
VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF SCIENCE
FALCULTY OF MATHEMATICS AND COMPUTER SCIENCE



FISH SPECIES CLASSIFICATION USING COLOR AND LOCAL FEATURES

PATTERN RECOGNITION



SUBMITTED BY:

Student Name

Student ID

Dương Quốc Đạt

1511058

Nguyễn Nhứt Sâm

1511261

JANUARY 1, 2018

Under the Guidance of

Dr. Trần Anh Tuấn

ACKNOWLEDGEMENT

This project would not have been possible without the support of our lecturer Dr. Trần Anh Tuấn who has provided us with many helpful knowledge and skills throughout the semester to help complete this project.

Contents

Acronyms and Abbreviations	4
List of Figures	5
Abstract	6
CHAPTER 1: INTRODUCTION	7
1.1 Motivation.....	7
1.2 Problem Description	7
1.3 Organization.....	7
Chapter 2: LITERATURE REVIEW	8
1. Introduction.....	8
2. Process for Fish Species Classification.....	8
3. Preprocessing	8
3.1 K-Means clustering algorithm	8
4. RGB - Color Feature	13
5. Overview of “Bag of Visual Words” Model (BoVW).....	13
5.1 The concept.....	13
5.2 Descriptors	14
5.3 Probability And Training Models	14
6. Bag of Visual Words in Fish Classification.....	15
6.1 Speeded Up Robust Feature (SURF)	15
6.1.1 Detection	15
6.1.2 Description.....	19
6.1.3 Implementation	20
6.2 Bag of Visual Words Model	22
6.3 Classifiers.....	23
Chapter 3: MEDTHOLOGY.....	24
1. Naïve Bayes Classifier.....	24
2. K-Nearest Neighbor Classifier (KNN).....	25
3. Support Vector Machine (SVM).....	27
4. Artificial Neural Network (ANN).....	29
Chapter 4: EXPERIMENTAL RESULTS.....	31
1. Dataset Overview	31
2. Experiments	32

2.1	Classification Methods.....	32
2.2	Dictionary Size.....	32
2.3	Parameters of the Classifiers.....	32
2.4	Results and Discussion	32
REFERENCES		34

Acronyms and Abbreviations

BoVW	Bag of Visual Words
SIFT	Scale – Invariant Feature Transform
SURF	Speeded Up Robust Feature
KNN	K-Nearest Neighbor
SVM	Support Vector Machine
ANN	Artificial Intelligence

List of Figures

Figure 1 (Left) Image before segment, (Right) Image segmented with K-Means	11
Figure 2 3 octaves with 3 levels.....	14
Figure 3 $L_{xx}(x, \sigma)$ and $L_{xy}(x, \sigma)$ Discretized Gaussians and the approximation D_{yy} and D_{xy}	16
Figure 4 Where SIFT (left) down-scale the image, SURF(right) uses larger and larger filters.	16
Figure 5 The scale(σ), the filter sizes and octaves on a logarithmic scale	17
Figure 6 A 20s areas is divided into 4×4 subareas that are sampled 5×5 times to get the wavelet response	18
Figure 7 The wavelet response. Black and white areas corresponds to a weight -1 and 1 for the Haar kernels. They are used with a filter size of 2s	19
Figure 8 Graphical representation of the BoVW model	20
Figure 9 Example of k-NN classification	25
Figure 10 Example of decision boundary and margin of SVM.....	27
Figure 11 Nonlinear SVMs decision boundary	27
Figure 12 An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain	29
Figure 13 Illustration of all species used in this project and its popular and scientific names.....	30
Figure 14 Description of Fish dataset	31
Figure 15 Words equal 64.....	33
Figure 16 Words equal 128.....	33
Figure 17 Words equal 256.....	33
Figure 18 Wrods equal 512.....	33

Abstract

In this project we will be looking into visual classification of objects, specifically fish species classification using visual features. This is a highly important area of computer vision and there is a lot of research on this topic as it has many applications in the real world. The feature extraction methods were Speeded Up Robust Feature (SURF) with Bag of Visual Words and RGB color feature. For the species classification, three methods were used for comparison: Naïve Bayes, Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Artificial Neural Network (ANN). In the experiments several parameters for the classifiers were tested in order to find the best results for classification.

CHAPTER 1: INTRODUCTION

1.1 Motivation

The aim of this project is to further develop an application for classifying fish species which can be use for many different purposes and in different locations such as to identify different types of fish species in an aquarium, classifying fish prolem in fish factories,...

1.2 Problem Description

The identification of fish species is not easy, even for humans. M. S. Nery reported in his paper that fish have , at least, 47 different characteristics that can be used in this proccess. Based on previous research, our approach is to set about creating a method to robustly and accurately classify fish species images using the Bag of Words principle. Images will be broken down to their component features, or keypoints and compared against the keypoints in the training data or ‘codebook’. This comparison aims to reveal the most probable image category that an object belongs to.

1.3 Organization

The rest of the report is organized as follows,

- In Chapter 2, we will present the introduction, the methods and process as well as the disussion about different techniques and technologies that could be used.
- In Chapter 3, we will explain more in detail about the algorithms used in the classifying process.
- In Chapter 4, we will present the data, the results and the comparision between different methods we used.
- In Chapter 5, we will dissuss about the benefits and limitations of the methods and conclusion.

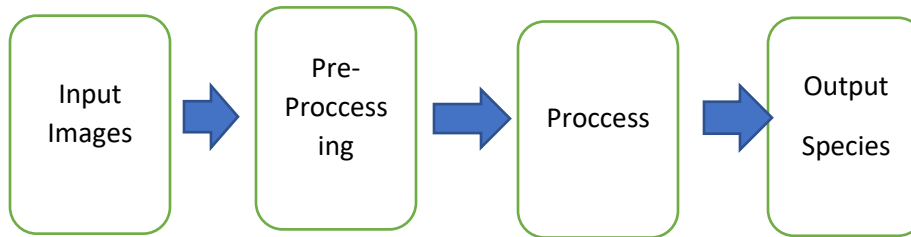
Chapter 2: LITERATURE REVIEW

1. Introduction

In this Chapter we will discuss about the process of classifying fish species and the approach methods we used and the details about them.

2. Process for Fish Species Classification

The figure below shows the description of the classifying process:



Pre-processing is the step of isolating the object, enhancing the contrast of the image,...

Process step consists of two phases: the first one is local feature extraction and the second one is classify.

3. Preprocessing

This is the very first and important step before doing classification because it can help improve accuracy. By enhancing and isolating the object from the background it can help make the classification step much easier. In this project we used K-Means algorithm to help segment the image.

3.1 K-Means clustering algorithm

The term "*k*-means" was first used by James MacQueen in 1967, though the idea goes back to Hugo Steinhaus in 1957. The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation, though it wasn't published outside of Bell Labs until 1982. In 1965, E. W. Forgy published essentially the same method, which is why it is sometimes referred to as Lloyd-Forgy.

In the K-means clustering algorithm, we do not know the label of each data point. The goal is to be able to separate data into different clusters so that data in the same cluster is of the same nature. The final goal of this clustering algorithm is to: From the input and the number of clusters we want to find, point to the center

of each cluster and divide the data points into the corresponding clusters. Suppose that each data point belongs to the same group.

Assume that there are N data points $X = [x_1, x_2, \dots, x_n] \in R^{d \times N}$ and $K < N$, is the number of clusters we want to divide. We find the centers $m_1, m_2, \dots, m_k \in R^{d \times 1}$ and the label of each data point.

