Sonar Image Processing

An Application of Template Matching Through Relaxation

Charles Thorpe

The Robotics Institute

Carnegie-Mellon University

Pittsburgh, PA 15213

9 October 1981

Abstract: This paper discusses the use of high-level templates and relaxation techniques in processing side-scan sonar images. The images examined do not have enough resolution to identify the objects individually. Distances and orientations among groups of objects are stable, however, and provide the information necessary to unambiguously identify each object. The identified targets are then used to provide navigation information for underwater vehicles. Much of the work described is an application and refinement of established techniques, most directly those of Davis. Low-level filtering, template matching, relaxation and M* search are all discussed.

This research was supported by The Robotics Institute, Carnegie-Mellon University, and, in part, by The Office of Naval Research.

Table of Contents

1. Introduction

This paper describes the use of template matching and relaxation in processing side-scan sonar images. Most of the basic ideas involved in combining template matching and discrete relaxation are refinements of the method discussed by Davis [2]. This study extends their use to processing sonar images and uses knowledge specific to the particular task. Target detection, the techniques of template matching, relaxation, and M* search, and contributions of this study are all discussed, as well as specific details and experimental results.

The system to be examined processes side-scan sonar images for an underwater channel conditioning robot. This robot will map the location of large obstacles on a relatively smooth channel bottom, then check periodically to see if any of the objects have shifted or if there are new obstacles. Sonar would serve two purposes: detecting the objects (called "targets" in the jargon), and navigating based on identifying targets as known objects on the map. In practice, the primary navigation would probably be done by an inertial system, with sonar providing backup and drift correction.

There are two basic parts to the sonar image processing. The first part is target detection: deciding which echoes are from targets and which are just noise, and reporting target positions. This is not in general possible to do completely accurately. Some false alarms will usually get through and some targets may be missed.

The second and more interesting part is target recognition. Given the location of detected targets, the system must identify each target as a particular object on the map or else classify it as a new object. Individual targets look very much alike, so identification must be based on distances and angles between groups of targets rather than on echoes from individual objects. It is here that template matching, at an abstract level, is used. If several identifications are possible, relaxation helps narrow down the possibilities. Finally, M* search picks the best identification for each target. In this case, "best" means not only the most likely for that single target, but also the most consistent for all targets.

2. Target Detection

The first task is to find the location of the targets in an image which is rather noisy by optical imaging standards. The images dealt with have been generated by a Westinghouse side-scan sonar. A side-scan sonar looks at right angles to the direction of travel of the vehicle, constructing an image row by row as the ship moves forward. The sonar looks down and out at an angle, so it misses objects directly below the vehicle but gets a good look at objects sticking up from the bottom off to the side. The images generated are 8 bit images, with about 6 bits of reliable data, and are 1000 pixels square. There is much noise, with scattered reflections off the bottom ranging over the whole dynamic scale, giving a strong salt and pepper appearance to the image. Various preprocessing filters and smoothing algorithms were used to try to clean up the image. None of them succeeded in eliminating noise without also eliminating targets.

Targets (reflections from objects) appear as bright blobs, about 5 or 6 pixels across. Each target has a long (up to 50 pixels or more) shadow trailing directly away from the sonar transducer. Length of the shadow is a function of a number of variables: height of the vehicle above the bottom, height of the obstacle, and distance from the vehicle to the object. Neither the bright targets nor the dark shadows by themselves are enough to pick the targets out of the noise. Together, however, there is enough information to do a fairly accurate job of detection.

