MULTI TARGET VIDEO TRACKING USING RESIDUAL VECTOR QUANTIZATION

Salman Aslam, Christopher Barnes, Aaron Bobick

Georgia Institute of Technolgy

ABSTRACT

In this paper, we use Residual Vector Quantization (RVQ) for multi-target tracking. To the best of our knowledge, this is the first reported application of RVQ to any form of video processing. We implement a completely automatic method of initializing RVQ encoder and decoder codebooks from targets in a crowded scenario, and then use those codebooks to subsequently detect and track targets.

Index Terms— Multi-target tracking, residual vector quantization, σ-tree classifier, pattern recognition, machine learning, source coding, compression.

1. INTRODUCTION

Multi-target tracking is one of the most complex tasks in video analytics. In many cases, it is the first step in more complex processing tasks, such as temporal recognition and classification. Many researchers have worked in this area.

1.1. σ-tree Classifier and Residual Vector Quantization

In this paper, we use the Image Driven Data Mining approach (IDDM)introduced in [1]. This approach is based on the σ-tree for data warehouse similarity searching within the pixel-block space of feature classes, and for extracting labels from knowledge base collections or ”aggregates”. These collections, in this paper are foreground blobs corresponding to tracked targets. A key difference is that the data warehouse is built on-line, and is updated every few frames as new observations of the targets under interest arrive. Another difference is that temporal correlations are used for unsupervised learning of target labels. To better understand the σ-tree classifier, we relate it with five well-known areas of information theory, pattern recognition and machine learning.

First, the σ-tree classifier is a type of variable rate vector quantizer [2]. Second, Truncel *et al* explore the relationship between rate-distortion theory and content-based retrieval in high dimensional databases [3]. Third, the σ-tree design problem parallels the joint codebook design problem addressed in the source coding literature for multiple stage vector quantization [4]. Fourth, the σ-tree classifier can be compared to dimensionality reduction methods such as Principal Components Analysis (PCA). PCA seeks to reorient the basis vectors in $\mathbb{R}^k$ and achieves compression by ignoring projected data components with least variances. Even though a σ-tree classifier can be used to achieve lower dimensionality, it primarily seeks to partition $\mathbb{R}^k$ like other well known classifiers such as neural networks and support vector machines [1]. Further, note that neural networks partition the decision space with hyperplanes or hypersurfaces, depending on whether or not hidden layers are used. Support vector machines also do partition the decision space, but with hyperplanes in a higher dimension space that separate the data with maximum margin. The σ-tree classifier tesselates the decision space $\mathbb{R}^k$ with k Voronoi cells, the geometric dual of delaunay triangulation in $\mathbb{R}^2$. The fifth and final comparison is with the well-known k-means algorithm, or the Linde Buzo Gray (LBG) algorithm as it is more commonly referred to in the source coding literature. It is also called the Generalize Lloyd Algorithm (GLA).

