*MVA'90* IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

# AN INTERACTIVE COLOUR LINE RECOGNITION SYSTEM FOR SEISMIC SECTION DIGITISATION

Jonathan Shapiro and Jin Zhengping

The Turing Institute
George House, 36 North Hanover St.,
Glasgow G1 2AD, Scotland.
E-mail {jonathan, jin}@turing.ac.uk

## ABSTRACT

In this paper, we describe a commercial vision system for the digitisation and extraction of horizons drawn with coloured pencils on seismic sections. The system combines image processing and statistical pattern recognition techniques together with a graphical user interface to provide a solution that is more accurate, faster and involves less user interaction than the manual method.

## INTRODUCTION

Seismic sections are interpreted by geophysicists who decide what geological events (referred to as horizons) are present on the section by drawing over them with coloured pencils. These horizons, once digitised, provide input to a 3D surface modelling package. The conventional method for this digitisation task involves placing the paper section onto a large digitising table on which a technical assistant indicates points of colour by clicking a mouse. This method is both labour intensive and tedious. As such, it is prone to inaccuracies which can lead to incorrect interpretation of the data.

In this paper, we describe an alternative approach to this task which combines image processing [2] and statistical pattern recognition [1] techniques together with a graphical user interface to provide a semi-automatic solution. The goals of the system, which was termed **OASIS**, (operator assisted seismic interpretation system), are to provide a digitisation method that is more accurate, faster and involves less user interaction which in itself, is less tedious than the conventional method. To achieve this, the system shifts the burden of accuracy from the user to the computer which is coupled to a fast co-processor enabling the system to run at high speed.

The system hardware comprises of a large, relatively low-cost, low spatial resolution (100 dpi), colour (24 bits per pixel), flat-bed scanner connected to a Sun-4/110 workstation via a separate frame store and fast microcode co-processor.

The main system software modules, described in the following sections are indicated below;

- data acquisition
- data calibration
- colour training
- classification
- line extraction
- editing
- conversion to UKOOA format

## DATA ACQUISITION

Seismic sections are digitised by the scanner into 3 images of size 2580 × 1720, each containing 8 bits per pixel, $s_h(x,y), h = r,g,b$, for red, green, and blue channels that span RGB space as

$$\mathcal{R}^3 = \left\{ \mathbf{s} \mid \mathbf{s} = \begin{bmatrix} s_r \\ s_g \\ s_b \end{bmatrix}, \begin{array}{l} 0 \le s_r < 256, \\ 0 \le s_g < 256, \\ 0 \le s_b < 256 \end{array} \right\}$$

Each of the image then subject to a normalisation process,

$$p_h = 255(s_h - b_h)/(w_h - b_h),\ h = r,g,b$$

where $b_h$ and $w_h$ are scanned images of black and white paper which model the underlaying illumination distribution of the scanner. The application of these images, through the normalisation process described above, eliminates the illumination unevenness introduced by the scanner. This process enables the system to work with a scanner that may not produce an even illumination distribution.

# CALIBRATION

Calibration involves the generation of a mapping that transforms pixels into real world values found on the section. A seismic section is made up of a number of traces (a reflected signal received from a controlled explosion). The horizontal axis of a typical seismic section represents the trace number or "shot point". The vertical axis represents time, or some other measure of depth. Calibration is an interactive process, whereby the user indicates at least two points along the shot point axis and at least one point on the time axis. The user tags these points with information from the section, (the shot point numbers for the points along the horizontal axis and the time for the points along the vertical axis). This data is used to generate a function that maps pixel positions $(x, y)$ to the dimensions of shot point and time $(s, t)$, such that,

$$\begin{aligned} s &= a_x[\cos(\theta)(x - d_x) - \sin(\theta)(y - d_y) + d_x] + b_x \\ t &= a_y[\sin(\theta)(x - d_x) + \cos(\theta)(y - d_y) + d_y] + b_y. \end{aligned}$$

# COLOUR TRAINING

The goal of colour training is to produce cubes that define sub-spaces of $\mathcal{R}^3$

$$\mathrm{C}(c) = \left\{ \mathrm{p} \mid \mathrm{p} \in \mathcal{R}^3, \begin{array}{c} r_{min}(c) \le p_r < r_{max}(c) \\ g_{min}(c) \le p_g < g_{max}(c) \\ b_{min}(c) \le p_b < b_{max}(c) \end{array} \right\}, \; c \in \mathcal{C}$$

for classification. Thus, pixel $\mathbf{p}(x, y)$ is classified as $c$ if $\mathrm{p}(x, y) \in \mathrm{C}(c)$. Cube generation is achieved through an interactive process and can be identified as consisting of four stages; namely,

