Master in Artificial Intelligence (UPC-URV-UB)

Master of Science Thesis

Automatic Demarcation for Videofluoroscopy Swallowing Study

Matias Lizana García

Advisor/s:
Ulises Cortés and Marcos Faundez

05/09/2012

Abstract

Videofluoroscopy tests are designed to analyze the swallowing response of the patient. Lot of patients die due to swallowing disorders, and physicians want to analyze this procedure to detect this swallowing problems. The principal problem is that the time spent to analyze a video is so long, and has to be done manually, and this derives to a high cost of time and money to the health system.

Hyoid Marker is an application that provides an automatic demarcation for videofluoroscopy studies, saving to the physician the time to mark important objects in the video, and allowing him to dedicate his time analyzing only the medical aspects of the video.

Acknowledgments

I want to thank my tutors Ulises Cortés (to his rapid management and interest for my work) and Marcos Faundez (to advise me during this work, and help me to choose this topic as the master thesis).
To Jordi Ayza, for his interest on my trajectory, and to introduce me to the world of health informatics.

Also I am grateful to my scientific friends Ruben Ortega, Alberto Viñas and Alberto Quintana, who discussed this work with me during our meetings.

Finally I want to thank my parents, who are always by my side and give me a lot of support.

Table of Contents

1 Introduction

This work shows how Computer Vision techniques can give support in the analysis of medical images, concretely in this case, in videofluoroscopy studies. These studies are used to detect different disorders in swallowing, like Oropharyngeal dysphagia. The study provides an x-ray video where the internal swallowing process can be viewed and analyzed. Medical technicians have to analyze this video manually, so the process tends to be slow and cumbersome.

The aim of Computer Vision in this project is to assess the medical technicians in this process, analyzing in a semi-automatic way the videos provided by the doctors, to create the statistics needed to analyze the patient swallowing.

1.1 Motivation

The aim of CCI[1] (Integration Competence Center), a group from Tecnocampus (that is a research, teaching and entrepreneurs building in Mataró) where the author of this thesis is working now, is to provide integration solutions to the health systems, more specifically in Catalunya. Also, signal processing research group[2] of Marcos Faundez also in Tecnocampus, try to impulse research projects with health professionals. Here is where Pere Clavé, a physician of "Hospital de Mataró"[3], specialized in digestive system and neurological disorders, propose the problem that he has with the videofluoroscopy videos analysis (explained in the next sections).

One of the important points in this project is the amount of fieldwork outside the theory and books, doing visits to the hospital, talking with the physicians and nurses, and seeing the real situation. The main goal of the project is to provide an application that solves a real problem.

1 http://cci.tecnocampus.cat
2 http://www.tecnocampus.cat/web/estudis-universitaris/248
3 http://www.csdm.es/

2 State of the Art

Before entering into technical details and discussions, it is important to know about the fields that have been studied in this project.

First of all, a description of the medical concepts are shown, which are swallowing procedure and disorders, and videofluoroscopy. After this, the technical part is introduced, presenting computer vision applications and techniques used in medicine.

2.1 Medical concepts

This paragraph contains all the concepts related to the medical part. It is important to know basic things about the medical area, related to swallowing procedures and videofluoroscopy technique, that help to adjust the algorithms, according to medical needs and specifications (explained in Chapter 3 : Problem Definition).

2.1.1 Swallowing Procedure

The main aim of this project is the analysis of the swallowing behavior, so first swallowing procedure has to be described, in a medical way.

Swallowing is known scientifically as deglutition, and it is the process of passing something from the mouth, trough pharynx to esophagus. The process can fail and the object can go to the trachea, and a pulmonary aspiration can occur. Epiglottis has the role of closing or opening the respiratory via, and this movement is provided by the hyoid bone (shown in Illustration 1 and Illustration 2), that is the main element studied in this work.

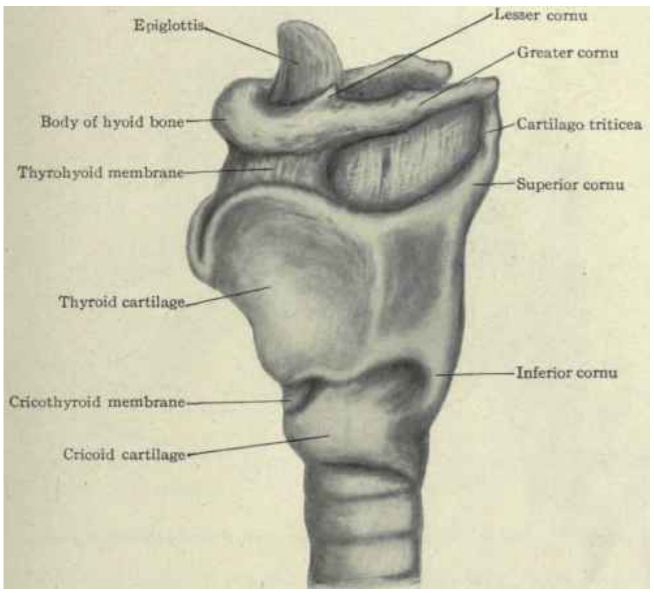


Illustration 1: Hyoid Bone Side View

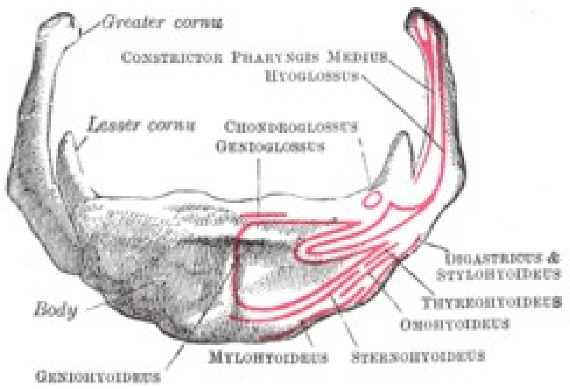


Illustration 2: Hyoid Frontal View

In Illustration 3 a representation of the superior digestive system is shown, with the important parts treated in this project marked.

