UCRL-CONF-220030

LAWRENCE LIVERMORE NATIONAL LABORATORY

Weighted Model Components for Gradient Direction Matching in Overhead Images

C. W. Grant, S. Nikolaev, D.W. Paglieroni

March 20, 2006

Optics and Photonics in Global Homeland Security II

SPIE Defense & Security Symposium, Orlando, Florida, April 20, 2006

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Weighted Model Components for Gradient Direction Matching in Overhead Images

Charles W. Grant, Sergei Nikolaev and David W. Paglieroni

Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA, 94551, USA

ABSTRACT

Gradient direction matching (GDM) is the main target identification algorithm used in the Image Content Engine project at Lawrence Livermore National Laboratory. GDM is a 3D solid model-based edge-matching algorithm which does not require explicit edge extraction from the source image. The GDM algorithm is presented, identifying areas where performance enhancement seems possible. Improving the process of producing model gradient directions from the solid model by assigning different weights to different parts of the model is an extension tested in the current study. Given a simple geometric model, we attempt to determine, without obvious semantic clues, if different weight values produce significantly better matching accuracy, and how those weights should be assigned to produce the best matching accuracy. Two simple candidate strategies for assigning weights are proposed – pixel-weighted and edge-weighted. We adjust the weights of the components in a simple model of a tractor/semi-trailer using relevance feedback to produce an optimal set of weights for this model and a particular test image. The optimal weights are then compared with pixel and edge-weighting strategies to determine which is most suitable and under what circumstances.

Keywords: image processing, computer vision, object recognition, relevance feedback, edge matching, model matching, gradient matching, component weighting

1. INTRODUCTION

The Image Content Engine (ICE) is an ongoing project at Lawrence Livermore National Laboratory to produce systems for extracting domain specific semantic information from large volumes of imagery and evaluating queries using this semantic information [1]. The initial application domain is aerial and satellite surveillance imagery of broad areas. The system will be able to combine imagery information from various observation modes, including optical, hyper-spectral, and synthetic aperture radar. ICE uses GDM for object matching when a sufficiently accurate solid model of the target objects is available and the image resolution is suitable. GDM is described in [2] and outlined below. Several areas are identified where it seems likely that modifications could improve the matching accuracy. The main subject of this paper is an experiment where we evaluate different model component weighting strategies by comparing the weight assignments with the weight assignments produced by using relevance feedback.

1.1. Gradient Direction Matching

GDM is a solid model-based edge-matching algorithm that does not does not require explicit extraction of edges from the source imagery. The edge identification is instead done to projections of 3D solid models, which are free of sensor and environmental noise. The purpose of GDM is to find target instances in imagery, based on a 3D solid model of the target. The operation of GDM is based on several assumptions about typical imagery and target objects.

1. The locations, orientations, obscuration and even existence of targets in the image are unknown.
2. The background, illumination conditions, and surface properties of the target objects are unknown, but we assume there will be some, unknown, intensity difference between surfaces and between surfaces and the background (because a majority of the edges will need to be visible).
3. On pixels near visible edges, the gradient of the image intensity will be approximately perpendicular to the edge.
4. The magnitude of the gradient is unknown and unimportant - in fact we usually want to consider gradients that differ in direction by 180° degrees as equivalent.
5. The intensity and the direction of the gradient of the intensity at all image locations which are not near edges are completely unknown, and can be (and should be) ignored.

1.2 Spatial Domain GDM Algorithm

Given these assumptions we can formulate the spatial domain GDM algorithm:

1. Calculate the gradient of the image intensity and save the direction at each pixel, discarding the magnitude.
2. Project the 3D solid model of the target, at a series of orientations, into the image coordinate system doing visible surface determination and building a set of "model gradient direction" maps.
3. For each pixel p in the source image and each model orientation, calculate a similarity match value $\mathbf{S}_p$, that represents how well the model gradient direction map for that orientation matches when centered on location p in the image. The matching function is based on the difference of the image gradient direction, θ, and the model gradient direction, β, at each corresponding pixel. The entire raster of similarity values is called the similarity surface. Instances of the target at particular locations in the source image are expected to produce local maxima in the similarity surface at those locations.
4. The similarity surfaces are combined by taking the maximum value over all orientations at each pixel.
5. Local maxima in the combined similarity surface are identified and disambiguated.
6. A list of image tiles centered on the local maxima is presented to the image analyst, sorted by match similarity.

