SEMANTIC SALIENCY OF VIDEO CONTENT: APPLICATION TO QUALITY ASSESSMENT

Ho Tien Lam[1*], Phan Ho Duy Phuong[2] Le Dinh Phu Cuong[3]
[1]UF-HCM – Ho Chi Minh National University [2]
Cuu Long University, [3] Yersin University, Dalat
*Corresponding Author: hotienlam@gmail.com

(Received: July 30th, 2016; Accepted: August 31th, 2016)

ABSTRACT

Enjoying video on digital television as well as Internet has become a significant demand at the present time. People demand not only as fast system response time but also a high video quality. However, through the transmission networks and processing system, some amounts of errors may be introduced in the video signal, so video quality assessing is an important problem. Many researchers proposed a lot of methods for evaluating video quality. Researchers at LaBRI have proposed to assess video quality taking into account visual saliency. Visual saliency of an item is the state or quality by which it stands out relative to its neighbors. Spatial and temporal saliency were be used for defining an objective quality assessment metric. The aim of this topic was to help in studying if a semantic saliency, based on faces, could improve this metric. In this topic studied how to create semantic saliency maps, if human faces on video frame attract viewer's attention, and how to use a face detection algorithm for automatic semantic map construction.

***Keyword**: LaBRI (Laboratoire Bordelais de Recherche en Infromatique) is a French research laboratory in field of computer science. It is associated with CNRS (UMR 5800), the University of Bordeaux 1, the IPB and the University of Bordeaux 2. It has been the partner of INRIA since 2002.*

TÓM TẮT

Thưởng thức video trên truyền hình kỹ thuật số cũng như Internet đã trở thành một nhu cầu quan trọng ở thời điểm hiện tại. Mọi người có nhu cầu không chỉ có thời gian đáp ứng hệ thống nhanh mà còn là một video chất lượng cao. Tuy nhiên, thông qua các mạng lưới và hệ thống xử lý truyền dẫn, một số lượng lỗi có thể được giới thiệu trong các tín hiệu video, do đó chất lượng video đánh giá là một vấn đề quan trọng. Nhiều nhà nghiên cứu đã đề xuất rất nhiều phương pháp để đánh giá chất lượng video. Các nhà nghiên cứu tại LaBRI đã đề xuất để đánh giá chất lượng video nhìn rõ ràng. Hình ảnh rõ ràng của một mục là trạng thái hay chất lượng mà nó đứng yên tương đối so với các hình ảnh kế cận với nó. Sự rõ ràng không gian và thời gian đã được sử dụng để xác định một chỉ số đánh giá chất lượng khách quan. Mục đích của đề tài này là để giúp trong việc nghiên cứu nếu sự rõ ràng ngữ nghĩa, dựa trên khuôn mặt, có thể cải thiện sự đo đạc này. Trong chủ đề này nghiên cứu làm thế nào để tạo ra bản đồ sự rõ ràng ngữ nghĩa, nếu khuôn mặt của con người trên khung hình video thu hút sự chú ý của người xem, và làm thế nào để sử dụng một thuật toán nhận diện khuôn mặt để xây dựng bản đồ ngữ nghĩa một cách tự động.

***Từ khóa:** LaBRI (Laboratoire Bordelais de Recherche en Infromatique) là một phòng thí nghiệm nghiên cứu của Pháp trong lĩnh vực khoa học máy tính. Nó được kết hợp với CNRS (UMR 5800), Đại học Bordeaux 1, IPB và Đại học Bordeaux 2. Nó đã là thành viên của INRIA kể từ năm 2002.*

INTRODUCTION

According to physiological and psychological evidences, we know that human beings do not pay equal attention to all exposed visual information, but only focus on certain areas known as focus of attention (FOA) or saliency regions [6]. The LaBRI's researchers have proposed a novel objective quality assessment metric using spatio-temporal saliency map [3]. In the literature, the saliency of the visual scene is characterized by two saliency maps called *spatial* and *temporal* saliency maps. The spatial saliency map is based on color or contrast, for example. The temporal saliency map is computed with the residual motion in the visual scene with regard to global and camera motion. And the spatio-temporal saliency map is the result of the fusion step.

These results will be integrated into the existing method [3] to assess video quality. Figure 1 gives an overview to the whole process. To enhance this method, the semantic saliency map is generated in two ways, manual way and automatic way. The detail of manual way is described in section II.1 and II.2. The results of this task will be used in automatic way which is described in II.4. Section II.3 will show a statistics of faces in saliency maps.

Below we introduce some important terms used in this report.

Saliency Map is used for representing the saliency areas in an image and guiding the selection of attended locations, based on the spatial distribution of saliency [4].

Gaussian Map is a gray-scale saliency map which assigns a saliency value to each pixel. This value is calculated by using the formula of Gaussian function. This function will be introduced in section II.2.3.

Error Map is a gray-scale image. The white areas in this map represent the error location in a distorted image. This map is produced by video decoder in processing systems.

Bounding Box is a rectangle which surrounds an object of interest. In context of this study, we are only concerned by bounding boxes around human faces. The purpose of these bounding boxes is to represent sematic saliency regions in an image. The bounding boxes may be generated manually by using BBBox Annotation Software, introduced in section II.1, or automatically by face detection algorithm, introduced in section 2.4.

Figure 2 illustrates an example of bounding boxes and the corresponding Gaussian map, and figure 3 gives an example of error map and ground-truth saliency map produced by eye-tracker. In Gaussian map, the white areas show the salient locations in the image. The pixel value becomes whiter when the point goes closer to the center of bounding box. This kind of map and the saliency map from eye-tracking device show visual areas which actually attract viewers' attention. However, these maps are generated manually by human or supporting device. So it is not effective for automatic video quality assessment. The purpose of using these maps is to compare with the quality assessment metric using automatically produced saliency maps.