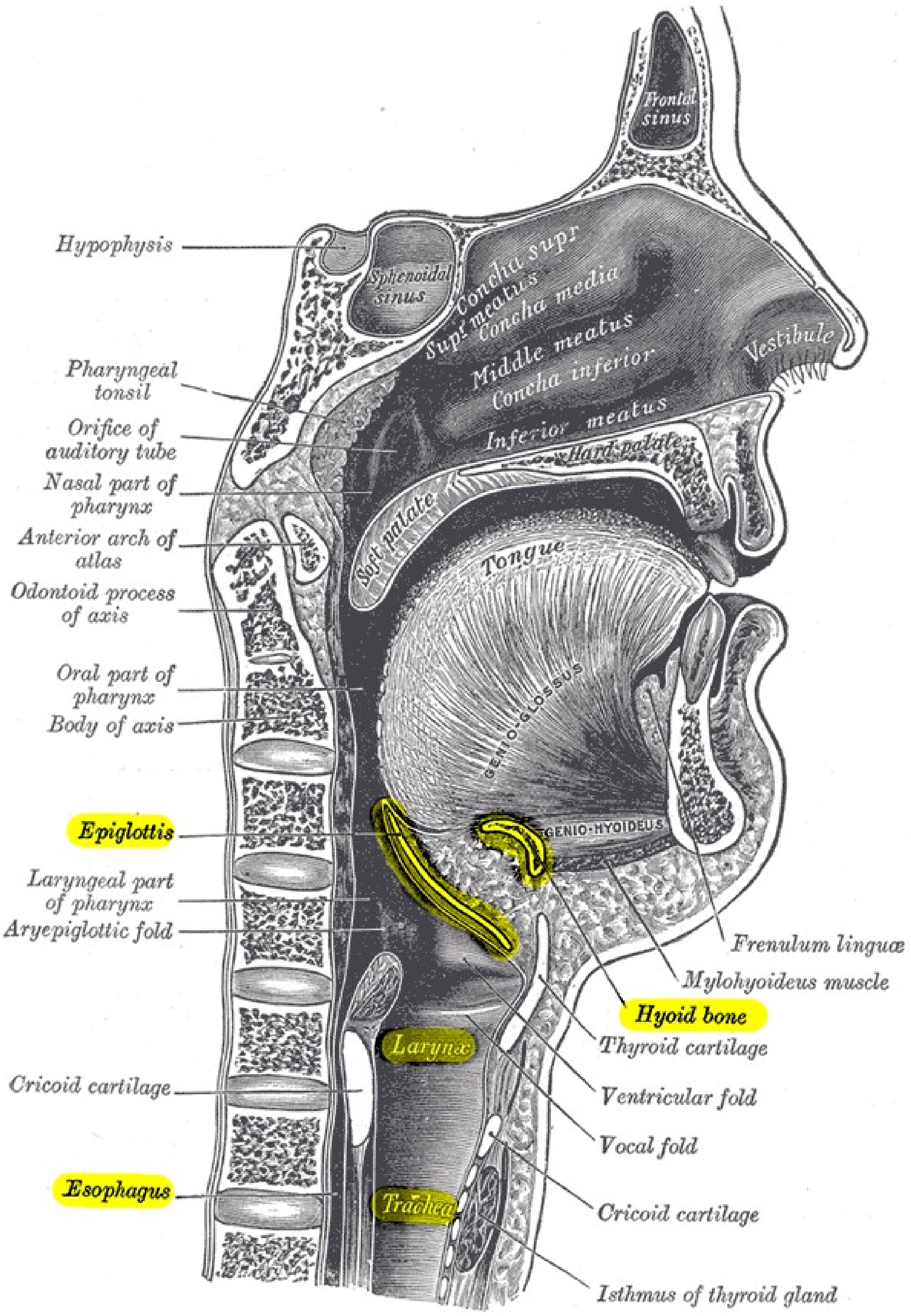


Illustration 3: Swallowing System

The next Illustration 4 and Illustration 5 show a diagram of the swallowing process (obtained from [1]):

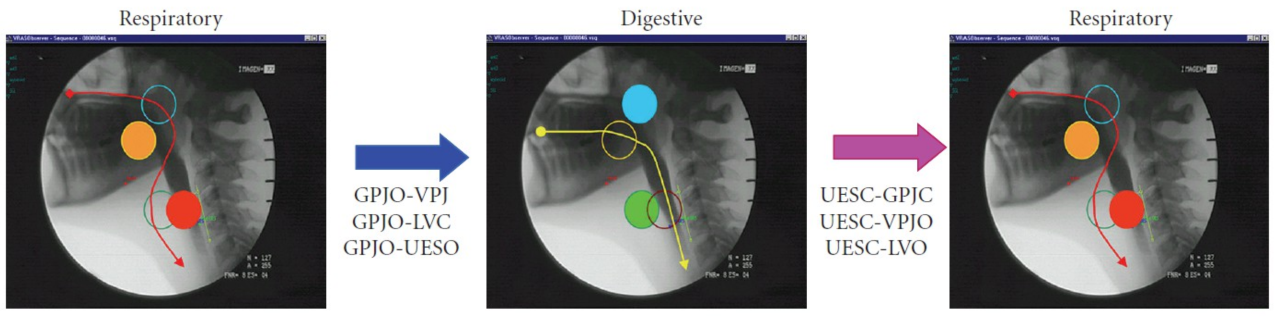


Illustration 4: Swallowing Procedure Steps

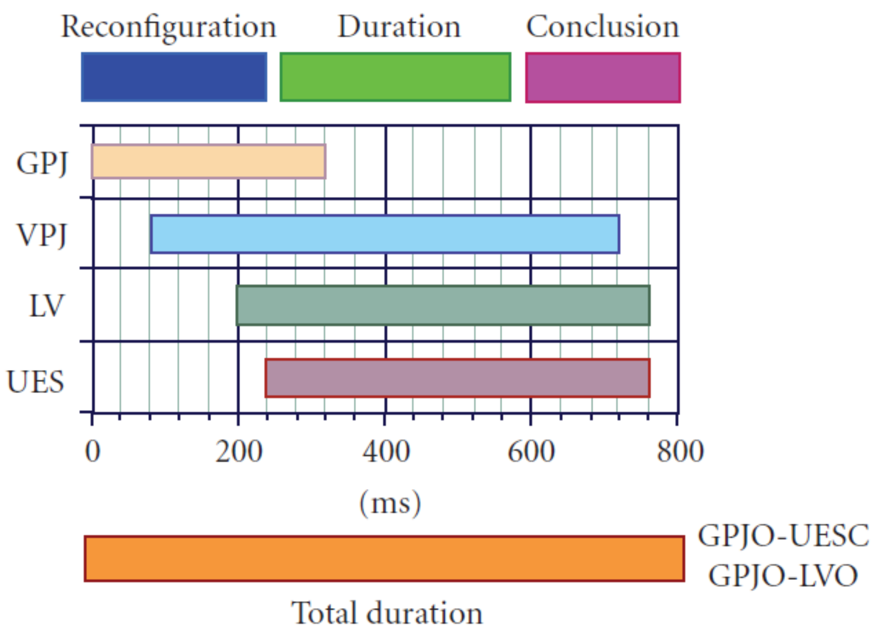


Illustration 5: Swallowing Procedure Diagram

The diagram is based on the different configurations of the oropharynx during swallowing response. Each time segment is defined by the opening and closing events in the different parts, that are GPJ (glossopalatal junction), VPJ (velopharyngeal junction), LV (laryngeal vestibule and UES (upper esophageal sphincter).

One of the analysis of swallowing procedure consists of measure that times in the diagram are correct (or similar to the normal times). Another part is analyzing the movement of the hyoid bone, to concord with this times and see if the movement is also correct. This is the main goal of this project.