The key here is to use a matching function based on the difference of the gradient direction angles of the image and the model (which ignores the magnitudes of the gradients) and to double the angles (which makes differences of 180° disappear). The formula for the similarity at a pixel (at one model orientation) in the spatial domain is:

$$S(c,r) = \frac{1}{2} + \frac{1}{2} \sum_{(j,k) \in m} a(c+j,r+k)\, b(j,k)\, cos[2\theta(c+j,r+k) - 2\beta(j,k)] \tag{1}$$

where:

a is the image pixel gradient direction weight (usually 1, but 0 if there is insufficient gradient magnitude at a pixel)
b is the model pixel gradient direction weight (between 0 and 1)
θ is the image gradient direction (angle)
β is the model gradient direction (angle)
(c,r) is the pixel location in the image
(j,k) is the pixel location in the model projection
m is the set of all pixels in the model projection with non-zero weight

1.3 Frequency Domain GDM

Spatial domain matching is far too inefficient for production use so a much more efficient FFT based implementation is used. The flow of information is shown in figure 1. The flow is the same for spatial domain processing, except cross-correlation is replaced by FFT based cross-correlation.

This transformation into the frequency domain is possible because equation (1) is equivalent to equation (2) below.

$$S(c,r) = \frac{1}{2} + \frac{1}{2} Re\left(\sum_{(j,k) \in m} a(c+j,r+k)\, e^{2i\theta(c+j,r+k)}\, b(j,k) e^{2i\beta(j,k)} \right) \tag{2}$$

which is in the form of a cross-correlation that can be calculated efficiently in the frequency domain using the Fast Fourier Transform:

(3) $$S(c,r) = \frac{1}{2} + \frac{1}{2} Re\,[\,FFT^{-1}\,[\,FFT\,[\,a(c,r)\,e^{2i\theta(c,r)}\,]\,x\,FFT\,[\,b(-c,-r)\,e^{2i\beta(-c,-r)}\,]\,]$$

The right half of the inner formula, $FFT\,[\,b(-c,-r)\,e^{i2\beta(-c,-r)}\,]$, is pre-calculated for each model orientation and stored. The left half, $FFT\,[\,a((c,r)\,e^{i2\theta(c,r)}\,]$, is calculated once per block of the image (this includes calculating the gradient directions, θ). The outer loop is one block multiplication and one inverse FFT to be performed once per block per orientation. Because there are many orientations (typically 60-100) and many image blocks (typically thousands) these inverse FFT operations dominate the run time of the GDM algorithm.

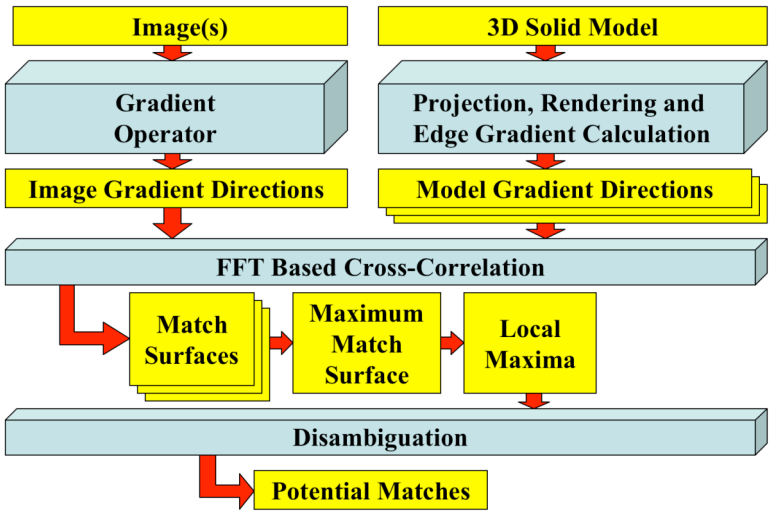


Figure 1: Gradient Direction Matching

1.4 Potential Performance Improvements

The cross-correlation core of GDM algorithm is strictly structured and does not allow for modifications without sacrificing the great efficiency of the FFT implementation. But the procedures used to generate the inputs and process the outputs of the cross-correlation are more flexible. Are there any pre-processing methods that can be applied to the input image to enhance the matching accuracy? Various methods of gradient determination in each image block are possible, which method gives the best matches? We usually use a 3x3 Sobel gradient operator with no preprocessing. How should the 3D solid model be structured to give the best matches? How should the projection and sampling of the desired gradient directions be done? Various methods of weighting the pixels in the expected gradient direction rasters are possible, which of these gives the best matches? Once the match surfaces for each orientation are generated, what is the best way to extract the potential match locations? Once the potential match locations are extracted, are there post-processing methods that will improve the ranking of valid matches while reducing the rankings of false alarms?

