize text labels in raster maps using morphological transforms. After applying a Mean-shift filter to smooth the image and reduce the noise, they performed color-quantization with a median-cut algorithm; the text was extracted by letting the user select a set of colors used on text in the map. To identify strings, they computed the width and height of characters for each font size from samples provided by the user; the dilation operator[3] was used to merge nearby characters and group them together. The orientation of the characters was calculated with the closing operator[4]. After string orientations have been identified, the strings are rotated clockwise and anti-clockwise and then passed to an OCR to

[1]Active contour model, also called snake, is an algorithm used to delineate an object outline from a possibly noisy 2D image.

[2]Generalized gradient vector flow controls curve contour features through appropriate restriction applied to deformable curves. It makes edge attraction flow smoothly to the image border, therefore it expands decisive range of attraction flow.

[3]The dilation operator gradually enlarges the boundaries of regions of foreground pixels in binary images. Thus, areas of foreground pixels grow in size, while holes within those regions become smaller.

[4]The closing operator preserves background regions that have a similar shape to the structuring element, while eliminating all other regions of background pixels.

determine their real horizontal orientation. Velasquez et al [34] computed the gray-scale image and performed segmentation using threshold values selected from a histogram of frequency gray values. To separate touching or overlapping characters, they used V-lines (4 horizontal lines measured at different height w.r.t the character) and V-curves (for curvilinear text) and used an artificial neural network for recognition.

The authors reported a good performance of character recognition for all the methods. However, they didn't focused on the separation of characters from other objects that appear in maps.

In conclusion, researchers focused on treating specific parts involved in the process of automatic interpretation of maps, such as character recognition or contour line extraction. However, a complete system that performs this task still doesn't exist. The necessity of a system that automatically interprets maps arises in a lot of different activities which require an automatic digitization of existing maps.

1.4.4 Spatial interpolation methods

The interpolation of spatial data has been considered in many different forms. Myers [27] and Li [26] give good overviews of the existing methods. They divide the methods into *deterministic or non-geostatistical methods* and *stochastic or geostatiscal methods.* Li also presents a comparison and evaluation of the methods. Some representative examples from both categories are summarized below.

Deterministic methods

1. *Kernel approximation*
   If the sample points are written as a sum of delta functions [1] , then the 'spikes' given by the delta function can be replaced with a kernel approximation or interpolation (for example, an exponential function). Usually, the kernel function has a parameter called *band width*, which determines the area of action of the kernel. By adjusting the band width, the interpolator can give greater or lesser weights to data in terms of how close the data locations are to the point where the interpolation is desired. Thus, it can be adjusted to perform a smooth or a strict interpolation.

2. *Inverse distance weighting*
   Inverse distance weighting can be seen as a special case of kernel approximation in

[1] The delta function can be viewed as the derivative of the Heaviside step function: if

$$H(x) = \begin{cases} a, & x < x_0 \\ b, & x = x_0 \\ c, & x > x_0 \end{cases} \tag{1.1}$$

then

$$\frac{d}{dx}H(x) = \delta(x) \tag{1.2}$$

See http://mathworld.wolfram.com/DeltaFunction.html for more details.

which the kernel function is given by the inverse of the distance between the point where the interpolation is desired and the sample points; this kernel function has infinite band width. The method is fully explained in Chapter 4.

3. *Triangular Irregular Networks*
   The triangular irregular network (TIN) [30] was developed by Peuker et al. in 1978. In a TIN model, the sample points are connected by lines to form triangles; within each triangle, the surface is usually represented by a plane. Most often, the triangles are formed based on a Delaunay's triangulation[1], ensuring that each triangle is empty so it does not contain any of the sampled points. The value of a point inside a triangle is usually estimated by linear or cubic interpolation.

4. *Splines*
   The splines are polynomials with each polynomial of degree p being local rather than global. The polynomials describe pieces of a line or surface (they are fitted to a small number of data points exactly) and are fitted together so that they join smoothly.

Stochastic methods

1. *Models*
   The interpolating function is expressed as a sum of functions, but those functions are not explicitly known nor specified a priori. Information contained in the data, such as spatial correlation, is used to determine the coefficients of the unknown functions.

2. *Spatial structure functions*
   The basic underlying premise is that values at locations that are close together are more similar and values at locations far apart are relatively independent.

3. *Kriging*
   Kriging [18] is a special case of a spatial structure function. It is based on the assumption that the parameter being interpolated can be treated as a regionalized variable. A regionalized variable varies in a continuous manner from one location to the next and therefore points that are near each other have a certain degree of spatial correlation, while points that are widely separated are statistically independent[2].

According to Li's evaluation, the IDW method works well with regularly spaced data, but it is unable to account for clustering. The disadvantage of the TIN method is that the result only depends on three samples; the estimated surface is continuous, but with abrupt changes in gradient at the margins of the triangles. Splines can be quickly calculated and predictions are close to the values being interpolated, but there are no direct estimates of the errors. Kriging methods provide the best linear estimate, but the result depends on the assumptions made, the definition of the model is time-consuming and it requires a large number of samples.

[1]See Annex A for an overview of Delaunay triangulation.

[2]In probability theory, two events (points here) are statistically independent if the occurrence of one event gives no information about the occurrence of the other event.

Li also mentions that all the deterministic methods are local methods in which the value of interpolated point is based on the sampled data nearby, while the stochastic methods are global methods. The TIN model produces a discrete and abrupt surface, while the others produce smooth and gradual surfaces. They all are exact methods generating an estimate with the same value as the observed value at a sample point or they can be forced to be exact.

In conclusion, spatial data interpolation can be performed in various ways by invoking different assumptions and models. The most suitable method for a specific problem depends on the data and on the desirable properties the interpolation result should have.

1.5 Designing a module for automatic conversion of scanned sea charts into 3D models

Even though some steps were taken in this direction, there is still no fully automatic algorithm for map interpretation or conversion into a 3D model. Designing and developing such a system is a complicated work, since there are many considerations to be made.

First of all, scanned sea charts are large complicated images, which display a lot of information and symbols that sometimes may touch or overlap. Thus, a good separation and identification of the symbols was required. Secondly, the conversion of a 2D map into a 3D model requires an interpolation of the spot soundings. The interpolation method has to fulfill some constraints, since the result has to be similar to a real terrain.

The first step was to divide the problem in two separate parts: *automatic sounding recognition* and *automatic interpolation of the extracted soundings*. The first part was further divided into:

1. *Segmentation* - the goal of this step is to remove all background pixels (belonging to land areas, water or submerged areas) and keep only pixels that belong to symbols;
2. *Labeling* - different symbols are separated by searching for connected components through the foreground pixels;
3. *Classification* - all the individual objects are given a class label based on the similarity with a certain type of symbol (characters, lines, curves etc.)
4. *Recognition* - all the objects belonging to the 'characters' class are passed to an OCR software to recognize the true soundings.

For the second part, several interpolation methods were considered and reviewed; based on the obtained results and certain constraints, an Inverse Distance Weighting interpolation method based on Delauney triangulation was chosen[1].

[1]Refer to Chapter 5 for a detailed explanation.

Part II

Automatic recognition of spot soundings

Chapter 2

Object identification

Scanned sea chart images represent the input data of this module. The images are color images which usually exceed sizes of (5000, 5000) pixels. Analyzing a large image like this is a tedious work which takes a lot of processing time. A simplification of the problem is to only consider those parts of the image that are of main interest (pixels that belong to a symbol) and to ignore pixels that belong to the background (land, water, submerged area). The separation of image pixels into foreground and background pixels is achieved through a binary segmentation method. The following chapters will present the step-by-step process of the automatic recognition of spot soundings in scanned sea charts, from the binary segmentation method, object labeling and classification to soundings recognition.

2.1 Binary segmentation

In a sea chart image, background areas may have different colors, depending on their type: blue for water, white for submerged areas, gray or yellow for land. However, the shades of these colors are different from chart to chart and this fact makes an automatic segmentation method in RGB-color space impossible.

Figure 2.2a shows an example of a sea chart color image and Figure 2.2b displays the corresponding gray-scale image. Looking at the gray-scale image, it is visible that background areas have light shades of gray, while symbols are represented by dark shades of gray. This observation can be used to develop an automatic binary segmentation: all pixels with light shades of gray can be labeled as background and all pixels with dark shades as foreground. The problem is how to decide which gray levels are considered light and which dark.

Partitioning all pixels in the image in two clusters using the K-Means algorithm gives the answer to the problem: the resulted cluster centers will give a threshold for light and dark shades of gray and all pixels will be labeled as background or foreground depending on their distance to the cluster centers.

2.1.1 K-Means algorithm

The K-Means clustering algorithm is used to partition n data points or observations into k disjoint clusters, c_j, in which each observation belongs to the cluster with the nearest mean. The partitioning is done so as to minimize the sum-of-squares optimization function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - \mu_j \right\|^2 \tag{2.1}$$

where $\left\| x_i^j - \mu_j \right\|^2$ is a distance measure between the data point x_i^j and the cluster center μ_j.

The algorithm is composed of the following steps:

1. Initialize the cluster centers, either randomly or by some heuristics:$\mu_1^0, \mu_2^0, ..., \mu_k^0$.
2. Assign each data point to the cluster that has the closest centroid:

$$c_j^t : \arg\min_i \left\| x_j - \mu_i^t \right\|^2 \tag{2.2}$$

3. Update the value of the k centroids:

$$u_j^{t+1} = \frac{\sum_{x_i \in c_j} x_i}{|c_j|} \tag{2.3}$$

4. Repeat step 2 and 3 until the assignments no longer change.

Although it is proven that the algorithm will always terminate, K-Means does not necessarily find the global minimum of the objective function [6]. The algorithm is also highly sensitive on the initial selected cluster centers. To reduce this effect, the algorithm can be run multiple times.

Figure 2.1[1] presents an illustration of the K-Means algorithm applied for a set of 20 2-dimensional data points with $k = 3$ and Euclidean distance as a distance metric. Figure 2.1a displays the data points and the initial cluster centers. After the first assignment step, data points are divided in three clusters as in figure 2.1b; the updated centroids are displayed in figure 2.1c. Figures 2.1d and 2.1e present the next iteration: the points are reassigned to the new clusters, the centroids are updated and the algorithm converges. The final solution is given in figure 2.1f.

2.1.2 Binary segmentation algorithm

The steps of the binary segmentation algorithm, illustrated in Figure 2.2, are:

1. Compute the gray-scale image (figure 2.2b) from the original image (figure 2.2a);
2. Compute the gray-scale image histogram, by counting the number of pixels for each gray level. Figure 2.2c displays the histogram of the image in Figure 2.2b.

[1]The images were created using [5].

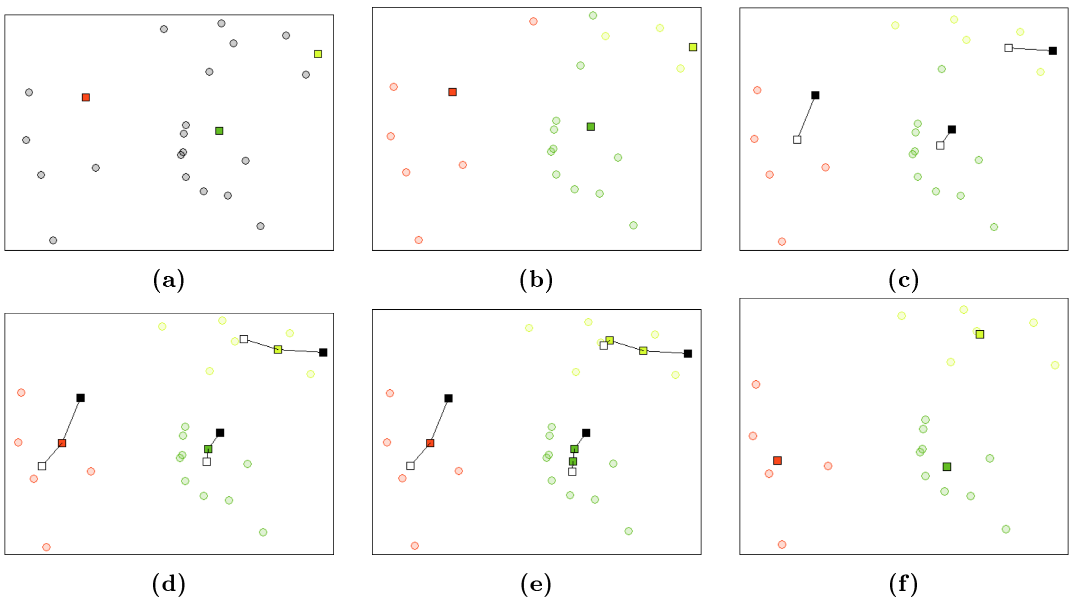


Figure 2.1. K-Means algorithm:a) Data points and initial centers, b) First iteration: assignment step, c) First iteration: update step, d) Second iteration: assignment step, e) Second iteration: update step, f) Final solution.

3. Run the K-Means algorithm to partition the pixels in 2 clusters, thus $k = 2$. The K-Means algorithm doesn't return the global optimum of the problem, but a local optimum, and the result is dependent on the choice of the initial centers. Because it is desired to cluster the pixels into light (close to white) and dark (close to black) shades, the centers are initiated with these extreme values: $\mu_1 = 0$, $\mu_2 = 255$. After applying K-Means clustering for the histogram in figure 2.2c, the resulted centers are: $\mu_1 = 70, \mu_2 = 209$.