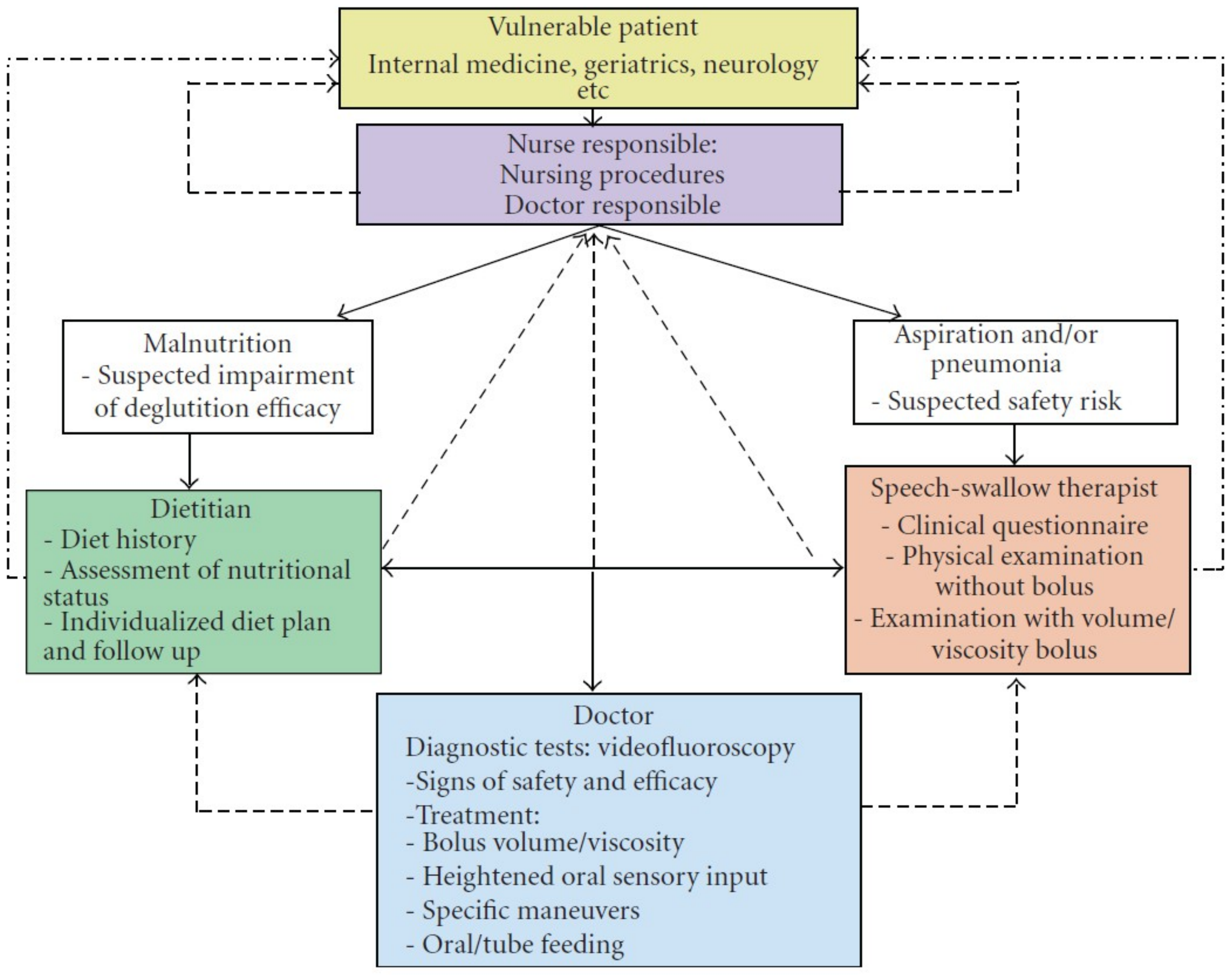
2.1.2 Swallowing Disorders

When this procedure doesn't work properly, it means that there are a malfunction in the swallowing system (see [1]). One type of malfunctions are structural alterations (esophageal tumors, neck osteophytes or Zenker's diverticulum among others, also a side effect in patients in head and neck cancer undergoing radiotherapy.

However, malfunctions in swallowing process (known as oropharyngeal dysphagia), are commonly associated to a functional disorder of deglutition, due to systemic or neurological diseases, elderly age or strokes. Dysphagia can lead to pneumonia complications or malnutrition, causing dead to the patient.

To diagnose the problem, a full team has to be selected, in which there are nurses, speech-swallow therapists, gastroenterologists, neurologists, surgeons, rehabilitation physicians, dietitians, radiologists and geriatricians, and develop functions like early identification of dysphagia, detection of bio-mechanical events responsible for functional dysphagia and design of a set of therapeutic strategies to provide effective deglutition to the patients.

In Illustration 6, the diagram to diagnose the dysphagia (provided by "Hospital de Mataró") is shown:



Vulnerable patient
Internal medicine, geriatrics, neurology etc
Nurse responsible:
Nursing procedures
Doctor responsible
Malnutrition
- Suspected impairment of deglutition efficacy
Aspiration and/or pneumonia
- Suspected safety risk
Dietitian
- Diet history
- Assessment of nutritional status
- Individualized diet plan and follow up
Speech-swallow therapist
- Clinical questionnaire
- Physical examination without bolus
- Examination with volume/ viscosity bolus
Doctor
Diagnostic tests: videofluoroscopy
-Signs of safety and efficacy
-Treatment:
- Bolus volume/viscosity
- Heightened oral sensory input
- Specific maneuvers
- Oral/tube feeding

Illustration 6: Dysphagia Diagnosis Diagram

2.1.3 Videofluoroscopy

Videofluoroscopy (VF) is a medical procedure used to analyze the internal behavior of a patient in a video format. This procedure is composed by two main elements:

First, the machine that provides the video, that is an X-Ray machine. This machine is commonly used in medical procedures to detect a lot of pathologies in the whole body (from broken bones to organ functionality).

Second, a solid or liquid bolus is administrated to the patient. This bolus is always combined with the radio-contrast barium sulphate material, which appears in the video as a low contrast element that can be easily detected.

In a visit to the “Hospital de Mataró”, the procedure has been analyzed in detail:

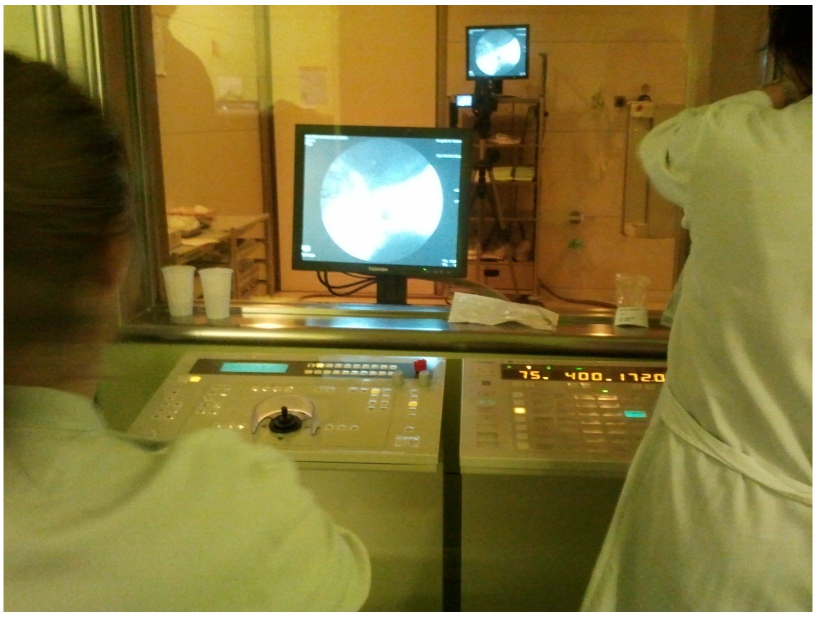


Illustration 7: Videofluoroscopy Study in Hospital de Mataró

A nurse selects the information of the patient from the worklist (the list used to store all the patient orders), provided by the RIS (Radiology Information System), and checks that the patient that comes to the procedure is scheduled in the procedures list. The patient enters to the X-Ray room, and meanwhile another nurse prepare different recipients with the bolus material, in different quantities (5, 10 and 20 ml, liquid and solid). When the bolus is prepared, the nurse enters to the room and administrate the bolus to the patient, which swallows it with the indications of the nurse (that is outside the room, recording the video take). Detailed procedure steps are shown in Illustration 8.