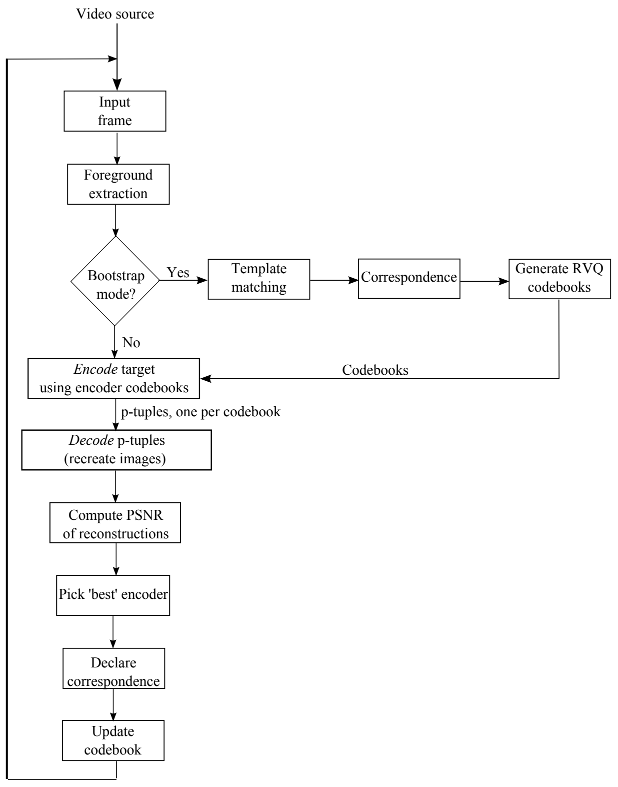


Fig. 1. Block diagram.

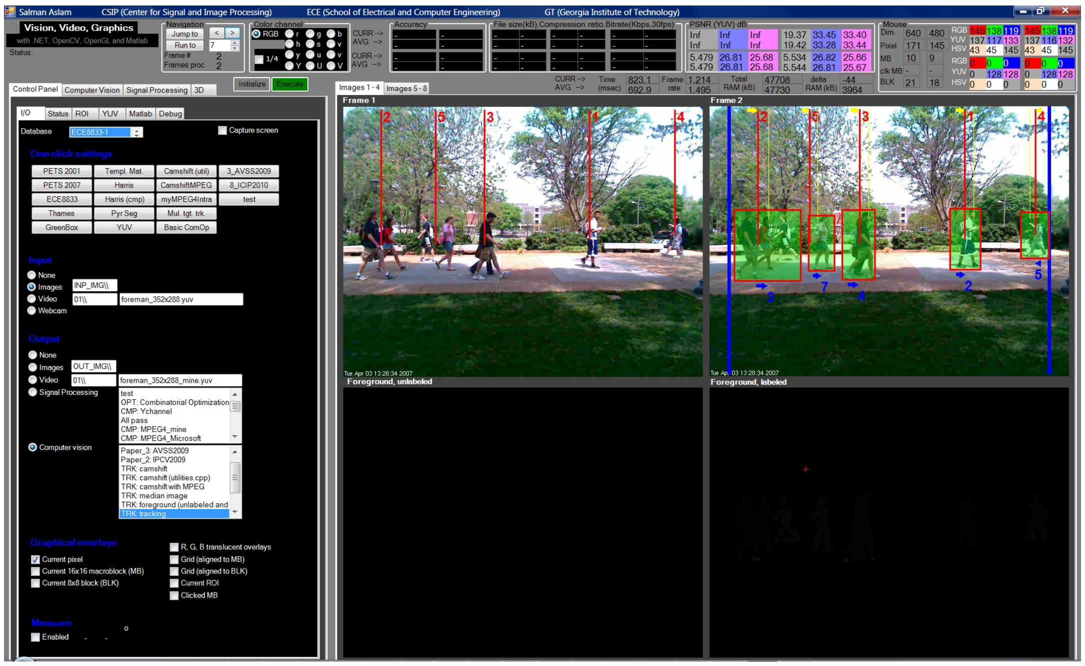


Fig. 2. Software.

Most vector quantization methods use this algorithm to create cluster centroids [5]. Indeed, the σ-tree classifier centroids can also be designed using this algorithm. For a concrete and simple example of a σ-tree, see [1].

The mechanism that allows a σ-tree to be built is multi-stage Vector Quantization (VQ), also called Residual Vector Quantization (RVQ) or Direct Sum Successive Approximation (DSSA) Vector Quantization. These three terms will be used interchangeably in this paper. Juang and Gray first proposed the RVQ structure [6] and suggested that RVQ stages be designed by sequential application of GLA. The main shortcoming of this approach is that each stage codebook is generated while considering only the error due to the previous stages (the *causal error*); the error due to subsequent stages (the *anticausal error*) is ignored. A joint design approach on the other hand takes into account both the causal and anticausal errors to reduce the *total error*. Various techniques have been proposed for joint optimization. We now introduce notation and define ”joint optimality.”

