ON VECTOR QUANTIZATION FOR FAST FACET EDGE DETECTION *

M. Y. Jaisimha Jill R. Goldschneider Alexander E. Mohr Eve A. Riskin
Robert M. Haralick
Department of Electrical Engineering, FT-10, University of Washington, Seattle, WA 98195

Abstract

We present an approach for performing edge detection which builds on our prior work in fast facet edge detection using tree-structured vector quantization (TSVQ). We first extend the approach by using larger image vectors to reduce computational complexity by performing edge detection on multiple pixels at once. We then reduce the computational complexity of the edge detector without sacrificing performance by pruning the TSVQ with an edge detection-based criterion. We present results of edge detector performance on a sequence of images obtained from a mobile robot.

1. INTRODUCTION

This paper presents extensions to our work using TSVQ [2] to speed up the process of facet edge detection [4, 5]. Because image VQ and facet edge detection operate on small block-based neighborhoods, VQ can be used to perform edge detection. The image is encoded with a VQ for which the edge/no-edge decision has already been made for each codeword. Thus edge detection becomes a simple lookup of this information. The algorithm behaves as a "trainable edge detector" which has the advantage of having lower computational complexity than the conventional facet edge detector.

To improve the computational speed of the edge detector, we first explore the use of larger image vectors to perform edge detection on more than one pixel per codeword. We briefly describe our previous work and the extension in Section 2. When edge detection is performed on the centroid associated with each node of the tree, we find that often, the leaves of an entire subtree of the TSVQ has nodes which are all labeled as edge or no-edge nodes. Thus we can prune the TSVQ based on this edge detection criterion without affecting the edge detection performance. We describe the approach to pruning TSVQ-based edge detectors in Section 3. We conclude in Section 4.

*This work was supported by an NSF Young Investigator Award, NSF Research Experiences for Undergraduates Supplement MIP-9110508-SI, and NASA Graduate Student Fellowship in Global Change Research 4112-GC93-0191.

2. FAST FACET EDGE DETECTION

The second directional derivative edge detector [4] models a digital image as being derived by sampling a continuous underlying graytone intensity surface. The surface can be represented by a bivariate polynomial. referred to as a facet. Each pixel is labeled as an edge pixel if the second directional derivative of the facet in the direction of the gradient has a negatively sloped zero crossing within a threshold radius of the center of the pixel and if the edge contrast exceeds a threshold value. The facet polynomial coefficients for a cubic facet are obtained by performing a least squares fit to a 5×5 neighborhood of pixels centered around a pixel with a suitable set of bivariate basis polynomials.

The overall approach presented in [5] which we extend in this paper is as follows: 1) VQ Codebook Design: The training set is formed from all overlapping 5×5 blocks of the first image of a motion sequence. It is then used to design an unbalanced TSVQ. 2) Facet Edge Detection on the VQ Codebook: For each vector in the codebook, the facet parameters are computed. Second directional derivative edge detection is then performed on each of the codewords and the edge/no-edge decision for the center pixel is stored with the codeword. 3) Fast Facet Edge Detection: Later images in the sequence are encoded using the VQ and the edge/no-edge decision is output for each center pixel.

For an average depth R TSVQ with vector dimension $M \times M$, edge detection on one pixel costs R $(M \times M)$-dimensional hyperplane tests. A rough estimate for this cost is RM^2 multiplications per pixel.

We first reduce the complexity of this algorithm by using vectors of size $(M+1) \times (M+1)$ to perform edge detection on the four center pixels of the larger vector. In this case, the complexity is now roughly $\frac{R(M+1)^2}{4}$ multiplications per pixel and for our case of $M = 5$, the complexity reduces from $25R$ to $9R$ multiplications per pixel. We see that increasing the vector size improves the edge detector in Section 2. Figure 1 is the original input image which is Frame 2 of a mobile robot motion sequence. The output of a TSVQ-based edge

detector using 6×6 vectors at a rate of 6.8 bits per vector appears in Figure 2; this rate TSVQ includes all perceptually meaningful edges while rejecting significant amounts of texture edges.

To provide a quantitative measure of edge detector performance, we compare the output of the edge detector against a "ground-truth" edge image, generated by a human operator, which contains the perceptually meaningful edges of the image (see Figure 3). The edge detector makes two kinds of errors - the first type, *misdetection errors*, occurs when a pixel that is labeled as an edge in the ground-truth image is labeled as no-edge; the second type, *false alarm errors*, occurs when a pixel that is labeled as no-edge in the ground-truth image is labeled as an edge.

