
# Automatic visual classification of events in underwater video

**Marco Aurélio G. Moreira, Universidade Federal de Minas Gerais**
***Mentors: Duane Edgington and Danelle Cline***
***Summer 2009***

**Keywords: AVED, object recognition, classification, color**

## ABSTRACT

This report addresses the problem of visual event classification. To do that we use a set of local descriptors invariant to rotation and scale. Each image is described by one feature vector. First, a set of invariant descriptors with respect to rotation and shift are computed for each pixel at different space scales. Then, through nonlinear operations, they are mapped to a higher dimensional space, where they are averaged. Fisher's Linear Discriminants are used to reduce the dimensionality of the feature vectors representing each image. Then the feature vectors distributions, for each class of events, are modeled using Mixtures of Gaussians. Differently from previous approaches, we take also color information into account when computing the feature vectors.

## I. INTRODUCTION

The classification of events in underwater video is a difficult task, even for experts. There are a lot of variables that influence the viewed image, such as lighting conditions and the animal pose, size, and distance from the camera. The annotator has also to deal with distracting particles, such as marine snow. When designing an algorithm for automatic detection and classification of events, all of these variables have to be taken into account.

The input for the classifier is the output of the AVED system (Automated Visual Event Detection) (Walther et. Al, 2004). The AVED system detects and tracks potential interesting events. New objects are detected using a saliency-based bottom-up attention system. The saliencies are computed using several channels, such as intensity contrast and blue/yellow double color opponencies. Once detected, the objects are tracked using linear Kalman filters. The sequence of frames in which an object is tracked constitutes an event. The goal of this project is to assign a proper class to each of the events.

## II. METHODS

To perform the visual classification, we use an algorithm based on and very similar to the algorithm presented in [3]. The following subsections present the key steps of the algorithm. To avoid repetition of the original paper, we try to present the concepts in a more intuitive rather than mathematical way.

### A. LOCAL INVARIANT DESCRIPTORS

Taken a point **p** from an object in an image **I** and its corresponding point **p'** in an image **I'**, there exist some local properties intrinsic to the point which are invariant to shift and rotation. The local average luminance and the square of the gradient magnitude are easy examples (the gradient of an image in a point is the vector which points in the direction of greatest increase rate of intensities and whose magnitude is the increase rate).

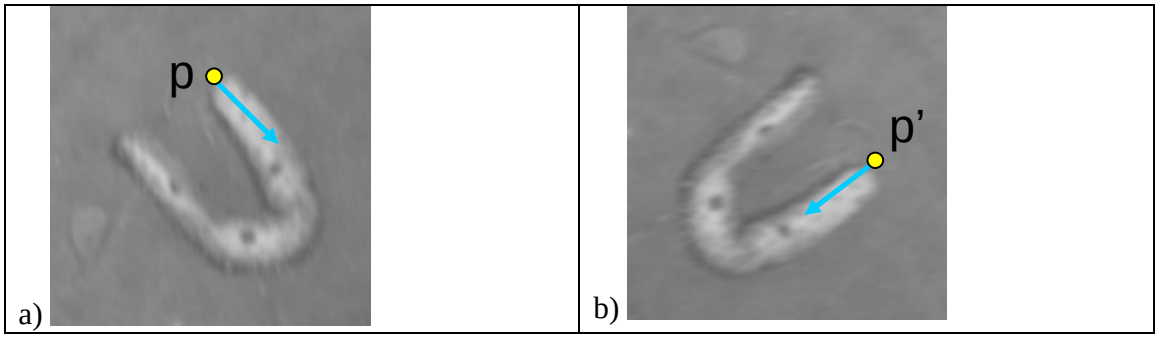
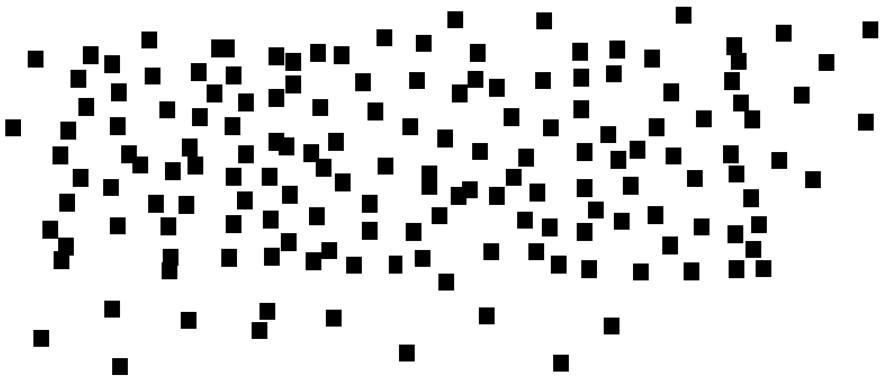