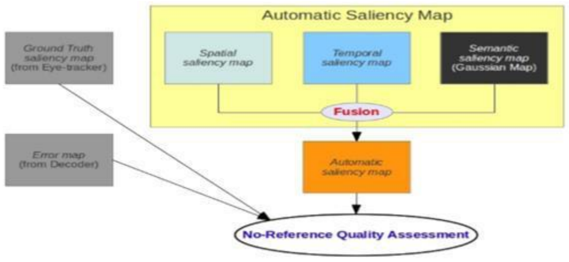


Figure 1. Process Overview

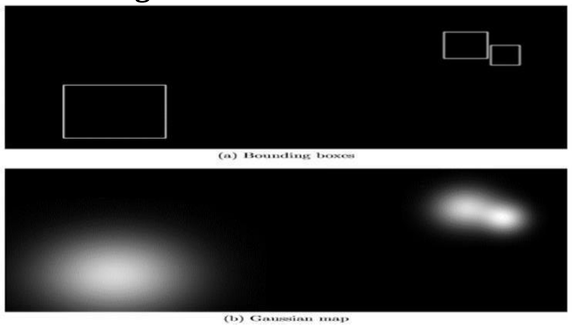


Figure 2. Example of Bounding boxes and Gaussian map

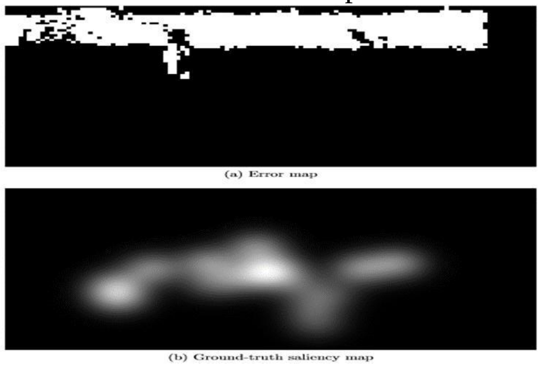


Figure 3. Example of Error map and Ground-truth saliency map

VISUAL SALIENCY DETECTION

The goal of us training placement is to create "semantic saliency" maps of HD video which express the presence of a semantic object- "face". We thus present related tasks below:

- Adapt software of face annotation for annotating HD videos.

- Annotate the ground truth: trace the bounding boxes of faces and store their coordinates observed in video frames by human.

- For manually annotated bounding boxes of faces, we compute the saliency maps with the help of Gaussian filtering.
- Detect faces automatically by using Viola-Jones face detector of OpenCV library.

In the following sub-chapter, we will thus describe our contribution into adaptation of face-annotation GUI.

BBBOX ANNOTATION SOFTWARE

To annotate videos, we used the BBBox software developed in LaBRI on the basis of Viola-Jones object detection algorithm. This application is written in C++, and it uses Qt[1] framework to build graphical user interface (GUI). The aim of this task is to annotate high-definition videos. In this task, we have to work with Qt framework which is described in section II.1.1.

INTRODUCTION OF QT FRAMEWORK

This section introduces Qt Framework which is free and open source software. Qt is a cross-platform application framework that is widely used for developing software with a GUI and non-GUI grograms. At first, Qt is produced by Trolltech, an Norwegean company, in May 1995. Currently, it is developed by Nokia's Qt Development Frameworks division after Nokia's acquisition of Trolltech. Since Trolltech's birth, Qt has become a product used by thousands of customers and hundreds of thousands of open source developers all around the world. The community licenses Qt under both open source licenses (LGPL and GPL), as well as a commercial license. Using Qt Framework, developers can build C++ application that run natively on many platforms such as Windows, Linux/Unix, Mac OS X, and embedded Linux, without making source code changes. For more information, please refer to the book "C++ GUI Programming with Qt 4" [1].

FUNCTIONALITY OF BBBOX SOFTWARE

BBBox is used for annotating the human faces in each frame of a video. The figure 4 shows the screenshot of this software.

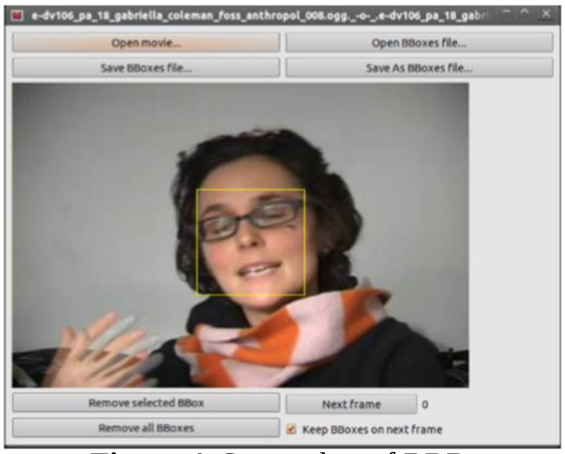


Figure 4. Screenshot of BBBox

MISSING FEATURES OF BBBOX SOFTWARE

Suppose that user open a full-HD video with display resolution (1920 x 1080) bigger than that of desktop screen – what will happen? The problem with the actual version of BBBox is that cannot adjust the window size to fit well in desktop screen. Moreover, the software does not scale the movie scene up or down when a user resizes the window. Solving these two problems is one of us tasks. For the first problem, we corrected completely so that BBBox window can fit in desktop screen. The second problem was partially solved. It was possible to scale the image up, while the act of scaling down was not successful. Section II.1.5 will elaborate more on the solution for these issues.

ARCHITECTURE OF BBBOX SOFTWARE

This section describes the architecture and function of all classes in BBBox. This software includes some classes which are implemented using object-oriented model. The hierarchy of classes is shown in figure 5.