For each data point x_i is set $y_i = [y_{i1}, y_{i2}, \dots, y_{ik}]$ is its vector label, where if x_i is assigned to cluster k then $y_{ik} = 1$ and $y_{ij} = 0, \forall j \neq k$. This means that one element of the vector y_i is equal to 1 corresponding to the cluster of x_i , the remaining elements are equal to 0. For example, if a data point has a vector label of $[1, 0, 0, \dots, 0]$ it belongs to cluster 1, $[0, 1, 0, \dots, 0]$ it belongs to cluster 2, ... the constraint of y_i can be written as:

$$y_{ik} \in \{0, 1\}, \sum_{k=1}^K y_{ik} = 1 \quad (1)$$

Loss function and optimization problem

If we consider center m_k as the center (or representative) of each cluster and *estimate all points allocated to this cluster by m_k* , then a data point x_i divided into cluster k would be error $(x_i - m_k)$. We want this error to have the absolute minimum value so we will find the following way to get the smallest value:

$$\|x_i - m_k\|_2^2$$

In addition, since x_i is divided into cluster k so $y_{ik} = 1$ và $y_{ij} = 0, \forall j \neq k$ hen the above expression is rewritten as:

$$y_{ik}\|x_i - m_k\|_2^2 = \sum_{j=1}^K y_{ij}\|x_i - m_j\|_2^2$$

The loss function is:

$$\mathcal{L}(Y, M) = \sum_{i=1}^N \sum_{j=1}^K y_{ij}\|x_i - m_j\|_2^2$$

Where $Y = [y_1; y_2; \dots; y_N]$, $M = [m_1; m_2; \dots; m_N]$ are matrices generated by the vector label of each data point and center of each cluster. The loss function in

our K-means clustering problem is the function $\mathcal{L}(Y, M)$ v with constraints as shown in equation (1).

In summary, we need to optimize the following problem:

$$Y, M = \operatorname{argmin}_{Y, M} \sum_{i=1}^N \sum_{j=1}^K y_{ij} \|x_i - m_j\|_2^2 \quad (2)$$

subject to $y_{ij} \in \{0,1\} \forall j, \sum_{j=1}^K y_{ij} = 1$

Optimal algorithm for loss function

Problem (2) is a difficult problem to find the optimal point because it has additional constraints. This problem is of mixed-integer programming type - it is difficult to find a global optimal point, which makes the loss function the smallest possible value. However, in some cases we can still find a way to find the approximate solution or minima.

One simple way to solve the problem (2) is to alternate Y and M when the rest of the variable is fixed. This is an iterative algorithm, which is also a common technique for solving optimal problems. We will in turn solve the following two problems:

Fixed M, find Y

Assuming that you have found the centers, look for the vector labels so that the loss function reaches the smallest value. This is equivalent to finding the cluster for each data point.

When the centers are fixed, the problem of finding the vector label for the whole data can be subdivided into the problem of finding the vector label for each data point x_i as follows:

$$y_i = \operatorname{argmin}_{y_i} \sum_{i=1}^N \sum_{j=1}^K y_{ij} \|x_i - m_j\|_2^2 \quad (3)$$

subject to $y_{ij} \in \{0,1\} \forall j, \sum_{j=1}^K y_{ij} = 1$

Since only one element of the vector label y_i is equal to 1, problem (3) can continue to be written in simpler terms:

$$j = \operatorname{argmin}_j \|x_i - m_j\|_2^2$$

Since $\|x_i - m_j\|_2^2$ is the square of the distance from point x_i to center m_j we can conclude that each point x_i **belongs to the cluster closest to its center! From this we can easily deduce the vector label of each data point.**

Fix Y , find M

Assuming that you have found the cluster for each point, find the new center for each cluster so that the loss function reaches the minimum value.

Once we have identified the vector label for each data point, the problem finding center for each cluster is reduced to:

$$m_j = \operatorname{argmin}_{m_j} \sum_{i=1}^N y_{ij} \|x_i - m_j\|_2^2$$

Here, we can find the solution by solving the function of zero, because the optimal function is a continuous function and has the derivative defined at every point. And more importantly, this function is a convex (convex) function in m_j so we will find the smallest value and the corresponding optimization point. Later on if possible, I will talk about convex optimization - an extremely important array in the optimization math.

Let $\ell(m_j)$ be the function inside the *argmin*, we have the derivative:

$$\frac{\partial \ell(m_j)}{\partial m_j} = 2 \sum_{i=1}^N y_{ij} (m_j - x_i)$$

Solving the derivative equation with 0 we have:

$$m_j \sum_{i=1}^N y_{ij} = \sum_{i=1}^N y_{ij} x_i$$

$$\Rightarrow m_j = \frac{\sum_{i=1}^N y_{ij} \mathbf{x}_i}{\sum_{i=1}^N y_{ij}}$$

Note that the denominator is the count of the number of data points in the cluster j the numerator is the sum of the data points in cluster j .

Or, in a much simpler way : m_j **is the mean of the points in cluster j**

Algorithm summary

Input : Data X and number of clusters to find K .

Output : Centers M and label vector for each data point Y .

1. Select any K points to do the initial center.
2. Assign each data point to the cluster closest to its center.
3. If the assignment of data to each cluster in step 2 does not change from the preceding loop, then the algorithm stops.
4. Update center for each cluster by taking averages of all the data points that were assigned to that cluster after step 2.
5. Go back to step 2.

We can guarantee that the algorithm will stop after a finite loop. Indeed, since the loss function is positive and after every step 2 or 3, the value of the loss function is reduced. According to the knowledge of sequence numbers in the high school program: if a series of numbers fall and is blocked under, it converges! In addition, the number of clustering for all data is finite so at some point, the loss function can not be changed, and we can stop the algorithm here.



Fig.1. (Left) Image before segment, (Right) Image segmented with K-Means

4. RGB - Color Feature

We used RGB color as a global feature for classification. We made a vector of 16 common colors. After analysing the color for each class, we calculate the color ratio of each fish image.

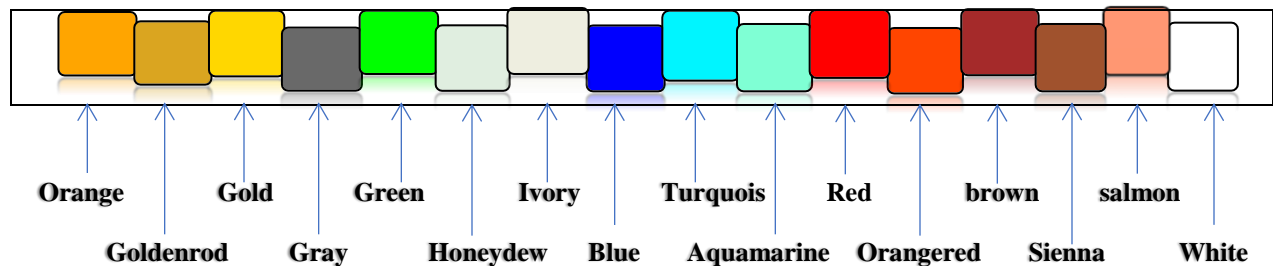
The 16 color vector is: orange, goldenrod, gold, white, gray, green, honeydew, ivory, blue, turquoise, aquamarine, orange-red, red, salmon, brown, sienna. We also combine with another feature, that is, the roundness of the object.

The calculation of roundness is based on the formula:

$$Roundness = \frac{4 * \pi * S}{P^2}$$

Where S is the object's area, P is the object's perimeter.

