Interpreting Images of Line Drawings

Sergey Ablameyko and Tony Pridmore

A Tutorial presented at the 16th International Conference on Pattern Recognition (ICPR), Quebec City, Canada, 11th August 2002
Preface

While development continues, multimedia tools for planning and recording the results of work on complex engineering products and projects are now widely available. These tools can significantly improve communication within project teams but suffer from an input bottleneck: most of the necessary 3D and other product/design information is readily available, but is typically in the form of paper documents, particularly drawings. Manual input of drawings into CAD, GIS and other systems is a possibility, albeit a slow and expensive one. This tutorial will focus on techniques for the interpretation of images of line drawings. It will cover the low level processes involved in and issues to be addressed during the segmentation and geometric description of line drawing images, consider the extraction of intermediate level entities (e.g. text, dimensions, crosshatched areas and physical outlines) and present and discuss current techniques for ground-truthing and performance evaluation.

Prof. Sergey Ablameyko is Head of the Image Processing and Recognition Laboratory and Deputy Director of the Institute of Engineering Cybernetics of the National Academy of Sciences of Belarus.

Dr. Tony Pridmore is Senior Lecturer in Computer Science, School of Computer Science and IT, University of Nottingham, UK, where he is a senior member of the Image Processing and Interpretation Research Group.

Prof. Ablameyko and Dr. Pridmore have a combined 20 years experience of line drawing image interpretation. They have published some 100 papers in the area (independently and together) and several books, most notably S. Ablameyko & T.P. Pridmore, "Machine Interpretation of Line Drawing Images" (Springer-Verlag, 2000).
Interpreting Images of Line Drawings: A Tutorial Introduction

Sergey Ablameyko\textsuperscript{1} and Tony Pridmore\textsuperscript{2}
\textsuperscript{1}Institute of Engineering Cybernetics of the National Academy of Sciences, Minsk, Belarus
\textsuperscript{2}School of Computer Science and IT The University of Nottingham, Nottingham, UK

Motivation

\begin{itemize}
\item Line drawings are used in a wide variety of disciplines to record, communicate and test ideas
\item Many who work with drawings also use computers that support the creation, manipulation and display of complex graphics
\item The full potential of the technology cannot be realised if the drawings to be manipulated are stored on paper
\item We need a way of extracting graphical information from paper drawings
\end{itemize}

\begin{itemize}
\item engineering drawings $\rightarrow$ CAD
\item maps $\rightarrow$ GISs
\item sketches and diagrams $\rightarrow$ Computer-supported Cooperative Work
\end{itemize}

One approach is via the interpretation of images of line drawings.
The Interpretation Problem

- How can high-level symbolic descriptions be extracted from images?
  - dimensioning and cross-hatched areas from engineering drawings?
  - contour lines and roads from topographic maps?
  - rows of houses from cadastral maps?

- The ideal system would be fully automatic, but a semi-automatic system would be acceptable in most domains

- Drawing standards vary dramatically; successful interpretation requires considerable knowledge of drawing type

- Line drawings can be very large, a successful system must be computationally efficient

This tutorial will focus on the interpretation of images of engineering drawings and maps

Engineering Drawings and Maps

- Mixed, possibly overlapping text and graphics

- Multiple object types: lines, symbols and regions

- Objects can appear anywhere, in any orientation and at any scale

- The original drawing could be coloured

- Poor quality originals lead to noisy images

- Images are large but key features can be very small
Five Stages of Line Drawing Interpretation

- The interpretation problem varies, but can be thought of as successive transformation from low to high level representations.
- Five steps are commonly recognised and occur in some form in most drawing interpretation systems.

The rigid, linear architecture implied by the figure is common, but not necessary.