4. Label all the pixels w.r.t their distance to the cluster centers, as in the pseudocode in Listing 2.1. Figure 2.2d shows the binary segmentation of the image in Figure 2.2a.

```
// G - grayscale image
// $\mu_1, \mu_2$ - final cluster centers
// B - final binary image
forall pixels (i, j) in G
        if abs(G[i, j] - $\mu_1$) < abs(G[i, j] - $\mu_2$)
                B[i, j] = 0;
        else
                B[i, j] = 255;
```

Listing 2.1. Label pixels as background or foreground

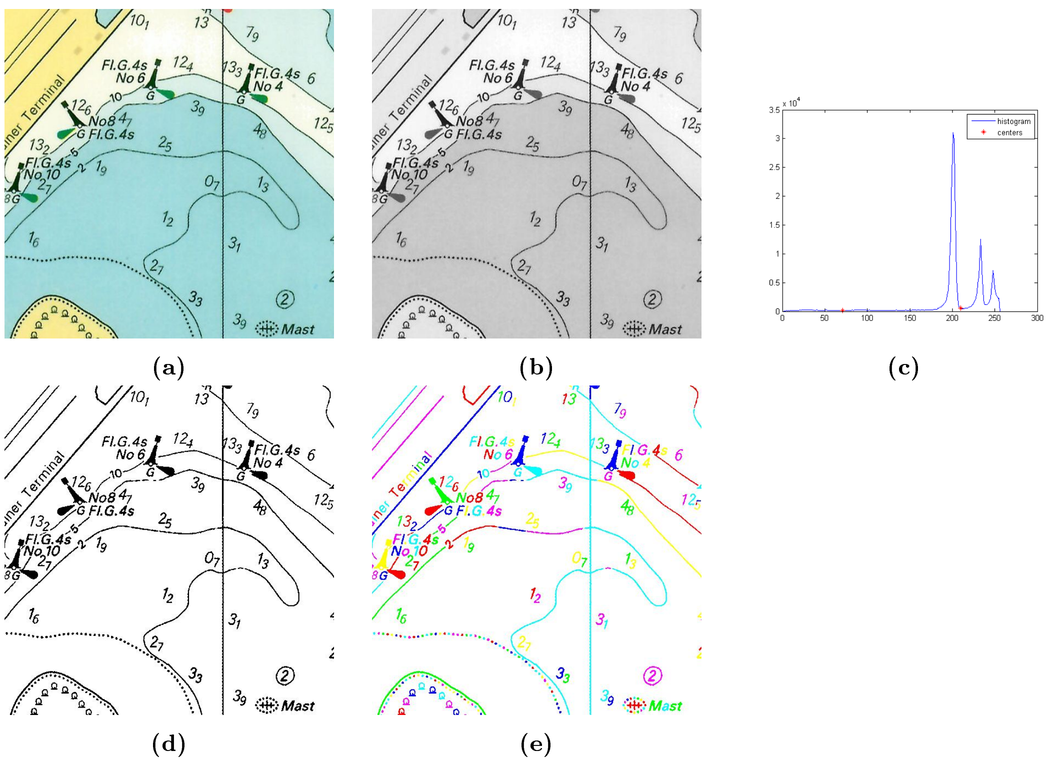


Figure 2.2. Object labeling steps: a) Original color image; b) Grayscale image; c) Corresponding grayscale histogram and centers found with K-Means; d) Binary segmented image; e) Individual objects (connected componets) in the image.

2.1.3 Evaluation

The performance of the binary segmentation method is influenced by the intensity of every pixel, since it is a graylevel-based segmentation. The threshold values delimiting the background and the foreground pixels are very important; they may give a good segmentation (according to the human visual perception) or they may cause different objects to be treated as background losing any information about them. The resolution of the image also plays an important role. Despite the fact that sea charts have a high resolution and the objects in the image seem to be represented by constant pixel values, they sometimes have small variations in their intensity values. That is why one object may end up being segmented in several parts.

Figure 2.3 shows several results. Image 2.3a contains four different objects (two 'one' digits and two lines) and the segmentation result in fig. 2.3c is accordingly. The other two images show the two type of problems that may be encountered:

- *Undersegmentation*
  Figure 2.3d is a case of undersegmentation. As can be seen in fig. 2.3e, the gray values of the horizontal lines are more similar to the gray values of the background and therefore are treated as background. Another problem here is that different objects may end up being segmented as a single object, depending on the gray values of the pixels in between them. Even though humans can distinguish eight different objects in the image (two digits, one curved line and five horizontal lines),

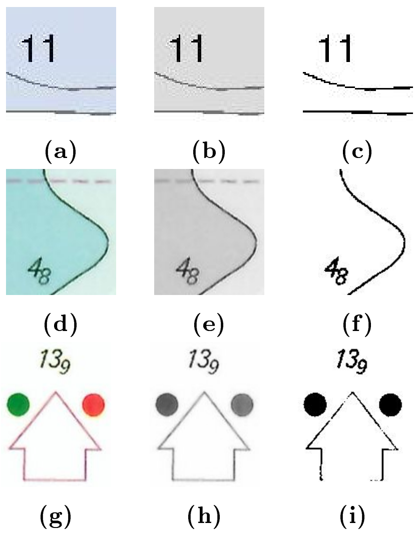


Figure 2.3. Segmentation results: a, d, g) Original images; b, e, h) Graylevel images; c, f, i) Corresponding binary segmentations.

the method only finds two (the curved line and the two digits glued together and treated as a single object).

- *Oversegmentation*
  Figure 2.3g is a case of oversegmentation: the gray values of the house-shaped object vary and some of the pixels are treated as background. This way, gaps are introduced in the object, making it to be segmented in several parts. Even though only six different objects are distinguishable in the scene (three digits, two discs and a house-shaped object), the method finds 16 different objects, by segmenting the house in 11 small parts.

For the purpose of the project, some of the segmentation problems can be ignored. The omission of certain objects that are not soundings and that are treated as background does not affect the overall performance, because only soundings are of interest. Also, the oversegmentation of objects other than soundings can be ignored. However, spot soundings can sometimes be affected by these problems: when the scanned sea chart is an old chart with blurred or faded colors digits may be lost during the segmentation process, resulting in an undersegmentation, or they may be oversegmented if gaps are formed. Another problem is when digits are glued together, causing them to lose their individual shape properties; in the classification step they will not be recognized as digits or characters.

2.2 Object labeling

The result of a binary segmentation is a binary image defined by the fact that all pixels are either black or white (0 and 1, 0 and 255). Usually, black pixels represent the foreground and white pixels the background. The goal of labeling binary images is to identify and separate individual objects in the scene. Thus, all pixels belonging to a certain object are marked with a unique label, different from the labels of other objects in the image. An individual object is a distinct connected component. The label may be a number or a string of characters.

Object labeling was performed using a method called *region tracking with correspondence tabels* [2]. This is a simple algorithm that consists in iterating through the image (from left to right and from top to bottom) and assigning a label to an object pixel w.r.t the labels of its predecessors. The predecessors in a V_8 neighborhood are shown in Figure 2.4. When the predecessors have different labels, the minimum label is assigned to the current pixel, but since all pixels actually belong to the same object, the correspondence between labels is written in a correspondence table. This way, the number of iterations through the image is limited to one.

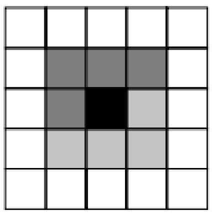


Figure 2.4. V_8 neighborhood (dark and light gray) and the predecessors of the curent pixel (dark gray).

Let T be the correspondence table. When the algorithm starts, each entrance in the table is initialized with its value $(T(j) = j)$. The table is updated in two situations:

- if the current pixel has no labeled predecessors it has to receive a new label which is pushed back in the table $(T(j_{max} = j_{max})$;
- if the current pixel has labeled predecessors it receives the minimum label and the correspondence between the other labels and the minimum one is written in T. If the current pixel has as predecessors P_1, P_2, P_3, P_4, then let $m = \min(P_i)$. The pseudocode for updating the correspondence table is given in Listing 2.2.

```
forall $P_i$
        while $T(P_i) \neq m$ do
                tmp = $T(P_i)$
                $T(P_i)$ = m
                $P_i$ = tmp
```

Listing 2.2. Updating the correspondency table

After the first iteration, T is updated such that each label corresponds to the final label of the object. The final labels are not necessarily in order. The algorithm ends by replacing each label j in the image with $T(j)$.

Figure 2.2e shows the result of this algorithm applied on the binary image in figure 2.2d.

2.2.1 Evaluation

The object labeling method correctly identifies all the connected components in the input binary image. As can be seen in figure 2.5, each connected component is identified as a different object given in a different color. In figure 2.5g, the digit *9* and the buoy are glued together in the segmentation method and now they are identified as a single connected component.

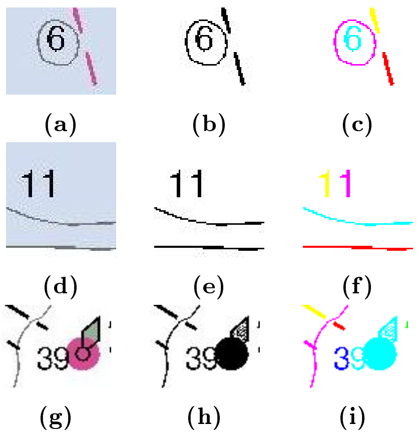


Figure 2.5. Object labeling: a, d, g) Original images; b, e, h) Binary segmentation; c, f, i) Individual objects given in different colors.

Chapter 3

Object classification

After applying the labeling algorithm, all the individual objects in the chart image are obtained. They further need to be classified into several classes according to the symbol they are most similar with. The classes are: *horizontal straight line*, *vertical straight line*, *oblique straight line*, *curved line*, *character* and *other symbols*. Some examples of symbols are given in Figure 3.1.

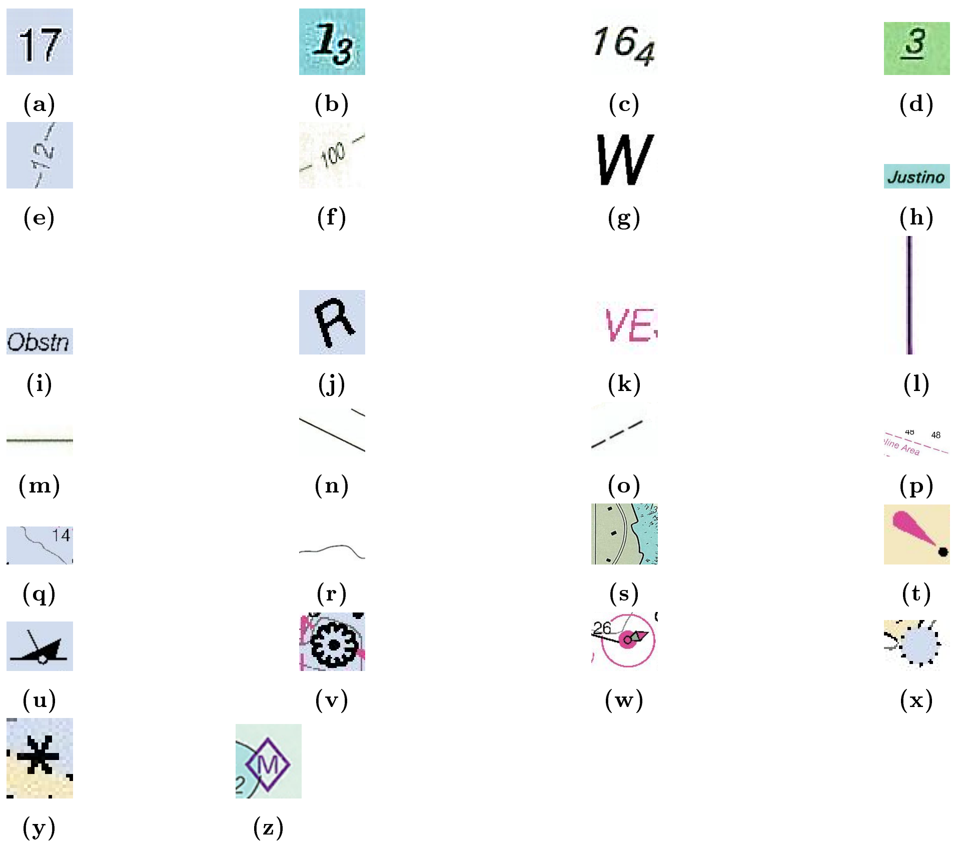


Figure 3.1. Symbols in nautical charts: a) - d) soundings, e) - f) depth of contour lines, g) - k) letters, l) vertical line, m) horizontal line, n) - p) oblique lines, q) - s) curved lines, t) - z) some examples of other symbols.

3.1 Features

After eliminating the noise resulted after the segmentation and object labeling steps based on a minimal surface criteria, geometrical features are calculated for the remaining objects. Some of the geometrical features are illustrated in Figure 3.2.

- *Area*
  The area of an object is equal to the total number of pixels that belong to that object.
- *Center of gravity*
  The center of gravity of an object is equal to the arithmetic sum of all object points.
- *Axis-aligned minimum bounding box*
  The axis-aligned minimum bounding box of an object is the tightest rectangle which includes the object. The rectangle is parallel to the x and y axes and is described by four numbers: the smallest x and y coordinates of the object points denoting the top-left corner of the rectangle, and the greatest x and y coordinates denoting the bottom-right corner.
- *Orientation*
  The orientation of an object is the imaginary rotation that is needed to move the object from a reference placement to the current placement. It is calculated with an iterative search: the object is rotated successively with angle values ranging from 0 to 2π and the width of the rotated object is calculated. The orientation is chosen to correspond to the minimal width of the object.
- *Arbitrarily oriented minimum bounding box*
  The arbitrarily oriented minimum bounding box is the minimum bounding box calculated subject to no constraints as to the orientation of the result.
- *Density*
  The density of an object is the ratio of the area of the arbitrarily oriented minimum bounding box and the area of the object: $\rho = A_{bb}/A_{obj}$.
- *Vertical and horizontal projection histograms*
  The vertical and horizontal projection histograms represent the projection of the object on the x ans y axes.

All these features are later used in the classification process.

3.2 Decision rules

As mentioned in Chapter 1, symbols of the same type have similar features and these features are different from features of symbols of some other type. If a symbol is properly described, the respective features could be used to build decision rules, which represent the basis of the classification process.