Figure 1: Image **b** represents image **a** rotated 90º clockwise. The points p and p' share common properties, such as the local average luminance and the square of the gradient magnitude (squared length of the blue vector).

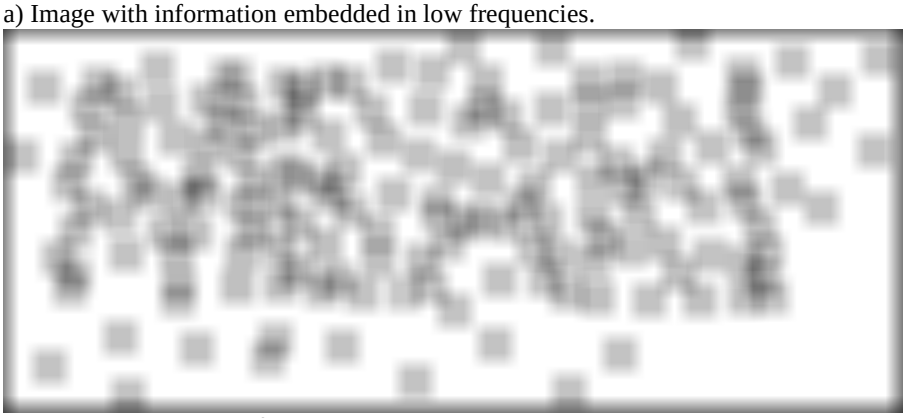
Schmid and Mohr [2] define a set of 9 local descriptors based on image derivatives which are invariant to shift and rotation, among them are the invariants depicted in Figure 1. The invariants are the same in **p** and **p'**, but the problem is that the position of **p'** in **I'** is unknown. The solution is to compute the average value of the invariants over the object of interest. It means that we somehow have to be able to estimate and discount the background information. The background information is estimated in a 10 pixel wide region along the image borders.

Another important variable that we have to deal with is scale. Scale changes in an animal class may occur because of the animal size or due to its distance from the camera. According to [3], the feature vectors are also scale-invariant, because of the average operation.

To capture different levels of detail, the invariants are computed at various space scales **t**. For each scale **t**, the original image is convolved with a Gaussian kernel before the computations. The widths of the kernels are t = {0, 0.5, 1, 2}. Figure 2 illustrates how low frequency information can be highlighted by blurring operations.



a) Image with information embedded in low frequencies.



b) The blurred version of image a.

Figure 2: effects of gaussian blurring.

## B. ADDING NONLINEARITIES

Ranzato [4] observed that adding some nonlinearities to the invariants can improve the performance of the classifier. Several nonlinearities were tested and the one which gave better results is described by:

$$F1(x, y) = f(x, y) - BG, \text{ if } f(x, y) > BG \quad \text{positive part}$$
$$= 0, \text{ otherwise}$$

$$F2(x, y) = BG - f(x, y), \text{ if } f(x, y) < BG \quad \text{negative part}$$
$$= 0, \text{ otherwise}$$

$$F3(x, y) = |f(x, y) - BG| + BG, \quad \text{absolute part}$$

where BG is the background mean, computed over the pixels that fall within a given distance from the image borders; f(x, y) is an invariant; and F1, F2 and F3 are, respectively, the positive, negative and absolute parts of the invariants with respect to the background. The complete monochromatic feature extractor is shown in Figure 3.