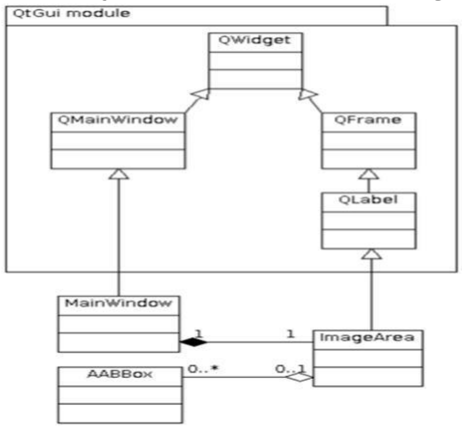


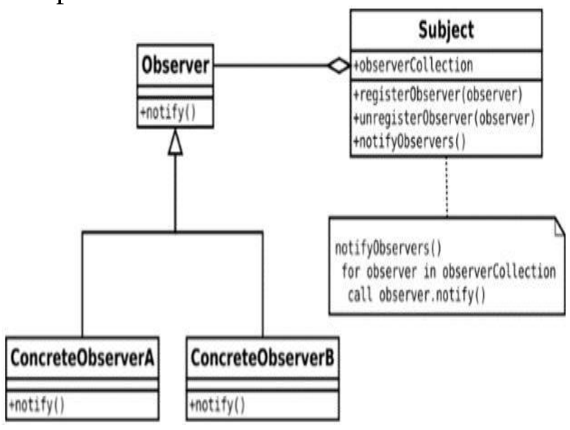
Figure 5. Class diagram of BBBox

In Qt Framework, the QWidget class is the base class of all user interface objects. The widget is the element of user interface: it receives mouse, keyboard and other events from the window system, and paints a representation of itself on the screen. A widget that is not embedded in a parent widget is called a window. In Qt, QMainWindow and the various subclasses of QDialog are the most common window types. An application's main window is created by subclassing QMainWindow.

The QLabel class is used for displaying a text or an image. To display an image, we use the QPixmap class as a paint device. In definition of QLabel class, there is a property, pixmap, which holds the label's pixmap. This property is modified through the setPixmap method.

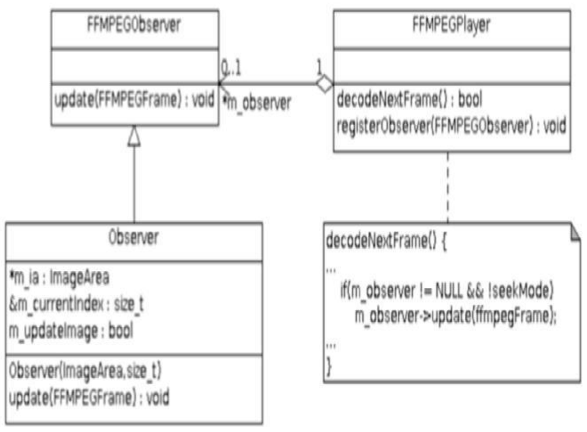
The ImageArea class is both a subclass of QLabel and a child widget of MainWindow. This widget will receive mouse and resize events from user. The way which ImageArea object handles events is similar to that of Java Swing[2]. Whenever a user clicks a button or adjusts window size, many objects in the application may need to react to the change.

For example, when a user clicks the "Next frame" button, the ImageArea object will have to update the new frame on its graphical representation. To implement this, the BBBox software uses the Observer pattern[3]. The structure of this pattern is illustrated in figure 6 taken form Wikipedia. Figure 7 show how this pattern is used in this software.



```
notifyObservers()
  for observer in observerCollection
  call observer.notify()
```

Figure 6. UML diagram of Observer pattern



```
decodeNextFrame() {
...
    if(m_observer != NULL && !seekMode)
        m_observer->update(ffmpegFrame);
...
}
```

Figure 7. UML diagram of Observer pattern used in BBBox

Besides the classes liste in figure 5, this software also uses some classes in another library called *libVideoFeatureBridge*. This library is developed by members in the AIV group, Hugo Boujut and Boris Mansencal. Some classes of libVideoFeatureBridge used in this software are FFMPEGPlayer, FFMPEGObserver, and FFMPEGFrame.

The word "AABBox" stands for Axis-Aligned Bounding Box. AABBox always has the rectangle shape; however, BBox (Bounding Box) may have the parallelogram shape. That is the difference between these two kinds of bounding boxes. Later on we only use the term "BBox", "bounding box" or "box" to mention Axis-Aligned Bounding Box. The AABBox class defines the attributes of a bounding box and some utility methods.

In summary, the MainWindow class takes charge of building user interface, receiving events, and event-handling. The ImageArea class is in charge of displaying each frame on main window and dealing with events. There may be zero or non-zero human face(s) on each frame. The ImageArea object, therefore, holds a list of AABBox objects.

MODIFICATION OF BBBOX SOFTWARE

As mentioned in section II.1.3, there are two problems with the actual version of BBBox. To solve the first problem, the flowchart in figure 8 is carried out. This is also the flowchart of the update (FFMPEGFrame) method in Observer class.

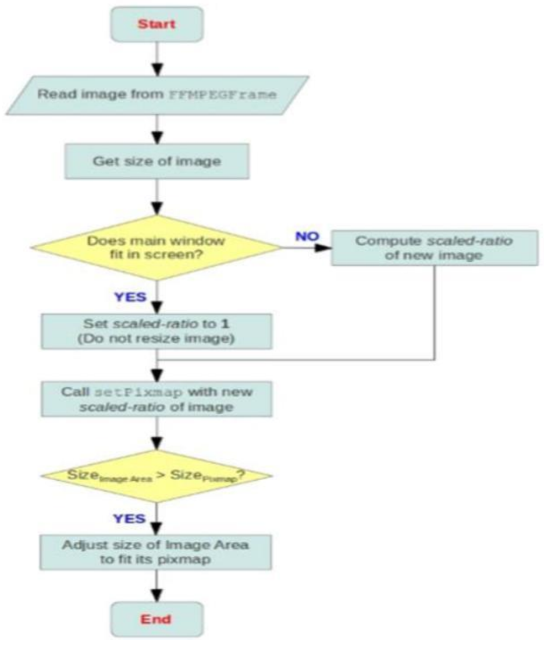

Start
Read image from FFMPEGFrame
Get size of image
Does main window fit in screen?
NO
Compute scaled-ratio of new image
YES
Set scaled-ratio to 1 (Do not resize image)
Call setPixmap with new scaled-ratio of image
Size_Image Area > Size_Pixmap?
YES
Adjust size of Image Area to fit its pixmap
End

Figure 8. Flowchart for solving the first problem

In order to check if main window fits in desktop screen, we need two input parameters. They are dimensions of desktop screen and image which will be shown on ImageArea. One difficulty that we had in solving this issue is that we have to calculate main window size before calling setPixmap() method on ImageArea. Suppose that we do in inverse way, i.e. this method is called before the computation of main window size. What will happen in this case? This will make main window bigger than screen. We have overcome this obstacle by computing approximately the main window size. This computation is done with dimension of displayed image, button height, height of title bar, space and margin values of layout.