- training data acquisition
- background filtering
- colour sample sorting
- threshold setting

**Training Data Acquisition:** Suppose lines of $N_c$ colours $\mathcal{C} = \{c_1, c_2, \cdots, c_{N_c}\}$ are to be extracted from a section. To train on a colour $c \in \mathcal{C}$, the user draws a region using a mouse on the seismic section covering a part of a line of $c$. This region is referred to as the "central box". As the user draws the region, the system displays up to 4 additional regions (see figure 1), that are referred to as "flanking boxes".

The user ensures that the flanking boxes do not contain any part of the coloured line present in the central box.

Training data for $c$

$$\mathcal{M}(c) = \left\{ \mathrm{p}(x, y|c) \mid \begin{array}{c} x = m_x, m_x + 1, \cdots, m_x + N_{c_x} - 1 \\ y = m_y, m_y + 1, \cdots, m_y + N_{c_y} - 1 \end{array} \right\}$$

$$\mathcal{F}(c) = \left\{ \mathrm{p}_i(x, y|c) \mid \begin{array}{c} i = 0, 1 \text{ or } 0, 1, 2, 3 \\ x = f_{i_x}, f_{i_x} + 1, \cdots, f_{i_x} + N_{i_x} - 1 \\ y = f_{i_y}, f_{i_y} + 1, \cdots, f_{i_y} + N_{i_y} - 1 \end{array} \right\},$$

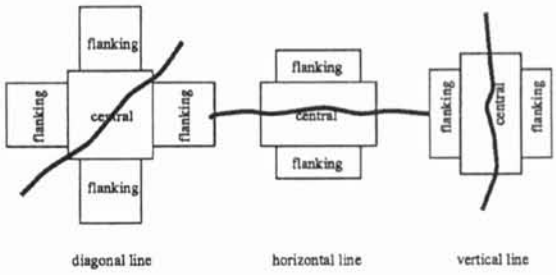


Figure 1: Central and flanking box configurations.

where $\mathcal{M}(c)$ are the data from the central box and $\mathcal{F}(c)$ the flanking boxes, are generated.

For the central region, all pixels of non-colour $c$ are identified and removed using the background filtering method described below.

**Background Filtering:** The data in $\mathcal{M}(c)$ consists of both colour $c$ and non-colour $c$ samples. The non-colour $c$ samples form a set $\mathcal{B}(c)$. Since the data in $\mathcal{F}(c)$ does not contain samples of $c$, it is used to approximate $\mathcal{B}(c)$, which in turn is used to differentiate colour samples from non-colour samples for $\mathcal{M}(c)$. In this system, $\mathcal{B}(c)$ is expressed as

$$\mathcal{B}(l, u|c) = \left\{ \mathrm{p} \mid \mathrm{p} \in \mathcal{R}^3, l(r, g|c) \le p_b \le u(r, g|c) \right\}.$$

where

$$l(r, g|c), \; u(r, g|c); \quad r, g = 0, 1, \cdots, 255,$$

are the lower and upper boundaries in the blue dimension defined over the red and green 2-dimensional planes. The advantages of this representation are that it is simple and that it can represent arbitrary shape as long as colour clusters are not sandwiched by $l(r, g)$ and$u(r, g)$, which is the case in this application.

The non-colour $c$ pixels in $\mathcal{M}(c)$ can now be filtered away through $\mathcal{B}(c)$: $\mathbf{p}(x, y) \in \mathcal{M}(c)$ is only classified as $c$ if $\mathbf{p}(x, y) \notin \mathcal{B}(c)$. All the pixels in $\mathcal{M}(c)$ thus classified as colour samples form the sample set

$$\mathcal{S}(c) = \{\mathbf{p} \mid \mathrm{p} \notin \mathcal{B}(c), \mathrm{p} \in \mathcal{M}(c)\}$$

for colour $c$ and are represented as a distinct cluster in colour space.

The input images contain noise which means that the background filtering process is not precise. Hence, the best number of samples picked out for the following processes may not be exactly $|\mathcal{S}(c)|$. Therefore the following sample sorting method is used to improve its result.

**Colour Sample Sorting:** The system sets the initial number of samples to $|\mathcal{S}(c)|$ and then passes fine tuning of this number to the user. The number of samples can be altered by the user through a graphical slider object. The slider ranges from 0 to the number of samples in $\mathcal{M}(c)$. By changing the value of the slider to $n$, the system chooses the best $n$ samples from $\mathcal{M}(c)$.