All targets are about the same size, and all their shadows fall in the same direction (directly "behind" the target as seen from the sonar transducer). This makes it possible to convolve the image with a filter that has maximum response when it overlays a target and shadow. A relatively crude step filter, with weights of +1, 0, and -1, is adequate and can be convolved efficiently. The filter can be described graphically as

```
+++000-----
+++000-----
+++000-----
```

where the value calculated is the sum of the values of the pixels underlying the +'s, minus the values of the pixels underlying the -'s. As the filter is moved across the image, the highest values result when the bright, high-valued pixels of a target lie under the +'s, and only dark shadow with low values lies under the -'s. The 0's are necessary because there typically is a fuzzy edge between target and shadow. If the value calculated for a given point exceeds some threshold, the coordinates of the center of the + region are reported as a possible target location. Another program takes this list of points and clumps adjacent candidate points into objects, reporting row and column coordinates and number of points in each object. This process also uses a threshold, rejecting objects with only a few points as probably being noise. Aspect ratio, left- or right-side scanning, and other sensor-dependent variables are taken care of at this point.

The straightforward approach to convolving the filter with the image is computationally expensive. Each point in the image would require, for the filter shown, 9 additions and 15 subtractions. A better algorithm keeps column sums and incrementally updates the total each time the filter is shifted [3]. This cuts the convolution down to only 3 additions and 3 subtractions per pixel for an arbitrarily large filter.

3. Template Matching

Matchmaker, matchmaker, make me a match ...

Once the targets are picked out, they need to be identified as some known object on the map. A target can almost never be identified in isolation. There may be more differences between the appearance of the same object in two different images than between different objects in the same image. There are problems with noise, specularity, different distances between target and transducer, and views from different directions. Consider, for example, the difference in echoes from a "D"-shaped object when viewed from the left or from the right. Since information about single targets is so unreliable, it is necessary to use other knowledge sources. Distances and orientations between targets are in general much more reliable and much more stable from one image to the next. This information is used to drive a high-level template-matching scheme.

Template matching works by comparing an image, or parts of an image, with known objects. Classical, low-level template matching works with the digitized image directly, comparing it pixel by pixel with a template. The image and the template match if the difference between their pixels is less than some threshold.

Template matching can also be used on a higher level, matching description to description rather than pixel to pixel. In sonar image processing, the actual image is abstracted to the coordinates of the targets, and forms a template described in terms of locations, distances, and orientations. Note that the image is used as the template in this case, since it is smaller than the map. The objects on the map are described in similar terms. In fact, the map may have been constructed from a previous pass with the same vehicle and sonar imaging system. Quality of a template match is then a function of how much the template (image description) must be stressed--stretched, twisted, and shifted--in order to line up with the map. Hence the name "rubber template" for this kind of matching. A fringe benefit of this kind of map and template is that very little storage is needed; only three integers (row, column, and intensity) per target, for about 10 to 20 targets per image, rather than the megabyte of memory necessary to hold the entire image.

A match between the template and the map is an identification of each target, or a decision that some targets are not on the map. In order to make the process more efficient, template matching is done in stages. First, a rough approximation is made using only part of the information available. This usually leaves several candidate matches, that is, more than one possible map identification for each target. Then, more constraints are applied using relaxation to discard the less likely matches. Finally, the remaining template matches are evaluated and the one that induces the least stress in the template description is chosen as most likely correct.

A few things are known *a priori*. Other navigation systems can provide a rough idea of present position and a quite accurate heading. So it is possible to calculate an approximate position for each target. This means that the map objects that have to be considered as possible identifications of each template (unknown) object can be limited to those within some small distance of where the robot thinks each object actually is.

The first step in template matching constructs a list of possible identifications, each a target-map object pair. A pair is created only if the position of that object on the map is sufficiently close to the calculated position of that target, and pairs are given a weight based on how close the actual distance is. Note that any given map object or target can be part of several different pairs, since a target may be near several map objects.

The pairs only contain information about the location of single targets and map points. This usually isn't enough to unambiguously identify targets. To incorporate more information and create a framework for further processing, the list of pairs is turned into a graph. Each pair becomes a node of the graph. Information about relationships between pairs of targets and pairs of map points goes into the edges of the graph. Two nodes are linked only if the corresponding distances are within some tolerance. In other words, if T_i-M_i and T_j-M_j are two nodes (that is, two target-map pairs), they will be linked if the distance T_i-T_j between targets is about the same as the distance M_i-M_j between corresponding map points. The idea is that if the distances are inconsistent, at least one of the identifications must be wrong.