Intermediate and Target Representations

- Images
  - colour, grey level or binary
- Vectors
  - unstructured collections of simple geometric primitives
  - describe either the contour or skeleton of each line
Intermediate and Target Representations

- Universal Drawing Entities
  - each drawing standard specifies a set of entities from which an acceptable drawing must be composed
  - entities do not necessarily represent specific objects (e.g., screws or cogs) but form the lexicon with which objects are described

- Two-Dimensional Objects
  - particular gearboxes and drive assemblies in mechanical drawings
  - particular rivers and village layouts (e.g., linear villages)

- The border between entities and specific objects is not always clear; is a roadbridge over a river an entity or a scene? Strictly, if it's not specified as an entity in the drawing convention, it's a scene.
Intermediate and Target Representations

Three-Dimensional Shape and Semantics
- 3D data is obtained either from the semantics of the drawing 
  e.g. Relief maps may be built from annotated contour lines 
  or from multiple projections, as elsewhere in computer vision

3D recovery from images is a complex process and remains an active research area
- It needs reliable vector and entity descriptions

The Tutorial

- Introduction
- Segmenting Line Drawing Images
- Vectorisation and Recognition of Geometric Primitives
- Entity Extraction
- Performance Evaluation Issues and Techniques

Examples will be drawn from map and engineering drawing interpretation, but (most of) the techniques described can be applied to (almost) any drawing type
Segmenting Images of Line Drawings

- Introduction
- Segmenting Colour Drawings
- Thresholding Grey Level Images
- A Taxonomy of Thresholding Techniques
- Document Image Statistics
- Some Effective Thresholding Algorithms
- Reducing Noise in Binary Images

Introduction

- Drawing images can take a number of forms
  - some maps are coloured, and scanned in colour → colour images
  - most drawings are monochrome and many scanners have the option of recording intensity data → grey level images
  - some drawings can be scanned directly to binary → binary images
- The goal is to separate ink marks and plain paper to produce a binary image; a distinct noise removal stage is often required

  Colour → Segmentation

  Grey level → Thresholding → Noise Removal

  Binary

After a brief look at colour drawings, attention will focus on thresholding grey-level images and removing noise
Segmenting Colour Drawings

- Many map types comprise several different coloured layers, but printed on separate sheets, so separation is trivial.
- Some coloured images do arise, and must be segmented.

- Maps use comparatively few colours that are designed to be easily separated.
- Colour segmentation is done by finding clusters of similar colour in a training set, then classifying each input pixel as belonging to one cluster.

---

- Interactive correction is provided by allowing the operator to add/delete clusters to/from the colour space.
- This improves performance on lower quality images.

Most drawings are monochrome, and most scanners produce grey level data, so thresholding is the most common form of segmentation used in line drawing interpretation.
Thresholding Grey Level Images

Let A \( a_{ij} \) be a grey level image and \( b_0, b_1 \) be a pair of individual grey levels. Thresholding produces a binary image B by generating a new binary pixel \( b_{ij} \) from each pixel \( a_{ij} \) of A as follows:

\[
\begin{align*}
    b_{ij} &= b_0, \text{ if } a_{ij} \leq t_y \\
    b_{ij} &= b_1, \text{ if } a_{ij} > t_y
\end{align*}
\]

where \( t_y \) is an appropriately selected threshold value.

This assumes that A may be completely represented by two sets of disjoint regions; objects (pen strokes) and background (plain paper).

Many thresholding schemes have been developed; the common issue is the identification of appropriate values of \( t_y \).

The nature and properties of the threshold(s) employed are often used to classify binarisation techniques.

A Taxonomy of Thresholding Techniques

Global vs Local

- in **global** thresholding, \( t_y = T \), object and background intensities must be consistent over the image for this to be effective.
- in **local** thresholding, the image is divided into a number of regions, and each is assigned a (potentially) different threshold.

Fixed vs Adaptive

- in **fixed** thresholding, user supplied threshold value(s) are applied regardless of image properties or contents.
- **adaptive** threshold values are defined as functions of the image region under consideration, and computed automatically.
A Taxonomy of Thresholding Techniques

- Static Adaptive vs Dynamic Adaptive
  - in a **static adaptive** method the threshold value depends only upon the distribution of grey levels within the region under consideration; no information regarding the properties of other regions is included.
  - in a **dynamic adaptive** method the threshold applied to a given region is a function of the position of that region relative to other regions having particular local properties

- Point-dependent vs Region-dependent
  - **point-dependent** methods are those which rely solely upon the grey level of each pixel
  - **region-dependent** methods also measure and exploit other image properties, e.g. higher-order derivatives of intensity

---

A Taxonomy of Thresholding Techniques

```
Thresholding
Local       Global
  Fixed      Adaptive  Fixed  Adaptive
Static      Dynamic   Static  Dynamic
  Point  Region  Point  Region  Point  Region
```
Document Image Statistics

- The most common approach is adaptive using grey level histograms
- Classic methods assume a bimodal histogram with two clear peaks
- In line drawing images the object peak is small to non-existent
- The area between 'peaks' is also flat; there is no reliable minimum
- It's tempting to think that any value in the middle will be OK; it's not
- Small changes in threshold greatly affect the detail in the binary image.