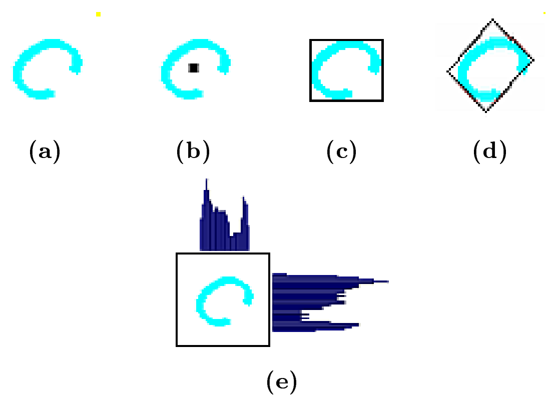


Figure 3.2. Geometrical features: a) Symbol, b) Center of gravity, c) Axis-aligned bounding box, d) Arbitrarily oriented bounding box, e) Horizontal and vertical projection histograms

The decision rules mainly focus on describing and classifying different types of lines and characters in a sea chart. The reason is the goal of the project: to recognize spot soundings and contour lines in a scanned sea chart.

3.2.1 Vertical straight lines (VSL)

The main and simplest features of a vertical straight line is a small width of its bounding box and an orientation angle of 0 degrees. To eliminate false positive results, a constraint on the height of the line is required: the height has to be big enough such that dots or other similar objects are not labeled as a vertical straight line. The examples in Figure 3.3 have a width equal to 1, respective 2, and both have an orientation angle of 0 degrees.

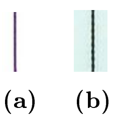


Figure 3.3. Vertical straight lines: a) width $= 1$, $\theta = 0$; b) width $= 2$, $\theta = 0$.

Therefore, the decision rule to identify vertical straight lines is: $(width < thresh_1)$ && $(height > thresh_2)$ && $(\theta = 0)$, which is also listed in the pseudocode in Listing 3.1.

```
if (obj.width < thresh1 && obj.height > thresh2 && theta == 0)
        return true;
return false;
```

Listing 3.1. VSL decision rule

3.2.2 Horizontal straight lines (HSL)

If vertical straight lines are characterized by a small width and an orientation angle of 0 degrees, horizontal straight lines are characterized by a small height of their bounding box and an orientation angle around 90 degrees. The lines in Figure 3.4 both have a width of 2 and an orientation angle equal to 90 degrees.

Figure 3.4. Horizontal straight lines: a) width = 2, $\theta = 90$; b) width = 2, $\theta = 90$.

A constraint on the width of the line is required: the line has to be wide enough such that dots are not labeled as horizontal straight line. Thus, the decision rule to identify horizontal straight lines is: $(height < thresh_1)$ && $(width > thresh_2)$ && $(\theta = 90)$, which is also listed in Listing 3.2.

```
if (obj.height < thresh1 && obj.width > thresh2 && theta == 90)
        return true;
return false;
```

Listing 3.2. HSL decision rule

3.2.3 Oblique straight lines (OSL)

The easiest way to identify oblique straight lines would be to calculate their arbitrarily oriented bounding box and apply the same decision rules as for vertical or horizontal straight lines. Since the calculation of the arbitrarily oriented bounding box depends on the line' s orientation angle which is calculated with an error of 1 degree, the result could be compromised. Therefore, another decision rule is required.

The most natural way to check if an object is a line is to find the endpoints of the object and compute the distance from all other points in the object to that line. If the distance is small enough, then it can be said that the points approximate a line.

Because we are dealing with straight lines, the easiest way to find the endpoints is to go through all the points of the object and keep the two points with minimum and maximum y coordinates, p_1 and p_2. Then, the distance from a point p to the line formed by p_1 and p_2 can be calculated using Eq. 3.1.

$$d = \frac{|(x_{p_2} - x_{p_1}) * (y_{p_1} - y_p) - (x_{p_1} - x_p) * (y_{p_2} - y_{p_1})|}{\sqrt{(x_{p_2} - x_{p_1})^2 + (y_{p_2} - y_{p_1})^2}} \tag{3.1}$$

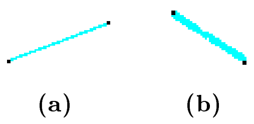


Figure 3.5. Oblique straight lines and the endpoints found: a) max_dist(p, line) = 1.51 b) max_-dist(p, line) = 2.69.

Figure 3.5 shows some examples of oblique straight lines cropped from sea charts, the endpoints found with the previous mentioned algorithm and the maximal distance from the other object points to the line. Of course, the maximum distance increases with the width of the line, but also with the error in approximating the endpoints of the line.

Thus, the condition for an object to be an oblique straight line is that the maximum distance from all object points to the line approximation to be smaller than some threshold. A pseudocode of the decision rule is given in Listing 3.3.

```
forall points $P_i$ in obj
        if (dist($P_i$, line) > thresh)
                return false;
return true;
```

Listing 3.3. OSL decision rule

3.2.4 Curved lines (CL)

Building a decision rule to identify curved lines in a sea chart image is a more delicate problem. A plane curve is the locus of points of coordinates x, y, such that $f(x, y) = 0$, where f is a polynomial in two variables defined over some domain, F, given here by the width and height of the image. The process of finding the function f that best fits the set of data points and possibly subject to some constraints is known as curve fitting. Polynomials, conic sections, trigonometric functions are generally used for curve fitting. However, in a sea chart image curved lines usually represent contour lines and there is no prior knowledge on what type of curves they could be. Thus, it is very hard to decide what type of curve fitting algorithm would be appropriate.

Another way to identify curved lines in sea chart images is based on properties of the shape and geometry of curves. For example, curves have bounding boxes with rather large area compared to their own area. The reason is due to the fact that curves are usually made of thin segments, but they are either long or wide. A consequence is visible in their projective histograms, on x or y axes: the maximum value of the x or y projection histogram is a relative small number.

Therefore, the properties of curved lines used in classification are:

- *The density of background pixels* inside the bounding box has to be greater than a threshold; the curves in Figure 3.6 have a density equal to 12.8, respective 10.55, which shows a relative high number of background pixels inside their bounding box compared to the number of foreground pixels.
- *The maximum values of the horizontal or vertical projection histograms* have to be smaller than a threshold. Because pixels are equally divided along the curved line (the curve has the same width along its path), in an ideal case[1] the values of each bin of the histograms are equal. This value is given by the ratio of the area and

[1] An ideal case is a curve shaped like a circle, without any loops or waves.

the width (or height) of the curve multiplied by a constant value[1] depending on how many waves or loops the curve has: $H_{max} = ct * \left\lceil \frac{area}{width} \right\rceil$, $V_{max} = ct * \left\lceil \frac{area}{height} \right\rceil$. In Figure 3.6a, the curve has a horizontal orientation and a threshold equal to 8, while the maximum value of the horizontal projection histogram is 3; the curve in Figure 3.6b has a vertical orientation, a threshold equal to 12 and a maximum value of the vertical projection histogram of 3. In both cases, the condition holds.

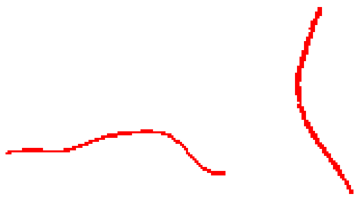


Figure 3.6. Curved lines: a) $\rho = 12.8$, $H_{max} = 8$, $h_{max} = 3$; b) $\rho = 10.55$, $V_{max} = 12$, $v_{max} = 3$.

Therefore, the decision rule for identifying a curved line in a sea chart is based on these two properties of the shape and a pseudocode is given in Listing 3.4.

```
if obj.density > $thresh$
        h_curve = false; v_curve = false;
        max = ct * ceil (area / width);
        forall bins i in $h_{hist}$
                if $h_{hist}$[i] > max
                        h_curve = false;
                        break;
        max = ct * ceil (area / height);
        forall bins i in $v_{hist}$
                if $v_{hist}$[i] > max
                        v_curve = false;
                        break;
        if h_curve || v_curve
                return true;
return false;
```

Listing 3.4. CL decision rule

3.2.5 Characters

Assuming all the vertical, horizontal, oblique straight lines and curved lines were correctly identified, the only remaining objects are characters (letters and digits) and other symbols with various shapes and sizes. Therefore, a decision rule to separate the characters from all these objects is needed.

[1] The constant value was chosen empirically and set to $ct = 4$.

Again, this is a delicate problem since there is no general way to describe the shape of all letters and digits. Furthermore, in different charts, characters have different font faces and sizes, which makes the problem even more difficult. The solution is to find similar features of all characters in maps and combine them to build decision rules that will separate characters from the other remaining symbols. The main concern is to classify spot soundings, thus the decision rules will be based on general features of digits by analyzing font types ans sizes.

The shape properties considered for this classification are mainly geometrical features of the object:

- *Area* - even considering all font faces and sizes used in charts, the area of a digit falls within some thresholds.
- *Orientation* - digits that represent spot soundings have a vertical orientation or they are slightly inclined, depending on the font face.
- *Density* - digits and characters in general can be seen as curved lines, thus they also have a large density (ratio between the amount of background and foreground pixels inside their bounding box).
- *Bounding box* - depending on the font face and size and, of course, on the digit, the size of the bounding box differs, but it is bounded by some thresholds.
- *Height-width ratio* - digits have a rectangular bounding box, with height greater than their width.

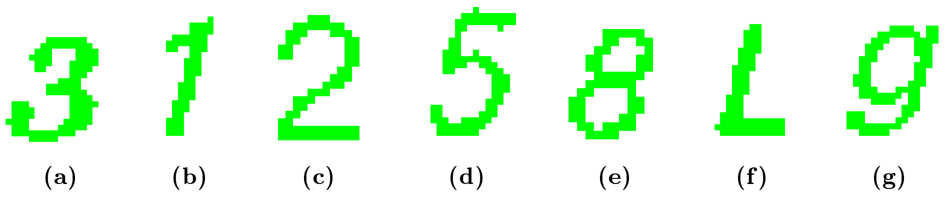


Figure 3.7. Characters: a) $area = 126$, $\theta = 0.26$, $\rho = 1.62$, $bb = (12, 17)$, $hw_{ratio} = 1.41$; b) $area = 66$, $\theta = 0.22$, $\rho = 1.43$, $bb = (5, 19)$, $hw_{ratio} = 3.8$; c) $area = 72$, $\theta = 0$, $\rho = 2.22$, $bb = (10, 16)$, $hw_{ratio} = 1.6$; d) $area = 98$, $\theta = 0.12$, $\rho = 2.35$, $bb = (11, 21)$, $hw_{ratio} = 1.9$; e) $area = 57$, $\theta = 0.34$, $\rho = 1.35$, $bb = (7, 11)$, $hw_{ratio} = 1.57$; f) $area = 84$, $\theta = 0.29$, $\rho = 1.92$, $bb = (9, 18)$, $hw_{ratio} = 2$; g) $area = 107$, $\theta = 0.2$, $\rho = 1.77$, $bb = (10, 19)$, $hw_{ratio} = 1.9$.

In Figure 3.7 there are some examples of digits and letters with different font sizes and faces; the corresponding feature values are listed. The feature values vary inside a certain range, therefore, using an empirical approach, the corresponding threshold values can be set. The decision rule for identifying a character in a sea chart is listed in Listing 3.5.

```
if      (InRange(obj.area) &&
        InRange(obj.theta) &&
        InRange(obj.density) &&
        InRange(obj.bb) &&
        InRange(obj.hw_ratio))
                return true;
return false;
```

Listing 3.5. Character decision rule

Combining all these features, a lot of symbols that don't represent characters are eliminated. The success of this decision rule highly depends on the set thresholds. The more strict the threshold is, the better the result. However, since more than one font is considered, the range of values increases and this may result in classifying other symbols as characters.

3.2.6 Other symbols

Since there is no interest in classifying other types of symbols, like buoys, lighthouses and so on, they will all be regarded as belonging to the same class. If all previous symbols were correctly classified, the remaining unclassified symbols all belong to the *other* class. The only mention is that, in order to distinguish between symbols on the chart and background pixels, the area of a symbol doesn't have to exceed a certain threshold. Therefore, the decision rule is only based on the area feature, as the pseudocode in Listing 3.6 shows.

```
if (obj.area < thresh)
        return true;
return false;
```

Listing 3.6. Other symbols - decision rule

3.2.7 Classification algorithm

After all individual objects have been identified in the sea chart image, the classification process starts by first testing for simple objects and continuing with more complicated ones. The order of testing is important; for example, if an object is first tested for the *other* class, all objects will end up being classified as *other* symbols. The pseudocode for classification is given in Listing 3.7.

```
if obj.isHSL()
        obj.type = HSL;
        break;
if obj.isVSL()
        obj.type = VSL;
        break;
if obj.isOSL()
        obj.type = OSL;
        break;
if obj.isCurvedLine()
```

```
        obj.type = CL;
        break;
if obj.isCharacter()
        obj.type = Character;
        break;
if obj.isOther()
        obj.type = Other;
else
        obj.type = BGR;
```

Listing 3.7. Classification - pseudocode

3.3 Evaluation

To evaluate the performance of the classification process, a set of labeled testing images was built to represent the ground truth. The testing images are small images cropped from the scanned sea charts and they usually contain a single type of objects. Figures 3.8 and 3.9 display some examples of testing images for all object types. The results and the corresponding threshold values are given in Table 3.1.

No.	Class	# test images	# correctly classified	Parameters
1	VSL	20	18	$thresh_1 = 2$ $thresh_2 = 4$
2	HSL	20	18	$thresh_1 = 2$ $thresh_2 = 4$
3	OSL	22	18	$thresh = 4$
4	CL	27	27	$ct = 4$
5	Soundings Other characters	32 23	30 13	$area \in (10, 160)$ $\theta \in (0, 0.8)$ $\rho \in (1.25, 2.5)$ $bb \in ((3, 10), (20, 25))$ hw-ratio > 1.1
6	Other	38	32	$thresh = 1000$

Table 3.1. Classification results

Figure 3.8 shows some classification results. For each type of object, one correct classification and some misclassification examples are given. Each type of object is represented by an unique color: VSL - magenta, HSL - yellow, OSL - cyan, CL - red, CH - green, Other - blue.