We modify the definition of false alarm errors to consist of edge pixels that are outside a two pixel radial tolerance of the ground-truth edge. To provide a quantitative measure of edge detector performance, we compute the probability of false alarms (P_{fa}) and misdetections (P_{md}) in the output of the edge detector. An ideal edge detector would have $P_{fa} = P_{md} = 0.0$.

In Table 1 we present the false alarm and misdetection probabilities for four frames of the sequence of images. The results are for the TSVQ-based edge detector with 6×6 vectors for a rate of 6.8 bits per vector of Figure 2; the facet edge detector; and the VQ-based edge detector (from [5]) with 5×5 vectors for a rate of 6 bits per vector. The edge contrast thresholds used in all three edge detectors are identical.

Comparing the VQ-based edge detectors to the facet edge detector, we see that the VQ-based edge detectors have substantially lower false alarm rates but higher misdetect rates. This would not adversely affect a scene interpretation system that fits lines and curves to the resulting edge data because such systems are more sensitive to high false alarm rates than to high misdetection rates.

In addition we find that the VQ-based edge detector with 5×5 vectors has a higher false alarm and misdetect rate than the one with 6×6 vectors. Upon visual examination of the edge detector output, we see that the 6×6 VQ-based edge detector detects low probability high contrast edges fairly reliably while rejecting higher probability low contrast edges. Also, as we mentioned earlier, the computational complexity of the 6×6 VQ-based facet edge detector is substantially lower than that of the original facet edge detector and the 5×5 VQ-based edge detector.

In Table 2 we present the false alarm and misdetection probabilities for the 6×6 TSVQ for Frame 2 with rates of four to nine bits per vector. Note that increasing the rate increases false alarms while decreasing misdetections. Choosing the TSVQ design rate is a tradeoff of these two quantities.

3. PRUNING OF TSVQ-BASED FACET EDGE DETECTORS

In our second extension, we prune the TSVQ-based edge detector using an edge detection criterion. We examine the TSVQ and if two siblings have the same edge/no-edge assignments, we prune them and assign the edge decisions to their parent node. This reduces the complexity and storage requirements by lowering the average rate of the TSVQ used to encode the images, but clearly the performance of the edge detector does not change. Usually, pruning is a tradeoff of decreasing the average bit rate and increasing the average distortion [3], but here we decrease the edge detector complexity without affecting its performance at all!

It is possible to predict the expected number of nodes pruned from a tree if we assume that we know the structure of the tree (i.e. the number of nodes at each level of the tree), and the probability that an edge assignment is an edge/no-edge. For example, assume that we have a balanced tree with N levels where level 0 is the root node and level $N-1$ is the leaf nodes. Also assume that any edge assignment at a leaf node is independent of other edge assignments and has the probability of being an edge or not an edge of $P(e)$ and $P(\bar{e})$ respectively, where $P(e) + P(\bar{e}) = 1$. Let $T[N]$ be the expected number of nodes pruned from an N level tree. We can write

$$T[N] = 2T[N-1] + 2((P(e))^{2^{N-1}} + (P(\bar{e}))^{2^{N-1}})^m$$

where $T[1] = 0$ and m is the number of edge decisions per node (i.e. $m = 1$ for 5×5 vectors and $m = 4$ for 6×6 vectors). In other words, the number of nodes pruned from a balanced tree with N levels is the number of nodes pruned from two balanced trees with $N-1$ levels (the subtree rooted at the children of the root node) plus two times the probability that the two children of the root node are pruned (all of the leaf nodes must have the same edge decisions). The expected percentage of nodes that are pruned is given by $Q[N] = \frac{T[N]}{2^N - 1}$. Notice that $Q[N]$ versus N is plotted in Figure 4 for $m = 4$. We can derive an asymptotic limit for $Q[N]$ by rewriting $T[N]$ as

$$T[N] = 2^N \sum_{i=1}^{N-1} 2^{-i}[(P(e))^{2^i} + (P(\bar{e}))^{2^i}]^m.$$

It is easily seen that $T[N]/2^N$ is absolutely convergent as N goes to infinity. Then

$$\lim_{N \to \infty} Q[N] = \sum_{i=1}^{\infty} 2^{-i}[(P(e))^{2^i} + (P(\bar{e}))^{2^i}]^m.$$

While this sum does not have an obvious closed form solution, it converges so quickly that a good estimate can be found by using only the first few terms of the sum. The expected percentages of nodes pruned for $m = 1$ and $m = 4$ are plotted in Figure 5.