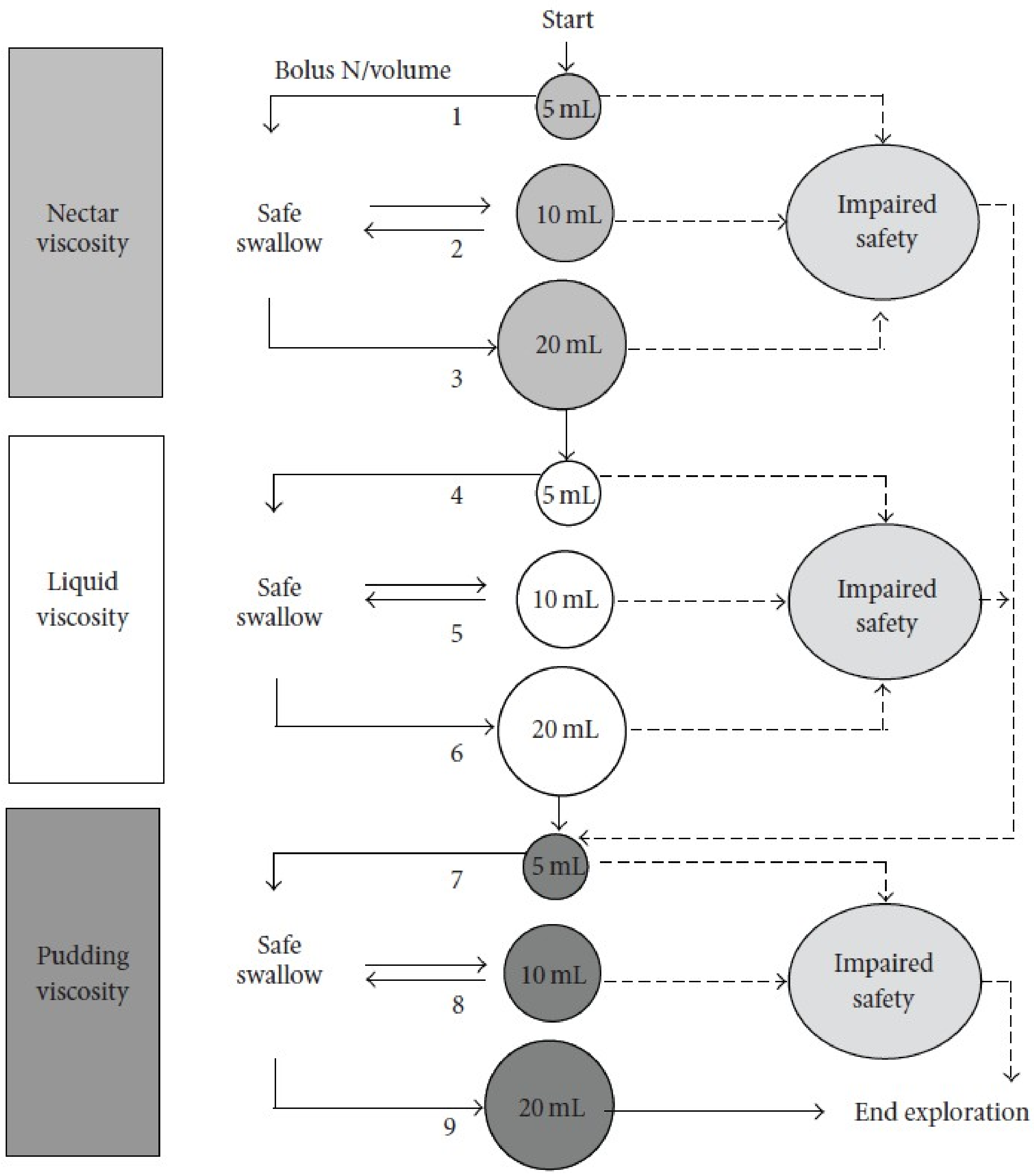


Illustration 8: Swallowing Procedure

When all video studies are captured, the camera is carried to a computer, where all the videos will be rendered and processed in digital format. In Chapter 3 this process will be explained in detail.

2.2 Computer Vision

Computer Vision is applied to many fields and types of medical assessment, but basically its main purpose is to provide information about the processed image. An image can be analyzed with multiple techniques, and each of them is a basic tool to analyze different aspects of this image. Most of computer vision algorithms use different combinations of these techniques to create a robust algorithm with an specific functionality.

In this sub-chapter basic techniques used in this work are explained, to deal later with more complex algorithms discussed in Chapter 4.

2.2.1 Histogram

Histogram is the basic representation of the intensities quantification of the pixels in an image, it means, is a vector that contains, for each possible color in an image, the number of pixels that deal with this concrete value.

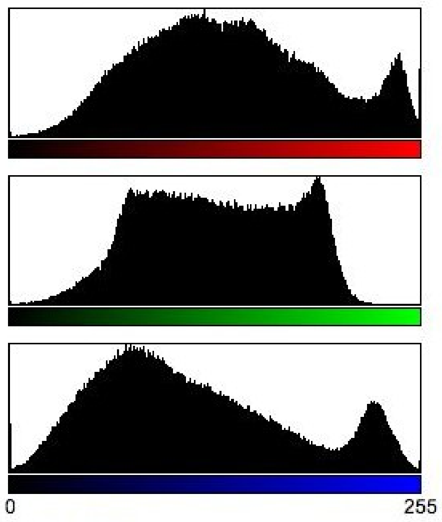


Illustration 9: RGB 8-bits per channel Histogram example

Histogram provides basically, the color distribution of an image, and with this it can be known if an image has more “dark” or “light” tones, from each color channel (normally RGB, or grayscale).

2.2.2 Binarization

This method converts a scale color image (for instance, 255 values for channel) to a binary scale of color (0 or 1). This is a mapping of one part of the colors to the 0 value, and the other part to the 1 value. To do this, a threshold is provided, which marks the value of this partition.

For instance, as shown in Illustration 10, binarization can be used to detect concrete objects in an image, in this case, text. Is easy to see that text is more darker than the background, so a threshold

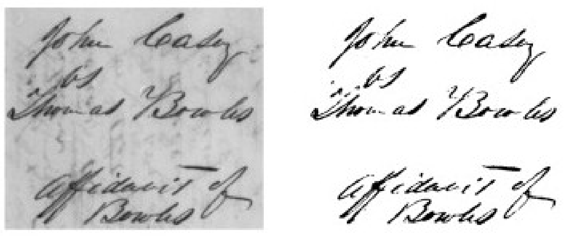


Illustration 10: Text binarization image extracted from Hedjam article (spatially adaptive statistical method for the binarization of historical manuscripts)

can be determined to separate what is text and what is background, using a concrete value of gray intensity. To determine this value, the histogram is needed. The histogram shows the distribution of the color, so text and background can be detected easily in this case, as shown in Illustration 11.

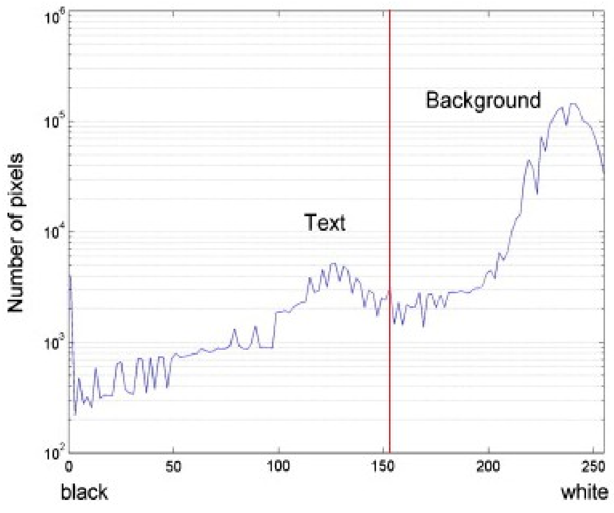


Illustration 11: Text histogram binarization

This technique is widely used in the computer vision area, because provides the way to work with a binary image, which is more easy to process and operate, concretely with morphological operators.

The main problem of binarization is that objects can only be segmented if the difference between object colors is noticeable. In Chapter 4, problems with binarization will be shown, due to absence of contrast in images.

2.2.3 Masks and Convolution Operation

Masks are used in computer vision to perform operations within images. The process is called Convolution, and it means that two functions (f and g) are combined with a set of rules to produce a third function (h), that is the resulting function.

Mathematically represented, as seen in Illustration 12, the function f is shifted over g, computing the product of the two functions, producing a third function h.

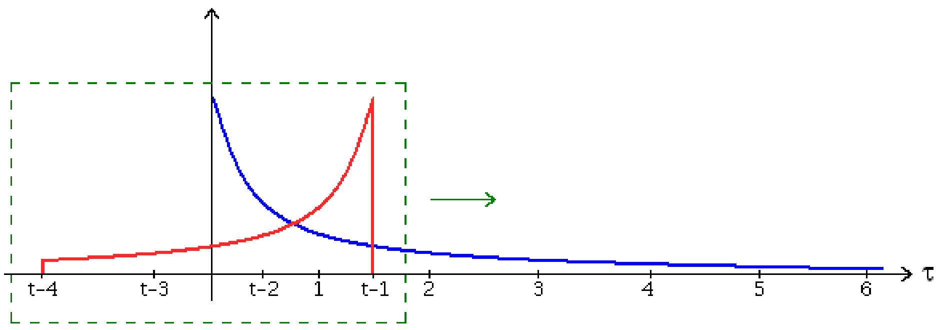


Illustration 12: Function convolution

In image processing, the first function is the mask, that will be convoluted over the second function, the image. The resulting function will be an image with the modified values due to the convolution (as it can be seen in Illustration 13). Mask is shifted over the image, and each mask pixel is multiplied with the corresponding image pixel, producing the convoluted image. The target pixel is computed using a function, that is normally the mean function.