The pth stage of a P-stage RVQ, with N codevectors per stage, is a k-dimensional VQ defined by the mapping

$$Q_p : \mathbb{R}^k \mapsto C_p \tag{1}$$

Here, $\mathbb{R}^k$ is the k-dimensional input space,

$$C_p = \{y_p(0), y_p(1), \ldots y_p(N_p - 1)\} \tag{2}$$

is the pth-stage codebook and $y_p(i_p) \in \mathbb{R}^k$ is a pth-stage *codevector*. Residual quantizer stage mappings Q_p are collectively equivalent to a single mapping,

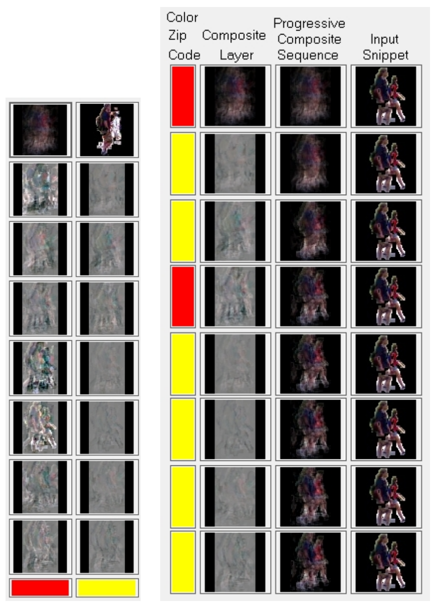
$$Q : \mathbb{R}^k \mapsto C \tag{3}$$

Here, C is the *direct sum codebook* and is the direct sum of the stage codebooks $C = C_1 + C_2 + \ldots C_p$. In practice, the stage mappings $Q_p(.)$ are realized as a composition of a stage encoder mapping $E_p = \mathbb{R}^k \mapsto I_p$, and a stage decoder mapping $D_p : I_p \mapsto C_p$, that is $Q_p(x_1) = D_p[E_p(x_1)]$, where x_1 is a realization of X_1, which is the random source output.

An RVQ is said to be jointly optimal if a local or global minimum value of the average distortion $d(E, D) = E[m(X_1, D(E(X_1)))]$ is achieved, where $m(.,.)$ is a distortion metric, and E[.] is the expectation operator. This shows that an RVQ is a product code VQ with a direct sum codebook structure [7]. Also notice that computation (search) and memory (storage) requirements are proportional to kNP. So, the cost functional grows linearly with classifier dimensionality k, branch multiplicity N, and the number of tree levels P. However, the number of indexing options is N^P. This computational advantage makes it possible to use RVQ in real time tracking, even though the data dimensionality is very high.

1.2. Background Subtraction

Several preprocessing steps may be required before tracking can be initiated. These could include image stabilization, normalization, and downsampling. The final preprocessing step in most cases is background modeling. Several algorithms have been developed for this task. Most of these algorithms



(a) Decoder codebooks. (b) Reconstructing target using direct sum successive approximation templates.

Fig. 3. Using decoder codebooks to reconstruct targets.

depend on appearance modeling. A detailed overview alongwith a graphical comparison of how these algorithms perform on the canonical problems in foreground detection is given in [8]. In this paper, we use the Multi Gaussian algorithm [9]. In this algorithm, the grayscale value of each pixel is tracked over time and represented as a mixture of K Gaussian distributions. The mean and variance of the Gaussians is tracked over time. This is a pixel based approach with no region based processing. However a post processing step of morphological operations, connectedness constraints, and area thresholding makes the combined approach quite robust to slow drifts in appearances. The probability of observing the current pixel value at time, x_t, is

$$p(x_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{4}$$

where K is the number of distributions, $\omega_{i,t}$ is an estimate of the weight of the ith Gaussian in the mixture at time t. The gaussian distributions are given by,

$$\eta(x_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1}(X_t-\mu_t)} \tag{5}$$