The performance of the classification method depends on the threshold values, which were empirically determined and have the same values for all charts. Because of the differences in fonts and styles from chart to chart, the threshold values have to be flexible enough to be able to correctly classify as many objects as possible and, in the same time, to minimize the classification errors over all sea chart images. The most important issue is to correctly classify the spot soundings.

For testing the decision rules for both *vertical and horizontal straight lines*, 20 ground truth images were used for each class. In both cases, there were two classification errors: the lines were classified once as oblique straight lines and once as other objects. They were misclassified as oblique straight lines because they are slightly inclined and the width of their bounding box exceeded the threshold value. This may not be a misclassification, but an error in human visual perception. Figure 3.8e shows a vertical line misclassified as other object; the reason is that, after segmentation, some parts of the texture in the background were touching the vertical line. Thus, they became one single object, with a different shape geometry. The same fact applies to the horizontal line in figure 3.8k: the segmentation changed the geometry of the line, resulting in a misclassification.

To test the classification of the *oblique straight lines*, 22 ground truth images were used resulting in 4 classification errors. The main reason for the classification errors is that the threshold values are very strict, because they are also meant to prevent classifying the digit *1* as an oblique straight line. As mentioned before, the correct classification of the spot soundings is the major priority. In all cases, the oblique lines were misclassified as curved lines.

The classification of *curved lines* was tested using 27 images containing curved lines and several other types of objects. The decision rules for this type of objects gave the best performance, since all curved were correctly classified. Some results are given in figures 3.8t, 3.8v, 3.8x.

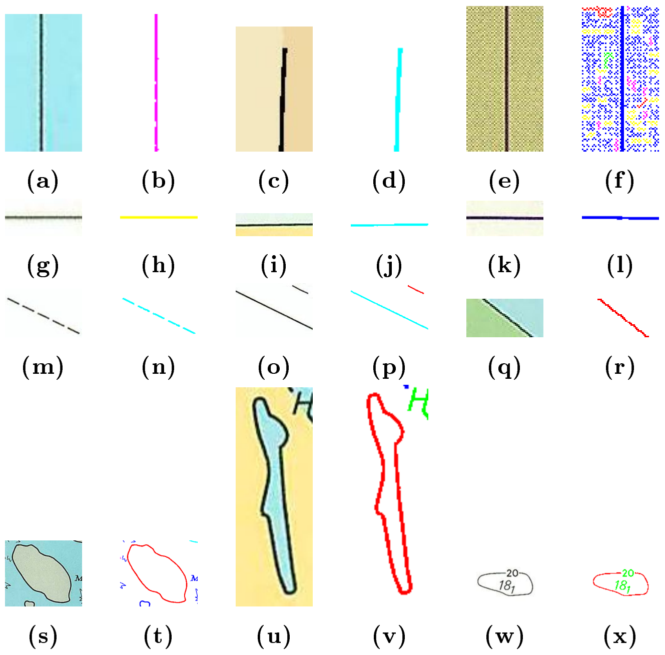


Figure 3.8. Classification results - VSL: a, c, e) Original images; b, d, f) Classification; HSL: g, i, k) Original images; h, j, l) Classification; OSL: m, o, q) Original images; n, p, r) Classification; CL: s, u, w) Original Images; t, v, x) Classification.

The performance of the decision rules for the *character* class was tested in two stages: first for spot soundings and then for other types of characters (letters, digits in different fonts). The reason is that the decision rules were build specifically to identify soundings in a sea chart image and they were based on the geometry of spot soundings (size, orientation etc.). Therefore, other types of characters may be misclassified.

32 images were used to test the performance of sounding classification resulting in 2 classification errors. The errors are due to the fact that in the sea chart image the digit representing the sounding was touching another symbol (in figure 3.9d, digits 0 and 7 are touching; the same happens for digits 4 and 1 in figure 3.9f); thus, they were interpreted as a single object with a different geometry.

When classifying other types of characters, the performance is not that good: out of 23 testing images, 10 were misclassified. The errors are due to the differences in fonts and styles along sea charts: in figures 3.9j and 3.9l the digits and letters have a bigger font size than the font size of the soundings and they are interpreted as *other objects*. Another error is the interpretation of letters like *l* or *i* as oblique straight lines. However, these errors do not affect the overall performance of the system, because the main interest is to recognize spot soundings.

The classification of symbols belonging to the *other objects* class was tested using 38 images containing different symbols that may be encountered in a sea chart. Out of all the symbols, 6 were wrongly classified. Some symbols have a shape similar to a curve and thus can be easily interpreted as curved lines, like in figure 3.9r. Other symbols are very similar to some characters (for example, the symbol in figure 3.9p is similar to a mirrored *3*, an *E* or an *m*), making them difficult to be correctly labeled based only on shape geometry. Because this type of error may affect the overall performance of the system, using a good Optical Character Recognition software is mandatory in order to eliminate symbols that were misclassified as characters.

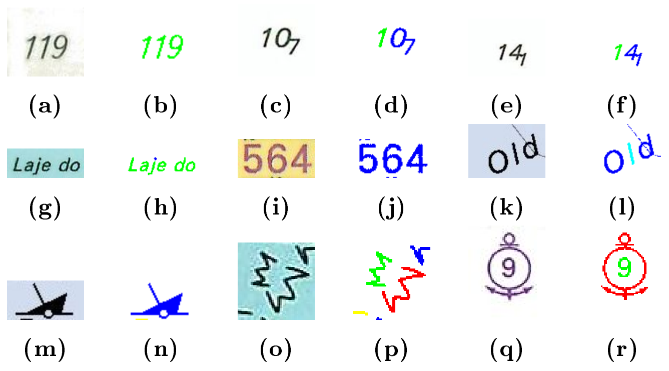


Figure 3.9. Classification results - CL: a, c, e) Original images; b, d, f) Classification; Soundings: g, i, k) Original images; h, j, l) Classification; Other characters: m, o, r) Original images; n, p, r) Classification; Other objects: s, u, w) Original images; t, v, x) Classification.

3.4 Object grouping

Individual objects that have the same class label and are close together usually form a group on the sea chart. This may happen for dotted lines, dots, characters. This part is very important for the proper recognition and interpretation of soundings that are represented by more than one digit or decimals.

The two most important grouping criteria are the type of the objects and the Euclidean distance between them. However, for spot soundings, the y-coordinate of the digit in the image is also important. The latter will prevent characters placed like in Figure 3.10g to be grouped together. The characters in the figure should form two groups: (6, s) and $(2, 1)$, but if the y-coordinate is not considered, they will form one single group: (6, s, $2, 1$), because they are close enough.

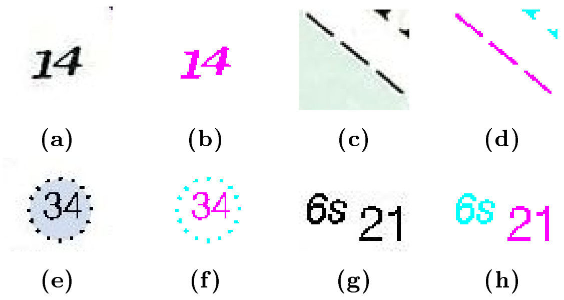


Figure 3.10. Object and corresponding groups; each group is given in different colors.

The pseudocode for object grouping is given in Listing 3.8. The method returns true if obj_1 and obj_2 should be grouped together.

```
if $obj_1$.type != $obj_2$.type
        return false;
if $obj_1$.type == Character &&
   abs($obj_1$.y - $obj_2$.y) > $th_1$
        return false;
if dist($obj_1$, $obj_2$) > $th_2$
        return false;
return true;
```

Listing 3.8. Object grouping

Another important issue is to set the sign of the spot sounding by checking whether there is a close horizontal line underneath it. If there is one, then the sounding is above sea level and its value will be $+|value|$, otherwise the value will be $-|value|$.

Chapter 4

Spot soundings recognition

The main purpose of this module is to recognize spot soundings in sea chart images. Having all the symbols in the image classified, the only remaining thing is to select all digits in the *characters* class and to identify their value. For this purpose an optical-character-recognition (OCR) engine is used: the *Tesseract OCR engine* [31].

4.1 Tesseract OCR engine

Tesseract is an OCR engine that was developed at HP Labs between 1985-1995. It was released as an open-source engine in 2005 and from 2006 its development has been sponsored by Google. Tesseract has no page layout analysis feature and it assumes as input a binary image with optional polygonal text regions defined. An overview of its architecture is given in Figure 4.1 [11].

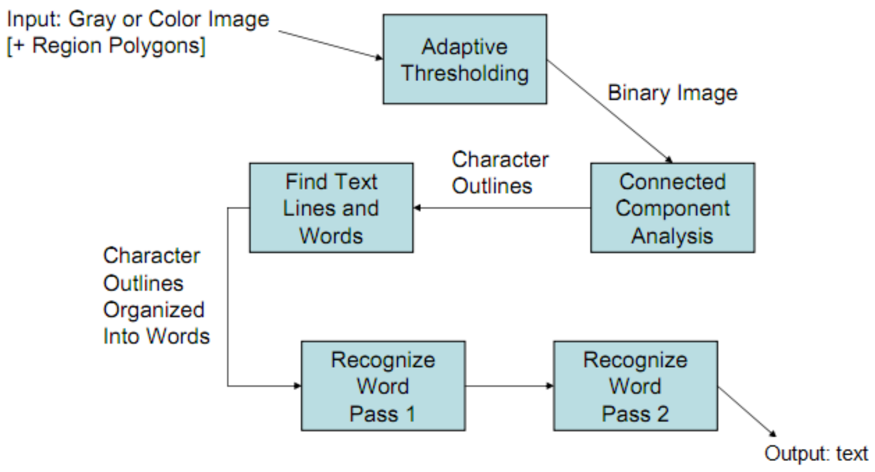


Figure 4.1. Tesseract OCR architecture

The first step in the processing pipeline is the *connected-component analysis*: the outlines of the components are stored and gathered together by nesting in structures called *blobs*. During the second step, blobs are organized into text lines. The text lines and regions are analyzed for fixed pitch and proportional text spacing, because for example italics, digits and punctuation create special-case font-dependent spacing. The text lines are broken into words according to the character spacing by measuring gaps in a limited vertical range between the baseline and the meanline, shown in Figure 4.2 [31] . The baselines are fitted using a quadratic spline model, which makes it possible to handle pages with curved baselines, affected by skew and curl.

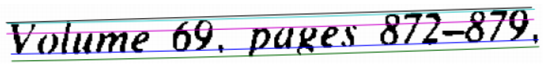


Figure 4.2. A line of text with a fitted baseline, descender line, meanline and ascender line; all these lines are parallel (with a constant y separation) and slightly curved.

To achieve a better performance, words are recognized in two steps. The first step attempts to recognize each word in turn. Two types of classifiers are being used: a static classifier and an adaptive one. The adaptive classifier is a more font-sensitive classifier and it is trained by the output of the static classifier. To improve the performance, each word is also passed as a training data; this way, the classifier gets a chance to more accurately recognize text lower in the page.

The static classifier uses two types of features, illustrated in Figure 4.3 [11]. During the training process, the features are represented by the segments of a polygonal approximation of the characters (fig. 4.3d), while for the recognition step features of a small, fixed length (in normalized units) are extracted from the outline (fig 4.3e) and matched many-to-one against the clustered prototype features in the training data (fig 4.3f and fig. 4.3g).

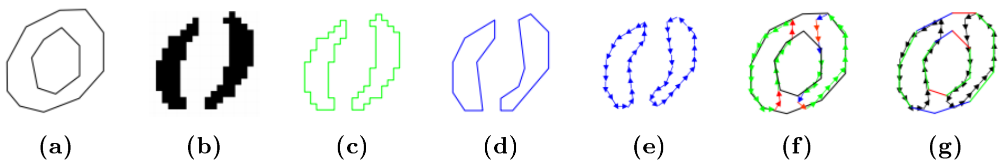
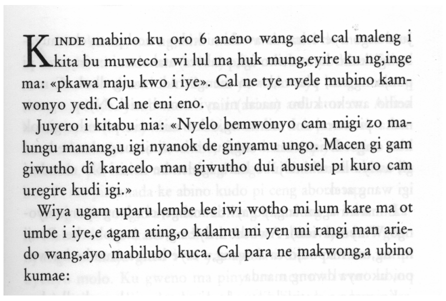


Figure 4.3. Recognition step: a) Prototype, b) Original image, c) Outlines of components, d) Polygonal approximation, e) Extracted features, f) Match of prototype to features, g) Match of features to prototype.

During the second step of the recognition process, words that were not recognized well enough are recognized again. The second recognition may improve, because the classifier may have learned something useful too late to make a contribution near the top of the page.

Tesseract may be run in several modes, depending on the type of characters it has to recognize. It normally recognizes all types of characters (letters, digits, punctuation), but it may be set to recognize only digits. However, the recognition performance depends on the font type it was trained for. Figure 4.4 displays some results. For more information on running Tesseract OCR engine, check [11].

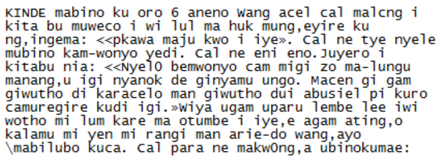


KINDE mabino ku oro 6 aneno wang acel cal maleng i kita bu muweco i wi lul ma huk mung,eyire ku ng,inge ma: «pkawa maju kwo i iye». Cal ne tye nyele mubino kam-wonyo yedi. Cal ne eni eno.

Juyero i kitabu nia: «Nyelo bemwonyo cam migi zo ma-lungu manang,u igi nyanok de ginyamu ungo. Macen gi gam giwutho dî karacelo man giwutho dui abusiel pi kuro cam uregire kudi igi.»