Some Effective Thresholding Algorithms

- Some algorithms have been designed specially for line drawing images, other general methods have been found to work well on them
- Algorithms widely used within the community include:
  - Otsu
  - Kittler and Illingworth's (1986) Minimum Error Thresholding
- Dunn and Joseph's (1988) algorithm is simple and effective
  - local/adaptive/static/point-dependent
  - explicitly assumes that the white peak is well defined but the dark peak is so small that it cannot be reliably located
  - compute histograms over 64x64 pixel sub-images, then smooth the histograms by local averaging to remove noise
  - locate the peak to +/- one grey level by seeking the modal grey level $g_m$
Effective Thresholding: Dunn & Joseph

- estimate the noise level of white paper by measuring the half-width of the white peak at half its height
- half-width = $1.2\sigma$ for a normally distributed peak of std. dev. $\sigma$
- threshold is set at $g_m - 3.6\sigma$
- half-widths can be averaged over sub-images to further reduce noise

Reducing Noise in Binary Images

- Noise can appear in several forms
  - isolated pixels of the opposite value salt-and-pepper noise (a)
  - small holes in objects/small spots on background areas (b)
  - distinct lines merging and single lines splitting (a,b)
  - contour protrusions and intrusions (c)
Reducing Noise in Binary Images

- Logical masks
  - simplest form of noise reduction
  - tuned by hand to specific situations
  - logical tests are performed on the neighbourhood of pixel $a_x$, the value of $a_x$ may be changed as a result
  - e.g. change $a_y$ if the number of black/white pixels in the 8 pixels surrounding $a_x$ (its 8-neighbourhood) is above some threshold

Reducing Noise in Binary Images

- Many (more principled) techniques are based upon the Distance Transform and/or Mathematical Morphology

- Distance transforms estimate the gap between each pixel in one set, e.g. B (black pixels) and the nearest pixel in another set, e.g. W (white)

- The resulting grey level image is known as a distance map

- The distance map can be computed for both B and W, W pixels are usually assigned negative distance values

- Any distance measure can be chosen, Euclidean distance is ideal but expensive so digital approximations (e.g. city block) distance are commonly used
Reducing Noise in Binary Images

- Simultaneous application of DTs to black and white pixels allows object protrusions and intrusions can be removed at the same time
  - compute distance map for both object (+ve) and background (-ve)
  - apply (separate) thresholds to the object and background regions of the distance map: set sub-threshold pixels to 0. This will create a channel of 0s between +ve and -ve regions
  - apply a reverse distance transform, propagating distance information into the zero channel

Reducing Noise in Binary Images of Line Drawings

- Distance Transforms (or morphology) can be used both to smooth boundaries and remove salt-and-pepper noise (a, b below)
- Distance transforms (or morphology) can also remove small holes, though the amount of smoothing needed to remove the blob and hole in b (below) may distort other areas of the image
- Logical masks can be targeted, avoiding widespread effects (b,c)
- Given medium-sized noise regions, logical tests may be applied to a contour description
Raster to Vector Transformation

- Contouring
- Skeletonisation
- Approaches to Vectorisation
- Global and Local Algorithms
- Direct Vectorisation

Contouring

- Extracts descriptions of the boundaries of black (or white) regions
- There are two common types of contouring algorithms: line following and scan-line

**Line following algorithms:**

1. raster scan the image looking for a start point
2. examine some neighbourhood of the current position (a pixel or pixel boundary), searching for candidate locations that might be included in the contour
3. select a candidate and go to 2
4. terminate when no unprocessed start points are left
Contouring

- **Scan-line algorithms:**
  1. examine a few lines of the input image at a time
  2. determine which of a number of possible situations is present in the stripe and select a method of resolving that situation

- These algorithms often work with run-length encoded images, using relations between strips of black and/or white pixels on adjacent rasters

- new object
- object continues
- object splits
- objects merge
- object ends

Skeletonisation

- The goal is to identify a set of pixels $S$ which comprise the **skeleton** of the set of black pixels $B$