After the computations, we have 9 x 4 x 3 = 108 values to describe each image. Comparing to a typical image matrix (100 x 100 pixels, for example), we have reduced significantly the amount of data to be analyzed. Besides that, each position of the new

feature vector has a proper meaning. However, to design a classifier, we still have to reduce the dimensionality of the feature vector.

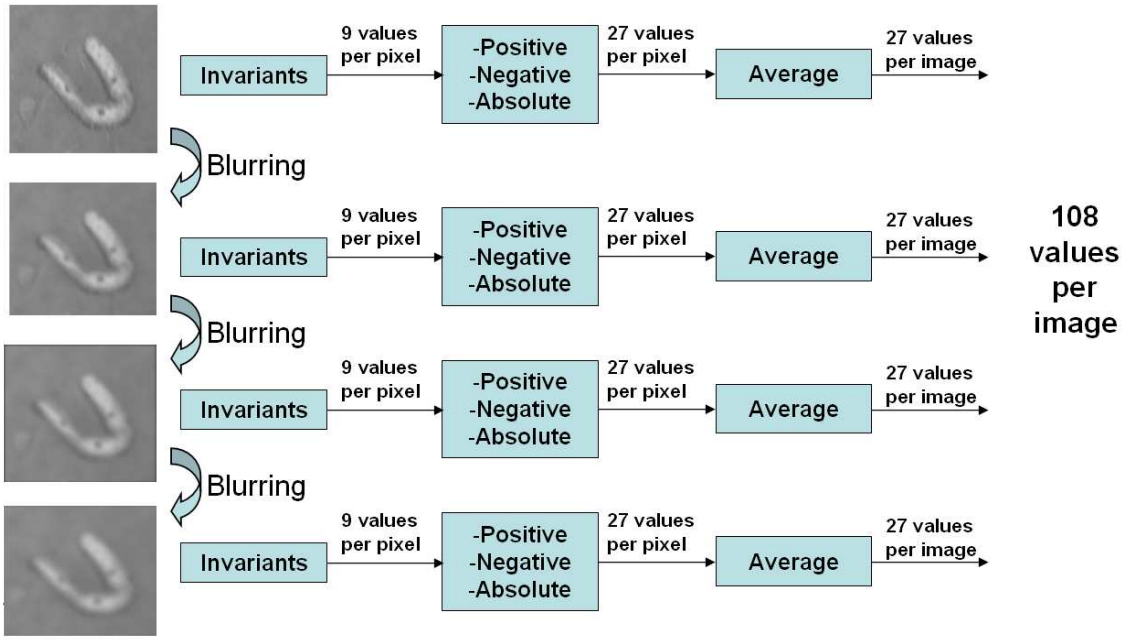



Figure 3: Monochromatic feature extractor

## C. FISHER'S LINEAR DISCRIMINANT (FLD)

To transform a data set from an N-dimensional space to a D-dimensional space, D<N, the Fisher's Linear Discriminant algorithm maximizes the distance between the means and minimizes the variance inter-class. One important constraint of the method is that we are unable to find more than K-1 linear features, where K is the number of classes [5]. To overcome this limitation, Ranzato [4] proposes to split the data of each class in a proper number of subcategories.

## D. ADDING COLOR INFORMATION

To consider color information, we run the monochromatic feature extractor three times, one in each image layer. The new feature size is then 108 x 3 = 324. The first attempt was to pre-convert the original images from RGB to YCrCb color space. By doing this, we separate the luminance values (Y component) from the chrominance values (Cr and Cb channels). The Y channel is similar to the grayscale image. Hence the feature vector contains the same information as in the monochromatic case plus color information. However, it turned out that using the original RGB color space gave us better results. As before, we run FLD to reduce dimensionality of the extracted features.