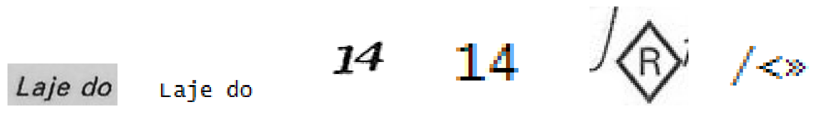
Wiya ugam uparu lembe lee iwi wotho mi lum kare ma ot umbe i iye,e agam ating,o kalamu mi yen mi rangi man arie-do wang,ayo `mabilubo kuca. Cal para ne makwong,a ubino kumae:

(a)



```
KINDE mabino ku oro 6 aneno wang acel cal malcng i
kita bu muweco i wi lul ma huk mung,eyire ku
ng,ingema: <<pkawa maju kwo i iye». Cal ne tye nyele
mubino kam-wonyo yedi. Cal ne eni eno.Juyero i
kitabu nia: <<Nyel0 bemwonyo cam migi zo ma-lungu
manang,u igi nyanok de ginyamu ungo. Macen gi gam
giwutho di karacelo man giwutho dui abusiel pi kuro
camuregire kudi igi.»Wiya ugam uparu lembe lee iwi
wotho mi lum kare ma otumbe i iye,e agam ating,o
kalamu mi yen mi rangi man arie-do wang,ayo
\mabilubo kuca. Cal para ne makw0ng,a ubinokumae:
```

(b)



(c) (d) (e) (f) (g) (h)

Figure 4.4. Results: a) Random text; b) Result for a); c), e), g) Symbols in sea chart images; d), f), h) Corresponding results.

4.2 Evaluation

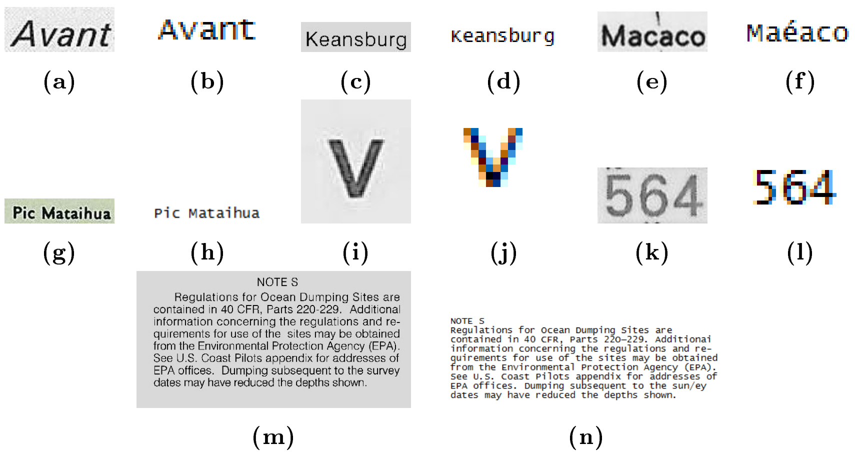
The performance of Tesseract OCR depends on the font styles it was trained for. The software was already trained to recognize English text and digits, therefore no extra training was performed. However, training it for the specific fonts that are used in sea charts could improve its performance.

The character recognition testing process was divided in two stages. Firstly, character recognition was tested, which implied all types of characters found in a sea chart (letters, digits with different fonts and sizes). Secondly, sounding recognition was tested by only considering soundings and setting Tesseract to interpret the input images as a single text line.

4.2.1 Character recognition

A number of images were passed to Tesseract to test its performance in character recognition. The images contained text and numbers having different font styles and sizes. Before passing them to Tesseract, they were converted to gray-scale images, because now Tesseract works only with gray-scale images. The results are shown in figure 4.5.

As can be seen, Tesseract can handle character recognition in a sea chart image when characters are identified beforehand. It can handle a variety of font styles and sizes. However, the recognition is not 100% correct. For example, the line in figure 4.5e represents noise but it is interpreted as an accent and together with the letter *c* are recognized as *é*. Other errors in figure 4.5m are the confusion between *l* and *i* in the word *Additional* or the bad recognition of the word *survey* as *sun/ey*.


Avant
(a)
Avant
(b)
Keansburg
(c)
Keansburg
(d)
Macaco
(e)
Maéaco
(f)
Pic Mataihua
(g)
Pic Mataihua
(h)
V
(i)
V
(j)
564
(k)
564
(l)
(m)
(n)

NOTE S
Regulations for Ocean Dumping Sites are contained in 40 CFR, Parts 220-229. Additional information concerning the regulations and requirements for use of the sites may be obtained from the Environmental Protection Agency (EPA). See U.S. Coast Pilots appendix for addresses of EPA offices. Dumping subsequent to the survey dates may have reduced the depths shown.

NOTE S
Regulations for Ocean Dumping Sites are
contained in 40 CFR, Parts 220–229. Additionai
information concerning the regulations and re-
quirements for use of the sites may be obtained
from the Environmental Protection Agency (EPA).
See U.S. Coast Pilots appendix for addresses of
EPA offices. Dumping subsequent to the sun/ey
dates may have reduced the depths shown.

Figure 4.5. Character recognition with Tesseract: a, c, e, g, i, k, m) Input grayscale images; b, d, f, h, j, l, n) Results.

4.2.2 Sounding recognition

The performance of Tesseract in sounding recognition was tested using images containing all digits from 0 to 9 in all available fonts. As mentioned before, when testing sounding recognition, Tesseract was set to interpret the images as a single line text. This prior assumption regarding the layout analysis improves the recognition rate and can be easily used when recognizing individual soundings. The outcome is given in figure 4.6.

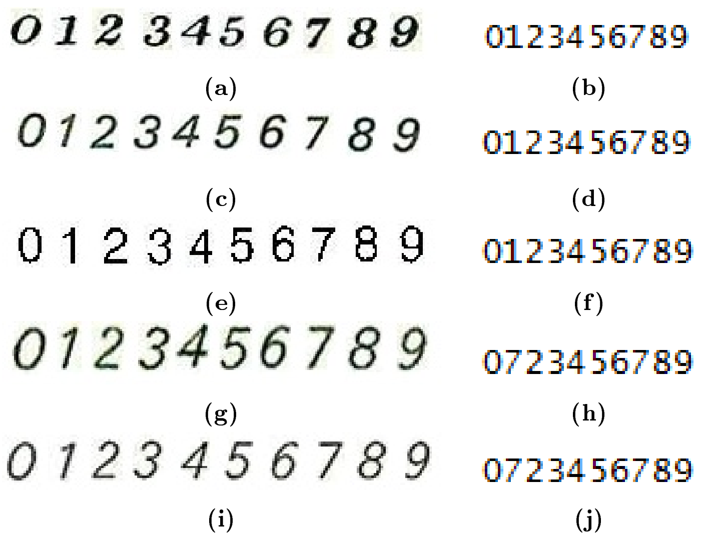

0 1 2 3 4 5 6 7 8 9
(a)
0123456789
(b)
0 1 2 3 4 5 6 7 8 9
(c)
0123456789
(d)
0 1 2 3 4 5 6 7 8 9
(e)
0123456789
(f)
0 1 2 3 4 5 6 7 8 9
(g)
0723456789
(h)
0 1 2 3 4 5 6 7 8 9
(i)
0723456789
(j)

Figure 4.6. Soundings recognition with Tesseract: a, c, e, g, i) Input images; b, d, f, h, j) Results.

The results show that Tesseract is able to correctly recognize the digits, even though different fonts are used. The only problem is that for some font styles Tesseract recognizes the digit *1* as a *7*. This confusion happens for fonts in images 4.6g and 4.6i, because those are slanted fonts and a slanted digit *1* becomes similar to a *7*. There are two possibilities to solve this problem: either the digit *1* is rotated before passing it to Tesseract or Tesseract is trained for this font style.

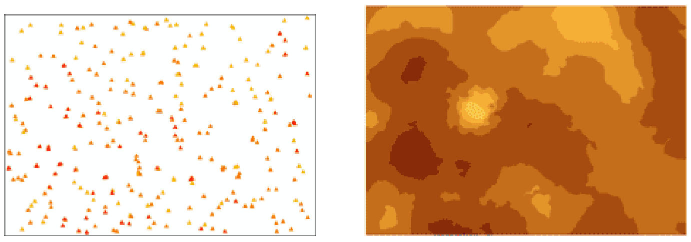
Part III

Automatic interpolation of spot soundings

Chapter 5

Automatic interpolation of spot soundings

Interpolation is a method of constructing new data points within the range of a discrete set of known data points. Suppose a variable has meaningful values at every point within a region, then given the values of that variable at a set of sampled points, an interpolation method can be used to predict values at *every* point. The value of any unknown point is calculated by taking some form of weighted average of the values at surrounding points. Figure 5.1 [1] illustrates an example of interpolation: given the points in fig. 5.1a, new values are calculated for all the points in the region, producing the output in fig. 5.1b.



(a) **(b)**

Figure 5.1. Interpolation: a) Input data, b) Interpolation result.

When dealing with spatial data, the interpolation problem is defined as follows [27]: there is an unknown function, f, values of f are known at finite number of points in space, and the objective is to produce an approximation to the values of f at one or more points where the value is not known. When the function is not known, the process usually begins with a model which includes one or more parameters that must be estimated from the data. Different models may produce different results, thus the model is chosen to ensure that the interpolation function has certain desirable properties.

5.1 Desirable features of the interpolation method

The input data of the interpolation module is given by the spot soundings recognized in the scanned sea chart image. The soundings are points in a 3D environment, each having an x and y coordinate and a depth value (z coordinate) associated with it. They further have to be interpolated to produce a 3D surface.

A number of spatial interpolation methods are known in the literature and each has its specific assumptions and features [26]. In order to decide which method is more appropriate for the current problem, the desirable properties of the result have to be described. These properties are:

1. *Local*
   The basic premise behind spatial interpolation methods is motivated by the first low of geography, given by Waldo Tobler in 1970: "*Everything is related to everything else, but near things are more related than distant things.*" [32]. Thus, near points generally receive higher weigths than far away points. Local methods which operate within a small area around the point being estimated (called neighborhood) and capture the local variation are a suited choice to respect this premise.

2. *Exactness*
   The method should generate an estimate that is the same as the observed value at the sampled point.

3. *Gradual*
   The method should produce a smooth and gradual surface, not an abrupt one.

5.2 Spatial interpolation methods

Spatial interpolation methods usually compute a new value for an unknown point as a weighted average of sample points. The general formula is given by Eq. 5.1.

$$F(x,y) = \sum_{i=1}^{n} w_i f_i \tag{5.1}$$

where n is the number of sample points taken into consideration, f_i are the depth values of the sample points, w_i are the corresponding weights given to each sample point and $F\,(x,y)$ is the new depth value calculated at coordinates (x,y). The weight functions are normalized so that the weights sum to unity.

To properly choose the technique that is best suited for the current problem, several interpolation methods were considered and evaluated.

5.2.1 Distance Transform Interpolation

The distance transform operator (DT) is a morphological operator normally applied to binary images. The result of the transform is a gray-scale image, called *image distance* in which the gray-scalel intensities of points inside foreground regions show the distance to the closest boundary from each point. In [15], de Souza and Banon gave an implementation for the DT transform based on the Chessboard metric[1] using only two iterations of the image: one in raster mode (from left to right and from top to bottom) and one in antiraster mode (from right to left and from bottom to top). However, for the current project the algorithm was adapted to the Chamfer distance. Given two 2-dimensional points, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, the Chamfer distance is calculated with Eq. 5.2 and represents the best approximation of the Euclidean distance in two-passes.

$$d_{(a,b)} = \max\{|x_1 - x_2|, |y_1 - y_2|\}\, a + \min\{|x_1 - x_2|, |y_1 - y_2|\}\,(b - a) \tag{5.2}$$

The Chamfer metric(a, b) has the constraint $0 < b < 2a$ and the best approximation for the Euclidean distance is given when $a = 1$ and $b = 1/\sqrt{2} + \sqrt{\sqrt{2} - 1}$ [14].

If E is a rectangle of $\mathbb{Z}^2$ and $K = \{0, 1\}$, then a binary image is the mapping from E to K, where for any point $(x, y) \in E$, $f(x, y) = 1$ if (x, y) belongs to the foreground and $f(x, y) = 0$ otherwise. If A is a subset of E and A^c its complement, then the image distance of A w.r.t the metric d is the mapping DT_A from E to $\mathbb{R}$ given by:

$$DT_A(p) = \begin{cases} d(p, A^c), & p \in A \\ 0, & otherwise \end{cases} \tag{5.3}$$

calculated for every p in E. The mapping $A \mapsto DT_A$ is called *distance transform*.

Consider E an $m \times n$ image and define two subsets of pixel values which represent the prior and posterior pixels of (x, y) in a V_8 neighborhood: $N_r((x, y), f) = \{f(x-1, y-1), f(x-1, y), f(x-1, y+1), f(x, y-1)\}$ and $N_a((x, y), f) = \{f(x, y+1), f(x+1, y-1), f(x+1, y), f(x+1, y+1)\}$. If M is a matrix with the same size as E, then the pseudocode for computing the DT transform of E is given in Listing 5.1.

```
/* Raster mode */
for $x$ from 1 to $m-2$ do
        for $y$ from 1 to $n-2$ do
                if $f(x,y) == 1$ then
                        $M(x,y) = \min\{N_r((x,y), M)\} + 1$
                else $M(x,y) = 0$;
/* Antiraster mode */
```

[1]Given two 2D points, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, the Chessboard distance between them is: $d(p_1, p_2) = \max\{|x_1 - x_2|, |y_1 - y_2|\}$.

```
for $x$ from $m-2$ to 1 do
    for $y$ from $n-2$ to 1 do
        if $M(x,y) > 1$ then
            $M(x,y) = \min\{M(x,y), \min\{N_a((x,y), M)\} + 1\}$
```

Listing 5.1. DT pseudocode

Before the calculation, the first and last columns and rows of M have to be initialized with the corresponding values of E. The pseudocode returns M, the image distance of E. Figure 5.3c shows the result of the DT transform applied to the binary image in fig. 5.3a having a curved line with 5 interior points as foreground objects: the whiter a pixel, the farther it is from an object.