To summarize, the terms "node", "target-map pair", and "possible identification of a target" are used interchangeably and all refer to the calculated location of a target. An example would be "target at row X_1 column Y_1 in the sonar image, object on the map at X_2 Y_2", which indicates that the particular echo may be coming from a certain rock that is at a particular place on the map. Furthermore, the existence of a link between two nodes implies that both nodes could be correct identifications of their respective targets without having to stretch distances too much.

4. Relaxation

The first step of the template matching uses only some of the information about an image and will almost always come up with several possible identifications for a given target. Recall that the information in the nodes themselves has only to do with calculated position of the targets. Relaxation is applied to use information from the links between nodes (that is, distances between pairs of targets and between corresponding pairs of map points) to delete nodes.

In relaxation the value of each node in a graph is a function of the values of its neighbors. For example, we can derive the temperatures at points inside a metal bar by modeling the bar as a lattice graph with known temperatures at the surface. A first approximation to the temperatures at any given point is the average of the temperatures of its neighbors. Successively finer approximations may be calculated by averaging the updated values of the neighbors until the change from one iteration to the next is sufficiently small.

The relaxation used in template matching is "discrete" rather than "continuous" as in the example above. Rather than a node's value being increased or decreased, the node is either retained or deleted. Again, this is a function of its neighbors, that is of the other nodes to which it is linked. A node is said to "be supported" by its neighbors if their weights are enough to prevent its deletion. Since deleting one node may remove enough support from one or more of its neighbors to cause their deletion, the process may be repeated until a stable graph remains.

If the position of each target were calculated exactly, there would be a node for each correct target-map pair. Moreover, each correct node would be connected to all of the other correct nodes, since the target-target and map-map distances would match exactly. There may be other spurious nodes, and they would probably have links to at least some of the correct nodes, but not to all. So the search could simply find the maximal clique (fully-connected subgraph) in the graph, report those nodes as the correct identifications, and discard all others.

In general, though, there will be nodes or links or both missing because of errors in calculated target positions. It will still be the case that almost all correct nodes will be linked to almost all of the other correct nodes, since the respective distances will be nearly correct. Discrete relaxation may be thought of as finding the largest almost-clique, useful for imperfect graphs. Each node is checked to see if it has enough links (that is, is consistent with enough other identifications); if not, it gets deleted. Furthermore, rather than just counting neighbors, the weights of the neighbors are summed and that total checked against a threshold. This way nodes that have a high probability of being correct based on position (that is, nodes with a high weight) contribute more to keeping a neighbor than do less certain nodes. Since deleting a node also deletes all links to that node, one or more of its neighbors may then fall below threshold. So the entire process is repeated until a stable subgraph remains.

Selecting a threshold poses somewhat of a problem. If there are a large number of spurious nodes at the beginning of the relaxation process, even incorrect nodes may have many neighbors. On the other hand, by the end of the deletions there would ideally be only the few correct nodes, each with only that number of neighbors. So a variable threshold must be used, set according to the number of

nodes remaining at each iteration. The threshold may be lowered automatically if a stable subgraph is reached with far too many nodes, or raised and the process retried if too many nodes are deleted.

Relaxation based on nodes and links will get rid of most of the incorrect nodes, but not all. The information used up to this point comes from calculated position of individual targets (creation of nodes), from distances between pairs of targets (creation of links), and from consistency of those distances (relaxation). The next stage looks at orientations and angles formed by 3 targets (two links from the same node).

The two template-map pairs at the ends of a link define a line segment in the template and an equivalent one in the map. Mapping the template line segment onto the map segment gives a four parameter transform: the line segment may be stretched, rotated, shifted in X and shifted in Y. These four parameters are calculated for every linked pair of nodes, and that information stored with each link; relaxation is then used on the links. Where before relaxation checked for consistency between nodes (position of targets), now it is used to check for consistency between links and therefore between groups of nodes.