In this paper we examine modifications to the model projection, rendering, and gradient extraction operation. Specifically we address the question of assigning weights to the pixels in the expected gradient direction rasters. We consider the idea that visible edges in the image might carry approximately the same amount of "matching information" suggesting that weighting pixels inversely proportional to projected visible edge length might give better matching accuracy. We also consider the idea that each pixel carries approximately the same amount of "matching information" and should be weighted approximately equally unless there is specific ad hoc knowledge that some parts of the model are more significant. We use the technique of relevance feedback to adjust the weights and improve the matching accuracy for some test images and then compare these improved weights with the performance of edge length based weighting and equal pixel weighting.

1.5 Weighting the Model Gradient Direction

The first weighting technique used with GDM is to weight all pixels equally if they are within a threshold distance of a visible projected model edge. All other pixels are weighted zero. This gives very reasonable matching performance, but is subject to considerable quantization noise. As the configuration, size, location, or orientation of the model is changed slightly, the weights of pixels switch suddenly from one to zero or zero to one. It is difficult to see if the similarity changes are due to a better or worse match, or just due to quantization noise.

The weighting technique was then changed so that pixels directly centered on a visible edge are weighted the highest and the weights of other pixels are based on how far the pixel center is from the visible edge using a smoothly varying function. This improved the matching accuracy slightly, but importantly eliminated many of the spatial quantization artifacts. Small changes in the model or projection no longer caused sudden jumps in the similarity value.

The weighting technique was then extended to allow the modeler to specify the significance of various components of the model when building the model. The projection and rendering code then multiplied the distance-based weight by the model-specified weight to get the final weight used for each pixel. A multi-level hierarchical scheme was adopted so weights could be given to sub-structures at various levels, down to polygons or even single edges. The weight of the lowest level component is the product of its weight and the weights of all of the higher level components to which it belongs. This introduces a great deal of flexibility in model building but leaves open the question of what weights should be used to give the best match.

One can always concoct situations where one type of weighting is better than another. Consider the case of a model that is a thin rectangle. There are two long sides with many pixels and two short sides with far fewer pixels. When looking for targets with this model, any two parallel line segments with the same spacing as the width of the rectangle will have a good match on most pixels (a large area with a uniform gradient will also be a good match). The thinner the rectangle is, the worse this problem becomes. The pixels on the short sides of the rectangle provide all the discrimination between rectangles of the correct length and longer objects. In some situations (such as plowed fields) weighting the short ends of the rectangle more heavily would clearly improve the relative matching of the desired shape compared to the many false alarms in the fields. Does this generalize to a mix of imagery? Perhaps weighting the pixels of all edges inversely proportional to their visible projected lengths (so that all edges contribute to the matching with equal total weight) gives better matching performance than weighting pixels independently of the visible projected edge length (so that all edges contribute to the matching with total weight proportional to their visible projected lengths). Or perhaps the plowed field background is atypical and such an approach is usually counterproductive leading to reduced accuracy.

The visible parts of a single component, when projected at different model orientations, may differ greatly. This includes differences in size and shape as well as possibly being partly or wholly obscured by other parts of the model. Assuming the component is visible at a given orientation, how should its contribution vary with orientation? Because our test images are near nadir looking and our model is so simple, the lengths of the projected edges and visibility of the edges will not vary significantly with model orientation so we cannot investigate this question directly. However we hope that this experiment will suggest a direct relation between edge length and best weight, and that relation might generalize to edge lengths that vary with orientation (a subject for further study).

2. EXPERIMENTAL METHODOLOGY

To investigate some of these questions we have selected a very simple model (figure 2) so that the effects of changes in the weighting strategy should not be hidden by the model complexity. Even though the model is very simple, it is still quite useful for some realistic search scenarios. Our model is of a large vehicle, a typical tractor/semi-trailer combination. The simplicity of the model is suitable for resolution levels of approximately 0.5 to 1.0 meters per pixel where much additional target detail is not visible. We have chosen to treat each of the eight edges of this model as a separate component. The aspect ratio of the semi-trailer is about 5.5 so we expect to observe some thin rectangle effects as described above. Near nadir looking imagery is used in this experiment so a full 3D model is not needed.