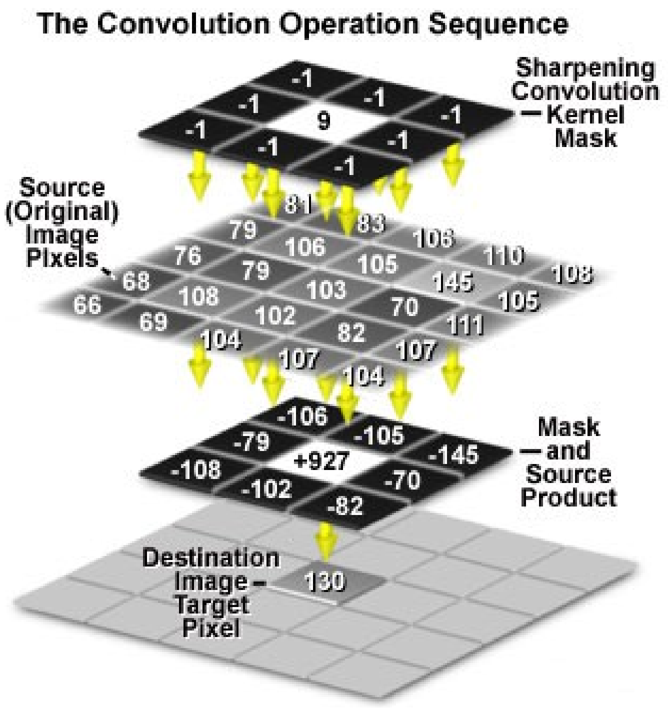


Illustration 13: Convolution Operation

Mean Filter

One typical case of convolution operation is the mean filter (the same as Illustration 13). The mean filter is a process that makes the mean between image pixels in the mask window, and provides a smooth effect, blurring areas and edges on the image. It is also a way to eliminate noise, as seen in Illustration 14. This technique will be used in this work, trying to delete the noise in the videos provided.

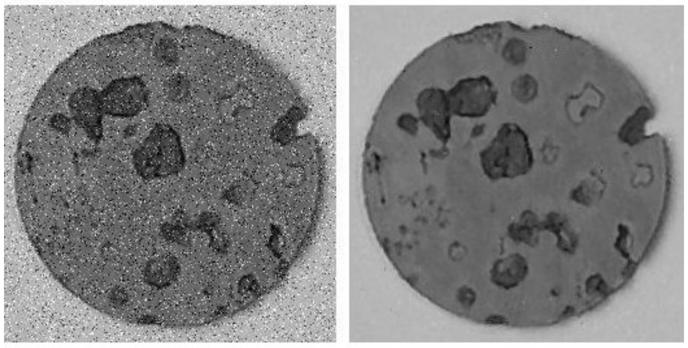


Illustration 14: Mean Filter image extracted from http://tracer.lcc.uma.es/problems/mfp/mfp.html

Gradient

Another technique widely used in computer vision is the gradient extraction of an image. The gradient wants to represent the “changes of intensity” in an image, represented as a vector for each pixel, which defines in which direction the change is more representative (vector direction), and with which intensity (vector module). This technique is designed to find the contours in an image.

To provide gradient information, one mask is provided for each dimension (in 2D images, there are one mask for x-coordinates and another for y-coordinates). Depending on the mask, the changes between intensities can be more or less important, and also the mask can take into account more than one dimension to calculate the gradient.

Sobel Operator

Sobel operator, is one of the most well-known techniques to calculate the contours. It takes two 3x3 masks, as shown in Illustration 15, that are multiplied with the image to obtain the gradient in each coordinate.

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

Illustration 15: Sobel masks

Once gradient is calculated, the vector for each pixel is computed with this formula:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

Illustration 16: Gradient vector calculation

The intensity of the vector module is represented in a grayscale image (Illustration 17), that shows finally the contours in the image.

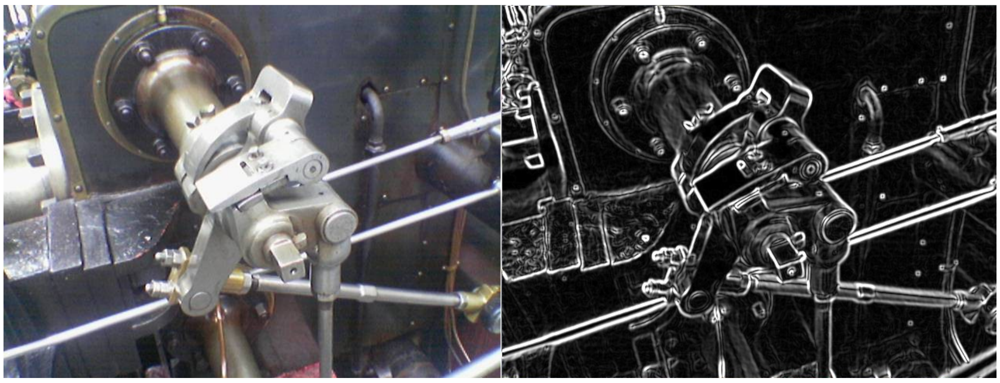


Illustration 17: Sobel operator

Canny Edge Detector

Another technique to find edges on images is the Canny technique [8]. It works like Sobel operator but with some modifications.

It uses another mask to compute the gradient in 3 directions (vertical, horizontal and diagonal), using a Gaussian mask that makes the method more robust to random noise (see Illustration 18):

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

Illustration 18: Gaussian Mask

The computation of the gradient module and angle is the same as Sobel, and the main difference is that finally, it uses a search method that find the “real” edges, based on the assumption that an edge is a “real” edge if its module is greater than the modules of the neighbor vectors (called non-maximum suppression). Also it uses the hysteresis threshold (a range with two thresholds, low and high). With high threshold the important edges are marked, and starting from these edges, the algorithm define paths through these edges, using the low threshold, that specify the correct direction of the path using the directions of the other edges.

This method is more complex than Sobel, but helps to detect better edges in noise images, and this will help in this work.

Basically, computer vision is used to extract information from an image, and this information wants to be exploited in some manner, so computer vision output data is closely related to data mining and machine learning techniques, those want to use the information extracted from the image so typically, computer vision is the input of machine learning algorithms.

3 Problem Definition

Videofluoroscopy study is a process that wants to detect the movement of the hyoid during the swallowing process from a patient. This will be done doing some marks on the video, which will be analyzed later with plots and statistics, to evaluate the state and responses of the patient.

Videos are obtained from the Videofluoroscopy procedure in the X-Ray machine. To obtain this videos, a rudimentary process is used, which is to record directly the monitor output with the camera. They do this because their system doesn't provide a digital output from X-Ray machine. Later, this videos are saved in a computer in a digital format, downloaded from the digital camera.

A physician starts to process the videos in this way:

He takes a video from a patient, and he puts the data related with this patient and his clinical order, to save in the hospital clinical history. When the video is identified, he starts to put temporal marks on the video, to define the starting and ending points of the swallowing process (called opening and closing).

Later, a reference point is needed to define the hyoid path (because patient can move head in all directions in the sagittal plane, and the position of the hyoid has to be referenced by a static point in patient's body, so vertebrae bones are marked to take this reference point. Later, the hyoid start position is marked too.

Finally, physician starts to follow the hyoid and vertebrae position, frame by frame, marking new positions on each frame. When all the frames are marked, a plot of the hyoid path is generated, showing to the physician the behavior of patient's swallowing. With this information, physician can make a diagnosis about the patient, and take decisions over his treatment.

So what is the problem in this situation? How computer vision can help in this procedure?

The principal problem that physicians need to solve is the time problem. Let's try to calculate the effort made by physicians with procedures done in a whole day (real data extracted from a journey in “Hospital de Mataró”):

A day of videofluoroscopy procedures contains over 5 studies in a day (made in 1 or 2 hours approximately). Each of this studies is composed by 9 takes (as explained in Chapter 2, see Illustration 8) that are three nectar, three liquid and three pudding takes, so in total, the number of videos captured (NV) are:

$$NV = Studies * Takes = 5 * 9 = 45$$

Now, as a mean statistic, each video can be processed and marked in the computer in 15 minutes minimum), so totally, a journey of videofluoroscopy studies can consume over 11 hours to a physician to mark all the videos, that is obviously a considerable waste of time.

The main aim of this work is to provide an automated way to mark the videofluoroscopy videos, to avoid this effort made by the physician, saving time and money to the hospital. In Chapter 4, methods tried to do this automatic marking are shown and discussed in detail, and difficulties discovered with the real videos provided are faced.

4 Research and discussion

There are so many possible approaches to solve the problem submitted in Chapter 3. In the first step, the research made in this area to discover possible techniques applied to this type of videos is shown. Once all theoretical techniques are studied, the reality has to be faced, seeing what techniques can be applied or not, studying the video format and quality, to know how possible is to work with this videos. The final step is to decide over them, which is the better method to use (testing them with the provided videos).