In a perfect match, each pair of correct nodes would produce exactly the same four parameters. In other words, if the template needed to be stretched, rotated, and shifted by certain amounts to match the map, the line segment connecting any two targets would itself have to be stretched, rotated, and shifted by those same amounts. If one of a linked pair of nodes represents an incorrect identification, then even though the length of the line segment may be nearly correct (it must have been, in order to get by the first relaxation step), the rotation and shifts will be wrong. So relaxation in this case compares a link with all other links having a common endpoint. If the four parameters of a transform attached to a link are each within some value of the parameters of an adjacent link, the links are consistent. If a link is consistent with enough other links, it survives; if not, it is deleted. Note that what is being looked at is not the value of an individual transform; rather, it is the difference between two transforms. A given line segment may be rotated and shifted by an arbitrarily large amount, as long as other line segments are rotated by nearly equivalent amounts. Just as in the case of relaxation over nodes, deleting a link may lead to the deletion of one of the adjacent links. So the relaxation is iterated until only stable links remain.

Deleting links reduces the number of neighbors of the formerly linked nodes. perhaps to the point that some of them may no longer have enough consistent neighbors. So the next step is to run the node relaxation process again. The result is a stable graph, usually with very few spurious nodes.

5. Search

Even after the template matching and relaxation steps, there may be more than one plausible identification for a given target. M* search is then used to pick the best set of identifications.

M* was first described by Barrow and Tennenbaum in [1]. It works as follows. Go through the list of nodes, looking for some target with more than one possible identification. Call the M* procedure recursively on a series of new lists, each the same as the original list except containing only one of the conflicting identifications for that target. When there is only a single node for each target, the remaining nodes describe a candidate match. This match is evaluated by assessing the stresses in the template as follows. The "cost" (stress) of the match is incremented 1) for each unidentified target, 2) as a function of less than perfect positional matches (from node weights), and 3) as a function of distances between a pair of targets not matching the distance between the corresponding map points (the link "stretch" parameter). If the total cost of this match is less than the total cost of the best match previously evaluated, this match replaces the former best match.

It may be that some nodes had survived the relaxation process by being linked to two conflicting nodes (that is, nodes with different identifications for the same target). This can be discovered, and those nodes deleted, by running the relaxation process each time the M* routine generates a new list of nodes. Often after relaxation the subgraph formed by that list of nodes and their links will collapse, ruling out that search path and saving search time.

Note that the M* search process would have worked correctly had it been invoked on the original graph. Relaxation is much less time consuming, though, and goes a long way towards reducing the size of the subgraph that must be searched. On the other hand, relaxation may occasionally throw out nodes that actually belong to the best possible match. So there is a tradeoff between running time and confidence in the result.

Finally, it is possible to use the results of all this processing to provide accurate information for navigation. Given identifications of several targets, it is relatively easy to calculate the four parameters of the global transform that maps each target as closely as possible to their map locations. Then, knowing the ship's location and heading relative to the image, the true map coordinates and heading of the ship are easily obtained.

6. Experimental Results and Further Research

The experimental study dealt with six sonar images of the same area: three generated on passes up the channel, and three in the opposite direction. One of the images was used to generate the map. Matching the other two images made in the same direction, with vehicle position known to within 100 pixels in each direction, produced the correct identification for each target, and gave heading and position correctly. Matching images made in the other direction was a little more difficult. The mathematics of flipping, rotating, and shifting the image posed no problem. But there were differences in the scaling of the two sets of images along the direction of travel, apparently caused caused by a current. Visual inspection of the images shows that the three taken in one direction are compressed in the direction of travel; while those in the other direction seem to have the vertical distances between targets stretched out. There are two solutions to this problem. The easiest, from an image processing point of view, would be to get more accurate information on absolute vehicle speed, and correct the images based on that. The other possibility would be to change the template to allow more stretch in the direction of travel than perpendicular to the vehicle, and to place more emphasis on consistent stretch and less on absolute positions.