Here is the color vector:



5. Overview of “Bag of Visual Words” Model (BoVW)

5.1 The concept

The concept of bag of words was originally devised for categorising books into the correct genre without human supervision. In essence it takes all the potentially meaningful words (i.e. excludes common words such as ‘at’ and ‘the’) and creates a ‘bag of words’. The bag of words contains no information about the order or position of the words in the book, simply how often each occurs. From training data it is possible to calculate the probability that each word will appear in a given genre of book, these probabilities can then be applied over the whole bag of words to calculate the most probable genre of the book. This technique was found to be very successful in identifying literature.

This concept can be applied to image classification with only minor changes. The main and most obvious change is that images do not contain words, instead we need to use ‘visual words’, these aim to be meaningful section of the image. To determine if a section of an image is meaningful we can use various techniques of linear filtering and edge detection, these are now well documented in various papers and text books. The meaningful sections could be anything from simple corners to something more specific such as a wheel or a nose. The next problem is how to compare one section to another as two pictures of a wheel can look very different.

Descriptors are used to ‘describe’ the section in the aim of keeping the essence of the section without holding too much specific data, this should allow for perspective and lighting changes, the different techniques for this are explained later. Once the bag of visual words is created and the words are matched to those in the training data, the probabilities are calculated and the most likely category can be identified, for example if the image contained two wheels and sharp corners it is more likely to be a picture of a car then a face, this is explained more later.

5.2 Descriptors

Feature Descriptors are used for detecting and describing local image features. The aim of a descriptor is to find an image feature and describe it in a way that is not affected by perspective, scale, occlusion or illumination. One of the most common methods for this is Scale-Invariant Feature Transformation (SIFT) which was developed by Lowe (1999), this is considered one of the most robust feature descriptors (Bauer, Sunderhauf and Protzel, 2007). The Speeded-Up Robust Features (SURF) descriptor developed by Bay, Tuytelaars and Van Gool (2006) is a method inspired by SIFT and it lives up to its name as it is considered to be equal to, if not more, robust than SIFT and is notably more efficient (Mikolajczyk and Schmid, 2005).

In this report we will be experimenting with SURF as our feature detector to detect and describing feature in fish images.

5.3 Probability And Training Models

Calculating the probability that an image segment is of a specific object is a crucial step in object classification, it requires a large set of training data that the image features can be compared to, the features are matched to possible categories

and the subject matter can be predicted. There are many statistical models can be used for this.

6. Bag of Visual Words in Fish Classification

6.1 Speeded Up Robust Feature (SURF)

As mentioned before, The Speeded-Up Robust Features (SURF) descriptor developed by Bay, Tuytelaars and Van Gool (2006) is a method inspired by SIFT and it is considered more robust than SIFT and notably more efficient.

To reacknowledge again, feature detection is the process where we automatically examine an image to extract features, that are unique to the objects in the image, in such a manner that we are able to detect an object based on its features in different images. This detection should ideally be possible when the image shows the object with different transformations, mainly scale and rotation, or when parts of the object are occluded.

The processes can be divided in to 3 overall steps.

Detection Automatically identify interesting features, interest points this must be done robustly. The same feature should always be detected irregardless of viewpoint.

Description Each interest point should have a unique description that does not depend on the features scale and rotation.

Matching Given and input image, determine which objects it contains, and possibly a transformation of the object, based on predetermined interest points.

This report will focus on the details of the first two steps with the SURF algorithm.

6.1.1 Detection

Scale-Invariant Feature Transform, SIFT is a successful approach to feature detection introduced by David G.Lowe. The SURF-algorithm is based on the same principles and steps, but it utilizes a different scheme and it should provide better results, faster.

In order to detect feature points in a scaleinvariant manner SIFT uses a cascading filtering approach. Where the Difference of Gaussians, *DoG*, is calculated on progressively downscaled images.

In general the technique to achieve scale invariance is to examine the image at different scales, *scale space*, using Gaussian kernels. Both SIFT and SURF divides the scale space into levels and octaves. An octave corresponds to a doubling of σ , and the the octave is divided into uniformly spaced levels.

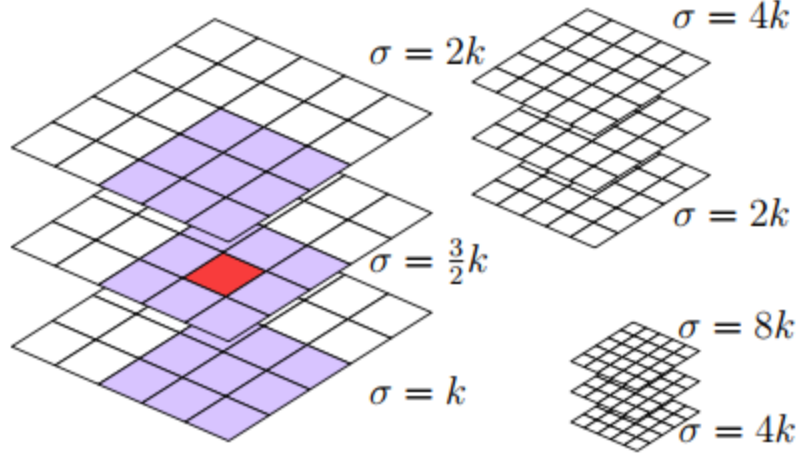


Fig. 2. 3 octaves with 3 levels.

Both approach builds pyramid of response maps, with different levels within octaves. The interest points are the points that are the extrema among 8 neighbors in the current level and its 2×9 neighbors in the level below and above. This is a non-maximum suppression in a $3 \times 3 \times 3$ neighborhood, the relation between levels, octaves and neighborhood is illustrated in Figure 2.

6.1.1.1 Hessian matrix interest points

SURF uses a Hessian based blob detector to find interest points. The determinant of a hessian matrix expresses the extent of the response and is an expression of the local change around the area.

$$\mathcal{H}(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (1)$$

Where

$$L_{xx}(x, \sigma) = I(x) * \frac{\partial^2}{\partial x^2} g(\sigma) \quad (2)$$

$$L_{xy}(x, \sigma) = I(x) * \frac{\partial^2}{\partial xy} g(\sigma) \quad (3)$$

$L_{xx}(x, \sigma)$ in equation 2 is the convolution of the image with the second derivative of the Gaussian. The heart of the SURF detection is non-maximal-suppression of the determinants of the hessian matrices. The convolutions is very costly to calculate and it is approximated and speeded-up with the use of integral images and approximated kernels.

An Integral image $I(x)$ is an image where each point $x = (x, y)^T$ stores the sum of all pixels in a rectangular area between origo and x (See equation 4).

$$I(x) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y) \quad (4)$$

The second order Gaussian kernels $\frac{\partial^2}{\partial y^2} g(\sigma)$ used for the hessian matrix must be discretized and cropped before we can apply them, a 9×9 kernel is illustrated in Figure 3. The SURF algorithm approximates these kernels with rectangular boxes, box filters. In the illustration grey areas corresponds to 0 in the kernel where as white are positive and black are negative. This way it is possible to calculate the approximated convolution effectively for arbitrarily sized kernel utilizing the integral image.

$$Det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (\omega D_{xy})^2 \quad (5)$$