4.1 Computer vision techniques

In this section the research about methods is discussed, to solve the videofluoroscopy problem. Obviously, the first idea is to search about videofluoroscopy papers, which can contain techniques that segmentate the hyoid (that is the main goal of the work). The next subsections explain all the techniques found. ASM is explained in more detail, because it is described in a paper related to videofluoroscopy videos, and has been implemented in the firsts stages of this work. The other methods are described as an overview, to take an idea of its functionality.

4.1.1 Active Shape Model

Aung's paper [2] uses the ASM (Active Shape Model) as the main technique to solve the problem. ASM is described in detail in [3] by Cootes. This method tries to define a general model structure, which can fit with new similar structures in the future with new images. First, a learning process is provided by the physician, which marks the model in multiple images as a training set, a model represented in Illustration 19.

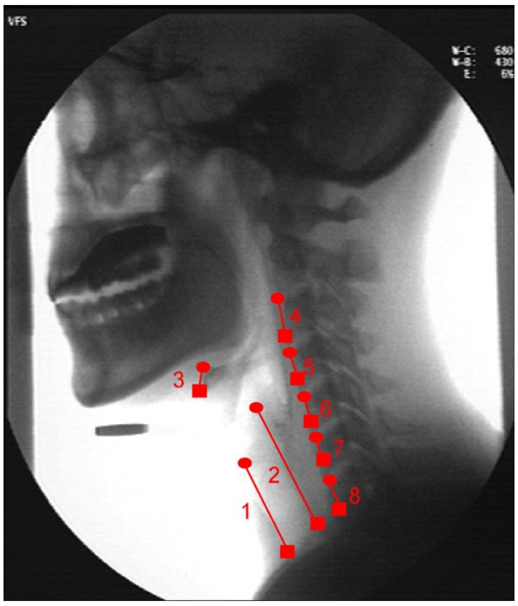


Illustration 19: 16-point model for ASM

When all the images are marked, a general model is computed with the mean of all model points, and saving the deviation between each of them. This deviation creates a matrix of possible deviations in each dimension, which will be used in the automatic labeling process, to adapt the shape form (see Illustration 20, contains the shape with different statistic deviations):

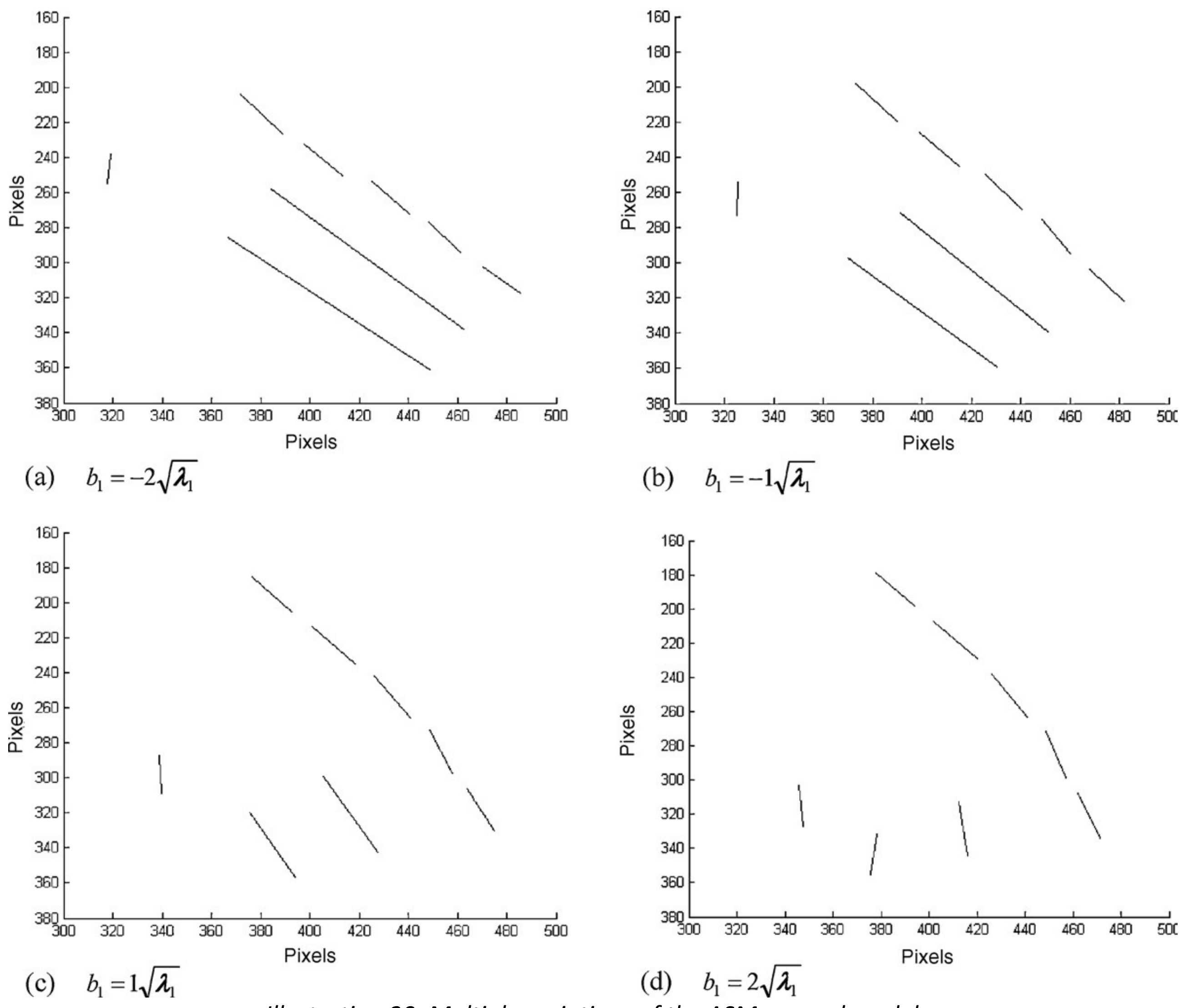


Illustration 20: Multiple variations of the ASM general model

When the general model is created, the next step is to detect the model in the new images provided. The model, using the variations calculated between all training images, can rotate, scale and move over the 2D plane. The goal is to find the better model parameters that fit with the new image. But to fit the model, first is needed to find the points to fit to. To do this, an edge detector is applied to find the possible model point candidates. With this points, the model is fitted searching for the gradient change through a perpendicular line in the model point (see Illustration 21).

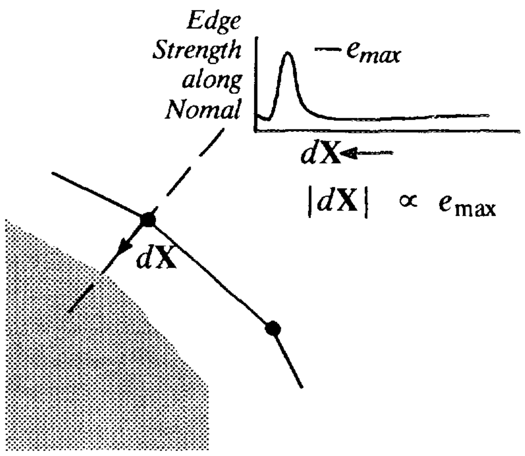


Illustration 21: Model point fitting for ASM

The process consist in iterate between all possible variations of the model, to find the best candidate. This process can be viewed in the next Illustration 22:

