Edge and Boundary Interpretation

Consistent line drawing labeling via backtracking

Project Report

Introduction to Computational and Biological Vision,
Computer Science Department, Ben Gurion University,

Guy Shtub
1. Introduction
This paper deals with line labeling of Trihedral (i.e. having been formed by three planes meeting at a point) blocks world images.

1.1 Edge labeling in the Trihedral block world
In the 1970’s Huffman and Clowes researched methods to teach a computer to recognize trihedral images as 3d scenes.

It is assumed that the image represents a general viewpoint. A viewpoint is general if a small perturbation of it would not affect the configuration of the line drawing. Line labeling means classification of edges in the target image corresponding to depth and surface discontinuities. Illumination and color discontinuities are ignored. Edge types include:

![Diagram of edge types]

And the corresponding edge types symbols:

![Diagram of edge symbols]

Edge intersections can be of 4 types:
There are only 16 possible edge intersections [1,2]. Each of the possible options is numbered.

A computer can use this knowledge of which different combinations of edge intersections is possible to map out an image, thereby determining what objects it contains. This process is a first step for computers towards understanding 3d dimensional space.

One method of labeling lines according to the Huffman and Clowes catalog is Backtracking.

1.2 Backtracking Algorithms
Backtracking algorithms attempt to solve Constraint Satisfaction Problems (CSP). CSP are problems that consist of a set of variables each of which must be assigned a value from a possible domain. Backtracking attempts to find a solution by trying all the combinations in order to obtain a solution.
Backtracking can be implemented in several ways. Prosser investigates and compares different backtracking algorithms [4].

1.2.1 Naïve backtracking
In naïve backtracking variables are incrementally given values from their domains. Whenever a variable X is given an assignment, the assignment is checked with all previous variable assignments. If the assignment is consistent the algorithm moves on to the next variable. Otherwise a different value for X is chosen. If the variable’s domain is empty then the variable that was given an assignment before X, namely Y is given a different assignment. If the first variable’s domain is emptied then there is no solution. If the last value receives a consistent assignment then a solution is found.

1.2.2 Forward Checking
Forward Checking (FC) attempts to solve a CSP by a “look ahead” method. When the search attempts a trial assignment for variable X it removes from the domain of all other variables, all values that are not consistent with the assignment for X. In this manner when we choose a value for a certain variable, we can be certain that it is consistent with the current assignment. If after a trial assignment a domain for some variable is emptied a new value is tried for the current variable. If no values remain the algorithm backtracks to the last variable that was given an assignment. FC performs more work then Naïve Backtracking with each assignment given. This is because it “wipes out” future values. On the other hand it attempts to make fewer assignments.
Prosser [4] proves that FC necessarily performs better then Naïve Backtracking.

2. Project goals
This project’s aim is to implement line labeling according to the Huffman and Clowes catalog, by using Naïve Backtracking and Forward Checking algorithms. I aim to develop a program that given a line labeling problem returns the correct labeling, or reports that no solution exists. In addition the performance of Forward and Naïve Backtracking, in the context of line labeling problems, will be examined.

3. Course of Action
In this section I explain the problem representation, the implementation of the algorithms, and the algorithm evaluation method.

3.1 Problem Representation
For a given target image each edge intersection is a variable. The domain size for each variable is 16, according to the Huffman and Clowes catalog. For each image with N variables, a NxN constraints matrix is built. This matrix represents which variables are constrained, i.e. which edge intersections are adjacent. Value 0 represents no constraint while value 1 represents a constraint. For example if variables 1 and 3 represent neighboring edge intersections then the value of the matrix at position [1][3] = [3][1] = 1. This is because the constraints are symmetrical. An additional conflicts matrix is built for each value 1 in the constraints matrix.
This conflicts matrix is of size 16x16. In this matrix the value in [X][Y] indicates if domain value X has a conflict with domain value Y. For example assume that variables 3 and 6 represent adjacent neighbors, therefore the value [3][6] = 1 in the constraints matrix, and they have a conflicts table. In this table, if [4][14] = 1 then the domain value 4 for variable 3 is not consistent with domain value 14 for variable 6. Performing a check if variable A with value X is consistent with variable B with value Y includes checking constraints[A][B]. If constraints[A][B] = 0 return true. Otherwise constraints[A][B] = 1, then check the A-B conflicts table. If conflicts[X][Y] = 1 return false, else conflicts[X][Y] = 0 and return true.