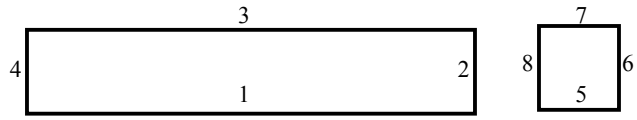


Figure 2. Simple tractor/semi-trailer model with 8 components

The procedure for this experiment is to use relevance feedback (RF) to optimize the component weights for best performance using a representative test image. The optimized component weights can then be used for matching the same type of target in other images. The values of the optimized weights should directly suggest which of the two ad hoc weighting strategies under consideration is superior. Our test image is a commercially available Digital Globe Quickbird image with a resolution of 16k×16k and ground sample distance of 0.6 meters of a section of Moscow. The image contains a mix of urban, forest and water areas, with many roads, cars, trucks and railroads.

2.1 Relevance Feedback Algorithm

The relevance feedback [3] algorithm is widely used in content-based image retrieval work. In the standard RF approach, there is a set of images tiles $\mathbf{T} = \{t_k\}_{k=1}^{K}$. Each tile in the set, t_k, is characterized by a feature vector $\mathbf{f}_k = \{\mathbf{f}_{k,i} = 1,N\}$, where N is the total number of features in the vector. Typically, the features $\mathbf{f}_{k,i}$ that constitute the feature vector represent various statistics (mean, variance, color histogram, response to edge detector, etc) that describe the tile at a range of image resolutions. In our application each feature will be a similarity measure for a model component. In the standard formulation there is a query tile $\boldsymbol{t}_0$ and its corresponding feature vector $\mathbf{f}_0 = \{\mathbf{f}_{0i}, i = 1,N\}$, and the goal is to find tiles in $\mathbf{T}$ that are "similar" to $\boldsymbol{t}_0$. This is done by tuning a set of weights, w_k, so that the norm of the weighted distances $\|w_k(f_{0k} - f_{i,k})\|$ is small for "relevant" tiles and large for "non-relevant" tiles. Any well behaved norm can be used. Define S_i to be the similarity of tile, t_i, to the query tile, $\boldsymbol{t}_0$, as:

$$S_i = \sum_{k=1}^{N} w_k \, \xi_k(i,0) \tag{4}$$

Where $\xi_k(i,0)$ is the normalized difference of the k-th feature between the i-th tile and the query tile. The distances are normalized so that the standard deviations are equal. The RF algorithm then adjusts the weights, w_k, so that the discrimination between relevant and non-relevant tiles is improved.

The RF algorithm can be written as the following sequence of steps:

1. Calculate the matrix of feature similarity values $\mathbf{S}_i(m,n)$, between i-th features of m-th and n-th tile
2. Calculate the mean $\mu_\mathbf{i}$ and the standard deviation $\sigma_\mathbf{i}$ for each column of $\mathbf{S}_i(m,n)$.
3. Initialize feature weights, $w_i = 1/N$, $\quad i = 1..N$
4. Compute the similarity values between each (k-th) tile and the query tile t_0, $s_i(k,0) = [\mathbf{f}_{ki} - \mathbf{f}_{0i}]^2$, and normalize them $s_i(k,0) \rightarrow \xi_i(k,0)$
5. For each tile in $\mathbf{T}$, calculate the overall similarity between the tile and the query as the linear combination of normalized feature similarities $\xi_i(k,0)$,

$$S(k) = \sum_{i=1}^{N} w_i \, \xi_i(k,0), \qquad k = 1..K$$

6. Rank all tiles t_k according to their similarity values $\mathbf{S}(k)$. The tiles are ranked in the order of decreasing similarity values and a predefined number of top matches are displayed to the human analyst who issued the query. We will call the set of displayed matching tiles $\mathbf{M}$. Clearly, $\mathbf{M} \subseteq \mathbf{T}$.