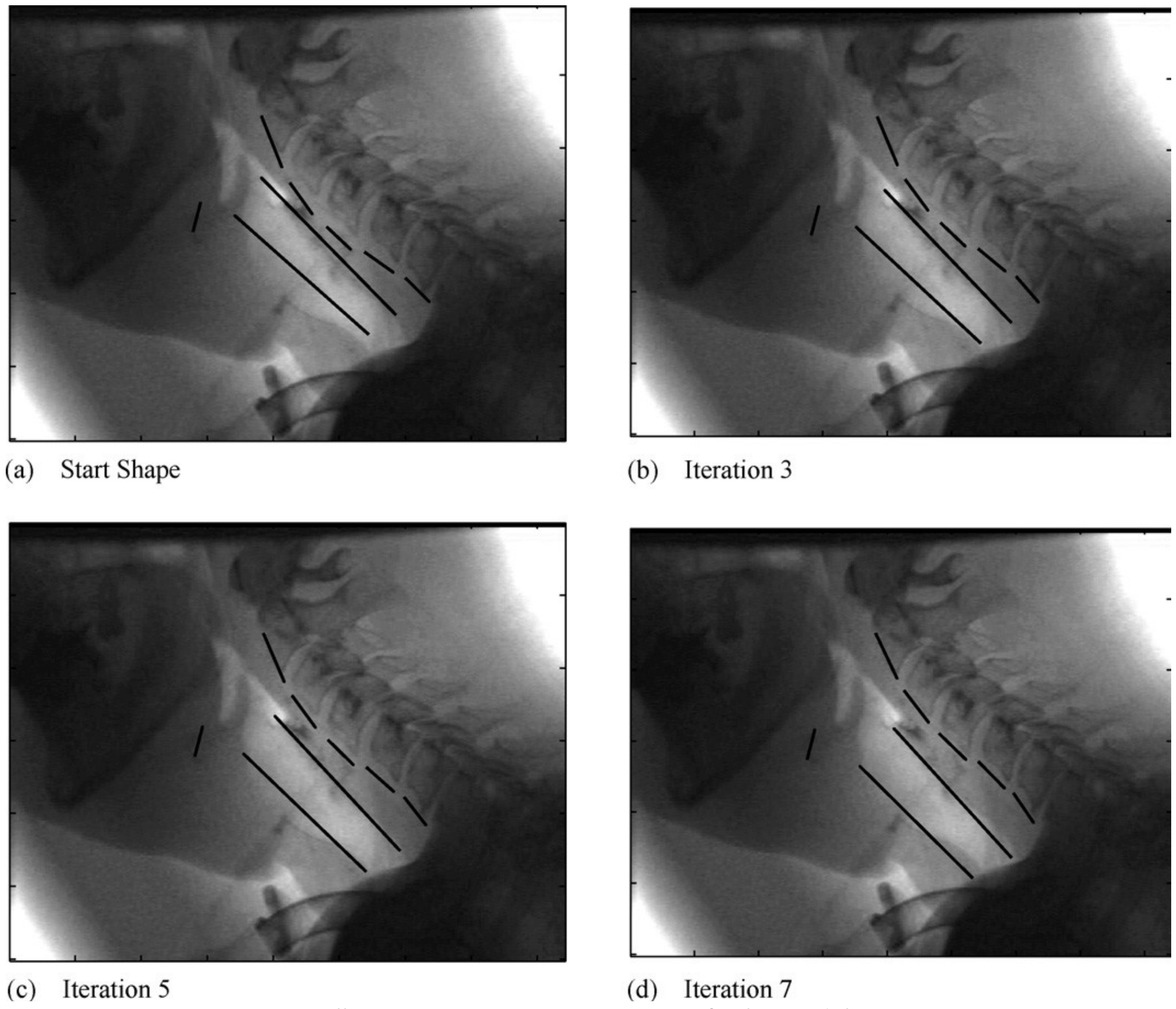


Illustration 22: ASM iterations to fit the model

4.1.2 DRLSE

Distance Regularized Level Set Function [4] is a technique that tries to segmentate objects in an image. The method consists on defining a function that starts growing or decreasing, updating its shape, with the goal of fitting to an object contour. See Illustration 23.

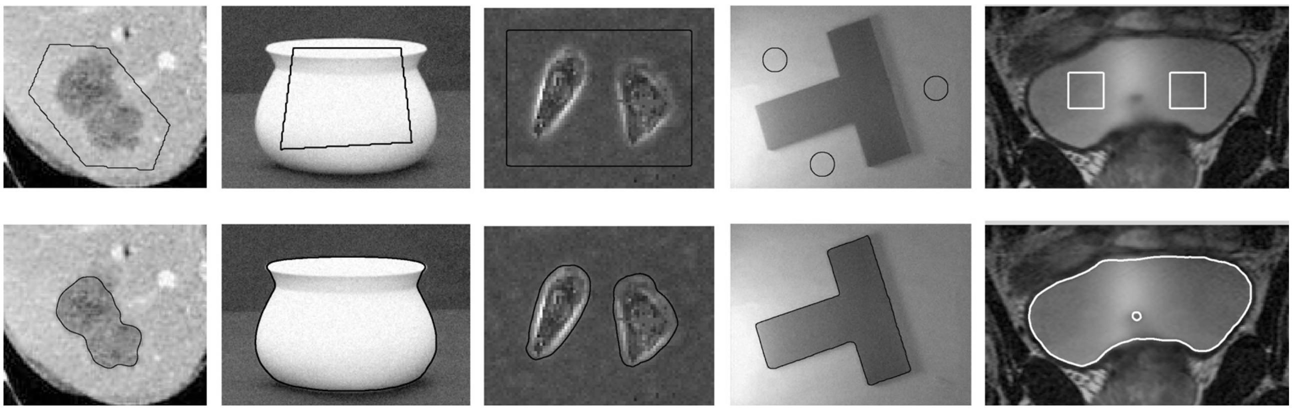


Illustration 23: DRLSE experiment images obtained from [4]

4.1.3 CAMShift

The CAMShift algorithm [5] is designed to follow color patterns. The goal of this method is to define a search window over the initial object, computing its color distribution, and in the next frames, the method tries to minimize the error between new color distributions and the initial pattern distribution. The process gives the possibility of track the object using translation and rotation.

4.1.4 HOG

The Histogram of Oriented Gradients is a widely used technique in object tracking, for instance, in pedestrian detection [6]. HOG consists on training a classifier using the gradients of an image, to detect new images in the future. First, a set of images are used to train the classifier. The object is selected, and divided in “bins” (a set of cells). For each bin, a histogram of oriented gradients is computed, this is, the x and y gradients are computed in the bin, and a histogram is made using the direction and intensity of each gradient computed. The result is a histogram that says, in general, which is the distribution of gradient directions in a bin. When the classifier is trained, the process consist on compare new images, searching for these HOG distributions, minimizing the error. As it can be seen in Illustration 24, each gradient is computed in the pedestrian image.

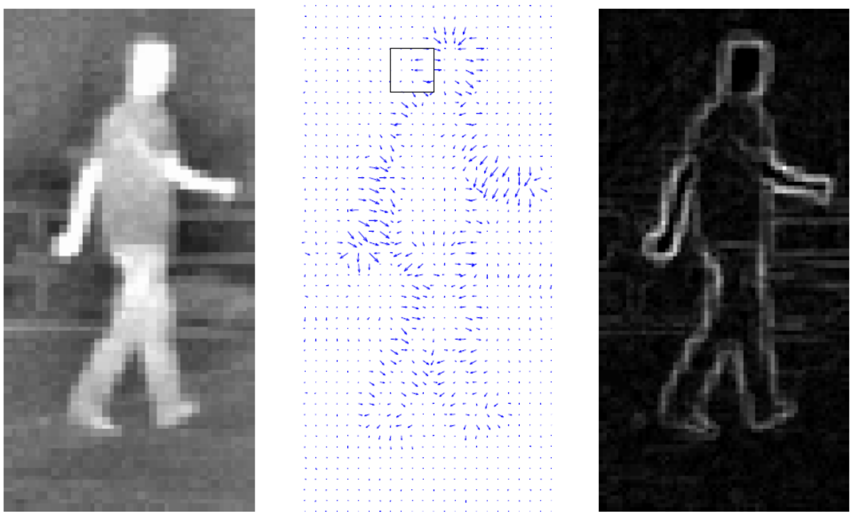


Illustration 24: HOG computation obtained from [6]

4.1.5 Temporal Filter

Another idea to detect the hyoid object in videofluoroscopy videos, is to use the movement as a tool to obtain the tracking of the object. The main idea is to subtract groups of frames to detect changes in the image. If one pixel is equal in the next frame, the image doesn't change, and it marks the pixel as a zero value (for instance, black pixel), but if the pixel changes its intensity, the absolute difference is represented with a value, represented in the grayscale, and detect if an object is moving in the image.