The flowchart in figure 9 describes the steps for solving second problem. It is also the flowchart of the resizeEvent(QResizeEvent) method in ImageAre class. In this method, we need to scale not, only the pixmap of ImageArea but also the bounding boxes displayed on this pixmap. The reason is that there are two different coordinates of a bounding box. One coordinate is used for displaying on the scaled pixmap, and the other is used for storing in BBoxes file. The latter one is the BBox coordinate on original video scene (pixmap). The task of transforming BBox coordinate between original size and displaying size is also used in the accessor methods of ImageArea class, getBBoxes() and setBBoxes(). The MainWindow class gets and sets BBox coordinates through these methods. Thus, the displaying size of BBox takes effect, only in the ImageArea class, and its original size takes effect in the MainWindow class.

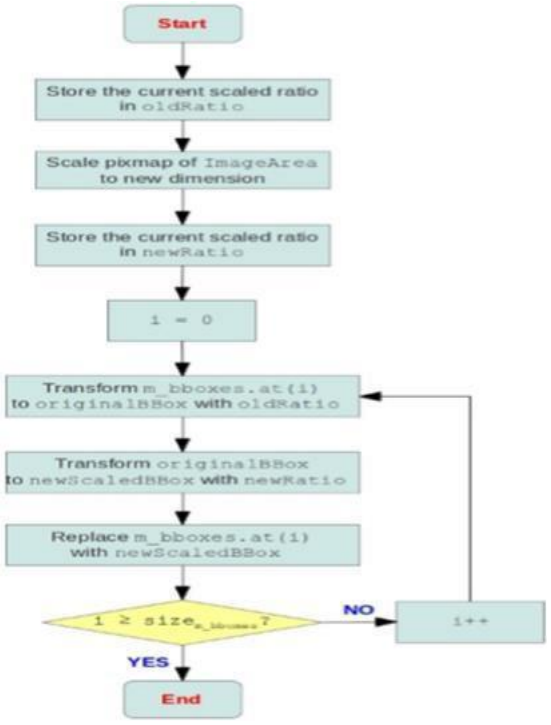

Start
Store the current scaled ratio in oldRatio
Scale pixmap of ImageArea to new dimension
Store the current scaled ratio in newRatio
i = 0
Transform m_bboxes.at(i) to originalBBox with oldRatio
Transform originalBBox to newScaledBBox with newRatio
Replace m_bboxes.at(i) with newScaledBBox
i ≥ size_m_bboxes?
NO
i++
YES
End

Figure 9. Flowchart for solving the second problem

GROUND-TRUTH FOR SEMANTIC SALIENCY

Gaussian Map, one kind of saliency map, is generated by using the results of annotating videos and Gaussian function. For this task and the task described in section II.3, we need to use OpenCV library in processing images. The following part provides some information on this library.

INTRODUCTION OF OPENCV LIBRARY

OpenCV[4] (Open Source Computer Vision) is an open-source library that includes hundreds of computer vision algorithms. OpenCV has many modules. Each module includes several shared or static libraries. Below we introduce some main modules in this library.

- Core: a compact module defining basic data structures, including the multi-dimensional array Mat and basic function used by all other modules.

- Imgproc: an image processing module that includes linear and nonlinear image filtering,

geometrical image transformations, color space conversion, histogram, etc.
video: a video analysis module that includes motion estimation, back ground subtraction, and object tracking algorithms.
- Calib3d: basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
features2d: salient feature detectors, descriptors, and descriptor matchers.
- Objdetect: detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, etc.).
- Highgui: an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.

ANNOTATE SOURCES OF VIDEOS

The aim of this task is to simulate perfect face detection, and the result of annotating video is on BBoxes file per video. With this result, we can construct perfect semantic saliency maps. There are several sources of videos. Each one contains many videos of same content, but the quality of videos is different. These videos contain the different amount of error in frames. We annotate them by using the modified version of BBBox software.

GENERATE GAUSSIAN MAP

Gaussian function[5] are widely used in statistics where they describe the normal distributions, in signal processing where they server to define Gaussian filters, in image processing where two-dimensional Gaussians are used for Gaussian blurs. In this task, Gaussian functions are used for generating saliency map. The definition of this function will be presented in the following section.

Introduction of Gaussian function

In mathematics, a Gaussian function is a function of the form:

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \quad (2.1)$$

for some real constants a,b,c > 0, and e ≈ 2.718281828 (Euler's number).

The graph of a Gaussian has a shape of symmetric "bell curve" that quickly falls off towards plus/minus infinity. The parameter a is the height of the curve's peak, b is the position of the center of the peak, and c controls the width of the "bell".

For creating Gaussian map, we used a particular form of two-dimensional Gaussian function:

$$f(x,y) = Ae^{-\left(\frac{(x-x_0)^2}{2\sigma_x^2}+\frac{(y-y_0)^2}{2\sigma_y^2}\right)} \quad (2.2)$$

In above formula, the coefficient A is the amplitude, x_0, y_0 is the center and σ_x, σ_y are the x, y dimension of the blob. The figure 10 and figure 11, taken from Wikipedia, give examples of one-dimensional Gaussian functions and two-dimensional Gaussian function.