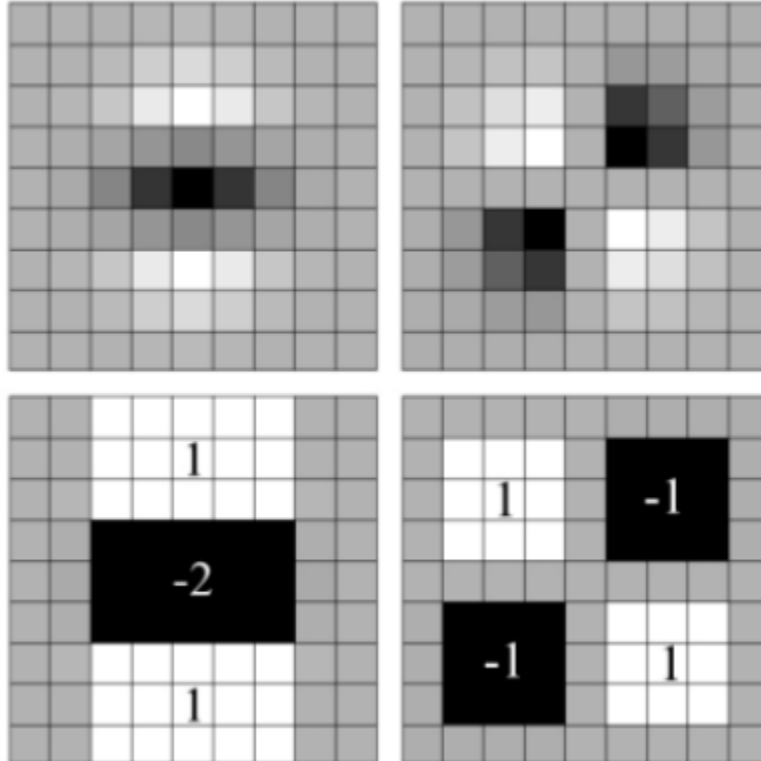


Fig. 3. $L_{xx}(x, \sigma)$ and $L_{xy}(x, \sigma)$ Discretized Gaussians and the approximation D_{yy} and D_{xy}

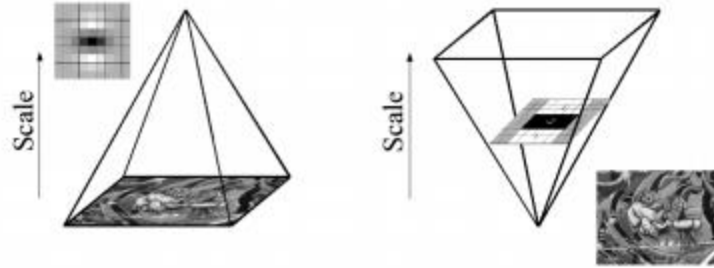


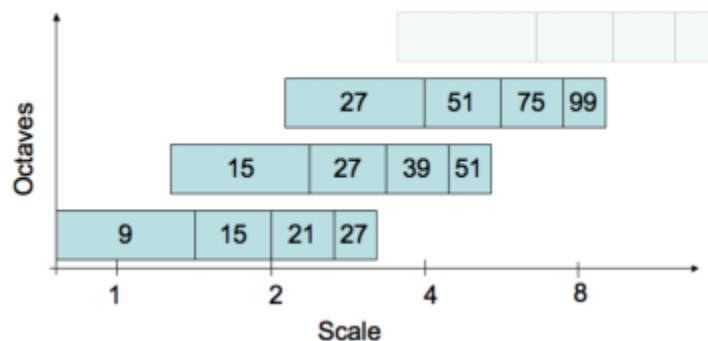
Fig. 4. Where SIFT (left) down-scale the image, SURF(right) uses larger and larger filters.

The approximated and discrete kernels are referred to as D_{yy} for $L_{yy}(x, \sigma)$ and D_{xy} for $L_{xy}(x, \sigma)$. The illustrated kernels corresponds to a σ of 1.2 and are the lowest scale that the SURF algorithm can handle. When using the approximated kernels to calculate the determinant of the Hessian matrix we have to weight it with w in equation 5, this is to assure the energy conservation for the Gaussians. The ω term is theoretically sensitive to scale but it can be kept constant at 0.9

To detect features across scale we have to examine several octaves and levels, where SIFT scales the image down for each octave and use progressively larger Gaussian kernels, the integral images allows the SURF algorithm to calculate the responses with arbitrary large kernels(Figure 4).

This does pose two challenges, how to scale the approximated kernels and how this influences the possible values for σ . The kernels has to have an uneven size to have a central pixel and the rectangular areas, the *lobes*, has to have the same size.

The SURF paper [2] goes into detail with these considerations. The result is that division of scale space into levels and octaves becomes fixed as illustrated in Figure 5. The filter size will be large and if the convolution were to be done with a



regular Gaussian kernel this would be prohibitively expensive. The use of integral images not only makes this feasible - it also does it fast, and without the need to downscale the image. It should be noted that this approach with large box filters can preserve and be sensitive to high frequency noise. When finding a extrema at one of the higher octaves the area covered by the filter is rather large and this introduces a significant error for the position of the interest point. To remedy this the exact location of the interest point are interpolated by fitting a 3D quadratic in scale space [4]. An interest point is located in scale space by (x, y, s) where x, y are relative coordinates($x, y \in [0; 1]$) and s is the scale.

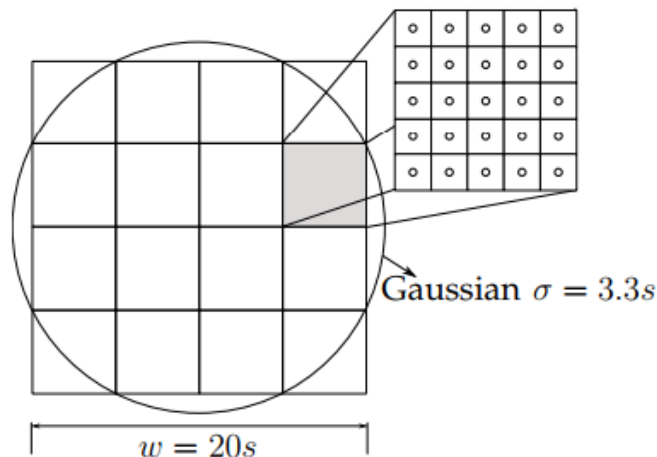
Fig. 5. The scale(σ), the filter sizes and octaves on a logarithmic scale.

6.1.2 Description

The purpose of a descriptor is to provide a unique and robust description of a feature, a descriptor can be generated based on the area surrounding a interest point. The SURF descriptor is based on Haar wavelet responses and can be calculated efficiently with integral images. SIFT uses another scheme for descriptors based on the Hough transform. Common to both schemes is the need to determine the orientation. By determining a unique orientation for a interest point, it is possible to achieve rotational invariance. Before the descriptor is calculated the *interest area* surrounding the interest point are rotated to its direction.

The SURF descriptors are robust to rotations and an upright version, U-SURF, should be robust for rotations $\pm 15^\circ$, without performing an orientation assignment [1].

The SURF descriptor describes an interest area with size $20s$. The interest area is divided into 4×4 subareas that is described by the values of a wavelet response in the x and y directions. The wavelet response in the x and y direction is referred to as dx and dy respectively, the wavelets used to calculate the response is illustrated in Figure 7. The interest area are weighted with a Gaussian centered at the interest



point to give some robustness for deformations and translations. The components involved in the calculations is illustrated in Figure 8.

$$v = \left\{ \sum dx, \sum |dx|, \sum dy, \sum |dy| \right\} \quad (6)$$

Fig. 6. A 20s areas is divided into 4×4 subareas that are sampled 5×5 times to get the wavelet response (Figure 7)



Fig. 7. The wavelet response. Black and white areas corresponds to a weight -1 and 1 for the Haar kernels. They are used with a filter size of 2s.

For each subarea a vector v (Equation 6) is calculated, based on 5×5 samples. The descriptor for a interest point is the 16 vectors for the subareas concatenated. Finally the descriptor is normalized, to achieve invariance to contrast variations that will represent themselves as a linear scaling of the descriptor.

Several schemes varying the size, number of samples and wavelet function has been tested. This setup has been experimentally found as optimal, taking performance and precision into account [1].