- $S$ should:
  - be (8-)connected
  - have the same number of connected components as (preserve the topology of) $B$
  - be centred within $B$

- Each pixel in $S$ should be labelled with an accurate estimate of its distance from $W$

- It should be possible to reconstruct $B$ from $S$

**Skeletons are usually obtained via thinning or the medial axis transform**
Thinning

- Thinning is an **iterative shrinking process**
  - on each iteration, each object boundary pixel is analysed and, if certain removal criteria are satisfied, deleted (i.e. turned white)
  - a pixel survives or not as a function of the configuration of pixels in its local neighbourhood
  - removal continues until no further pixels satisfy the removal criteria
  - the remaining pixels should form a unit width string, though the precise form of this string varies with the removal criteria

- Pixel removal criteria are expressed as 2D masks: templates **partially** specifying neighbourhoods from which the central pixel should be removed

```
  0 * *
  0 1 1
  * 1 *
```

```
  0 0 0
  * 1 *
  1 1 *
```

```
  0 0 0
  * 1 *
  * 1 1
```

(0 or 1 means that pixel must be 0 or 1; * means it can be either)

---

Thinning

- Thinning algorithms may be **parallel** or **sequential** in nature

- In **parallel thinning algorithms**
  - pixels survive or are deleted on the basis only of the results of the previous iteration.
  - contour pixels are examined independently and in parallel and, if appropriate, modified independently and in parallel

- In **sequential thinning algorithms**
  - contour pixels are examined in some predetermined order
  - deletion or survival of a given pixel is again a function of its neighbourhood, but some members of that neighbourhood will already have been considered and perhaps modified during the current iteration, while others will not.
Thinning

- A Thinning Algorithm for Line Drawings (Ablameyko et al 1994)
  - takes a scan-line approach, scanning the image once
  - 3 x 3 masks mark pixels for removal
  - the stripe is w+3 lines high (w is the expected maximum line width)
  - lines within the stripe are numbered, bottom to top, thinning proceeds from the first (bottom) to the last (top) line
  - pixels are processed a varying number of times, depending on their vertical position in the window. In one application of the window, pixel runs in the top line are thinned completely while those in the bottom line remain unchanged. Central lines are partially thinned.

Thinning

- each line in the stripe has two descriptions: a primary, run-length description and an auxiliary, pixel array representation
  - the first allows easy separation of black segments requiring processing from white and completely thinned runs
  - the second records thickness and forms representations of pixel neighbourhoods that can be matched to a look-up table of 3 x 3 masks
  - the pixel representation also records which pixels are marked for deletion, allowing parallel and sequential operations to be combined
The Medial Axis Transform

- Medial axis transforms use distance maps to produce a skeleton
- They generally provide more accurate line width estimates than thinning
- The basic idea is that each object can be represented by a set of maximal discs

A maximal disc is a circle which fits over the input object and is not completely overlapped by any other such disc

- The medial axis transform is defined by the centres of the maximal discs and their radii
- It is a compact representation, though the discs can overlap significantly

The medial axis is usually thin, but is not guaranteed to be one pixel wide

Removing Noise from Thin Data

- Thinning algorithms and medial axis transforms both leave small defects and noise in the skeleton
- Preprocessing the binary image can reduce, but not remove, this
- Noise can be removed from the skeleton
  - avoids vectorising spurious data and reduces the risk of poor vectors
  - but the image must be inspected several times, so it can be expensive
- Removing noise from the vector representation may be easier; vectorisation adds information, so erroneous data is more readily identified
- All the noise removal methods in common use are "empirically derived"

Noise is removed from the skeleton by a process known as "pruning"
Removing Noise from Thin Data

- Several types of skeleton defect are commonly observed:
  - small gaps between the end points of successive segments
  - additional short branches attached to skeleton nodes
  - neighbouring nodes connected by implausibly short arcs
  - spurious, free-standing short loops
  - spurious isolated segments

- **Pruning** may involve shortening or deletion of a branch

- **Logical tests** are applied to representations of e.g. branch length, area and/or elongation, and changes made as a result

Approaches to Vectorisation

- **Vectorisation systems** segment and describe thin data

- **Global methods** have simultaneous access to the entire pixel string

- **Local methods** access thin data sequentially, taking a **line following** or **scan-line** approach and constructing a vector description as they go

- Global methods may be **iterative**, **feature-based** or **hybrid**
Global Vectorisation

- **Iterative methods** are derived from the Ramer's (1975) "iterative end-point fit algorithm"
  - compute the parameters of a straight line between the end-points of the input pixel chain
  - compute some measure of the goodness of fit between the proposed line segment and the original data
  - if the error level is below some threshold, accept the proposed line
  - if the proposed line is unacceptable, segment the pixel string at the point furthest from the line segment and apply the algorithm to each of the resulting substrings.