3.2 Algorithm implementation
Both algorithms were implemented in Java. The implementation includes 4 classes. Problem class represents the problem. Table class represents the data structure used to build problems. AlgoForwardChecking and AlgoBackTracking classes implement the respective algorithms.

3.3 Evaluation Method
In evaluation of Backtracking algorithms a Constraint Checks (CCs) measure is commonly used. This measure counts the number of times the algorithm checks the problem to see if two variables and their values are consistent. The advantage of using such a measure is that it is not implementation dependent. This means that the algorithm can be implemented in different programming languages and run on different machines, but the measure would still be valid. In the code the class Problem includes a counter that is incremented whenever the function “check” is called.

4. Results
The program was tested on 3 different problems. For each problem, the edge intersections were arbitrarily numbered. The domain for each variable was 0-15, each number representing the respective edge intersection type from the numbered catalog (section 1.1). The received results are in form of pairs, where the first element is the edge intersection number and the second is the domain value assigned to this variable.
4.1 Pyramid Problem
The target image:

The received result:
Assignment=<0,2>,<1,9>,<2,2>,<3,9>

4.2 Test Problem
The target image:
The received result:
Assignment=<0,2>,<1,9>,<2,7>,<3,11>,<4,2>,<5,11>,<6,6>,<7,7>,<8,11>,<9,9>,
<10,2>
4.3 Block Problem

The target Image:

![Image](image1.png)

The received result:

Assignment=<0,9>,<1,2>,<2,9>,<3,2>,<4,1>,<5,12>,<6,9>,<7,2>,<8,9>,<9,7>,<10,2>

4.4 Number of CCs

<table>
<thead>
<tr>
<th>Problem</th>
<th>Number of CCs – Naïve Backtracking</th>
<th>Number of CCs – Forward Checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pyramid</td>
<td>18633</td>
<td>130</td>
</tr>
<tr>
<td>Test</td>
<td>4201144</td>
<td>1036</td>
</tr>
<tr>
<td>Block</td>
<td>25228136</td>
<td>747</td>
</tr>
</tbody>
</table>
5. Conclusions
The program provided consistent line labeling for the given test problems. As expected in all tests the FC algorithm performed considerably better than the Naïve Backtracking algorithm. Future work includes comparison of these results to consistent line drawing labeling via relaxation labeling, as well as other methods.

6. Appendix - Program Output on test problems
run:

Assignment=<0,2>,<1,9>,<2,2>,<3,9>
BACKTRACKING ans for pyramid problem = solution
Number of Checks: 18633

Assignment=<0,2>,<1,9>,<2,2>,<3,9>
FC ans for pyramid problem = solution
Number of Checks: 130

Assignment=<0,2>,<1,9>,<2,7>,<3,11>,<4,2>,<5,11>,<6,6>,<7,7>,<8,11>,<9,9>,<10,2>
BACKTRACKING ans for test problem = solution
Number of Checks: 4201144

Assignment=<0,2>,<1,9>,<2,7>,<3,11>,<4,2>,<5,11>,<6,6>,<7,7>,<8,11>,<9,9>,<10,2>
ans for test problem = solution
Number of Checks: 1036

Assignment=<0,9>,<1,2>,<2,9>,<3,2>,<4,1>,<5,12>,<6,9>,<7,2>,<8,9>,<9,7>,<10,2>
BACKTRACKING ans for block problem = solution
Number of Checks: 25228136

Assignment=<0,9>,<1,2>,<2,9>,<3,2>,<4,1>,<5,12>,<6,9>,<7,2>,<8,9>,<9,7>,<10,2>
ans for block problem = solution
Number of Checks: 747

7. References
3. Depth and Shape Inference (II), Introduction to Computational and Biological Vision, Computer Science Department, BGU, Ohad Ben-Shahar
4. P. Prosser: "Hybrid algorithms for the constraint satisfaction problem",