7. The human analyst interactively marks tiles in $\mathbf{M}$ as "relevant" (set $\boldsymbol{M_R} \subseteq \boldsymbol{M}$), or (optionally) "non-relevant" ($\mathbf{M_{NR}} \subseteq \mathbf{M_R}$). Some tiles can be left unmarked; they will be considered "neutral" and will not affect the procedure.
8. The feedback from users is incorporated by adjusting the weights w_i in equation (4) to emphasize features which are seemingly more important for the matching. This is done as follows:

   a) Reset all weights to zero, $w_i = 0$, $i = 1..N$
   b) Analyzing the tile rankings based on the similarity values for individual features $\xi_i(k,0)$. For each feature i, define the set of best matches ranked according to $\xi_i(k,0)$ as $\mathbf{M_i}$.
   c) Count the number of times the relevant tiles from $\mathbf{M_R}$ appear in $\mathbf{M_i}$. For each occurrence, increment w_i by one.
   d) (Optional, if $\mathbf{M_{NR}}$ is defined.) Count the number of times the non-relevant tiles from $\mathbf{M_{NR}}$ occur in $\mathbf{M_i}$. For every such occurrence, decrement w_i by one.
   e) Check if the resulting weight w_i is negative. If so, reset to zero.
   f) Re-normalize the weights, $w_i = w_i / \sum_{j=1}^{N} w_j$

   This procedure will assign more weight to features for which the set intersection $\mathbf{M_i} \cap \mathbf{M_R}$ is maximized.
9. Go to step 5 if the weights have not converged.

The sequence of steps 5-9 is repeated until weights w_i converge. Given a good descriptive set of image features and consistency of user's feedback, the weights should converge to a set of values that improve the retrieval precision for this particular query t_0. The resulting set of weights can be stored for future queries related to the same tile t_0, or similar tiles.

2.2 Applying RF to GDM Weights

The summation in equation (1) runs over all projected edge pixels. This can be rewritten as sums over components as follows:

(5) $$S_{(c,r)} = \frac{1}{2} + \frac{1}{2} \sum_{c=1}^{N_c} \sum_{(j,k)\in \mathbf{P}_c} a_{(c+j,r+k)} b_{(j,k)} \cos 2[\theta_{(c+j,r+k)} - \beta_{(j,k)}]$$

where N_c is the total number of components in the model, and $\mathbf{P}_c$ is the set of edge pixels corresponding to c-th component. Factoring out the component weights $\hat{w}_j$ so that $b_{(c,r)} = \hat{w}_j b'_{(c,r)}$ for all pixels constituting a given component j yields

(6) $$S_{(c,r)} = \frac{1}{2} + \frac{1}{2} \sum_{k=1}^{N_c} \hat{w}_k \sum_{(i,j)\in \mathbf{P}_k} a_{(c+i,r+j)} b'_{(i,j)} \cos 2[\theta_{(c+i,r+j)} - \beta_{(c,r)}]$$

Finally, we can denote

(7) $$\xi_k = \frac{1}{2} \sum_{(i,j)\in \mathbf{P}_k} a_{(c+i,r+j)} b'_{(i,j)} \cos 2[\theta_{(c+i,r+j)} - \beta_{(c,r)}]$$

where ξ_k is the similarity value for k-th component between the model and the thumbnail. Substituting (7) into (6), we obtain

(8) $$\mathbf{S}_{(c,r)} = \frac{1}{2} + \sum_{k=1}^{N_c} \hat{w}_k \xi_k$$

which is nearly identical to equation (4), for which we know the RF procedure. The feature (or component) similarities ξ_k are still normalized to the interval [0,1]. The only difference between equation (4) and equation (8) is an extra

additive constant of 1/2. Note however, that the algorithm remains unchanged if $\mathbf{S}(k)$ in equation (4) has an additive constant, because any such constant will preserve the relative ranking of the tiles.

The RF algorithm applied to GDM an be summarized as follows:

1. Run normal GDM to find local maxima of the similarity surface $\mathbf{S}(c,r)$.
2. Disambiguate candidate matches and display the cleaned candidate list of thumbnails ranked according to their similarity values. Denote the returned list of thumbnails $\mathbf{M} = \{t_k\}_{k=1}^{M}$
3. For each *i*-th candidate thumbnail, compute the component similarity values $\xi_{k(i)}$ using equation (7), for all components in the model. These are computed separately in the spatial domain, since in normal GDM $\mathbf{S}(c,r)$ is calculated by FFT. It is important to realize that the component-based similarity values ξ_k *do not* change during RF iterations. It therefore makes sense to store them in computer memory. As we mentioned above, the values ξ_k are already normalized to the interval [0,1], so there is no need for explicit normalization at each step here.
4. Initialize component weights, $\hat{w}_k = 1/N_c$, where N_c is the total number of components in the phase boundary set **P**. This assigns equal weight to all components of the model.
5. Compute the total similarity $S(i)$ for each *i*-th thumbnail using current weights $\hat{w}_k$, according to equation (8).
6. The set of relevant thumbnails, M_R, are marked.
7. For each component *k*, sort the similarity values ξ_k in the descending order, and call the set of top-ranked values $\mathbf{M}_k$ Reset the corresponding weight to zero, $\hat{w}_k = 0$, and then follow the increment/decrement strategy for weight updating. After updating the weights, reset negative ones back to zero, and normalize so that $1 = \sum_{k=1}^{N_c} \hat{w}_k$.
8. Go to step 5.