## E. TRAINING

After collecting the feature vector in the 324-dimensional space, they are projected in a D-dimensional space using FLD. Then a mixture of Gaussians (MoG) probabilistic model is adjusted for each class using the NETLAB package for Matlab [6]. Several parameters must be chosen for the training stage:

- dimension of the final feature space;
- type of covariance matrix;
- number of Gaussians for the MoG model;

F. TESTING

For each frame we compute the probability of the event to belong to each class. The probabilities are normalized, yielding the sum of all the probabilities equal to one. If the maximum probability is greater than a threshold (fixed to 0.9), then that frame is assigned to the corresponding class; otherwise the frame class is said to be unknown. The majority rule is then used to assign the event to the winner class.

## III. RESULTS

To assess the performance of the algorithms we use a dataset with 1087 events: 69 “” (5288 images), 840 “other” (5939 images) and 178 “rathbunaster” (1022 images). These classes are shown in Figure 4. For each class we select 90% of the events for training and 10% for testing. We do this 100 times and then compute the average performance of the algorithm.


Rathbunaster


Parastichopus


Other

Figure 4: Classes used for assessing the performance of the algorithms.

The original algorithm uses Grayscale color space, reduction of dimensionality from 108 to 8 features, 4 gaussians and full covariance matrices (Table 1). We see from Table 2 that adding color information significantly improves the results for the class “Parastichopus”. In our tests, the classifier was not very sensitive to changes in parameters, as long as the color space remains the same.

We cannot visualize the feature vectors distribution in more than 2 dimensions. However, if we show that individual features are separable in one dimension, they will also be important to separate classes in N-dimensions, N>1. Figure 5 shows the 1-D independent feature distribution when using RGB color space and reducing the dimensionality from 324 to 8 features. Only the 2 first features seem to be important to

distinguish among the classes. This can be related to the restriction of the FLD algorithm. Since we are using only 3 classes, FLD is unable to find more than 2 linear features. This can also indicate that splitting classes into subcategories may be mathematically feasible but does not add much information to find extra features. Similar results are extracted when we use images in the Grayscale or YCrCb color spaces. Results in tables 1 to 5 also support this proposition.

Supposing that we cannot find more than 2 linear features using FLD, we show the features distribution for the three classes in figure 6 and 7, using, respectively, Grayscale and the RGB color space. Comparing the features distributions, we see that adding color reduces the covariance interclass and increases the distance between classes.

| Event-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 91.667 | 0.000 | 0.262 |
| Paras. | 0.000 | 68.571 | 1.393 |
| Other | 2.833 | 8.000 | 79.024 |
| Unk. | 5.500 | 23.429 | 19.321 |

| Frame-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 91.667 | 0.245 | 2.462 |
| Paras. | 0.065 | 63.523 | 3.885 |
| Other | 3.486 | 8.196 | 66.098 |
| Unk. | 4.782 | 28.037 | 27.555 |

Table 1: Confusion matrix using Grayscale color space, 8 features, full covariance matrices and four Gaussians.

| Event-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 95.611 | 0.000 | 0.274 |
| Paras. | 0.000 | 92.857 | 0.476 |
| Other | 0.333 | 3.000 | 96.571 |
| Unk. | 4.056 | 4.143 | 2.679 |

| Frame-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 96.278 | 0.089 | 1.122 |
| Paras. | 0.005 | 93.365 | 0.744 |
| Other | 0.535 | 1.074 | 89.539 |
| Unk. | 3.181 | 5.472 | 8.596 |

Table 2: Confusion matrix using YCrCb color space, 3 features, full covariance matrices and four Gaussians.

| Event-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 99.389 | 0.000 | 0.321 |
| Paras. | 0.000 | 94.286 | 0.512 |
| Other | 0.333 | 4.286 | 97.857 |
| Unk. | 0.278 | 1.429 | 1.310 |

| Frame-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 97.988 | 0.186 | 0.540 |
| Paras. | 0.019 | 94.019 | 1.073 |
| Other | 0.795 | 2.636 | 92.978 |
| Unk. | 1.198 | 3.159 | 5.409 |