- **Feature-based Methods**
  - identify points at which the local properties of the data suggest a discontinuity in the underlying curve
  - geometric approximation is applied, later, to the substrings linking pairs of discontinuities
  - discontinuities are usually identified by consideration of local curvature, measured as a function of arc length
  - some systems also seek extrema in the 1st derivative of curvature
  - noise is often reduced by weighted averaging before curvature is estimated

- **Hybrid systems** adopt the iterative end point algorithm structure, but choose a segmentation point on the basis of local curvature, not relative to the current line segment
A Local Vectorisation Algorithm

- Adopts the scan-line approach because of its high speed and limited memory requirements
- Each raster is represented by an ordered list of the x co-ordinates of black (white) pixels, which have white (black) pixels on their left
- Input is a labelled skeleton in which every run contains a label corresponding to the thickness of the original image object
- Two basic processes, situation extraction and segment following, performed simultaneously on three image lines held in a buffer
- Situation extraction detects: line beginning, end of line, continuation, lines merging, lines branching, a node, or an isolated point

As soon as the context of the first black pixel is determined, segment following records segment relations and prepares segment information for inclusion in a vector database
- Segments are traced during the downward motion of the stripe
- Special buffers (one per segment) are initialised, merged and/or split during scanning depending upon the situation encountered
- Each image object is divided into sub-objects at its feature points and ultimately represented as a graph.
Direct Vectorisation

- All the processing discussed so far has been bottom-up.
- An alternative approach is to accept that higher-level knowledge is required even in the early stages of interpretation and work top-down.
- Joseph's (1988) direct vectorisation method traces over the image, extracting vectors without producing any intermediate representations.

Initial circular search produces 2 spots

1st track

2nd track

Transverse track provides further black steering spot

- Basic operation is to test sequences of pixels and, while they satisfy a blackness criterion, extend the track.

Direct Vectorisation

- The system only examines pixels on or near a hypothesised line, so the computations performed are both simple and fast.
- As the tracker focuses on one line at a time it is naturally insensitive to the distractions posed by junctions and other intersecting linework.
Entity Extraction

- Introduction
- Extracting Entities from Vectorised Engineering Drawings
- Direct Extraction of Engineering Drawing Constructs
- From Engineering Drawings to 3D Objects
- Recognising Map Objects
- Interactive Map Interpretation

Introduction

- Vector representations cannot be used directly in e.g. CAD or GIS systems
- Vectors and other low-level primitives must be combined to form high level entities and/or signs
- We will consider entity extraction from engineering drawings and maps
- Two contrasting approaches to engineering drawing interpretation will be presented

The first question is how can entities be extracted from vectorised engineering drawings?
Extracting Straight Lines and Arcs

- Identify individual graphical primitives, **ignoring** any relations between them
- Join and align primitives, **exploiting** the relations between them

- produce a file of relations between graphical primitives (except those located at the start and end points) and a list of adjoining primitives
- straight lines that are nearly horizontal or vertical are made exactly horizontal or vertical
- adjoining primitives are joined

Extracting Dashed Lines

- A two-stage process
  - look for sequences of segments which have the local properties expected of a dashed or a dot-dashed line
  - seek an extension of each line, resolving ambiguous cases (processing gaps, nodes, etc.)