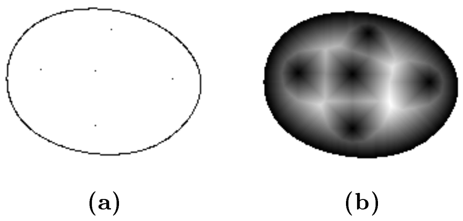


Figure 5.2. Distance trasform: a) Binary image: background = white, foreground = black, b) Image distance.

Even though the DT operator gives a good estimate of the distance between objects (or soundings) in an image, it is not intuitive how to incorporate the z-coordinate given the current algorithm and use it for interpolation. The DT operator is applied on binary images and it only considers a pixel value as being background or foreground. One idea to use the DT operator for interpolation would be to create a 3D function, $f : G_1 \times G_2 \times G_3 \rightarrow \{0, 255\}$, defined as follows:

$$f(x,y,z) = \begin{cases} 255, & I(x,y) \neq z \\ 0, & I(x,y) = z \end{cases} \tag{5.4}$$

where I is the original image containing soundings, x, y are pixel coordinates in I and z is the value of I at given coordinates x and y. By applying the DT operator on this 3D function the minimum distance to the closest object is obtained, while keeping track of the depth value of the closest object by using the z-coordinate. Therefore, all coordinates can now have an interpolated value inverse proportional to the minimum distance to the closest object and direct proportional to the depth value of the closest object: $interpolation(x,y) = \frac{z}{\min_z df(x,y,z)}$.

In 2004, Felzenszwalb et al. gave an algorithm to compute the DT of a one-dimensional function using Euclidean distance [21]. In the same article, he mentions that the DT of an n-dimensional function can be computed by performing one-dimensional transforms along each dimension. Figure 5.3 shows a distance image obtained by applying Felzenszwalb's algorithm and the corresponding interpolation. The drawbacks are that the interpolation result is not smooth, the influence of a spot sounding is radial and symmetrical and in order to calculate the interpolation value only one sounding is taken into account (the closest one). Even though the method is local and exact, it is an abrupt method and doesn't correspond with what it is expected.

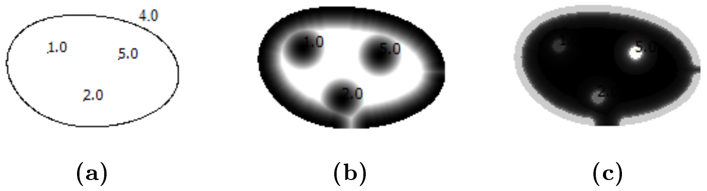


Figure 5.3. Distance trasform: a) Binary image: background = white, foreground = black, b) Image distance, c) Interpolation result.

5.2.2 Inverse Distance Weighting (IDW)

Inverse distance weighting is based on the assumption that the interpolating surface should be influenced most by the nearby points and less by the more distant points. The interpolating surface is a weighted average of the sample points and the weight assigned to each sample point diminishes as the distance from the interpolation point to the sample point increases. The weights can be expressed as in Eq.5.5 [26]:

$$w_i = \frac{\frac{1}{d_i^p}}{\sum_{i=1}^{n} \frac{1}{d_i^p}} \tag{5.5}$$

where d_i is the distance between f_0 and f_i in the neighborhood, p is a positive real number called the power parameter and n represents the number of sampled points used for the estimation. The weight function varies from a value of unity at the scatter point to a value approaching zero as the distance from the scatter point increases. The value of the power parameter p is the main factor influencing the accuracy of IDW: the smoothness of the estimated surface increases as the power parameter increases.

The choice of the neighborhood determines which points are included in IDW. Based on this choice, two methods of IDW were considered and are given in the following subsections.

5.2.3 IDW - Shepard's method

Shepard's method is the simplest form of IDW. Originally, it is a global method, which takes into account all the sample points when computing the value of an unknown point. Nevertheless, it can be localized by either defining the weights to be zero outside some disk of given radius or simply by considering the closest n points. The latter method was chosen for the implementation.

In Shepard's method, the weights are computed as in Eq. 5.5 using the Euclidean distance between the unknown point, f_0 of coordinates (x_0, y_0), and the sample points, f_i of coordinates (x_i, y_i):

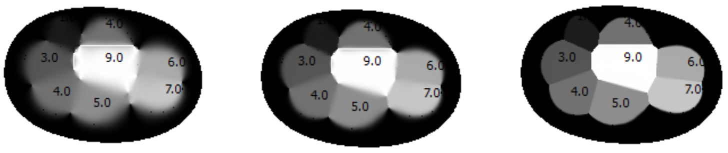
$$d_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2} \tag{5.6}$$

However, Franke and Nielson [22] found another equation to give superior results:

$$w_i = \frac{\left(\frac{R-d_i}{Rd_i}\right)^p}{\sum_{j=1}^{n}\left(\frac{R-d_j}{Rd_j}\right)^p} \tag{5.7}$$

where R is the distance from the interpolation point to the most distant sample point.

The weight function is a function of Euclidean distance and is radially symmetric about each sample point. As a result, the interpolating surface is somewhat symmetric about each point and tends toward the mean value of the sample points. The power parameter also influences the result: greater values of p assign greater influence to values closest to the interpolated point with the result turning into nearly a constant value for large values of p. Figure 5.4 shows this artifact: when $p = 2$ some gradual passing between values is visible, but when $p = 16$ the areas seem to have constant values with abrupt passings.



(a) **(b)** **(c)**

Figure 5.4. Shepard's method with $n = 3$: a) $p = 2$, b) $p = 4$, c) $p = 16$.

5.2.4 IDW - Natural neighbors

The set of three vertices of the triangle containing the interpolation point is another way of defining the neighborhood. When the triangles are given by a Delaunay triangulation, the Delaunay point group represents the *natural neighbors* of the sample point.

The weight function assigned to each sample point is a cubic function. Given a point (x, y) enclosed in a triangle T with vertices (i, j, k), the weight of vertex i is:

$$\begin{aligned} w_i(x,y) = \quad & b_i^2(3-2b_i) + 3\frac{b_i^2 b_j b_k}{b_i b_j + b_i b_k + b_j b_k} * \\ & \left\{ b_j \left[\frac{\|e_i\|^2 + \|e_k\|^2 - \|e_j\|^2}{\|e_k\|^2}\right] + b_k \left[\frac{\|e_i\|^2 + \|e_j\|^2 - \|e_k\|^2}{\|e_j^2\|}\right]\right\} \end{aligned} \tag{5.8}$$

where $\|e_i\|$ is the length of the edge opposite vertex i and b_i, b_j, b_k are the *area coordinates* or *barycentric coordinates* of the point (x, y) w.r.t triangle T. The latter describe the position of the point within the interior of the triangle relative to the vertices of the

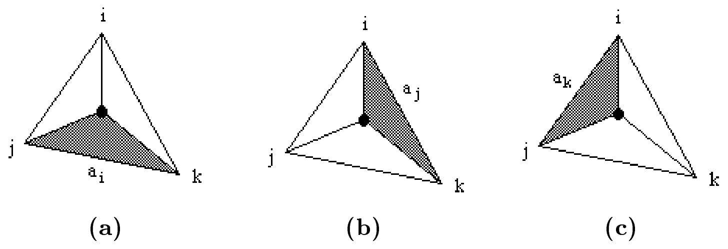


Figure 5.5. Barycentric coordinates for a point in a triangle.

triangle. The magnitude of the coordinates corresponds to area ratios as shown in Figure 5.5.

The coordinates of the interior point can be written in terms of the coordinates of the vertices as follows:

$$x = b_i x_i + b_j x_j + b_k x_k \tag{5.9}$$
$$y = b_i y_i + b_j y_j + b_k y_k \tag{5.10}$$
$$1 = b_i + b_j + b_k \tag{5.11}$$

Solving the above equations for b_i, b_j, b_k yields:

$$b_i = \frac{1}{2A}\left[(x_j y_k - x_k y_j) + (y_j - y_k)x + (x_k - x_j)y\right] \tag{5.12}$$
$$b_j = \frac{1}{2A}\left[(x_k y_i - x_i y_k) + (y_k - y_i)x + (x_i - x_k)y\right] \tag{5.13}$$
$$b_k = \frac{1}{2A}\left[(x_i y_j - x_j y_i) + (y_i - y_j)x + (x_j - x_i)y\right] \tag{5.14}$$

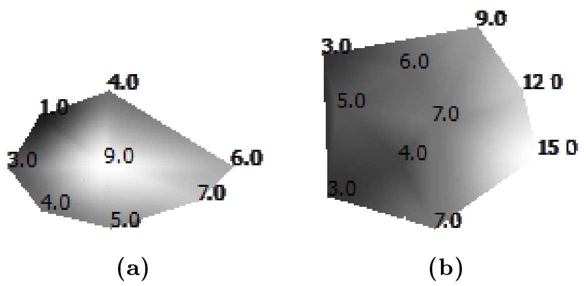


Figure 5.6. IDW with natural neighbors.

Figure 5.6 shows some results of IDW interpolation using natural neighbors. The result is visibly improved with a smooth and gradual passing between soundings.

5.3 Evaluation

Figure 5.7 shows the interpolation results of several user defined spot soundings using the three methods mentioned above. After analyzing the methods and the results they produce, some conclusions were drawn w.r.t the desirable properties they should meet:

1. *Interpolation with DT* is an exact and local method, but the interpolated value only depends on the closest spot sounding. Only close points are affected by a sounding because the minimum distance has a rapid growth in a 3D space and the denominator in eq. 5.4 increases very fast, making the result to tend to 0. The interpolation is not smooth, nor gradual.

2. *IDW with Shepard's method* is a local method, exact, but the radial symmetry of the Euclidean distance prevents it from having a gradual passing between soundings. Also, the computational time of the algorithm is large because for each interpolated point the closest n-points have to be found.

3. *IDW with Natural neighbors* fulfills all the desired properties: local, exact, gradual and smooth.

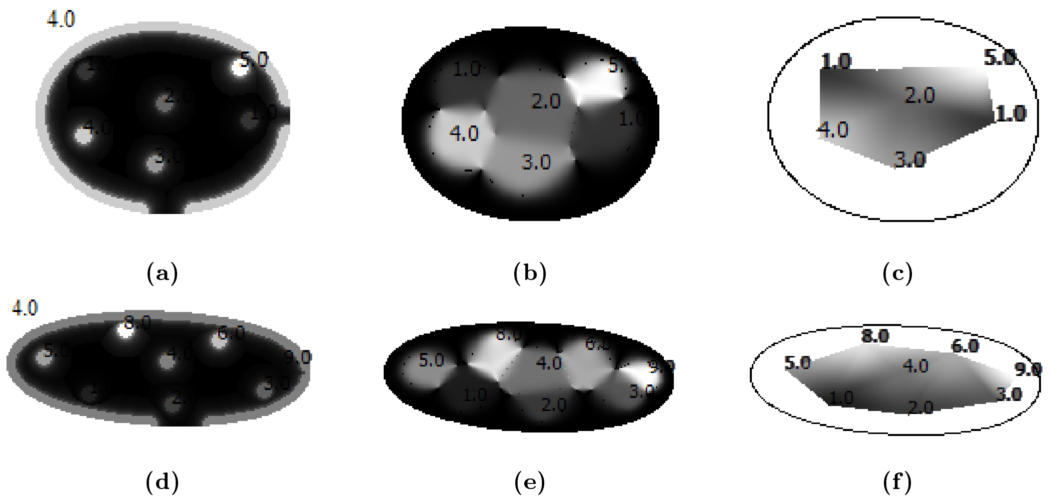


Figure 5.7. Interpolation methods: a, d) DT interpolation, b, e) IDW - Shepard's method, c, f) IDW - Natural neighbors.

Therefore, IDW with natural neighbors was chosen as the best interpolator; it gives a quick interpolation from sparse data, producing a smooth surface, being simple and fast.

Part IV

System overview

Chapter 6

System overview

In the previous chapters, all the methods used in the project were described and evaluated. However, the evaluation was per method and not per system as a whole. Thus, an evaluation of the entire system is needed. This chapter focuses on analyzing the behavior of the complete system for automatic conversion of scanned sea charts into 3D models.

6.1 Appearance

The GUI interface of the final product facilitates the manipulation of sea chart images. Figure 6.1 shows the appearance and some of the features of the system.

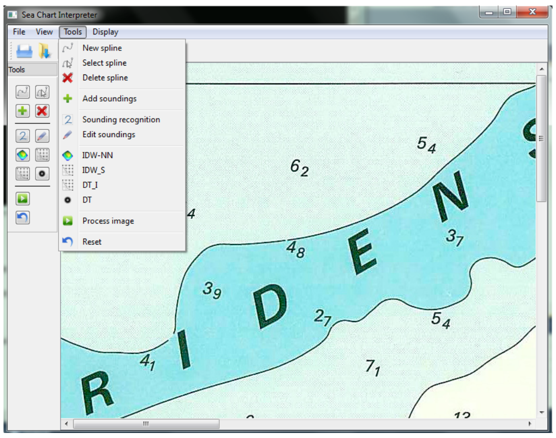


Figure 6.1. GUI interface.

Besides the already mentioned features that concern spot sounding recognition and interpolation, additional features give the possibility of creating and adjusting splines that simulate contour lines and the possibility of manually adding spot soundings that can be interpolated. Other features regard visualization options (zoom in, zoom out, normal view, fit-to-window view) or display options (the possibility to display only individual objects, groups, recognized soundings or the interpolated surface). The user also has the possibility to edit (add, delete or modify) spot soundings after the recognition process. This feature is useful to correct possible errors and obtain a correct 3D model of the seabed.

6.2 Performance

The module for converting 2D scanned sea chart images into 3D models was implemented in C++, using Qt [10] as a GUI library and the GDAL library [3] to write GeoTiff files.