In order to do this, the system first applies a scoring system to $\mathcal{M}(c)$. For $\mathbf{p}(x,y) \in \mathcal{M}(c)$, its score $t(\mathbf{p})$ is defined as,

$$t(\mathbf{p}) = -\sum_{h=r,g,b} [(p_h - m_h(c))/d_h(c)]^2,$$

where

$$m_h(c) = \frac{1}{|\mathcal{S}(c)|} \sum_{p \in \mathcal{S}(c)} p_h$$

$$d_h(c) = [\frac{1}{|\mathcal{S}(c)|} \sum_{p \in \mathcal{S}(c)} (p_h - m_h(c))^2]^{\frac{1}{2}},\ h = r,g,b.$$

$\mathcal{M}(c)$ is then sorted with its first element assigned the greatest (best) score and the last element the least (worst) score.

Corresponding to any positive number $n \leq |\mathcal{M}(c)|$, a best subset $\mathcal{M}'(c) \subseteq \mathcal{M}(c)$ is determined by extracting the foremost $n$ elements from $\mathcal{M}(c)$. The user adjusts $n$ by observing the classification quality on $\mathcal{M}(c)$ which is carried out instantly whenever the slider is adjusted.

**Threshold Setting:** Cube generation then becomes a matter of setting upper and lower bounds (referred to as thresholds), that defines a sub-space of $\mathcal{R}^3$ from the samples in $\mathcal{M}'(c)$ by,

$$h_{min}(c) = \min\{p_h \mid \mathbf{p} \in \mathcal{M}'(c)\},$$
$$h_{max}(c) = \max\{p_h \mid \mathbf{p} \in \mathcal{M}'(c)\},\ h = r,g,b.$$

Initally, cubes are generated for each colour independantly using separate regions $\mathcal{M}(c)$. Such an approach may result in cubes that classify their own regions perfectly but have the side effect of misclassifying pixels in other regions. To correct this, these misclassified negative samples are collected once all the colours have been trained and the central regions for all the colours, $\mathcal{M}(c), c \in \mathcal{C}$ are available by the operation,

$$\mathcal{N}(c) = \{\mathbf{p} \mid \mathbf{p} \in \mathbf{C}(c), \mathbf{p} \in \mathcal{M}(c'), c' \in \mathcal{C}, c' \neq c\},$$

and used to reduce $\mathbf{C}(c)$ to exclude some or all of the pixels in $\mathcal{N}(c)$.

**Batch Processing:** The colour training phase of the system's operation is the most user-intensive. Since one of the goals of the system is processing speed, we have incorporated a mechanism in which the trained data (cubes) can be stored for later recall in order to process similar sections that contain horizons drawn with the same colours with little or no adjustment.

Assuming that the main variation among sections is brightness and tint, the adjustment can be made by comparing the background statistics between sections. For this purpose, a small background region is stored together with the cubes. Thus, denote $(m_r, m_g, m_b)$ as the means of the background for the RGB channels, and $\mathbf{C}(c), c \in \mathcal{C}$ cubes for the trained section. Once the means of the background of the RGB channels for the current section, $(m'_r, m'_g, m'_b)$, are available, the cubes $\mathbf{C}'(c), c \in \mathcal{C}$ for the current section can be obtained through,

$$h'_{min}(c) = h_{min}(c) + \alpha_h(m'_h - m_h),$$
$$h'_{max}(c) = h_{max}(c) + \alpha_h(m'_h - m_h),\ h = r,g,b$$

where $\alpha_h$ are parameters adjusting the amount of compensation for individual RGB channels.

# CLASSIFICATION

Once the cubes have been generated, they are applied to the entire section. For each pixel $\mathbf{p}(x,y)$, its colour class is determined by,

$$s(x,y) = \begin{cases} c, & \text{if } \mathbf{p}(x,y) \in \mathbf{C}(c), c \in \mathcal{C} \\ b, & \text{otherwise} \end{cases},$$

where $b$ is the background colour.

**Fail-safe Processing:** There are some cases when the colour training process described above will not be able to distinguish adequately between colours. This has been shown to occur when pencils of very similar colours are used. Since this is a commercial system where system failure is very expensive and time consuming, a fail-safe mechanism has been built into the system that will ensure success. To operate the fail-safe, the user indicates a region and a colour that the region is not supposed to contain. The system will suppress this colour in this region.