The sequence of steps 5-9 is repeated until the weights $\hat{w}_k$ converge. The resulting final values of $\hat{w}_k$ can be stored for future searches of the same object model.

The use of an RF procedure in an automated test procedure that does not prompt for the selection of relevant tiles at each iteration implies the knowledge of the exact locations of true objects in the images is available to the system. This information is derived by visually examining the images and marking the locations of all trucks. This information is stored and used on each successive RF iteration to derive the relevant/non-relevant sets of thumbnails. For the RF procedure, we used the top 500 thumbnails with respect to their overall similarity values computed by GDM algorithm. This represents the base set, **M**. The size of the sets $\mathbf{M}_k$ which are used to update the component weights, is 100 tiles each.

3. RESULTS

The initial weights for the truck model were chosen so that each edge contributed an equal amount to the matching. Thus each pixel on a edge was weighted roughly inversely proportional to the length of the edge. This initial weighting corresponds to the horizontal dotted line in figure 3a. The second dotted curve in figure 3a shows the values of the weights after one iteration of RF. The solid line represents the converged weights after several iterations. The final values are a reasonable approximation of weighting each edge proportional to the length of the edge (equivalently weighting each pixel about the same). This tends to validate the second proposed ad hoc weighting strategy - uniform pixel weighting, while tending to discredit the first ad hoc weighting strategy - uniform edge weighting. The only significant variation from the uniform pixel weighting for this model was that the front edge of the truck cab (edge 6) was found to be much more significant than the rear edge of the truck cab (edge 8) and more significant than sides of the cab (edges 5 and 7). The long edges of the semi-trailer contributed the most to the performance of the matching. One might assume that this was due to higher similarities (better matches) of that component in the image, but in fact after examining the data we found that long edges had significantly lower similarity values in the non-relevant thumbnails. They were the best rejecters of false alarms.

It is apparent that the RF algorithm converges very quickly on this test data. The results were nearly ideal after only one iteration. This was not very significant to us in this application as GDM is much slower than RF.

As a means to compare the performance of the algorithms, we use Receiver Operational Characteristics (ROC) curves for the detections. ROC curves show the cumulative number of true reported matches as a function of the number of returned thumbnail pages. This quantifies the number of image thumbnails a human analyst would have to review, before a specified fraction of true matches to the model are be found. Figure 3b shows the ROC curves for the training thumbnails used in the RF training. A significant improvement in the ROC curve is observed.

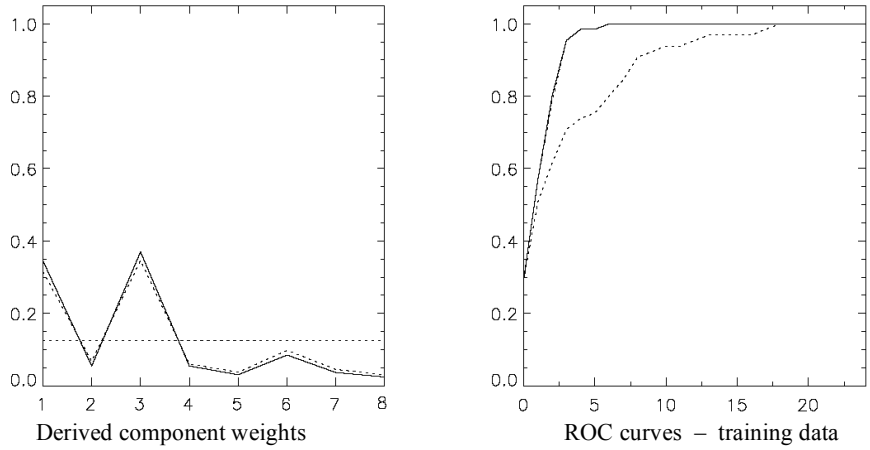


Figure 3. Relevance Feedback results.
The solid line denotes the final (converged) solution, while the dashed lines show successive iterations.