The running time of the algorithm depends on the size and complexity of the input image. If for a small image of 200×200 pixels the result is given in several seconds, for an image of 2000×2000 pixels the algorithm takes about forty seconds to output the result in Release mode. The algorithm was tested on a laptop with Intel Core i3 processor running at 2.27 Ghz, 4 GB RAM and Windows 7 32-bit operating system.

6.3 Testing procedure

In order to analyse the performance of the algorithm, the testing process was divided in two stages. In the first stage, small parts of a scanned sea chart image were given as an input to the algorithm. This testing stage gives a better overview and the possibility of identifying more drawbacks, because it is easier to analyse smaller parts. In the second stage, an entire sea chart image was given as an input to analyse the performance of the system in a real situation.

6.3.1 Testing small parts of a scanned sea chart image

For this part of the analysis, three images were cropped from a scanned sea chart; they are displayed in figures 6.2a, 6.2b and 6.2c.

The result of the classification module is shown in figures 6.2d, 6.2e, respective 6.2f. Each color represents a label for the type of the object: *yellow* - horizontal straight lines, *magenta* - vertical straight lines, *cyan* - oblique straight lines, *red* - curved lines, *green* - characters, *blue* - other objects; all other colors represent background objects which are not processed. Figures 6.2g, 6.2h and 6.2i show the recognized soundings. On a first glance, it can be seen that the more simple an image is, the better the results. The type of errors that can affect the result of the algorithm are:

1. *Errors due to overlapping or touching symbols.*
   For example, the digit *7* in figure 6.2d is touching an oblique straight line, thus they are identified as a single object in the image and recognized as a curved line. The same thing happens for some digits in figure 6.2f: digits *7*, *1*, *3*, *2* etc. are overlapping a curved line.

2. *Errors due to the classification process.*
   For example, three digits *1* in figures 6.2d and 6.2e are wrongly classified as *other objects* and are not passed further to Tesseract. The reason is the shape of the

digits: in figure 6.2e, the wrongly classified *1* is thicker than usual, while in the other images, the digits are thinner.

3. *Errors due to the OCR engine.*
   In figure 6.2e, number *26* is recognized as a group of characters and passed to Tesseract OCR engine. However, because digit *2* is touching a small oblique line, the number is not properly recognized by the engine and the sounding is lost. Another problem is the confusion between the digit *0* and the letter *o* in figure 6.2i.

4. *Errors due to the misinterpretation of other numbers displayed on the sea chart as soundings.*
   In figure 6.2f there is a string of characters *G"31"* in the bottom right side of the image. Digits *3* and *1* are labeled as characters and properly recognized, ending up as a sounding with value *-31.0* on the chart. However, it does not represent a spot sounding, but other information.

Figures 6.2j, 6.2k and 6.2l show the result of the interpolation algorithm and figures 6.2m, 6.2n and 6.2o display the corresponding 3D views of the interpolated surfaces[1]. The resulting surfaces are smooth and the interpolated values create a gradual passing between different spot soundings. A disturbing artifact could be the visibility of the Delauney triangles or edges in the 2D images of the interpolated surfaces. However, when displaying the 3D view of the corresponding surfaces, the Delaunay edges are not visible anymore and the result can be interpreted as a terrain or an earth surface.

6.3.2 Testing full scanned sea chart images

Given the nautical chart in figure 6.3a as an input, the results of the spot sounding recognition algorithm are displayed in figure 6.3b. Figure 6.3c shows a 3D view of the surface created by interpolating the recognized soundings and figure 6.3d shows the 3D view of the interpolated surface obtained after the wrongly recognized soundings were corrected. The size of the original image was (2143×1356) pixels and the program needed about 25 minutes to process it.

By analyzing the output of the algorithm, we see that the main problems are caused by the bad results of the OCR engine. The confusion between a slanted digit *1* and digit *7* is the most important problem, being the cause of the big holes in the output surface in figure 6.3c. For example, a sounding with value *-13.0* is recognized as *-73.0*; the difference between values is very big comparative to other soundings on the chart, thus the large holes in the 3D surface.

Other problem is that the original interpolated surface is not entirely smooth like the surface created after the correction step. The reason for this is the bad recognition of some of the soundings. However, the difference in absolute values is not as big as earlier mentioned: for example, the sounding *6.5* is recognized as *5.5*. A confusion of the classification algorithm is another reason for a non-smooth surface. Some digits that appear on the chart and do not represent soundings are similar in font and size with spot soundings and end ep being classified as soundings. For example, digits that appear in

[1]For a better view, the z-coordinates were exaggerated in the 3D models.

a string of characters that give information about buoys are classified as soundings and recognized accordingly.

The same type of errors can be observed by analyzing the chart of the coast of Rio de Jaineiro in figure 6.4. The majority of the errors are caused by the OCR engine, but there are also errors due to overlapping symbols or to false soundings which, as mentioned before, are digits similar in shape with the spot soundings, but represent other information on the chart.

By using the *Edit soundings* option after the spot sounding recognition process has ended, all the errors can be corrected to obtain an accurate 3D model of the seabed.

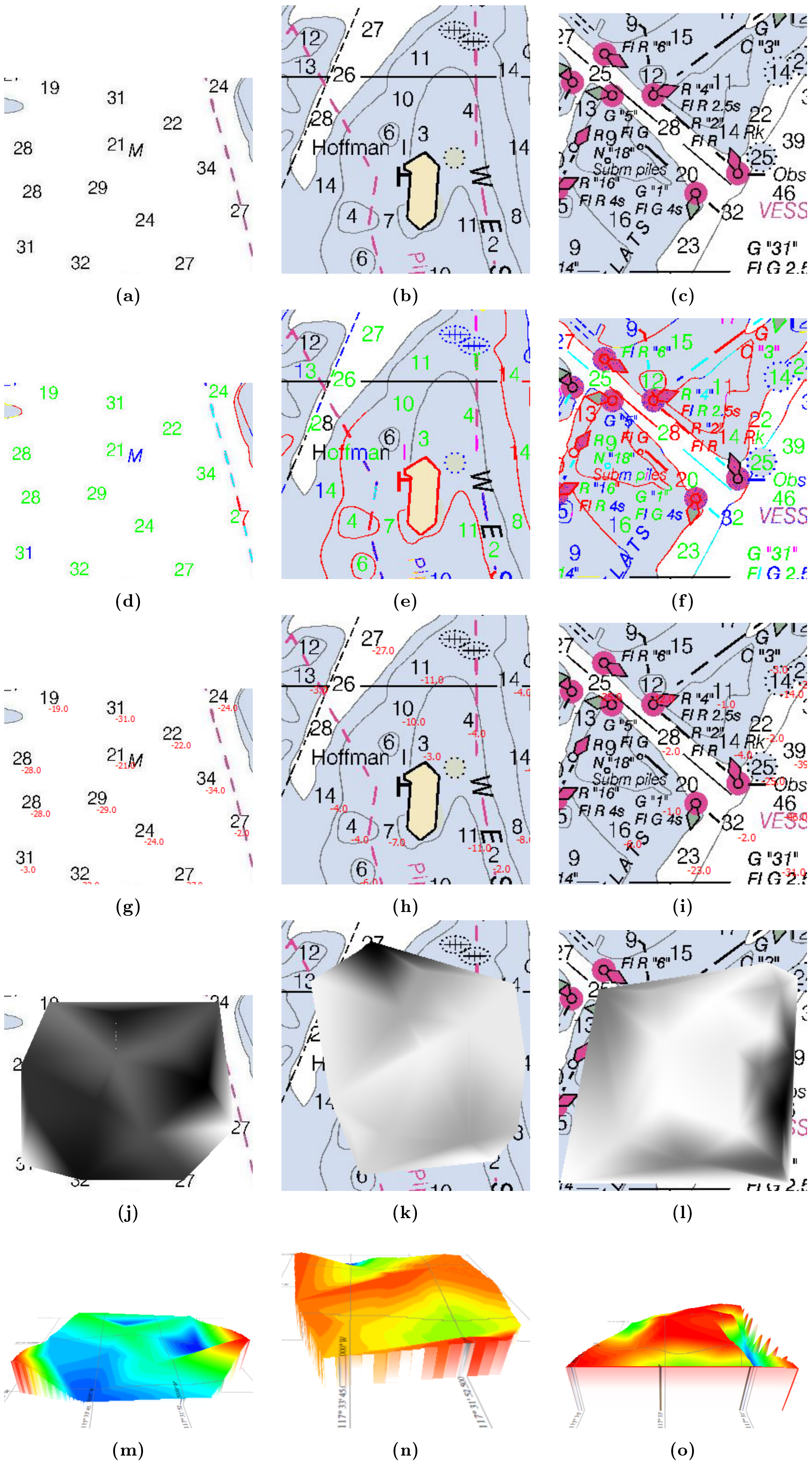
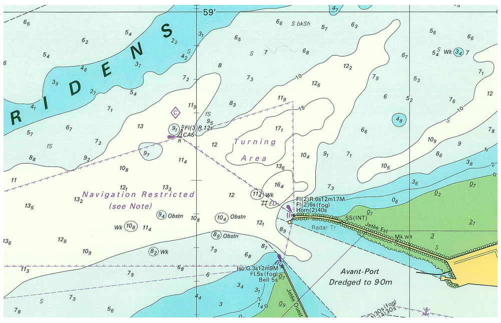
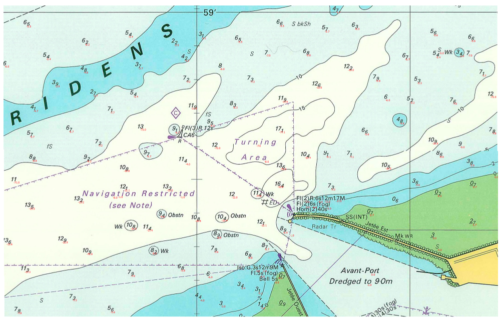
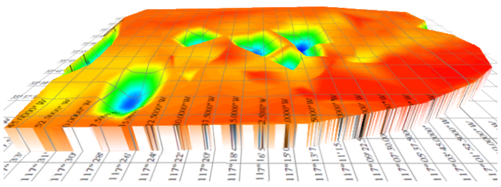


Figure 6.2: a–c) Original images, d–f) Classification, g–i) Sampling positions included, j–l)


(a)


(b)


(c)

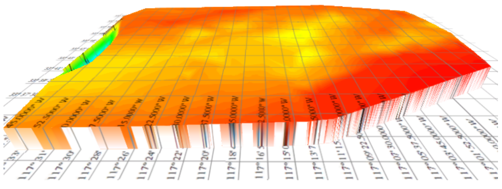
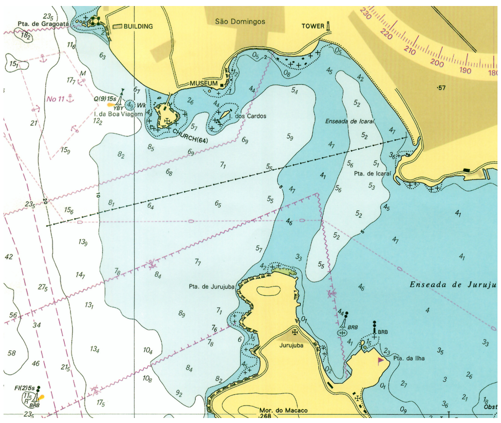
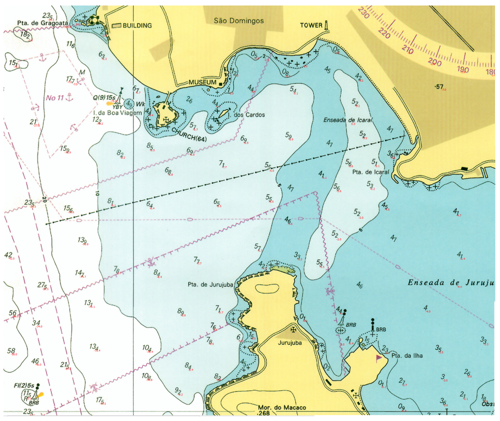
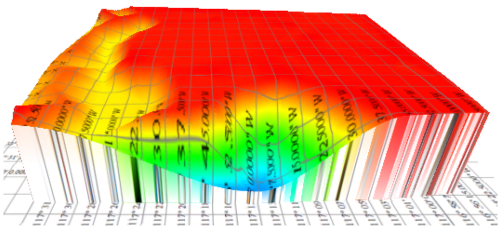

(d)

Figure 6.3. a) Original scanned sea chart image; b) The result of the spot sounding recognition method; c) 3D view of the resulting interpolated surface; d) 3D view of the interpolated surface after the wrongly recognized spot soundings were corrected.


(a)


(b)


(c)

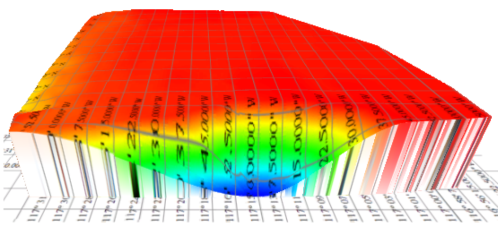

(d)

Figure 6.4. a) Original scanned sea chart image; b) The result of the spot sounding recognition method; c) 3D view of the resulting interpolated surface; d) 3D view of the interpolated surface after the wrongly recognized spot soundings were corrected.

Part V

Conclusions

Chapter 7

Conclusions

The digitization of nautical charts is a problem that was often approached by researchers. It has a variety of applications in navigation or in the development of navigational software. Digital nautical charts are necessary for the conversion of maritime terrain in 3D models. The latter are useful for 3D applications like training simulators used in maritime schools, games, movies etc. However, there is still no complete system that handles the digitization of nautical charts w.r.t all symbols encountered or that is capable of converting a 2D nautical chart in a 3D model.

The current project gives a partial solution for both digitizing sea charts and converting them into 3D simulation models. Given a scanned sea chart image, the algorithm first separates all different symbols from the background by using a threshold-based segmentation method. The thresholds are found with the K-Means algorithm applied on the grayscale image histogram. Usually, the segmentation method correctly separates the foreground objects, but it is unable to handle overlapping or touching symbols. Another problem is that symbols which are represented with colors that have high intensity values are interpreted as background and are lost during the segmentation process.