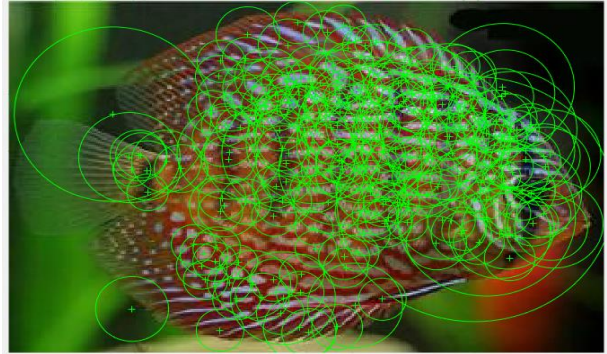
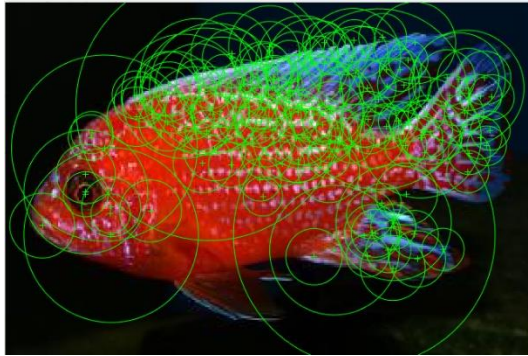
Having calculated the descriptors finding a match between two descriptors is a matter of testing if the distance between the two vectors is sufficiently small. The SURF algorithm does add another detail to speed up matching, that is the sign of Laplacian.

$$\nabla^2 L = tr(\mathcal{H}) = L_{xx}(x, \sigma) + L_{yy}(x, \sigma) \quad (7)$$

The laplacian is the trace of the hessian matrix (Equation 7) and when calculating the determinant of the hessian matrix these values are available. It is a matter of storing the sign. The reason to store the sign of the Laplacian is that distinguishes between bright blobs on dark backgrounds and vice versa. It is only necessary to compare the full descriptor vectors if the have the same sign, which can lower the computational cost of matching.

6.1.3 Implementation

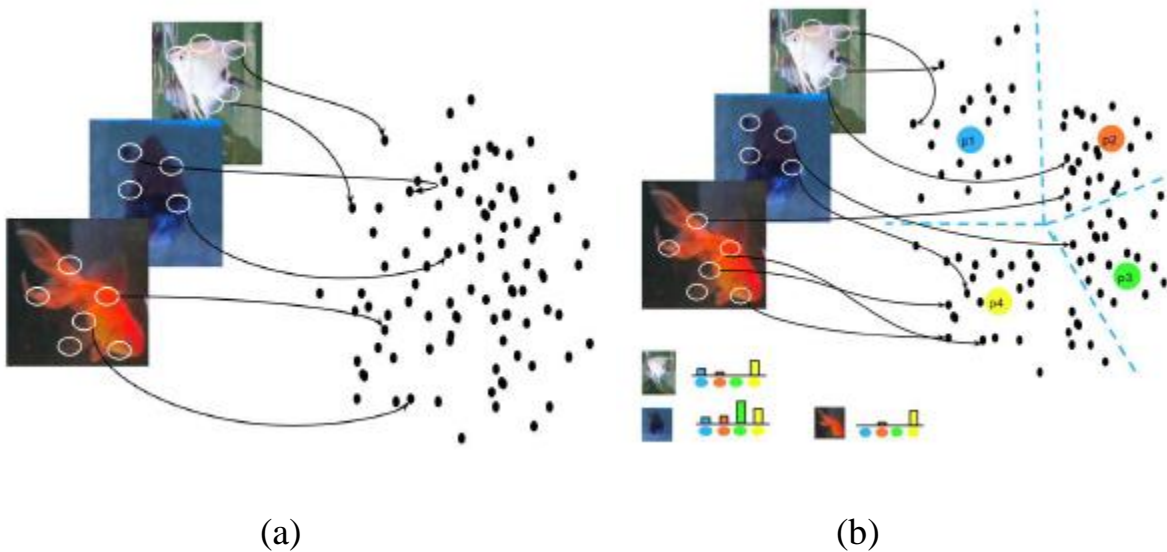
We tested keypoints extraction with SURF provided by Matlab R2017. Here are some results of the keypoint extraction on fish images.



6.2 Bag of Visual Words Model

Given the keypoints and descriptors extracted using SURF, or another local feature extractor, the BoVW generates a fixed sized vector the can be used as a global descriptor for an image and feed a feature vector based machine learning algorithm.

BoVW can be described in **4 steps**, as shown in Figure 8. The first step detects and describes all keypoints for all the training images (Figure 8 (a)). Given the descriptor vectors for all these keypoints, a clustering algorithm, such as k-means, is performed to partition the set of keypoints into k clusters (Figure 8(b)). Each of these k cluster is called a visual word and k , which is a parameter calculate experimentally, is the dictionary size. To describe a new image, the keypoints are extracted and assigned to one of the k clusters (Figure 8 (c)) using some similarity measure between descriptor vectors. In this way, each keypoint is associated with a visual word. Finally, a histogram of size k that counts the frequency of each visual word occurring in the image is built (Figure 8 (d)).



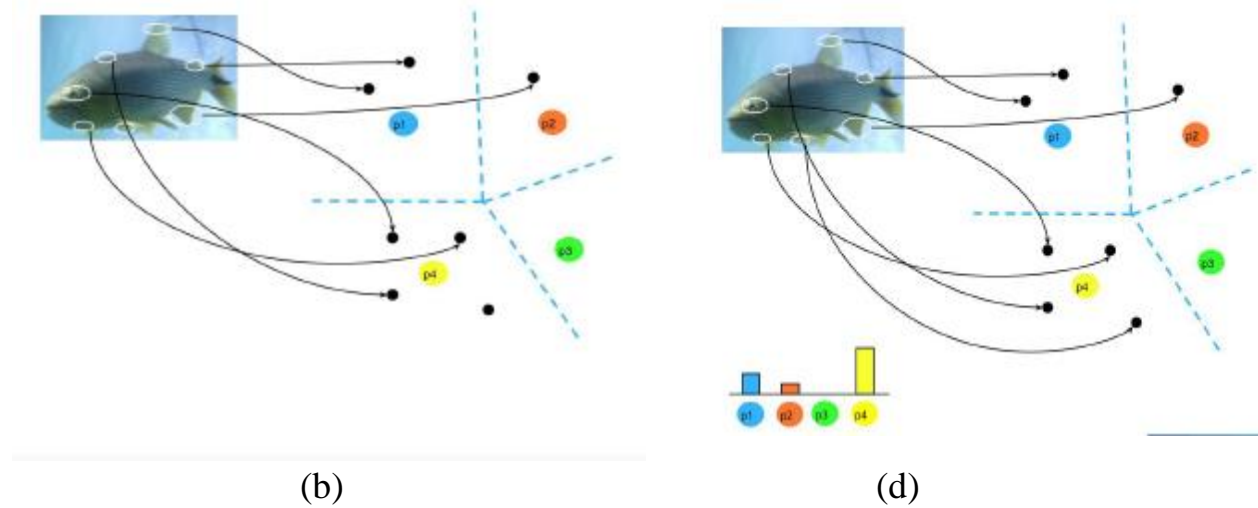


Fig.8. Graphical representation of the BoVW model.

- (a) Illustrates the keypoints extracted from all training images.
- (b) Shows four clusters found by the K-Means to compose the visual dictionary.
- (c) Illustrates the assignment of the keypoints to the clusters of an image.
- (d) Shows the histograms that counts the occurrence of each cluster.

Once the similar histogram data has been calculated for each species that we wish to identify we can begin to classify the images based on this data.