- A line segment bounded by at least one end point is chosen as the starting segment of each dashed line
- The assembly of a chain begins at the end point
- When the discontinuities of all pure chains have been analysed, an attempt is made to combine the resulting complex graphic primitives. To be combined these primitives must be colinear.
Recognising Crosshatched Areas

- Some guiding principles
  - use a priori knowledge of document type
  - seek groups of parallel lines in the raster image of the document
  - form, and analyse the attributes of, closed areas in the vector data

- Another two-stage process
  - hypotheses are formulated as to whether line segments belong to an area boundary or an internal line based on a local analysis of relations between segments
  - final assembly of areas then takes place, taking into account the pure chains assembled during the first stage

Dimensioning and Text

- Most types of dimensioning incorporate text and arrowheads
- Dimensioning text must first be separated from graphics and arrowheads must be identified
Recognising Dimensions

- Dimension recognition relies heavily on prior knowledge of dimensioning style
- A search for an initial dimension component is followed by parsing in compliance with a formal language used to describe dimensions
- During parsing, hypotheses are consecutively advanced and tested for acceptance or rejection. Spatial and logical relations between dimension components are used to formulate hypotheses.
- Hypotheses are accepted or rejected upon reference to template descriptions of dimension components held in the knowledge base. Acceptance of a hypothesis implies that parsing should continue at the next lower level.

Direct Extraction of Engineering Drawing Constructs

- Anon (Joseph and Pridmore 1992)
- Cyclic architecture
- Drawing entities represented by object/schemata hierarchies
- Each schema contains an entity description and indices to low level analysis tools
- Control system is an LR(1) grammar applied via a YACC-generated parser
An Example: Chained Lines

• Grammar rules aggregate and classify low level primitives

[Rule 1] broken_start: line BREAK LINE [broken_start.instantiate()];
[Rule 2] broken_start:broken_start BREAK LINE [broken_start.addon()];
[Rule 3] chain: broken_start ISCHAIN [chain.instantiate()];

Chained Lines

• Image search varies with current schema

[Rule 4] chain: chain BREAK LINE [chain.addon()];
Some Results

After User Intervention

- In automatic mode Anon generates initial searches randomly, these can miss small constructs.
- The user can (only) specify start points for the grammar-driven search.
Towards Scene Formation

Later implementations of Anon merge similar objects and detect potential conflicts, but cannot resolve them

Coincidence links produce a rich structure on which to build

From Engineering Drawings to 3D Objects

1. Interactive removal of annotation (text, dimensions, sizes, symbols, etc.)
2. Interactive determination of boundary projection rectangular and extruding projection high
3. Automatic recognition of outer contour projection and its editing if needed
4. Automatic inner contours projection recognition and their editing if needed
5. Automatic project contours extruding on the corresponding high
6. Automatic subtraction of inner solids from outer solid obtained from projections
7. Automatic turn, align and interfere of solids obtained from projections
8. Visual inspection of the 3D model by the user
From Engineering Drawings to 3D Objects

From Engineering Drawings to 3D Objects
Recognising Map Objects

- Some guiding principles
  1. Combined automatic/interactive interpretation
  2. Sequential processing of map layers
  3. Proceed from the simple to the complex
  4. Make maximal use of knowledge
  5. Explicit multi-level recognition
  6. Proceed from local to global analysis

Recognising Isolines

- To recognise isolines, attribute grammars are used
  \[ G = \{ V_n, V_t, P, S \} \]
  \[ S = \{ L_1, \ldots, L_n \} \] - set of lines under recognition.

- All \( V_t \) are divided into two groups: **constructive** and **connecting**
  - **constructive** \( V_t \) provide the geometrical coordinates of line objects
  - **connecting** \( V_t \) link the constructive \( V_t \) to form chains

\[ V_t = V_{\text{constr}}. \cup V_{\text{connect}}. \]

\[ V_{\text{constr}}. = \{ \text{segment, point, ...} \} \]

\[ V_{\text{connect}}. = \{ \text{gap, knot, ...} \} \]

- Each \( V_t \) has some associated attributes e.g. length, thickness, etc
- \( V_n \) and \( P \) are specific for every type of line
Recognising Isolines

- Line types are specified using different combinations of \( Vt \). For example, an isoline formed by the repetition of open line segment, point and gap is described by the following rule:

\[ P : G \rightarrow sgpG \]

where: \( G \) - nonterminal element and starting symbol

- \( Vt \) have the following attributes:
  
  s (segment) =: <thickness><type><length>;

  g (gap) =: <length>;

  p (point) =: <diameter>.

Recognising Roads

1. Extract and vectorise the contour of each image object
2. Divide objects into two classes: elongated (roads) and not elongated (buildings),
3. Extract the middle lines of the elongated objects
4. Record the results information in the output database
Recognising Roads and Correlated Objects

- Input to road recognition is the \( Mv(r) \) model obtained from the previously processed road layer and the vectorised model \( Mv(b) \) of the black layer. \( Mv(r) \) contains road skeletons while \( Mv(b) \) describes both sides of the roads.