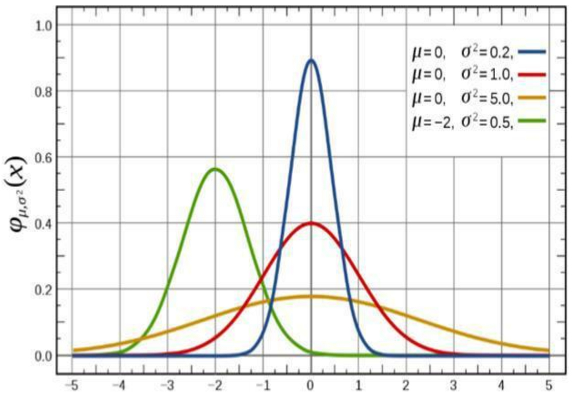


Figure 10. 1D Gaussian function

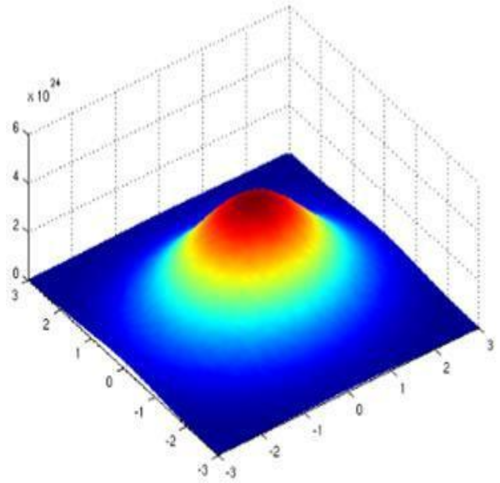


Figure 11. 2D Gaussian function

Program Description

The program for generating Gaussian Map uses the result of section 2.3 and OpenCV library. The figure 12 summarizes the processing steps.

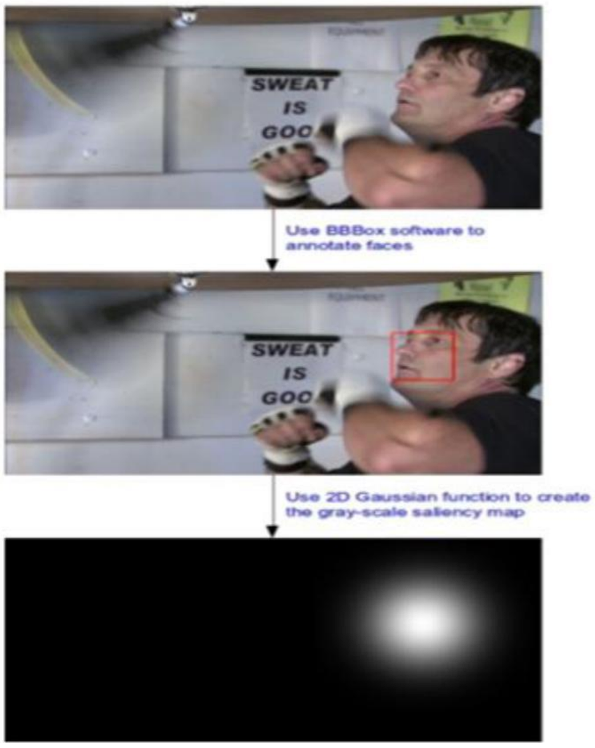

SWEAT
IS
GOO
Use BBBox software to annotate faces
Use 2D Gaussian function to create the gray-scale saliency map

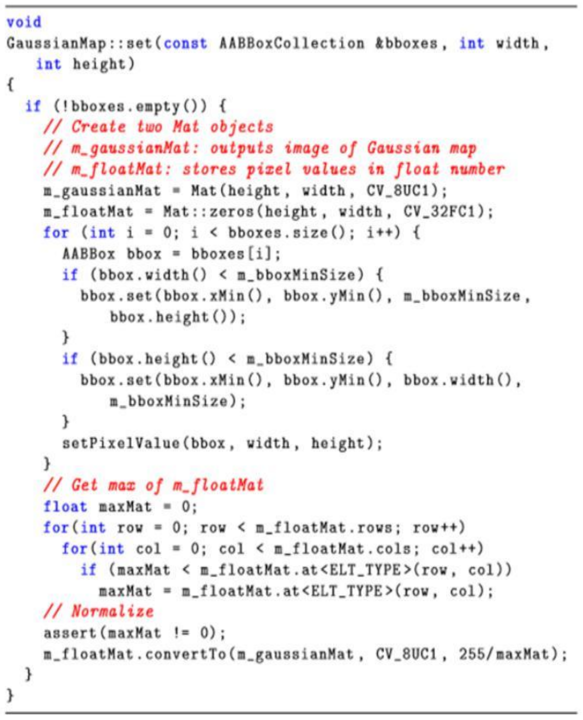
Figure 12. Processing steps for creating Gaussian Map

In this program, we created the following classes.

Table 1. Description of some important classes

Class name	Description
AnnotatedFrame	Stores a frame number and all coordinates of bounding boxes on that frame
BBoxFileReader	Opens a BBoxes file and creates an object of AnnotatedFrame class from information in each text line
GaussianMap	Computers pixel values by using formula of 2D Gaussian function and outputsagray-scale image of Gaussian map

For the task of constructing Gaussian map and outputting a corresponding image, we used the matrix structure. The class Mat represents a 2D numerical array that can act as a matrix. There are many different ways to create Mat object. Here is the way we used inside the GaussianMap class:



```
void
GaussianMap::set(const AABBoxCollection &bboxes, int width,
   int height)
{
  if (!bboxes.empty()) {
    // Create two Mat objects
    // m_gaussianMat: outputs image of Gaussian map
    // m_floatMat: stores pixel values in float number
    m_gaussianMat = Mat(height, width, CV_8UC1);
    m_floatMat = Mat::zeros(height, width, CV_32FC1);
    for (int i = 0; i < bboxes.size(); i++) {
      AABBox bbox = bboxes[i];
      if (bbox.width() < m_bboxMinSize) {
        bbox.set(bbox.xMin(), bbox.yMin(), m_bboxMinSize,
           bbox.height());
      }
      if (bbox.height() < m_bboxMinSize) {
        bbox.set(bbox.xMin(), bbox.yMin(), bbox.width(),
           m_bboxMinSize);
      }
      setPixelValue(bbox, width, height);
    }
    // Get max of m_floatMat
    float maxMat = 0;
    for(int row = 0; row < m_floatMat.rows; row++)
      for(int col = 0; col < m_floatMat.cols; col++)
        if (maxMat < m_floatMat.at<ELT_TYPE>(row, col))
          maxMat = m_floatMat.at<ELT_TYPE>(row, col);
    // Normalize
    assert(maxMat != 0);
    m_floatMat.convertTo(m_gaussianMat, CV_8UC1, 255/maxMat);
  }
}
```

Listing 1. A method used for constructing Gaussian map

After creating Mat objects, we scan the list of bouding boxes, **bboxes**. This list is stored in AnnotatedFrame object. For each box, we modify its width and height if they are smaller than a predefened value, m_bboxMinSize. Then we call the method setPixelValue() with box coordinate and image dimension (width, height). This method is used for computing all pixel values inside the area of bounding box. The pixel value is calculated on the basis of formula 2.2 with A=255. Then we find the maximum of m_floatMat matrix to convert into a matrix of unsigned char type (m_gaussianMat).