6.3 Classifiers

The idea of a classifier is to identify which is the most probable species for an image given in the test data that has already been processed. To do this we need to detect the features in the same manner as for generating the visual dictionary. Once the features are detected we need to create a histogram, again in the same way we did for the dictionary. We can then compare this histogram to the histograms for each species in the dictionary.

In this report, we experiment on 4 different classifiers: Naïve Bayes, Support Vector Machine (SVM), k-Nearest Neighbors (KNN) and ANN (Artificial Neural Network).

Chapter 3: MEDTHOLOGY

In this Chapter, we will present about the classification algorithms we used for our project. We will discuss about the design, the kernels,...and how they work. For testing purpose, we used the supported functions in Matlab.

1. Naïve Bayes Classifier

The naive Bayes classifier is designed for use when predictors are independent of one another within each class, but it appears to work well in practice even when that independence assumption is not valid. It classifies data in two steps:

- Training step: Using the training data, the method estimates the parameters of a probability distribution, assuming predictors are conditionally independent given the class.
- Prediction step: For any unseen test data, the method computes the posterior probability of that sample belonging to each class. The method then classifies the test data according the largest posterior probability.

The class-conditional independence assumption greatly simplifies the training step since you can estimate the one-dimensional class-conditional density for each predictor individually. While the class-conditional independence between predictors is not true in general, research shows that this optimistic assumption works well in practice. This assumption of class-conditional independence of the predictors allows the naive Bayes classifier to estimate the parameters required for accurate classification while using less training data than many other classifiers. This makes it particularly effective for data sets containing many predictors.

The Distributions

The training step in naive Bayes classification is based on estimating $P(X|Y)$, the probability or probability density of predictors X given class Y . In Matlab, the naive Bayes classification model **ClassificationNaiveBayes** and training function **fitcnb** provide support for normal (Gaussian), kernel, multinomial, and multivariate, multinomial predictor conditional distributions. To specify distributions for the predictors, use the DistributionNames name-value pair argument of fitcnb. You can specify one type of distribution for all predictors by supplying the string corresponding to the distribution name, or specify different distributions for the predictors by supplying a length D cell array of strings, where

D is the number of predictors (that is, the number of columns of X). In this project we experiment on Kernel Distribution only.

Kernel Distribution

The 'kernel' distribution (specify using 'kernel') is appropriate for predictors that have a continuous distribution. It does not require a strong assumption such as a normal distribution and you can use it in cases where the distribution of a predictor may be skewed or have multiple peaks or modes. It requires more computing time and more memory than the normal distribution. For each predictor you model with a kernel distribution, the naive Bayes classifier computes a separate kernel density estimate for each class based on the training data for that class. By default the kernel is the normal kernel, and the classifier selects a width automatically for each class and predictor.

2. K-Nearest Neighbor Classifier (KNN)

The KNN Classification principle is associated with finding a predefined number of neighbors in training set of a new example and rank it. The majority class among neighbors classifies the new instance. The KNN has one of its main advantages simplicity, plus a quick learning phase, it get good results in serveral problems.

Prediction Procedure

KNN predicts the classification of a point X_{new} using a procedure equivalent to this:

- Find the NumNeighbors points in the training set X that are nearest to X_{new} .
- Find the NumNeighbors response values Y to those nearest points.
- Assign the classification label Y_{new} that has the largest posterior probability among the values in Y .

In Matlab we can use *predict* function for prediction with k-nearest neighbor.

predict classifies so as to minimize the expected classification cost:

$$\hat{y} = \arg \min \sum_{k=1}^K \hat{P}(k|x)C(y|k),$$

Where

- \hat{y} is the predicted classification
- K is the number of classes
- $\hat{P}(k|x)$ is the posterior probability of class k for observation x .
- $C(y|k)$ is the cost of classifying an observation as y when its true class is k .

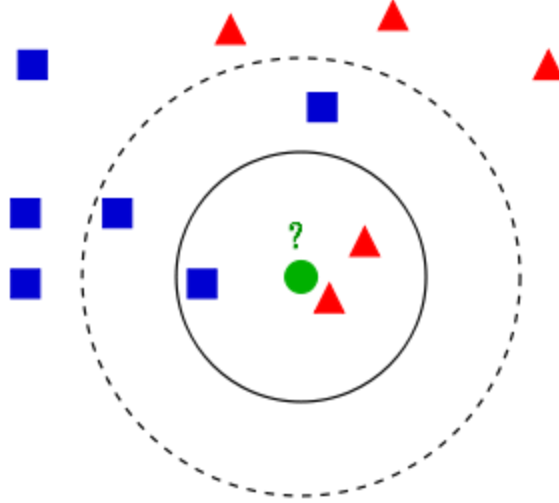


Fig. 9. Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or the second class of red triangles. If $k = 3$ (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If $k = 5$ (dashed line circle) it is assigned to the first class (3 squares vs 2 triangles inside the outer circle).

Posterior Probability

For a vector (single query point) X_{new} and model mdl , let:

- K be the number of nearest neighbors used in prediction, $mdl.NumNeighbors$
- $nbd(mdl, X_{new})$ be the K nearest neighbors to X_{new} in $mdl.X$
- $Y(nbd)$ be the classifications of the points in $nbd(mdl, X_{new})$, namely $mdl.Y(nbd)$
- $W(nbd)$ be the weights of the points in $nbd(mdl, X_{new})$
- $prior$ be the priors of the classes in $mdl.Y$

If there is a vector of prior probabilities, then the observation weights W are normalized by class to sum to the priors. This might involve a calculation for the

point X_{new} , because weights can depend on the distance from X_{new} to the points in $mdl.X$.

The posterior probability $p(j|X_{new})$ is

$$p(j|X_{new}) = \frac{\sum_{i \in nbd} W(i) 1_{Y(X(i)=j)}}{\sum_{i \in nbd} W(i)}.$$

Here $1_{Y(X(i)=j)}$ means 1 when $mdl.Y(i) = j$, and 0 otherwise.

3. Support Vector Machine (SVM)

SVM is a supervised set of methods used for classification and regression. SVM's Classifiers are based on a maximum margin between classes to classify new examples, and maximizing the margin, improves the generalization at the classification stage. The separation of the classes can be done by linearly or polynomially kernels, depending on the dataset used. Some advantages can be found in the use of classifiers based on support vector machines:

- It is very effective in large dimensional spaces, that is, when the amount of attributes for the problem approached is large.
- It is still effective even in cases where the number of dimensions is larger than the examples.
- Different kernels can be specified as decision functions become a versatile SVM classifier. If necessary, it is possible to set a customizable kernel.

Throughout training phase SVM takes a data matrix as input data and labels each one of samples as either belonging to a given class (positive) or not (negative). SVM treats each sample in the matrix as a row in an input space or high dimensional feature space, where the number of attributes identifies the dimensionality of the space. SVM learning algorithm determines the best hyperplane which separates each positive and negative training sample. The trained SVM can be deployed to perform predictions about test samples (new) in the class.

Nonlinear problems in SVM are solved by mapping the n - dimensional input space into a high dimensional feature space. Finally in this high dimensional feature space a linear classifier is constructed which acts as nonlinear classifier in input space.