Recognition of Area Objects

- Texture and black objects on maps
- The main feature of textured regions is that the frequency of texture elements (TEs), their regularity and the borders of textured regions are of great importance, while the form of the TE is unimportant.
- A distance transform based algorithm is used to extract texture borders
  1. Apply a DT to the image background; background pixels receive negative values
  2. Reset the pixels in the distance map with values between 0 and \( h \) to 0 where \( h \) is a chosen threshold (producing a zero channel between black pixels and the \( h \)-wave.)
  3. Compute the reverse DT for the channel in the inward direction with the \( h \)-wave. The sign is also expanded.
Recognising Symbols

- Three types of information are used for symbol recognition
  - raster object representation
  - contour representation (shape)
  - skeleton object representation

- A set of features is usually used to recognise symbols

- The feature vector could include:
  1) Shape features: circularity, number of holes, number of sides, number of lids, side chain length, side cycle length, etc.
  2) Skeleton features: number of segments, number of feature points, location of feature points, symbol width and height, segments' geometry, etc.

The Role of Knowledge

- There are two main benefits of using knowledge in a map interpretation system
  - exploiting knowledge in the analysis of complex situations increases the proportion of objects that can be recognised automatically
  - representing knowledge in a separate knowledge base can make the system applicable to a variety of map types and scales

- A priori and a posteriori (virtual) knowledge
  - A priori knowledge reflects human perception of maps and is represented as expert knowledge in a concrete problem field. It includes any and all information that can be extracted from the map before processing and stored in the knowledge base.
  - Virtual knowledge appears and is accumulated during the interpretation process and comprises real information about objects, primitives, and points. Virtual knowledge can be also be thought of as cartographic data obtained during the interpretation process.
Interactive Interpretation of Vectorised Maps

- A modified automatic object recognition procedure is applied under operator control. It comprises two main tasks:
  - automatic segment analysis and object tracing. The algorithms used are approximately the same as those used in automatic recognition but employ less rigid constraints
  - interaction with the operator in places where the program cannot make a reliable decision
- Several forms of elongated object extraction have been realised. For simplicity they are referred as Draw, Pick, Go, Run, and Jump modes.

Interactive Raster/Vector Interpretation

- Comprises the following stages
  - rough extraction of object metrics
  - manual input of object characteristics
  - object recognition exploiting both automatically extracted vectors and the results of the previous stages.
- Rough metric information is obtained using either a manual digitiser or a mouse and the screen of a standard monitor
- Object characteristics are input via a keyboard
- Correspondence is then established between the metric information and vectors and the rough metric information is refined.
Examples of Digital Maps

From top left to bottom right: an initial map (it was originally colored), the digitized black layer, the hydrography layer, the isoline layer, the road layer, and the forest layer.
Performance Evaluation and Conclusion

- Performance Evaluation
- Current and Future Issues
- Conclusion

Performance Evaluation

- As the field matures, performance evaluation has become a key issue in the wider field of image analysis and machine vision
- Performance evaluation is an important step in line drawing interpretation system development
- Appropriate and systematic evaluation
  - makes explicit the strengths and weaknesses of proposed techniques
  - avoids redundant effort
  - plots the advancement of the field

To date, many drawing interpretation systems have been evaluated either qualitatively or quantitatively on limited test sets which are often not widely available.
Performance Evaluation

- Performance evaluation of line drawing interpretation systems usually comprises two stages:
  - Interactive **ground truthing** of a set of test images
  - **comparison** of the results of interpretation with the ground truth data

- Comparison requires a **distance metric** to be defined

- Standard test sets exist for e.g. OCR, and there is clear agreement as to what correct character recognition is

- Line drawings are much more complex and open to varying interpretation by human observers

**Performance evaluation of line drawing interpretation systems can be very subjective**

See Lopresti and Nagy, GREC 2001
Performance Evaluation

Many questions remain:
- what input should be used?
- who should do the evaluation?
- what model should he/she use?
- how should the model be represented?
- should the entry system be "dumb" or have knowledge of the model?
- should the evaluation function emphasise correct description or correct classification?
- should it be domain-specific or more generic?
- how do you evaluate a complex, multi-stage interpretation system, at the component or system level?