Table 3: Confusion matrix matrix using RGB color space, 8 features, full covariance matrices and four Gaussians.

| Event-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 100.000 | 0.000 | 0.286 |
| Paras. | 0.000 | 94.857 | 0.988 |
| Other | 0.000 | 3.286 | 96.940 |
| Unk. | 0.000 | 1.857 | 1.786 |

| Frame-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 98.390 | 0.608 | 1.007 |
| Paras. | 0.050 | 93.416 | 1.211 |
| Other | 0.716 | 1.294 | 93.741 |
| Unk. | 0.844 | 4.681 | 4.042 |

Table 4: Confusion matrix using RGB color space, 3 features, diagonal covariance matrices and one Gaussian.

| Event-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 99.556 | 0.000 | 0.298 |
| Paras. | 0.000 | 96.286 | 0.750 |
| Other | 0.000 | 3.571 | 96.655 |
| Unk. | 0.444 | 0.143 | 2.298 |

| Frame-wise | | | |
|---|---|---|---|
| Est\True | Rath. | Paras. | Other |
| Rath. | 98.119 | 0.388 | 0.857 |
| Paras. | 0.065 | 95.747 | 1.400 |
| Other | 0.688 | 1.131 | 92.520 |
| Unk. | 1.128 | 2.734 | 5.223 |

Table 5: Confusion matrix using RGB color space, 2 features, full covariance matrices and one Gaussian.

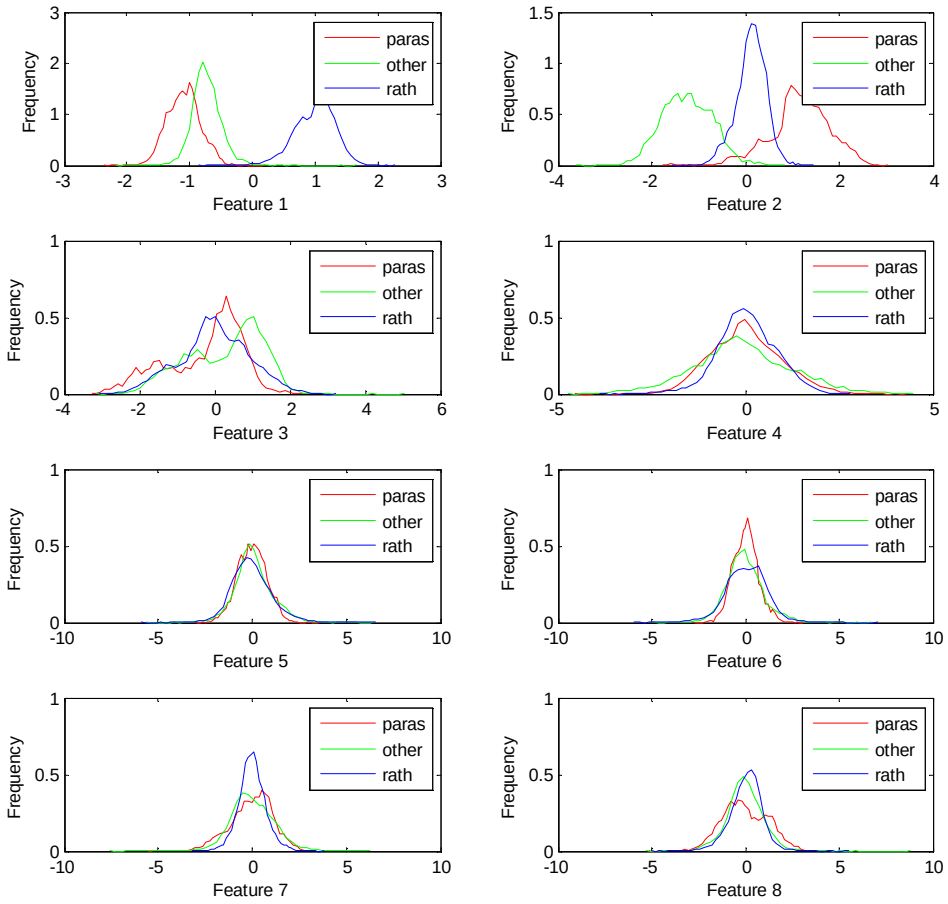



Figure 5: 1-D independent feature distribution using the RGB color space reducing dimensionality from 324 to 8 features.