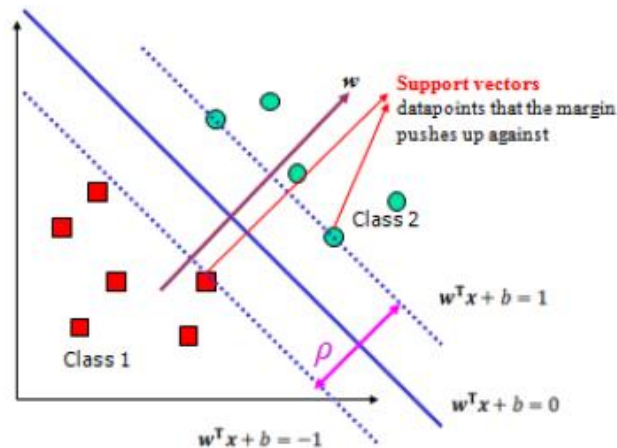


Fig. 10. Example of decision boundary and margin of SVM

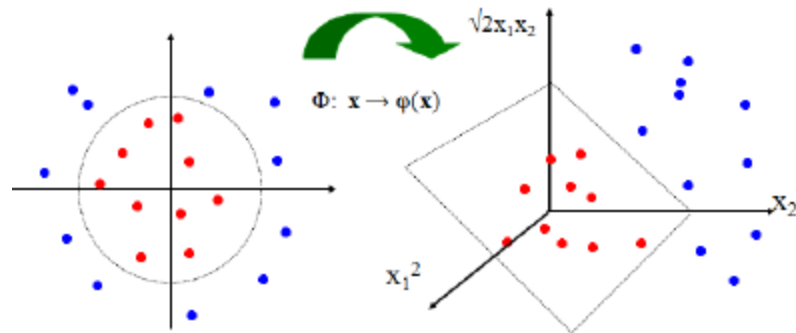


Fig. 11. Nonlinear SVMs decision boundary

In case of Multi-class classification problems, the issue becomes more complex because the outputs could be more than one class and must be divided into M mutually exclusive classes. In fact, there are many ways to solve Multi-class classification problems for SVM such as Directed Acyclic Graph (DAG), Binary Tree (BT), One-Against-One (OAO) and One Against-All (OAA) classifiers.

The four commonly used kernels are:

- Linear kernels
- Polynomial kernel
- Radial basis function (RBF) kernel
- Sigmoid kernel

4. Artificial Neural Network (ANN)

An (artificial) neural network is a network of simple elements called neurons, which receive input, change their internal state (activation) according to that input, and produce output depending on the input and activation. The network forms by connecting the output of certain neurons to the input of other neurons forming a directed, weighted graph. The weights as well as the functions that compute the activation can be modified by a process called learning which is governed by a learning rule.

Components of an artificial neural network

Neurons

A neuron with label j receiving an input $p_j(t)$ from predecessor neurons consists of the following components:

- An *activation* $a_j(t)$ depending on a discrete time parameter,
- Possibly a *threshold* θ_j , which stays fixed unless changed by a learning function,
- An *activation function* f that computes the new activation at a given time $t + 1$ from $a_j(t)$, θ_j and the net input $p_j(t)$ giving rise to the relation $a_j(t + 1) = f(a_j(t), p_j(t), \theta_j)$,
- And an *output function* f_{out} computing the output from the activation $o_j(t) = f_{out}(a_j(t))$.

Often the output function is simply the identity function.

An *input neuron* has no predecessor but serves as input interface for the whole network. Similarly an *output neuron* has no successor and thus serves as output interface of the whole network.

Connections and weights

The *network* consists of connections, each connection transferring the output of a neuron i to the input of a neuron j . In this sense i is the predecessor of j and j is the successor of i . Each connection is assigned a weight w_{ij} .

Propagation function

The *propagation function* computes the *input* $p_j(t)$ to the neuron j from the outputs $o_i(t)$ of predecessor neurons and typically has the form $p_j(t) = \sum_i o_i(t)w_{ij}$.

Learning rule

The *learning rule* is a rule or an algorithm which modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. This *learning* process typically amounts to modifying the weights and thresholds of the variables within the network.

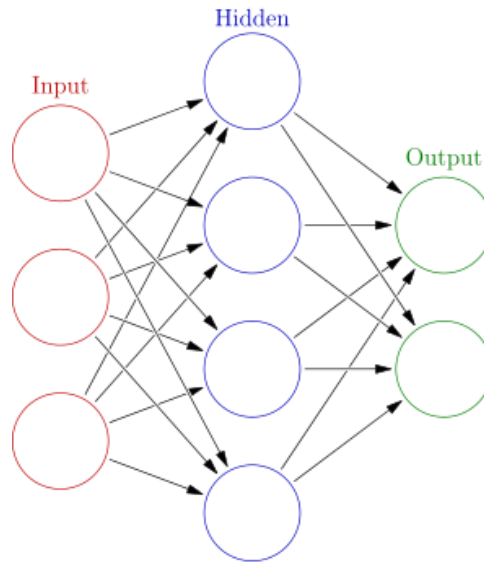


Fig. 12. An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

Chapter 4: EXPERIMENTAL RESULTS

1. Dataset Overview

- The Fish Species Dataset is composed by 40 different photos for each of the 5 fish species. All of the species used in this project and their respective informations are illustrated on Figure 13.
- All the images were taken from various sources on the Internet.
- Images were resized to 200×300 pixel and segmented manually using K-Means to give the best experiment.
- We used 70% of the dataset for training data, and the remainder 30% for the test data. Details are illustrated on Figure 14.






Image	Popular Name	Scientific Name
	Aulonocara Firefish (Peacock Cichlids)	Aulonocara nyassae Regan, 1922
	Discus	Symphysodon aequifasciatus Hackel, 1840
	Flame angelfish	Centropyge loriculus Günther, 1874
	Yellowtail kingfish	Seriola lalandi Valenciennes, 1833
	Yellow Molly	Poecilia Sphenops Valenciennes, 1846

Fig. 13. Illustration of all species used in this project and its popular and scientific names

Class	Total images	Training images	Test images
Aulonocara Firefish	40	1-28	29-40
Discus	40	1-28	29-40
Flame angelfish	40	1-28	29-40
Yellowtail kingfish	40	1-28	29-40
Yellow Molly	40	1-28	29-40

Fig. 14. Description of Fish dataset

2. Experiments

2.1 Classification Methods

The performance of BoVW with SURF extractor algorithms was evaluated by using four classification methods: Naïve Bayes, Support Vector Machine (SVM), k-Nearest Neighbors (KNN) and Artificial Neural Network (ANN). For the implementation environment we used Matlab R2017 with the **Statistics and Machine Learning Toolbox** and **Neural Network Toolbox**.

2.2 Dictionary Size

For the dictionary size k of BoVW to represent the characteristics of the fishes , the following values were evaluated: 64, 128, 256, 512. These values were varied in this way following the variations found in the literature reviews. For each dictionary, parameters of the classifiers were varied to the description of the next section.

2.3 Parameters of the Classifiers

The parameter of the classifiers were varied as follows:

- Naïve Bayes: The ‘kernel’ distribution was used.
- KNN: the K values tested were 1, 3, 5, 7, 9, 11, 13, 15. The metrics used to calculate the distance of the points was Euclidean.

SVM: the kernels used were linear, polynomial, and Gaussian RBF.

2.4 Results and Discussion

Here we show the comparison on the performance of the four methods using Bag of Visual Words (BoVW) ,Color Feature and roundness of object and with different dictionary sizes to represent the characteristics of the fishes features.