Further, all the individual objects in the segmented binary image are identified using a labeling method called *region tracking with correspondence tables*. The method is able to correctly identify all connected components in the image in only one passing. In this research a method has been developed to classify individual objects into: *vertical straight lines*, *horizontal straight lines*, *oblique straight lines*, *curved lines*, *characters* and *other symbols*. Geometrical features of the objects like area, center of gravity, orientation, density, axis-aligned and arbitrarily-oriented bounding boxes are used to create innovative decision rules used in the classification process. The decision rules are based on threshold values that were empirically determined and had fixed values over all charts. The thresholds had to be flexible enough to correctly classify the symbols along different charts with different font and styles, but also strict enough to make a difference. Overall, the classification rate is higher than 80%. Possible errors are due to overlapping or touching symbols that are not separated during the segmentation process or to the threshold values. The output of this step is the digitized version of the nautical chart given as input.

To recognize spot soundings in the chart, all symbols in the *character* class are passed to Tesseract OCR engine. Tesseract has a good performance in character recognition, but in some cases it can fail because of font styles or confusions between characters (for example, slanted *1* is confused with *7*, *0* sometimes is confused with *o*).

The recognized soundings are then interpolated to obtain 3D simulation surfaces. This research gives a comparison between different interpolation methods: interpolation using the distance transform operator, inverse distance weighting with Shepard's method and inverse distance weighting with natural neighbors. After analysing the methods in terms of the desirable features they should fulfill (exact, local, gradual), the Inverse Distance Weighting with Natural Neighbors was chosen as the best interpolator. The method is based on the assumption that an interpolated value should be mostly influenced by nearby points and less by further away points. The nearby points are the vertices of the Delauney triangle containing the point (called natural neighbors) and the weights are inverse proportional to the corresponding distances from the point to the triangle vertices. IDW with Natural Neighbors is an exact method, because the estimate value is the same as the observed value at a sample point. It is a local method by only considering the natural neighbors of a given point. It is also a gradual method, producing a smooth surface. The interpolated surface meets all the required features and it can be interpreted as a 3D terrain.

The performance of the whole system was analysed in the previous chapter and all possible drawbacks were reviewed. Even though the output is, in general, accordingly to what it is expected, errors due to the misinterpretation of symbols and the bad recognition of spot soundings may sometimes happen.

The goal of this research was to design and implement a tool which would facilitate the work of VSTEP in the process of creating a realistic seabed for their maritime training simulator, *NAUTIS*. At the moment, this process is done by manually tracking the contour lines in the sea charts and interpolating their depth values. The 3D surface obtained from the interpolation of spot soundings would increase the accuracy of the seabed model by overlaying the two surfaces and thus considering sounding information as well. Even though the result of this study is not a perfectly working system that converts scanned sea chart images into 3D models, it represents a good basis for further development and shows that an automatic conversion system is possible.

Chapter 8

Future work

As mentioned before, the algorithm for converting a 2D scanned sea chart image into a 3D model has certain drawbacks. The performance of the algorithm can be improved and the remaining open issues that need further work are:

- *Overlapping / touching symbols*
  Overlapping or touching symbols represent a major problem for the performance of the algorithm. Usually, overlapping symbols have different colors, but they are hard to detect in a graylevel-based segmentation method and without any user intervention. Incorporating color information in the segmentation method or using other color spaces (for example, Lab color space) could improve the result of the algorithm. Another possibility is to allow the user to select from a list the colors used to represent the symbols that he is interested in.

- *Object features*
  The features used in the classification process are geometrical features of the symbols. The performance of the classification could be improved by taking into account other types of features, like color features, or more complex features like the Zernike moments or the Fourier descriptors.

- *Decision rules*
  Another problem that affects the result is the recognition of false soundings on the chart. These are digits that have a similar font and size with the spot soundings, but represent other type of information (for example, buoy information). New decision rules should be build to separate these false soundings from the real ones. They could be based on the analysis of the surrounding area to check whether they belong or not to a group of characters.

- *Choice of OCR engine*
  The OCR engine is responsible for the correct recognition of characters and spot soundings, therefore the performance of the system is directly affected by the choice of the OCR engine. Tesseract is rated to be a good OCR engine, but its performance could be improved by training it for specific font faces that are used in sea charts. Also, other OCR engines could be tested and compared with Tesseract in terms of recognition rates.

- *Contour lines extraction*
  In order to complete the project proposed by VSTEP, the contour lines should be extracted and saved to an ESP file. The decision rules for curve classification are valid, but the remaining problem is the fact that curves often intersect other symbols on the chart. Besides this, a contour line reconstruction algorithm would be necessary to recover the entire curved line.

- *Eliminating grids*
  Charts usually display grids which are a set of horizontal and vertical parallel straight lines which often intersect other symbols on the map. Eliminating these line would facilitate the classification of the remaining symbols.

- *Interpolation method*
  To increase the accuracy of the interpolated surface, contour lines and their corresponding depth should be taken into account when interpolating spot soundings. A new method for spatial interpolation that fulfills this constraint should be developed.

Appendix A

Delaunay Triangulation

Triangulation is the division of a surface or plane polygon into a set of triangles, usually with the restriction that each triangle side is entirely shared by two adjacent triangles [12]. It was proven that every surface has a triangulation, but it might require an infinite number of triangles [35]. Figure A.1 shows an example of triangulation of a polygonal region.

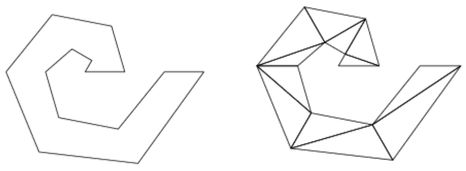


Figure A.1. a) Polygonal surface, b) Triangulation

In [25], Barry Joe defines a *valid triangulation* of a polygonal region as a collection of triangles that form a 'tiling' of the region without overlaps or gaps. Given a set V of vertices in the plane that are not collinear, Joe states that a *Delaunay triangulation* of V is a valid triangulation in the convex hull of V[1] which satisfies the *max-min angle criterion*: for any two triangles in the triangulation that share a common edge, if the quadrilateral formed from the two triangles with the common edge as its diagonal is strictly convex, the replacement of the diagonal by the alternative one does not increase the minimum of the six angles in the two triangles making up the quadrilateral. In other words, the Delaunay triangulation is the triangulation which maximizes the minimum angle in the triangles globally as well as locally in any two adjacent triangles which form a strictly convex quadrilateral.

Consider $V = \{p_i, p_j, p_k, p_l\}$ as in Figure A.2 [19]. The quadrilateral can be triangulated by either adding the edge $\overline{p_i p_j}$ or by removing this edge and inserting $\overline{p_k p_l}$ instead. The latter procedure is called an *edge flip*. It is visible that the triangulation given by inserting the edge $\overline{p_k p_l}$ is a Delaunay triangulation, while the other one is not, because the minimum angle of the triangles $p_i p_k p_l$ and $p_j p_k p_l$ is greater than the minimum angle of the triangles $p_i p_j p_l$ and $p_i p_j p_k$.

[1]The convex hull of a point set V is the smallest convex set containing all the points in V.

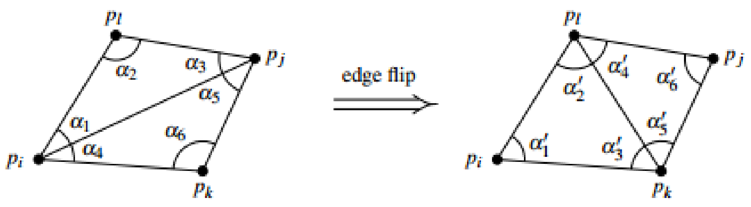


Figure A.2. Flipping an edge

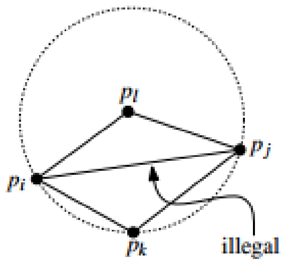


Figure A.3. Circle criterion

A Delaunay triangulation also satisfies the *circle criterion*: the circumcircle of any triangle in the triangulation contains no vertex of V in its interior. This property is useful to find a Delaunay triangulation of a given set of vertices by searching for Delaunay edges $\overline{p_ip_j}$, where $p_i, p_j \in V$. An edge is a Delaunay edge if and only if there exists a point c such that the circle centered at c and passing through p_i and p_j does not contain any other vertex of V in its interior. Given the triangulation in Figure A.3 [19], the circle C is the circle through p_i, p_j, p_k. The edge $\overline{p_ip_j}$ is an illegal Delaunay edge, because the point p_l lies inside the circle C.

For the current project, the Delaunay triangulation was calculated using the GEOMPACK library [4], a free C++ library that handles both 2- and 3-dimensional triangulations. The Delaunay triangulation method is implemented inside GEOMPACK using an incremental approach: one vertex is added at a time and the affected parts are retriangulated.

Bibliography

[1] *A brief introduction to spatial interpolation, http://www.bisolutions.us/a-brief-introduction-to-spatial-interpolation.php.*

[2] *Etichetarea imaginilor binare, http://alpha.imag.pub.ro/ro/cursuri/archive/etich.pdf.*

[3] *Gdal - geospatial data abstraction library, http://www.gdal.org/.*

[4] *Geompack - delaunay triangulations, http://people.sc.fsu.edu/ jburkardt/cpp_ src/-geompack/geompack.html.*

[5] *Kmeans - interactive demo, http://home.dei.polimi.it/matteucc/clustering/tutorial_-html/appletkm.html.*

[6] *Kmeans tutorial slides, http://www.autonlab.org/tutorials/kmeans.html.*

[7] *Learn about nautical charts, http://www.nauticalcharts.noaa.gov/mcd/learn_-aboutcharts.html.*

[8] *Nautical chart, http://en.wikipedia.org/wiki/nautical_ chart.*

[9] *Nautical charts, http://www.sailingissues.com/navcourse2.html.*

[10] *Qt, http://qt.nokia.com/products/.*

[11] *tesseract-ocr, http://code.google.com/p/tesseract-ocr/.*

[12] *Triangulation, http://mathworld.wolfram.com/triangulation.html.*

[13] S. Ablameyko, V. Bereishik, M. Homenko, N. Paramonova, and O. Patsko, *Automatic/interactive interpretation of color map images*, Pattern Recognition, International Conference on **3** (2002), 30069.

[14] M. Akmal Butt and P. Maragos, *Optimum design of chamfer distance transforms*, Trans. Img. Proc. **7** (1998), no. 10, 1477–1484.

[15] Gerald J.F. Banon Arley F. de Souza, *Distance transform as a basis for contour line interpolation and intermediate line generation*, (2004).

[16] Patrice Arrighi and Pierre Soille, *From scanned topographic maps to digital elevation models.*

[17] Yao-Yi Chiang and Craig A. Knoblock, *An approach for recognizing text labels in raster maps*, ICPR, 2010, pp. 3199–3202.

[18] John C. Davis, *Statistics and data analysis in geology*, John Wiley & Sons, Inc., New York, NY, USA, 1973.

[19] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf, *Computational geometry: algorithms and applications*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[20] Line Eikvil, Kjersti Aas, and Marit Holden, *Tools for automatic recognition of character strings in maps*, CAIP, 1995, pp. 741–746.

[21] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, *Distance transforms of sampled functions*, Tech. report, Cornell Computing and Information Science, 2004.

[22] Richard Franke and Greg Nielson, *Smooth interpolation of large sets of scattered data*, International Journal for Numerical Methods in Engineering **15** (1980), no. 11, 1691–1704.

[23] Steffen Frischknecht, Entela Kanani, and Alessandro Carosio, *A raster-based approach for the automatic interpretation of topographic maps*, In: IAPRS **32** (1998), 523–530.

[24] Tudor Ghircoias and Remus Brad, *A new framework for the extraction of contour lines in scanned topographic maps*, IDC, 2010, pp. 47–52.

[25] Barry Joe, *Delaunay triangulat meshes in convex polygons*, SIAM J. Sci. Stat. Comput. **7** (1986), 514–539.

[26] J. Li and A.D. Heap, *A review of spatial interpolation methods for environmental scientists*, Geoscience Australia, Canberra, 2008.

[27] D. E. Myers, *Spatial interpolation: an overview*, Geoderma **62** (1994), 17–28.

[28] LL.D. Nathaniel Bowditch, *The american practical navigator: an epitome of navigation*, 1995.

[29] Department of Commerce National Oceanic, Department of Defense National Imagery Atmospheric Administration, and Mapping Agency, *Nautical chart systems, abbreviations and terms*, 1997.

[30] R.J. Fowler J.J. Little D.M. Mark Peuker, T.K., *The triangulated irregular network*, American Society of Photogrammetry: Digital Terrain Models (DTM) Symposium (1978), 516–540.

[31] R. Smith, *An overview of the tesseract ocr engine*, Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02 (Washington, DC, USA), ICDAR '07, IEEE Computer Society, 2007, pp. 629–633.

[32] Waldo R. Tobler, *A computer movie simulating urban growth in the Detroit region*, Economic Geography **46** (1970), 234–240.

[33] Øivind Due Trier, Anil K. Jain, and Torfinn Taxt, *Feature extraction methods for character recognition-a survey*, Pattern Recognition **29** (1996), no. 4, 641–662.

[34] Aurelio VelÃązquez and Serguei Levachkine, *Text/graphics separation and recognition in raster-scanned color cartographic maps.*, GREC (Josep LladÃşs and Young-

Bin Kwon, eds.), Lecture Notes in Computer Science, vol. 3088, Springer, 2003, pp. 63–74.

[35] Jeff Weeks and George Francis, *Conway's ZIP Proof*, The American Mathematical Monthly **106** (1999), 393–399.

[36] Dongjun Xin, Xianzhong Zhou, and Huali Zheng, *Contour line extraction from paper-based topographic maps*, 2006.