Another problem to be worked on is automatic determination of thresholds. The first place this shows up is in target detection. Apparently the calibration of the scanners used in one direction was significantly different than the calibration of the other scanners, so the second set of images were of much higher intensity. The target detection parameters, carefully tuned to the first set of images, produced more spurious targets than real ones when applied to the second set. Adjustment of the parameters to suit this set of images was no big problem, but it was impossible to find a set of parameters that worked correctly on both images. Some sort of histogram-driven automatic threshold would be a good idea. Other variables that have been experimented with, but could use more examination, include size of the filter and number of points necessary in a clump before reporting it as a target.

In the relaxation steps, there should be a way to adjust thresholds as a function of confidence in the reported positions. If the vehicle's position is accurately known, ocean currents are at a minimum, and the image is sharp and well detailed, it should be possible to restrict the range of possible identifications for each target. Some of this is done manually now, by asking the user of the program for maximum allowable stretch, for instance. Also, node and link deletion thresholds, which are now only a function of the number of nodes, should be adjustable if too many or not enough nodes are left at the end of the process.

There is also a problem weighting the cost function for the M^* search. How much more cost should be assigned, for example, for an unidentified target than for an identification that stretches distances? Currently, the weights are set in a rather *ad hoc* fashion. There may be no good, scientific way to set them; if not, trial and error will have to suffice.

Finally, the whole process should run faster. As an experimental project, much of the coding is done in a straightforward, understandable, but inefficient style. In order to run this on board an autonomous vehicle, there will have to be reductions in running time as well as storage space. Most of this will be relatively easy once the design is frozen.

7. Conclusions

Template matching and relaxation provide an accurate and efficient way to recognize objects in certain kinds of images. The important image characteristics for the style of processing described in this paper are 1) image reliably separable into individual parts 2) those parts not recognizable by themselves and 3) some relationship between the parts that is stable in different views of the same area while being distinctive enough to provide reliable identification.

Davis, on whose work much of this is based, used template-driven relaxation to recognize contours of islands. He approximated the contours as polygons and did his recognition based on relative locations of vertices and their angles. There were problems reliably decomposing the image: a small change in curvature could cause a large shift in the location of a vertex of the polygon. His data could also be arbitrarily rotated, which added another complication. Even so, the distances between vertices, the size of the angles, and the ordering of the vertices along the contour were enough for accurate identification.

Sonar image processing as used in channel conditioning seems a natural application for these techniques. The image decomposes very nicely into separate targets, each nearly indistinguishable from the others. Orientation and position are roughly known, so there needn't be concern about rotation invariance. Nor should there be gross differences in scale. As a result, positions of targets and distances between them provide clues not available in Davis's island contours. Perhaps most importantly, the distances and angles between targets, perhaps corrected for ocean currents, are stable, reliable, and recognizable.

8. Acknowledgements

Thanks go to Art Nelkin and Jim Brinsley of Westinghouse Oceanic Services for providing me with the sonar images. The CMU Vision Group gave me many helpful and encouraging comments, both as a group and individually. Thanks also to my advisor, Raj Reddy, for introducing me to the world of image processing. Hans Moravec provided me with the mathematics in calculating the global transforms. And a special thanks goes to Mark Stehlik for his proofreading and editing.

References

[1] Harry G. Barrow and Jay M. Tenenbbaum.
MSYS: A System For Reasoning About Scenes.
Technical Report 121, Artificial Intelligence Center, Stanford Research Institute, April, 1976.

[2] Larry S. Davis.
Shape Matching Using Relaxation Techniques.
Technical Report TR-480, Computer Science Center, University of Maryland, September, 1976.

[3] Keith Price.
Change Detection and Analysis in Multi-Spectral Images.
PhD thesis, Carnegie-Mellon University, December, 1976.
Appendix B discusses his efficient smoothing algorithm.