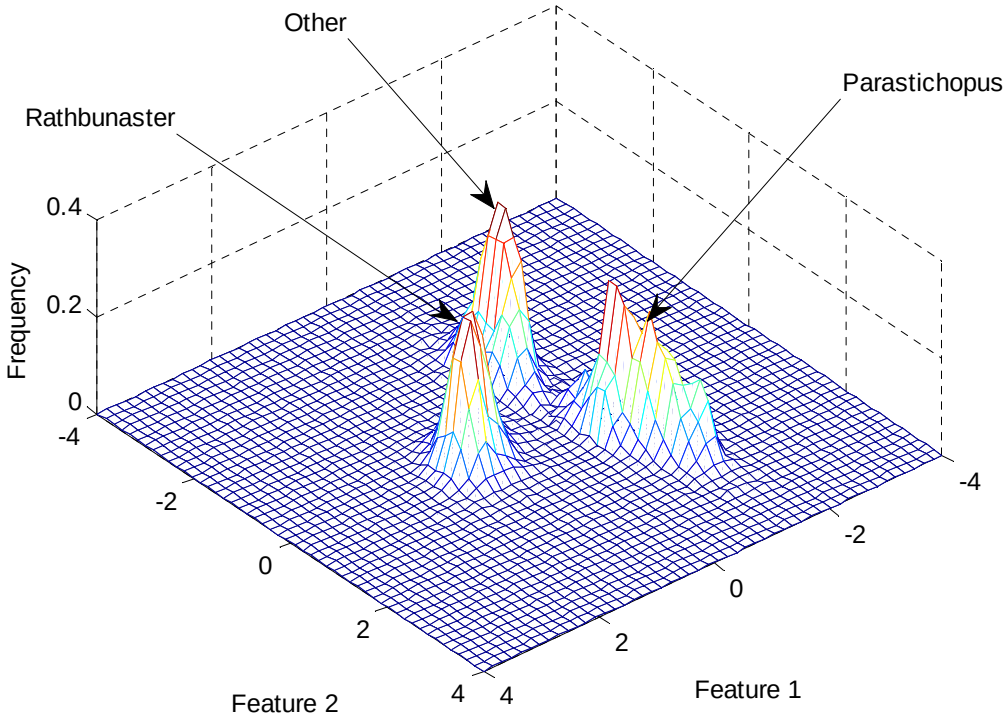



Figure 6: Feature Distribution using RGB color space and reducing dimensionality from 324 to 2 features.