STATISTICS OF FACES IN SALIENCY MAP

Statistics on faces in saliency maps will let us know whether human faces attract viewers' attention. Or how many errors appearing on faces and where the errors happen on frames would actually attract viewers' attention? In [6], the authors assume that an error that appears on a saliency region is much more annoying than an error appearing in an inconspicuous area.

Input data that we use in this task includes:

BBoxes files: are the results of annotating videos through the BBBox software. See section II.2.2 for more details.

Saliency maps: are created by using eye-tracking device. They will let us know the areas on one video frame to which viewers pay more attention.

Error maps: are extracted from video decoder in processing systems. These maps will show the distorted areas on each frame.

By using these kinds of map and faces' locations in BBoxes file, we compute one value per video according to the methods introduced in next section. After that, we plot all these numbers on one chart for comparison.

COMPUTATIONAL METHODS

There are three different computation methods. We can see the new terms, *Overlapped BBox* and *Overlapped pixel*, in these methods.

Overlapped pixel is the common pixel of both saliency area (white area) and bounding box. Thus, it has positive (>0) value.

Overlapped BBox is the bounding box which overlaps with saliency area.

To check if one BBox overlaps with saliency area, we read all pixel values of saliency map within area of BBox. If there exists one pixel having positive value, that BBox is counted as a overlapped BBox. In figure 13, the yellow BBox on the left does not overlap the saliency area, and the green one on the right is an overlapped BBox.

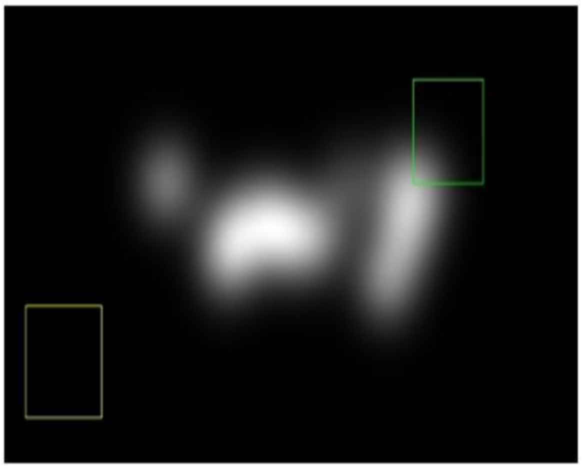


Figure 13. Example of overlapped BBox

The calculation of three methods is presented in following sections. For all methods, we compute firstly value f_i for the *i-th* frame in the video. Then we calculate the mean value of all previously-computed values. The only difference between these methods is the calculation formula of f_i. The mean value is computed in following equation.

$$f = \frac{\sum_{i=0}^{N} f_i}{N} \quad (2.3)$$

One remark is that the N number in all equations denotes the *number of processed frames*, not the number of frames contained in a video. This means that we do not process all frames, but we process the frames containing faces. To easily remember, we call method 1, 2, 3 by shortened names: *COB* (Count Overlapped BBoxes), *COP* (Count Overlapped Picels), and *FOB* (Find Overlapped BBoxes) respectively. These names represent the main task in each computational procedure.

Method COB

This method computes the value f_i for frame *i* as follows.

$$f_i = \frac{\textit{Number of overlapped BBoxes}}{\text{Number of BBoxes on frame}}, \; i = 0..N \quad (2.4)$$

Method COP

This method computes the value f_i for frame *i* as follows.

$$f_i = \frac{\textit{Total percentage of overlapped pixels}}{\text{Number of BBoxes on frame}}, \; i = 0..N \quad (2.5)$$

In above formula, we compute percentage p_i of overlapped pixels for one BBox. Then we get the numerator in equation 2.5 by calculating the sum of all p_i values.

$$p_i = \frac{\textit{Number of overlapped pixels in one BBox}}{\text{AreaBBox}} \quad (2.6)$$

and

$$Area_{BBox} = width_{BBox}.height_{BBox}$$

Method FOB

This method computers the values f_i for frame *i* as follows.

$$f_i = \begin{cases} 1 & , \textit{ if there exists one overlapped BBox} \\ 0 & , \textit{ otherwise} \end{cases} \quad (2.7)$$

STATISTICAL RESULTS

This section illustrates computed results in graphs, and the Gnuplot[6] tool is used for plotting. As mentioned above, we calculate one per video by using the methods introduced in previous section. There are many sources of videos. We choose one of

sources which contain nine videos. The best quality video is video 9. However, the remaining videos have worse quality, and the amount of noise appearing on one frame in every video is different. After the computation step is done, we plot all these numbers on one diagram for comparison. The statistical results in this part will be used for later comparison in section 4.3.

Firstly, through saliency maps, we examine the percentage of faces appearing in the saliency areas. Three methods are used for computing mean values.

Observing the graph in figure 14, we see that method COB and FOB have produced the same values. However, the values produced by method COP are slightly different because we compute the percentage of overlapped pixels for one bounding box. With this computational way, we know exactly how many percent of faces overlap with saliency areas. The mean values range from 0.954194 to 1 (100%). This result shows that viewers pay more attention to faces on frame.

Secondly, through error maps, we examine the percentage of faces appearing in the distorted areas. Only method COB and COP are used for computing mean values. These values are plotted on the graph in figure 15.