These results are encouraging, but will the derived weights help identify trucks in a random image? After all, the weights were derived on a limited set of thumbnails (500 total), which may not represent the whole variety of thumbnails in the image. Moreover, the set of weights was derived from a single image, which can be thought of as a "training set". There is no guarantee that the derived set of weights will work for all images taken from different viewing angles, under different lighting conditions, regardless of terrain types, etc. These are all valid concerns that needed to be addressed.

.

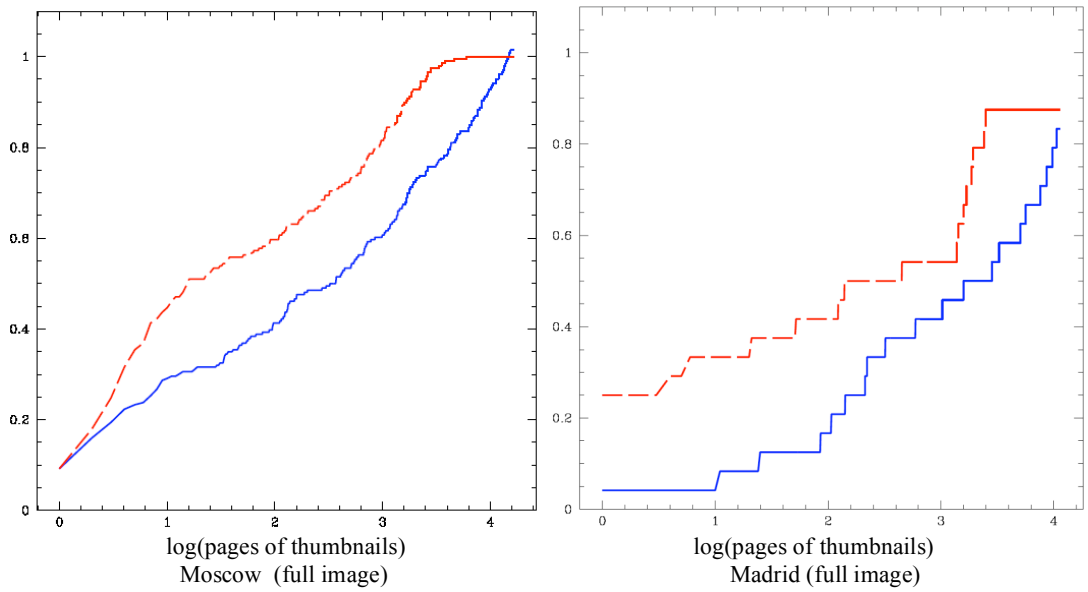
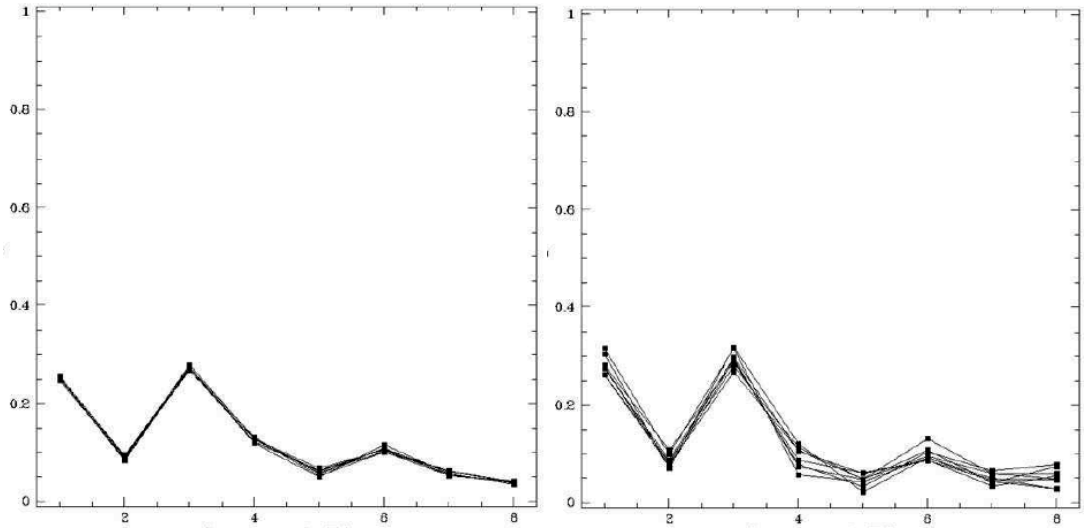