Shows a comparison between 4 classification methods

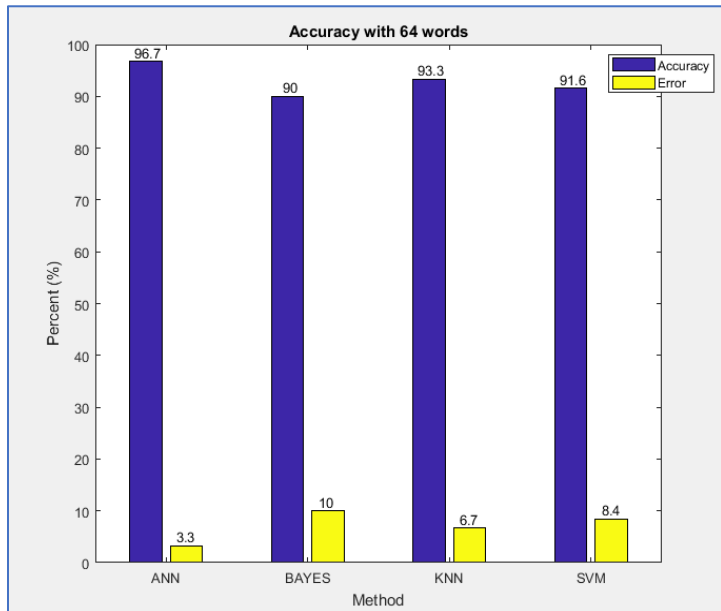


Fig. 15. Words equal 64.

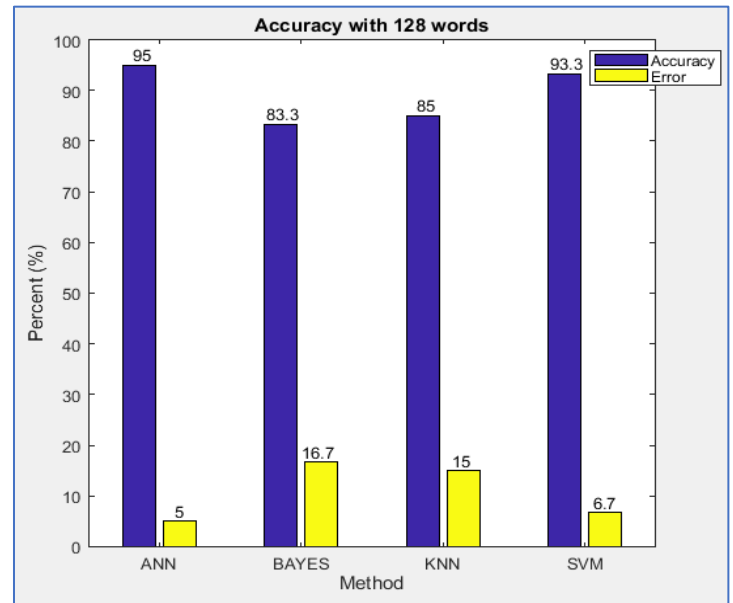


Fig. 16. Words equal 128.

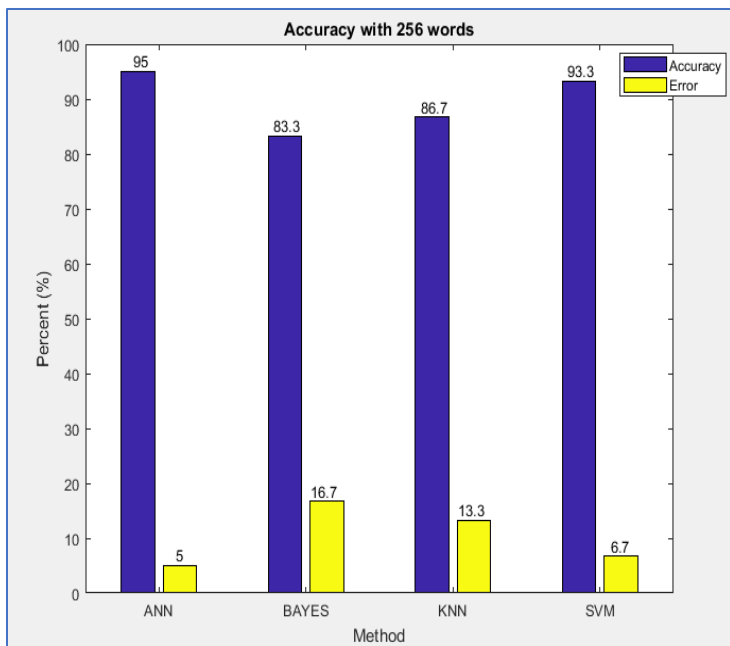


Fig. 17. Words equal 256.

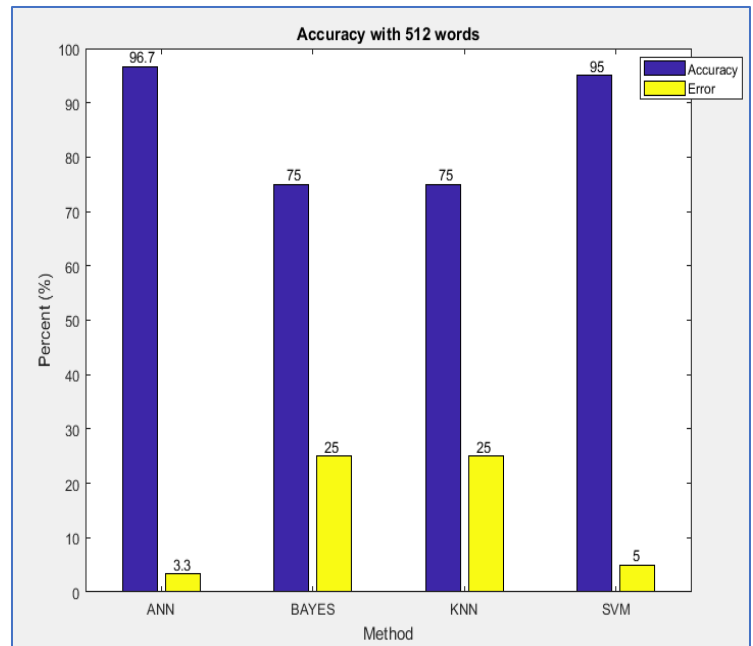


Fig. 18. Words equal 512.

REFERENCES

- [1] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, Speeded-Up Robust Features (SURF), *Comput. Vis. Image Underst.*, vol. 110, p. 346-359, n. 3, 2008.
- [2] Wikipedia, “Blob detection — Wikipedia, the free encyclopedia,” 2011, [Online]. Available: https://secure.wikimedia.org/wikipedia/en/wiki/Blob_detection
- [3] Benjamin Deavin, “Bag of Words: Automated Classification of Images”, May 7, 2010
- [4] M. B. David Lowe, “Invariant features from interest point groups,” *BMVC*, 2002.
- [5] U. Freitas, W. N. Gonçalves, E.T. Matsubara, J. Sabino, M.R. Borth, H. Pistori, “Using Color for Fish Species Classification”.
- [6] M. S. Nery, A. M. Machado, M. F. M. Campos, F. L. C. Pádua, R. Carceroni and J. P. Queiroz-Neto, Determining the Appropriate Feature Set for Fish Classification Tasks, *SIBGRAPI Conference on Graphics, Patterns and Images*, p. 173-180, n. 1530-1834, 2005
- [7] Wikipedia, “Artificial Neural Network” [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network
- [8] Wikipedia, “K-means Clustering”, [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering
- [9] MathWorks Documentation , “Image Category Classification Using Bag of Features”.
- [10] Wikipedia, “Feature Detection – Wikipedia, the free encyclopedia”, 2011. [Online]. Available: https://secure.wikimedia.org/wikipedia/en/wiki/Feature_detection%28computer_vision%29
- [11] H.Yao, Q.Duan, D.Li, J.Wang, “An improved K-means clustering algorithm for fish image segmentation”
- [12] Support Vector Machine, Slide from Prof. Pham The Bao

[13] MathWorks Documentation, “Supervised Learning Workflow and Algorithms”.