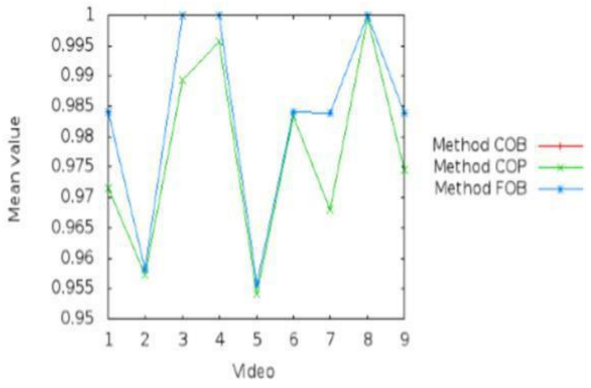


Figure 14. Statistics of faces in saliency maps

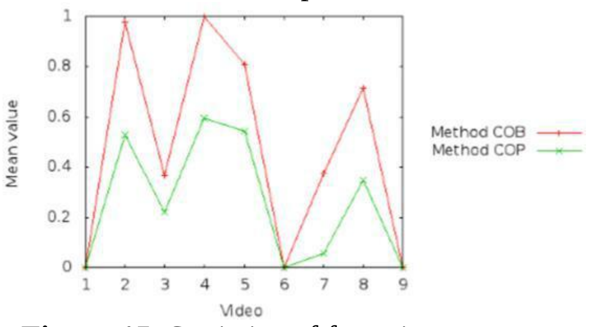


Figure 15. Statistics of faces in error maps

Finally, we build a combination of saliency map and error map. This job is performed by calling the **mul** method on **Mat** object as in following code snippet.

```
// Get list of BBoxes on AnnotatedFrame object
AABBoxCollection bboxes = aframe.getBBoxes();
// Performs an element-wise multiplication of
// the two matrices (errorMap & eyeTrackerMap)
Mat A = errorMap.mul(eyeTrackerMap);
// Compute value for one frame
float v = compute(A, bboxes, method);
```

Listing 2. A method used for constructing Gaussian map

Using these combined map, we examine how many errors appearing on faces and the percentage of errors would attract viewers' attention. Method COB and COP are used for computing mean values, and these values are plotted on the graph in figure 16. The videos having high mean values (video 2, 4, 5, and 8) will make viewers more annoyed than other videos.

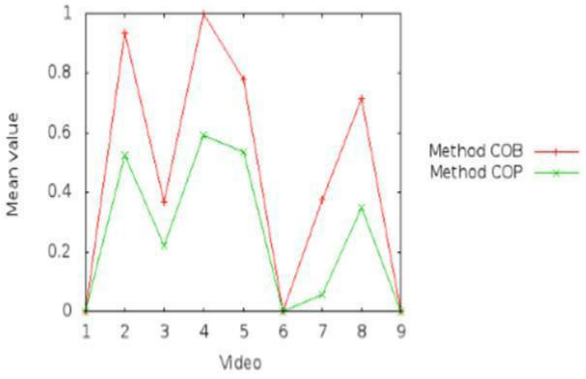


Figure 16. Statistics of faces in combination of saliency maps and error maps

AUTOMATIC FACE DETECTION

The purpose of this task is to add automatic face detection for the automatic semantic map construction. We give a brief introduction of Viola-Jones object detection algorithm in below section. This algorithm is implemented in an example program[7] coming with OpenCV library.

INTRODUCTION OF VIOLA-JONES ALGORITHM

This algorithm was proposed by Paul Viola and Michael Jones [5]. The proposed object, detection framework provides competitive object detection rates in real-time. There are three main contributions of this framework. We will introduce these ideals briefly below.

The first contribution is a new image representation called an integral image that allows for very for very fast feature evaluation. Their object detection framework classifies images based on the value of simple features. One critical motivation for using features rather than the pixels directly in that the feature-based system operates much faster than a pixel-based system. With the use of the integral image, rectangular features can be evaluated in constant time.

The second contribution is a method for constructing a classifier by selecting a small number of important features using AdaBoost[8]. In order to select the features and train the classifier, Viola-Jones use a modified version of the AdaBoost algorithm developed by Freund and Schapire in 1995.

The third major contribution is a method for combining successively more complex classifiers in a cascade structure which increases the speed of detector by focusing attention on promising regions of the image. The cascaded classifier if composed of stages. Each one contains a strong classifier. The job of each stage is to determine whether a given sub-window is definitely negative (non-face) or maybe positive. The detection process is shown in figure 17. This figure is taken from the article of Viola-Jones.

The strong classifiers are arranged in order of complexity. If a sub-window passes the first classifier, it will be evaluated at the second classifier. A positive result from the second classifier triggers a third classifier, etc. The initial classifier discards a large number of negative sub-windows with very little processing. Successive classifier discards additional negatives but require more processing. Through this model, the numbers of sub-windows have been decreased considerably.

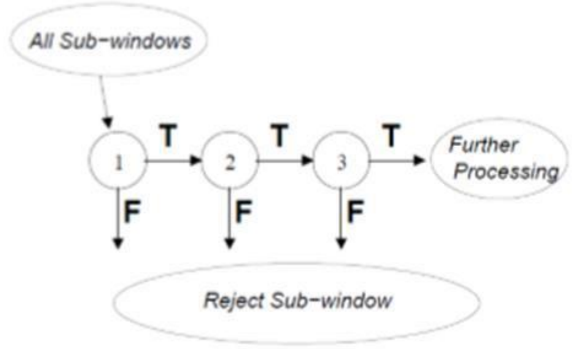


Figure 17. Cascade Architecture

The cascade structure reflects the fact that within an image an excessive amount of sub-windows are negative. In stead of finding faces, the algorithm should discard as many negatives as possible at the earliest stage. While a positive instance will trigger the evaluation of each classifier, this event happens rarely.