# LINE EXTRACTION

After the pixels have been classified into colours and background, they are grouped into blobs which are then grouped into lines, from which equally spaced points are chosen. The procedures to achieve this are described in the following sub-sections.

**Blob Linking:** Pixels of $c$ are grouped into a blob if they connect to each other. Neighbouring blobs of the same colour are then linked together if the linkage is deemed to be smooth. For blobs $b_1$ and $b_2$, they will be linked if the following condition is met.

$$e(b_1,b_2) = w_1m_1 + w_2m_2 + w_3m_3 < \mathbf{T}$$

where $w_i$ ($w_i > 0$, $i = 1,2,3$, and $\sum_{i=1,2,3} w_i = 1$) are weights, $m_1 = |e_1 - e_2|$ is the distance of the closest two end-points $e_1$ and $e_2$ of $b_1$ and $b_2$, $m_2 = \max\{\alpha_1, \alpha_2\}$ is the larger angle spanned by the linking line and the central parallel line through one of the blobs, and $m_3 = d$ is the distance between the two extended central parallel lines measured in the middle between the two blobs, and $\mathbf{T}$ is the threshold determined by experiments.

After this operation, $L$ chains of linked blobs $l(i)$, $i = 1,2,\cdots,L$ are obtained which will be further processed

to produce $L$ lines. For the convenience of processing, lines are categorised into two types – horizontal and vertical.

**Skeleton Generation:** The linked blobs $l(i)$, $i = 1, 2, \cdots, L$ are then thinned. For a horizontal line, pixels from a chain are arranged as

$$\{\mathbf{s}(x_1, *), \mathbf{s}(x_2, *), \cdots, \mathbf{s}(x_{n_i}, *)\}$$

where

$$\mathbf{s}(x_j, *) = \{s(x_j, y_{x_j 1}), s(x_j, y_{x_j 2}), \cdots, s(x_j, y_{x_j n_i})\}$$

The medians of $\mathbf{s}(x_j, *)$, $j = 1, 2, \cdots, n_i$ consist of the skeleton for the blob chain,

$$\mathcal{L}(i) = \{s(x_1, y_1), s(x_2, y_2), \cdots, s(x_{n_i}, y_{n_i})\},\ i = 1, 2, \cdots, L,$$

where

$$y_j = \text{median}\{y_{x_j 1}, y_{x_j 2}, \cdots, y_{x_j n_i}\}.$$

**Point Selection:** At this stage, many pixels in $\mathcal{L}$(i) may lie close together resulting in a large data set. To reduce the size of the data set, pixels of most regular interval are selected from $\mathcal{L}(i)$. This generates data that is of a similar size to that generated by the conventional (manual) digitisation method.

# EDITING

An editing tool has been written that allows the user to modify and add information to the horizons. Some of the modification functions are listed below;

- join segments
- split segment
- delete segment
- delete point
- add point

If the interpreting geophysicist has used the same colour pencil to represent more than one horizon, the editing tool is used to disambiguate between the possible horizons. Suppose there are $n$ horizons that have been drawn with the same colour. The user indicates the location of the first $n-1$ horizons by enclosing them within a region. All remaining segments are assumed to belong to the remaining horizon.

Throughout this editing phase, the user is able to view the digitised horizons overlaid on top of the colour input image. This serves as a final on the system's accuracy.

# CONVERSION TO UKOOA FORMAT

Horizons for seismic sections are encoded using the standard oil industry UKOOA format. Each point of each extracted horizon is converted into a line in the UKOOA file that contains entries for the following fields;

- section line name
- shot point number
- shot point distance
- time (or depth)
- segment start or end marker (if appropriate)
- event (horizon) number
- segment number
- fault code (if appropriate)

A user supplied description is placed into the the UKOOA file header which is prepended to the UKOOA data.

# CONCLUSION

In this paper, we have described OASIS: an interactive, user-driven commercial computer vision system that performs automatic digitisation and extraction of coloured horizons drawn on seismic sections. On an example, OASIS was used to process a seismic section of dimensions $650mm \times 450mm$ containing 8 horizons drawn with 6 colours in approximately 8 minutes. Using the batch mode, subsequent similar sections of the same size can be processed in approximately 5 minutes.

# ACKNOWLEDGEMENTS

This work has been carried out at the Turing Institute and funded by Enterprise OIL PLC. The authors would like to thank Peter Mowforth, Dave Rhodes, Sue Baxter and Mike Whyatt for their valuable contributions and enthusiasm.

# References

[1] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis.* Wiley, New York, 1973.

[2] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, Orlando, 1982.