The pixel process is not stationary, and so one cannot resort to maximizing the likelihood of the observed data using the Expectation Maximization (EM) algorithm. Instead, the weights of the Gaussians are adjusted as follows,

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha M_{k,t} \tag{6}$$

where α is the learning rate, and $M_{k,t}$ is 1 for the model which matched and 0 otherwise. A match is defined as a pixel within 2.5 standard deviations of a distribution. In this paper, we use a value of $K = 3$.

2. EXPERIMENTS

We track multiple targets using multi-stage, direct sum successive approximation RVQ. A block diagram of our approach is shown in Figure 1 while a sample tracking scenario is shown in Figure 2. We track several targets in a dense environment. Background maintenance is done using a Multi Gaussian background subtraction algorithm. Tracking intialization is done using template matching. When enough training examples of a particular target have been attained, they are input into a multi stage residual vector quantizer to train the encoder and decoder codebooks. During the bootstrapping process, this process is repeated for all targets. Subsequent foreground objects are tested against the codebooks of every target. The codebook that most explains the target is chosen. This is equivalent to Probabilistic Multiple Hypothesis Testing (PMHT) with hard decisions. In subsequent work, we will demonstrate soft decisions alongwith temporal evolution of the codebooks in real-time. In this paper, we limit ourselves to demonstrating the effectivness of using RVQ in the multi-target tracking scenario.

For this purpose, we have developed the software shown in Figure 2. It is integrated with Matlab and Intel OpenCV for basic mathematical, image processing and visualization purposes. Where more elaborate 3D visualization is required, for instance in viewing error surfaces, where each point on the surface corresponds to a pixel in an image, OpenGL is used for rendering a height map over the texture mapped image. The software GUI is developed in Visual Studio.NET.

3. RESULTS

Figure 3(a) shows the direct sum templates of the decoder codebook. We use 2 codevectors per stage ($N = 2$), and a total of 8 stages ($P = 8$). Although this number is somewhat arbitrary, empirical analysis shows this to be a reasonable setting. A total of N^P, or 256 unique images can be represented with this codebook, but the computational cost of computation and storage is $O(NP)$ rather than $O(N^P)$. Once the codebooks have been created for time $t = 1$ to $t = T$, subsequent targets at times $t = T+1, T+2, \ldots$ are encoded using the codebooks. The codebook that results in the least reconstruction error is chosen and the target corresponding to that codebook is matched. Figure 3(b) shows reconstruction of a target using the codebooks. In Figure 4 we show the PSNR value of target reconstruction in a window centered around the target. PSNR values are low if the target under test and the codebooks being used are mismatched. However, if they are matched, the PSNR is sufficiently high for unambiguous correspondence. This is shown in tabular form in Table 5 where PSNR values at the center of the targets are reported.

4. CONCLUSIONS

We demonstrate the first reported application of residual vector quantization to any form of video processing. In this paper, the application we focus on is multi-target tracking. We demonstrate initial results of a completely automatic method of training RVQ encoder and decoder codebooks, and then using them to detect targets. We show that this method can track targets in a difficult multi-target tracking scenario. In future work, we will focus on temporal evolution of the codebooks to generate time evolving desriptions of the targets under test.

5. REFERENCES

[1] C. F. Barnes, "Image-driven data mining for image content segmentation, classification, and attribution," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, pp. 2964–2978, 2007.

[2] Riten Gupta, Alfred Hero, Alfred, and O. Hero, "High rate vector quantization for detection," *IEEE Transactions on Information Theory*, vol. 49, pp. 1951–1969, 2003.

[3] E. Tuncel, P. Koulgi, and K. Rose, "Rate-distortion approach to databases: storage and content-based retrieval," *Information Theory, IEEE Transactions on*, vol. 50, no. 6, pp. 953–967, 2004.