STATISTICAL RESULTS WITH AUTOMATIC DETECTION

In section II.3.2, we have shown that faces attract more viewer's attention and thus face is a good idea for semantic maps. The result of this work interests us because we want to know if automatic face detection is good enough, compared to manual ground-truth face detection. In this section, we perform the same experiment that we have done in section II.3. We also use the similar input data as in previous experiment. The implementation of Viola-Jones algorithm in OpenCV has been modified in order to produce the BBoxes file. After the face detection is done, we use the produced BBoxes file as input data of computation program. The content of this file is described in section II.1.2.

One problem with face detection is that a detector trained on frontal faces is unable to detect profile faces. Therefore, instead of using a frontal-face detector or a profile-face detector, we use both of them to detect as many faces as possible. In this case, we will have two BBoxes files generated by both detectors. Thus, we will have to merge files into one file.

Observing the graph in figure 18 and the one figure 14, we see that the variation of mean values is quite considerable. For all methods, the mean values in the former are greater than in the latter. The percentage of variation is shown in table 2.

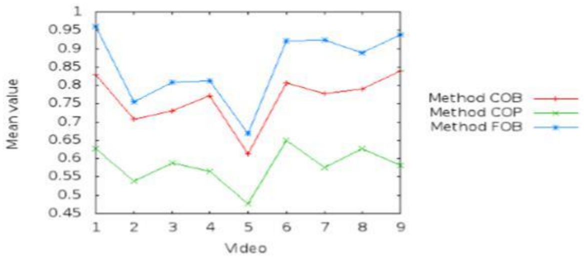


Figure 18. Statistics of faces (automatic detection) in saliency maps

Table 2. Percentage of variation in mean values

Video	Method COB (%)	Method COP (%)	Method FOB (%)
1	15.5218	34.5074	2.269
2	25.1572	41.8283	20.3616
3	27.031	40.1353	19.1257
4	22.9167	43.0406	18.75
5	34.3675	47.6441	28.6868
6	17.8407	33.4175	6.1338
7	20.6969	39.3025	6.0099
8	21.0391	37.1462	11.1111
9	14.4102	39.3511	4.6759

With the comparison in above table, we can see that the reliability of automatic face detection is not high. By looking at "Method COP" curve in figure 18, we see that the mean values approximately range from 0.45 to 0.65. In figure 14, these values approximately range from 0.95 to 1. These ranges let us know that sometimes the detection is false. Using the results of unreliable detection, we will compute the lower mean values. For the two remaining statistics, the mean values are slightly different from those in previous experiment (manual detection).

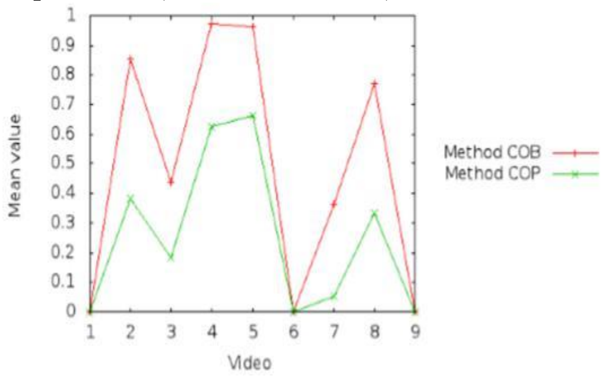


Figure 19. Statistics of faces (automatic detection) in error maps

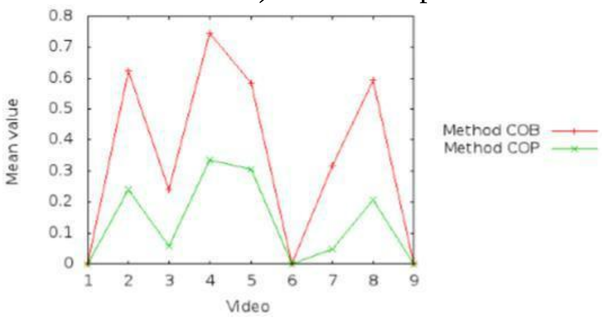


Figure 20. Statistics of faces (automatic detection) in combination of saliency maps and error maps

CONCLUSION

This goal has been achieved by:

Modifying BBBox software: BBBox is used for annotating the human faces in each frame of a video. There are some problems with the actual version of BBBox. This task must be done so that this software is able to annotate High-definition videos.

Annotating videos: We use the modified BBBox software to perform this task. The results are BBoxes files which are created by BBBox software.

Generating Gaussian map: This map is generated by using the results of annotating videos and the formula of Gaussian function. It lets us know the positions on a video frame where human faces are located in. In Gaussian map, the white show the salient locations in the image.

Examining statistics of faces in saliency map: Using the methods, we compute the values for all videos and plot all these numbers on one diagram for comparison. This statistics will let us know if human faces attact viewer's attention. Or how many percent of errors appearing on faces and where the errors happen on frames would actually attract viewers's attention?

Automatic Face Detection: This task helps us detect human faces automatically. The produced results are the BBoxes files containing locations of faces.

Examining statistics of automatically-detected faces in saliency map: We are interested in the output of this task. Based on this output, we can compare with the output in previous statistics. As a result, we can evaluate the effectiveness of using the automatic detection.

BIBLIOGRAPHY

JASMIN BLANCHETTE AND MARK SUMMERFIELD, "C++ GUI Programming with Qt 4", Prentical Hall PTR, Upper Saddle River, NJ, USA, 2006.

Gary R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface", 2006.

H. BOUJUT, J. BENOIS-PINEAU, O. HADAR, T. AHMED, AND P. BONNET, “Weighted-MSE based on Saliency map for assessing video quality of H.264 video streams”, IS&T / SPIE Electronic Imaging, San Francisco: United State, 2011.

L. ITTI, C. KOCH, AND E. NIEBUR, “A Model of Saliency-Based Visual Attention for Rapid Scene Analysis”, IEEE Trans, On Pattern Analysis and Machine Intelligence, 20(11):1254-1259, 1998.

PAUL VIOLA, MICHAEL JONES, “Robust Real-times Object Detection”, In International Journal of Computer Vision, 2006.

X. FENG, T. LIU, D. YANG, AND Y. WANG, “Saliency Based Objective Quality Assessment of Decoded Video Affected by Packet Loss”, In ICIP, pages 2560-2563, 2008.