In general, we are using unbalanced trees, so finding a closed form expression of the expected number of nodes pruned from any tree is difficult. Instead, we assume that we know the number of nodes at each level and again that any edge decision made at a leaf node is independent of other edge decisions and has probability of edge/no-edge assignment, $P(e)$ and $P(\bar{e})$ respectively, where $P(e) + P(\bar{e}) = 1$. Given any node n from the tree, we say that $T[n]$ is the number of nodes pruned from the tree descended from node n, and $P_n(e)$ and $P_n(\bar{e})$ are the probabilities that an edge decision at node n is an edge or not an edge respectively. Note that $P_n(e) + P_n(\bar{e}) \leq 1$ since it is possible that there may not be any edge decision due to conflict among the node's children. We can find the expected number of edges pruned from a tree with a known structure as follows. If node n is a leaf node,

$$P_n(e) = P(e), P_n(\bar{e}) = P(\bar{e}), T[n] = 0.$$

If node n is not a leaf node, then

$$P_n(e) = P_{n_l}(e)P_{n_r}(e), P_n(\bar{e}) = P_{n_l}(\bar{e})P_{n_r}(\bar{e}),$$

$$T[n] = T[n_l] + T[n_r] + 2[P_n(e) + P_n(\bar{e})]^m,$$

where n_l is the left child of n, n_r is the right child of n, and m is the number of edge decisions per node. In other words, the number of nodes pruned from a tree descended from node n is the number of nodes pruned from the trees descended from node n's children plus two times the probability that the two children of node n have the same edge decisions. Note that this is similar to the balanced case.

Results for TSVQs with 6×6 vectors at various rates are displayed in Table 3, where N is the number of nodes in the tree, $P(e)$ is the probability that an edge assignment at a leaf node is an edge, Q_p is the predicted percentage of nodes pruned, Q_a is the actual percentage of nodes pruned, and R is the effective rate of the TSVQ after pruning. While the analytic results of expected pruning may seem pessimistic, we have found that in practice the results of pruning far exceed our expectations. The reason is that in our analysis we have assumed for simplicity that all edge decisions are independent, even when there are multiple edge decisions per node. Since many of the edges we detect are 5 pixels wide, the actual edge decisions are highly correlated.

Figure 1: Original Image of Frame 2.

4. CONCLUSIONS

We have reduced the complexity of our fast facet edge detector in two ways. In the first, we increased the size of the VQ vector so that edge detection could be performed on multiple pixels at a time with lower encoder complexity. In the second, we pruned the TSVQ based on a new edge detection criterion that reduces storage requirements and further lowers encoder complexity without changing the performance. Finally, we presented a quantitative measure of edge detector performance for false alarm and misdetect rates relative to a ground-truth image generated by a human operator.

5. REFERENCES

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* The Wadsworth Statistics/Probability Series. Wadsworth, Belmont, California, 1984.

[2] A. Buzo, A. H. Gray Jr., R. M. Gray, and J. D. Markel. Speech coding based upon vector quantization. *IEEE Transactions on Acoustics Speech and Signal Processing*, 28:562–574, October 1980.

[3] P. A. Chou, T. Lookabaugh, and R. M. Gray. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Transactions on Information Theory*, 35(2):299 – 315, March 1989.

[4] R. M. Haralick. Digital step edges from zero crossings of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:58–68, January 1984.

[5] M. Y. Jaisimha, E. A. Riskin, and R. M. Haralick. Fast facet edge detection in motion sequences using vector quantization. In *Proceedings of ICASSP*, volume III, pages 441–444, 1992.

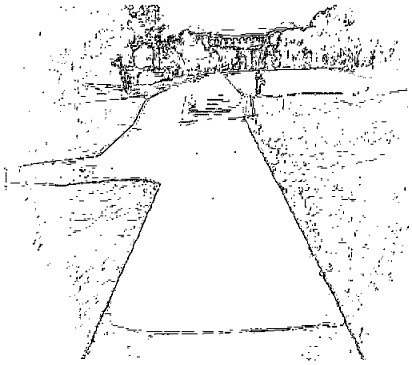

Figure 2: VQ-based edge detector at rate 6.8 bits per vector for Frame 2 using 6×6 vectors.