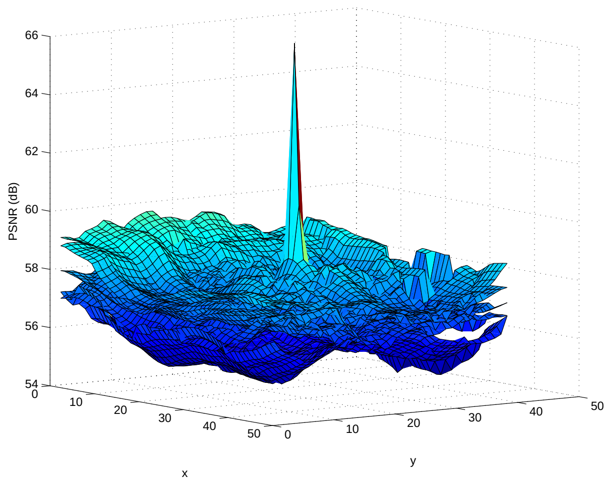
[4] C. F. Barnes, S. A. Rizvi, and N. M. Nasrabadi, "Advances in residual vector quantization: a review," *Image Processing, IEEE Transactions on*, vol. 5, no. 2, pp. 226–262, 1996.

[5] Allen Gersho and Robert Gray, *Vector Quantization and Signal Compression (The Springer International Series in Engineering and Computer Science)*, Springer, 1991.

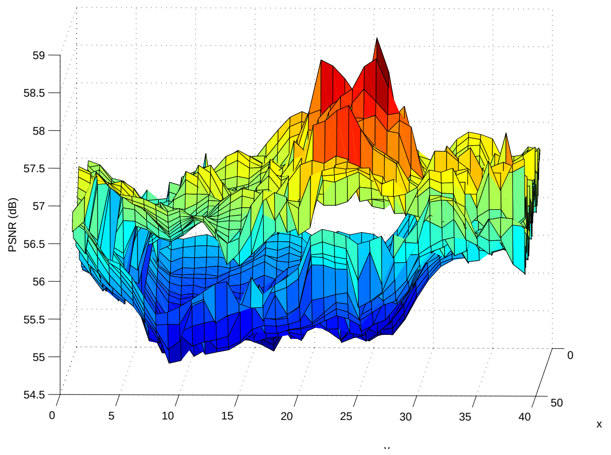
[6] Juang Biing-Hwang and Jr. Gray, A., "Multiple stage vector quantization for speech coding," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, 1982.

[7] C. F. Barnes and R. L. Frost, "Vector quantizers with direct sum codebooks," *IEEE Transactions on Information Theory*, vol. 39, no. Copyright 1993, IEE, pp. 565–80, 1993.

[8] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers, "Wallflower: Principles and practice of background maintenance," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV'99)*, 1999.



(a) For a given codebook C_i, a target from within the training set used to generate C_i produces a high PSNR as compared to targets that do not belong to the training set.



(b) Here the target under test was not used to generate the codebook C_i. Nevertheless, it belongs to the same class as C_i but is 3 frames temporally advanced, i.e. $t = T + 3$ where C_i was generated over $t = 1 \dots T$. Notice that it still has significantly higher PSNR than targets outside the class.

Fig. 4. Computing PSNR for target reconstruction.

	Codebook		
	Target 2	Target 5	Target 9
Target 1	58.77	59.66	60.15
Target 2	66.06	59.40	59.63
Target 3	59.72	59.47	60.10
Target 5	57.92	63.73	57.85
Target 9	59.14	58.41	62.44
Max diff	6.34	4.07	2.29

Fig. 5. Maximum PSNR values, in dB, of all targets for given codebooks. Also shown is the difference between the two highest PSNR values.

[9] Chris Stauffer, W. Eric, and Eric, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 747–757, 2000.