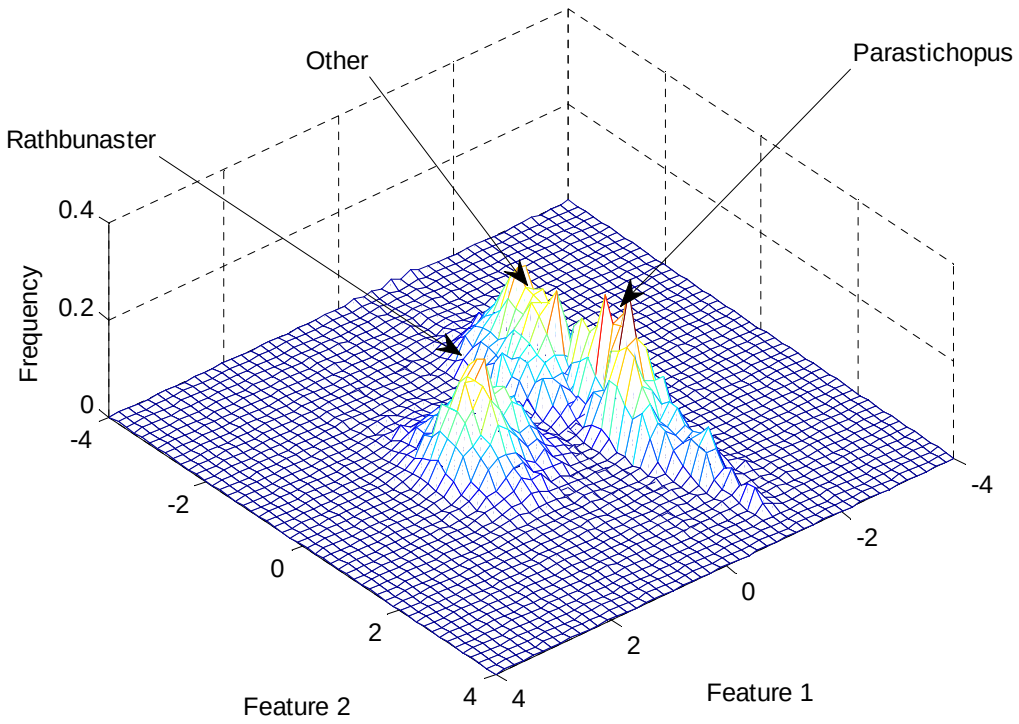



Figure 7: Feature Distribution using only grayscale information and reducing dimensionality from 108 to 2 features. Overlap occurs between classes “Other” and “Parastichopus”.

## IV. CONCLUSIONS/RECOMMENDATIONS

The automatic classification of animals in underwater video via computer vision is a relatively hard task, because animals can be subjected to all kinds of rigid and non rigid transformations. Despite of that, we have shown a good performance using a modified version of a classifier originally designed to distinguish biological particles. We have shown that adding color information can substantially improve the correctness of the classifier. Also we showed that only 2 numbers (features) were sufficient to describe the current classes.

The classes we worked with are very different to each other. When dealing with a greater number of classes, more strong similarities may arise and we may need to use extra descriptors. Several techniques have been studied to create mathematical tools invariant to scale, rotation and translation, such as SIFT descriptors, Fourier-Mellin transform and Hu Moments. These techniques can be combined to Ranzato’s descriptors and would probably add robustness to the current classifier.

Another option when dealing with few classes is to design a classifier specific for each class. Thus we would have, for example, a classifier to distinguish “Rathbunaster” from “Other” and a different classifier do distinguish between “Parastichopus” and “Other”. This way one could concentrate on the particular important characteristics of each class.

In spite of the simplicity of the task accomplished by our classifier, the reached results encourage us to think that it can be used with AVED to automate detection, tracking and classification of underwater events in well controlled environments, such as the Station M images [7].

**ACKNOWLEDGEMENTS**

I would like to thank my mentors, Danelle Cline and Duane Edgington, for their support, directions, suggestions and patience. Thanks to Marc'Aurelio Ranzato for his suggestions and source code. Thanks to George Matsumoto and Linda Kuhnz for preparing and coordinating this wonderful internship program. And thanks to all the interns for the unforgettable summer.

**References:**

[1] Walther, Dirk, D. R. Edgington, and C. Koch (2004). Detection and Tracking of Objects in Underwater Video. cvpr, vol. 1, pp.544-549, 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04) - Volume 1, 2004.

[2] Schmid, C., and R. Mohr (1997). Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19: 530 - 535

[3] Ranzato, M., P.E. Taylor, J.M. House, R.C. Flagan, Y. LeCun, and P. Perona (2007). Automatic Recognition of Biological Particles in Microscopic Images. *Pattern Recognition Letters*, 28: 31-39.

[4] Ranzato, M. Automatic Visual Recognition of Biological Particles (2004). Master Thesis.

[5] Bishop, C. M. (2006) Pattern Recognition and Machine Learning. Springer-Verlag.

[6] www.ncrg.aston.ac.uk/netlab/

[7] Cline, D. (2008). Demonstration Study on Applying AVED to Still Images from Station M.