Figure 4. ROC curves for uniform edge weighted components (blue) and RF weighted components (red)

To address these concerns, we conducted a series of tests. First, we considered whether the derived set of weights does improve the performance of the GDM algorithm on the remaining thumbnails in the test image. To do that, we fixed the component weights in GDM to the RF-derived values, and re-ran the GDM procedure obtaining the new list of matches sorted by decreasing similarity. Figure 4a shows the corresponding ROC curves for uniform (bottom curve) and non-uniform (top curve) sets of weights. The improvement due to weighting components is clear, and reaches as much as 20% for a given number of thumbnail pages. Conversely, for a given fraction of true hits recovered, the number of thumbnail pages that need to be examined by a human analyst can be reduced by as much as a factor of 10 compared to the uniform edge weighting case. The overall number of returned true hits is similar, 206 for non-uniform weighting and 209 for uniform edge weighting. In addition a second image was processed using the original and RF derived component weights. The results of those runs are shown in figure 4b. This second image was a section of Madrid, which contained, by visual inspection, 24 trucks. The RF derived weights produced better results, similar to the Moscow image. The conclusion here is that the component weights derived from a rather limited set of thumbnails (500 total) do go a long way in improving performance of the GDM algorithm.

Next we addressed the issue of whether the weights would change given a different training image (terrain types, lighting conditions, viewing angles, etc). The direct test would, of course, be to repeat the RF procedure on a significantly different image. However, indirect evidence that these factors will not matter much comes from the following test: instead of replacing the training image, we replace the set of thumbnails from which the weights are derived. Originally, we ran the RF procedure on 500 thumbnails with top similarity values. According to our "ground truth" information, there were 194 relevant thumbnails among these 500. In two additional tests, we randomized the non-relevant component, i.e. selected random non-relevant thumbnails from the whole set of GDM reported hits. Moreover, we also reduced the number of relevant thumbnails among the 500 to only 50, instead of 194. Both these steps were implemented to significantly change the training thumbnails, to see whether this affects the weights. Each test was run several times, using different random samples. The results are shown in Figure 5. The results indicate that the component weights remain rather stable even in the presence of random non-relevant components, and even when the "signal" (i.e. the number of true hits in the training set **M**) is low. This result lends indirect evidence that some variation in the training image will not affect the component weights significantly.



194 relevant 306 non-relevant 50 relevant 450 non-relevant

Figure 5. Component weights derived with random sampling of 500 thumbnails

4. CONCLUSIONS

We have applied the technique of relevance feedback to the selection of component weights for gradient direction matching.

1. Changing the component weights used by GDM definitely alters matching performance significantly. For one test image, we report the improvement of as much as 20% in terms of the detection efficiency (number of true hits with respect to the number of thumbnails examined), or a factor of 10 fewer thumbnails to examine, for the same detection probability.
2. Relevance feedback is very effective in determining component weights for GDM. It converges quickly in as little as 1 or 2 iterations in our case. It also appears to be tolerant of varying the non-relevant parts of the test image.
3. Training the weights on one image and using those weights on a different image was successful. The images were of similar resolution and quality so this result does show that more divergent images can continue to benefit from training on different images.
4. The proposed uniform edge weighting ad hoc strategy was shown to be clearly inferior to the proposed uniform pixel weighting ad hoc strategy for this model and these images.
5. Several other areas in the GDM process were identified where performance improvements might be made. Some additional research in these areas is planned.

ACKNOWLEDGMENT

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

REFERENCES

1. G. Weinert, J. Brase, D. Paglieroni, “Computer-aided content-based cueing of remotely sensed images with the image content engine”, Proc. IGARSS 2004, September 2004.
2. D. Paglieroni, W. Eppler and D. Poland, “Phase Sensitive Cueing for 3D Objects in Overhead Images”, SPIE Defense & Security Symposium; Signal Proc., Sensor Fusion and Target Recognition XIV, Proc. SPIE, Vol.5809, 28-30 March 2005, Orlando, FL.
3. Y. Rui, T. S. Ortega, M. Mehrotra, “Relevance feedback: a power tool for interactive content-based image retrieval,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 8, No. 5, p. 644, 1998.