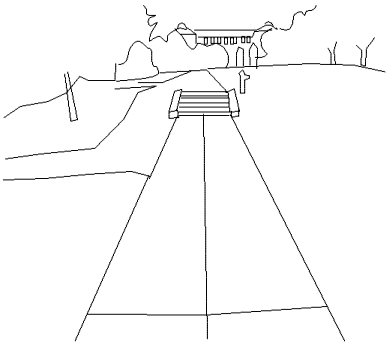

Figure 3: Ground-truth edge image for Frame 2.

Table 1: Comparison of edge detector errors for 6×6 TSVQ (6.8 bpv), original facet, and 5×5 TSVQ (6 bpv).

F	6×6 TSVQ		Facet		5×5 TSVQ	
#	P_{fa}	P_{md}	P_{fa}	P_{md}	P_{fa}	P_{md}
2	0.564	0.745	0.832	0.471	0.602	0.757
3	0.559	0.706	0.832	0.434	0.599	0.719
4	0.587	0.695	0.863	0.402	0.648	0.733
5	0.555	0.744	0.839	0.480	0.603	0.750

Table 2: 6×6 TSVQ edge detector errors vs. Rate for Frame 2.

Rate	P_{fa}	P_{md}
4	0.2740	0.9401
5	0.4366	0.8373
6	0.4418	0.7935
6.8	0.564	0.745
7	0.5870	0.7391
8	0.6208	0.6986
9	0.6918	0.6693

Table 3: Pruning Results for 6×6 TSVQ.

Rate	N	$P(e)$	Q_p	Q_a	R
4	345	0.5491	3.18	19.13	3.67
5	2501	0.4538	3.22	13.99	4.18
6	3583	0.4795	3.14	15.52	4.94
6.8	4919	0.4860	3.13	14.76	6.21
7	5347	0.4845	3.13	14.63	6.57
8	7717	0.4977	3.13	14.05	7.62
9	10587	0.5035	3.13	13.05	8.69

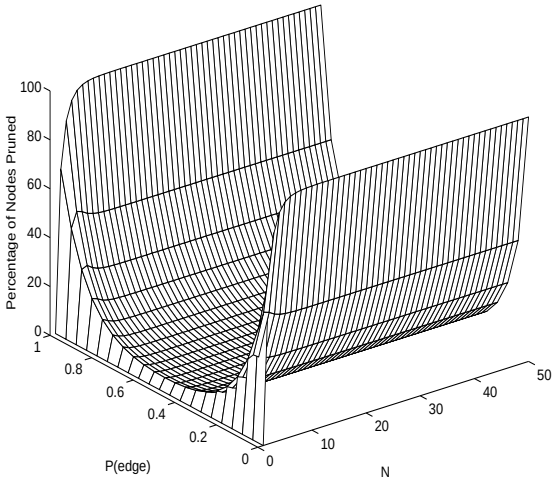

Figure 4: The expected percentage of nodes pruned vs. the depth of a balanced TSVQ with 6×6 vectors ($m = 4$).

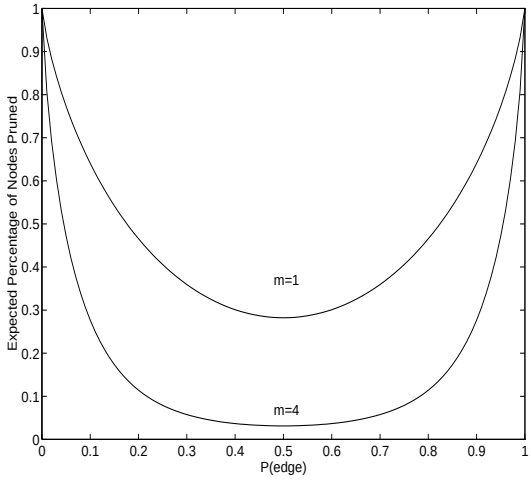

Figure 5: Expected percentage of nodes pruned from a balanced TSVQ with 5×5 vectors ($m = 1$) and 6×6 vectors ($m = 4$).