**Performance evaluation is one of several issues attracting attention within the drawing interpretation community**

Current Issues

1. **Improved image analysis** - e.g. thresholding with hysteresis
   - most thresholding algorithms use a single value
   - elsewhere Canny's two-level thresholding is a de facto standard
   - accepts pixels as black if they're below a very low threshold
   - pixels above low threshold are also black if they're below a high threshold and connected to a pixel below the low threshold
   - designed for 'thin' data, it can be applied to line drawing images using idempotent morphology (Pridmore 2001)

   **Low level drawing interpretation tools are well-developed, but there is always room for improvement**
Current Issues

2. **Better understanding of the role of knowledge**
   - drawing interpretation systems need a priori knowledge
   - despite developments in knowledge engineering (e.g. KADS/CommonKADS) most drawing interpretation systems employ unstructured 1st generation KBS methods
   - Pridmore, Darwish & Elliman (2000) analysed well-known systems from a KE perspective
   - goal was mappings between existing systems and KADS/CommonKADS models of expertise, a la Clancey (1983)

3. **Better understanding of the role of the user**
   - interactive systems exist, but none are truly co-operative at the higher levels of interpretation
Future Issues?

- Richer and more generic intermediate representations
  - there is a large representational (semantic) gap between vectors and drawing entities
  - systems like Anon use domain-dependent object hierarchies
  - intermediate entities are not extracted independently, but during higher level entity extraction
  - is there a role for domain-independent intermediate representations? E.g. Saund's (2000) minimum description length perceptual grouping approach?

Much has been done, much remains to be done

Conclusion

- Line drawing interpretation is a difficult and time-consuming process
- Digitized line drawing should include maximal possible object characteristics and, at the same time, it should be stored in minimal possible data volume
- There are no completely successful, fully automatic line drawing interpretation systems in the world
- Line drawing interpretation technology consists of two parts: automatic image vectorization and automatic/interactive object recognition
- Interactive techniques must be user-friendly and make use of available automatic recognition elements
- For objects to be input there should be found a compromise between automatic and interactive techniques to provide the required processing time and drawing input accuracy
Bibliography

As line drawing interpretation systems and techniques have developed over the years many research papers have been produced. Several excellent bibliographies and edited collections of those papers now exist and provide a valuable route into the wider literature.

Papers on the interpretation of document or line drawing images now appear regularly in the image processing, image analysis and computer vision literature. Since the early 1990s, issues of several international journals have been devoted to line drawing interpretation. Special issues of the Computer Magazine (July 1992), Machine Vision and Applications (1992, 1993) and the International Journal of Pattern Recognition and Artificial Intelligence (1994) describe both systems and techniques for line drawing interpretation. A new journal devoted to the field, the International Journal on Document Analysis and Recognition appeared for the first time in 1998. There are now regular conferences and workshops in this area; descriptions of many drawing interpretation systems and methods can be found in the proceedings of the:


The items listed below represent only a small fraction of the literature on line drawing interpretation and its component operations. In putting this list together we have simply tried to provide those interested in the area with a starting point; there are many other excellent publications in the area. We hope you enjoy finding and reading them.

Books


Overview Papers


**Thresholding**


**Vectorisation**


**Engineering Drawing Interpretation Systems**


**Map Interpretation Systems**


Performance Evaluation


Ablameyko, S., Belarusian Academy of Sciences, Minsk, Belarus
Pridmore, T., University of Nottingham, UK

Machine Interpretation of Line Drawing Images
Technical Drawings, Maps and Diagrams
3-540-76207-8
£ 45.00 DM 139.00 FF 524.00 öS 1015.00 SFr 126.50
Lira 153,520

Line drawing interpretation is a challenging area with enormous practical potential. At present, many companies throughout the world invest large amounts of money and human resource in the input of paper drawings into computers. The technology needed to produce an image of a drawing is widely available, but the transformation of these images into more useful forms is an active field of research and development.

Machine Interpretation of Line Drawing Images
- describes the theory and practice underlying the computer interpretation of line drawing images and
- shows how line drawing interpretation systems can be developed.
The authors show how many of the problems can be tackled and provide a thorough overview of the processes underpinning the interpretation of images of line drawings.

FOR YOUR COPY, PLEASE COMPLETE THE ORDER FORM OVERLEAF

www.springer.co.uk
www.springer.de