4.2 The real problem: Source videos

4.2.1 Format

"Hospital de Mataró" provide 4 videos of real patients to use in this work. Obviously, the number of videos is limited, due to the variability in them. A frame of this videos is shown in Illustration 25:

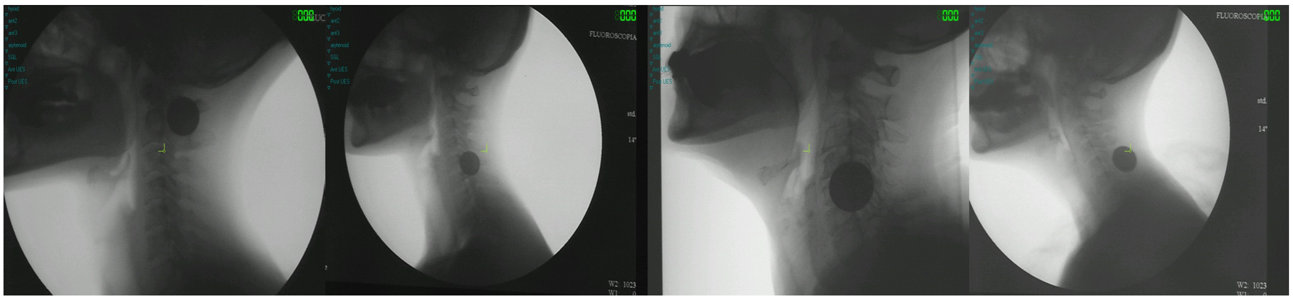


Illustration 25: Source Videos

Videos are not recorded directly from the X-ray machine, because it doesn't has digital output, only monitor output. This means that the only way that physicians can record videofluoroscopy, is using a camera that records the video from the monitor. This is a rudimentary procedure, but they haven't any other way to do it.

The good points are that the camera records in a sufficient good quality (768x576), and it has anti-flickering (it means, the camera does not record the flicker effect while recording to a monitor). But also bad points exists, the first, the position of the camera. Each time a video is recorded in a different session of videofluoroscopy procedure, the camera is readjusted to the monitor, so different sizes and zooms in the video are recorded. But the most problematic thing is the noise created by the camera while recording the video.

4.2.2 Test

To explore the possibilities of the video, different tests are tried. The most important element that has to be detected is the hyoid bone, so tests are applied to the entire image and to the hyoid bone concretely. Tests are made using the video 1 from resources.

The main point that its important to try is to segmentate the image. First, a Sobel operator is applied trying to obtain the contours of the image. After trying with different thresholds to define contours in a binarized image, the best result obtained is shown in Illustration 26:

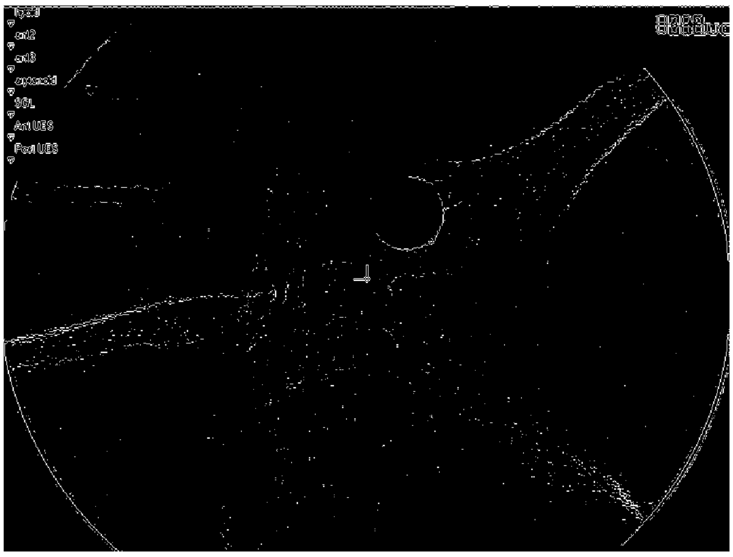


Illustration 26: Sobel test

As it can be seen, the result is very bad, and this is caused by the noise in the video (lot of disconnected segments, and contours poorly defined). Due to this, the next test is to try with canny edge detector method, that applies first the Gaussian filter that can provide better results to edge detection. The best try obtained is shown in Illustration 27:

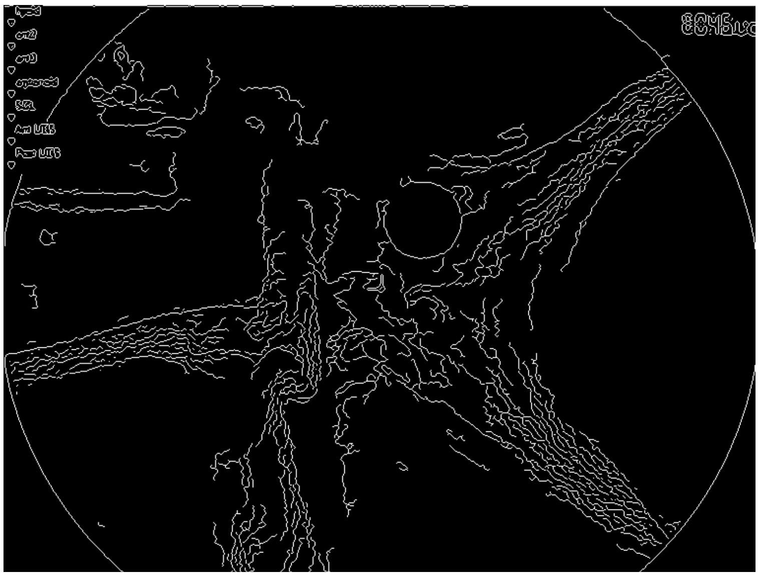


Illustration 27: Canny test

Now the results are much better. The Gaussian filter make its effect, and contours are more detailed and defined. But the problem continues, there are too much noise to get better defined contours, and its not only a problem of the noise, but also the contrast and definition of the objects in the image.

To see the real situation, the hyoid is analyzed in detail as shown in Illustration 28. In the left image, the hyoid is presented. As it can be seen, the contrast is bad, and details are difused. When the gradient is extracted on that image, the right image shows the result, almost intractable.

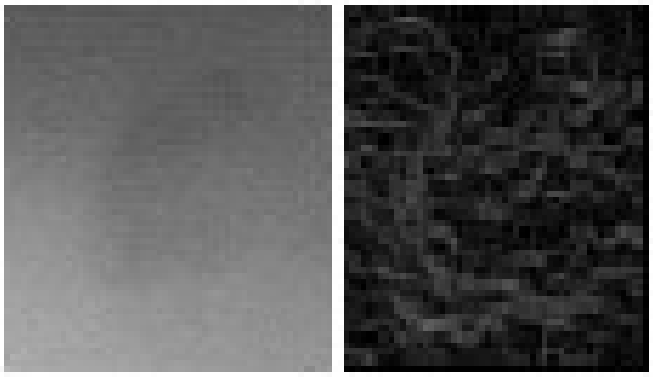


Illustration 28: Hyoid image and gradient extraction

4.3 Discussion over researched techniques

This section wants to demonstrate why none of these techniques work with the proposed problem, and why this argument its totally related with the previous section (quality of source videos).

As described before, the principal problem of this videos, due to the noise and the poor definition and contrasts in the image, is the capacity of extracting contours in the image.

ASM is the first technique proposed, but needs an accurate detection of the contours. This is because it needs to fit the model to the edges found in the image, so applying ASM in an image like Illustration 27, it is quite impossible, because it detects lot of disconnected edges, that can bring to multiple variations to fit the pattern model. This technique is implemented in MATLAB (see resource [R1]). The video shows a result using training data as test data too (only 1 video used), obviously it adjust quite well (unless the hyoid position), but the problem is that in the other videos, the same model doesn't work. Obviously, it is not only a problem of contours: With the four videos provided, a poor model can be computed, and its impossible that fits correctly in all the videos. To solve that, not only contours have to be improved, but also more videos are needed to compute the pattern model, for instance, hundreds of videos. This is why, in this situation, this method is discarded.

DRLSE and HOG are discarded due to the same problem, the contour definition. In DRLSE, the function cannot fit correctly to the shape, and in HOG, the gradients found are so sparse, and creates lot of noise in the histogram. Also, DRLSE is discarded by its computation cost.

To improve contours (that is the main problem), other techniques are tried, like Hough transform [7]. Hough transform tries to reconstruct the contours using detected edges, and tracing a line that defines the contour. But also, contours are poorly defined, and lines traced can be distorted a lot. Also other references are studied like Chord-to-Point Distance [9] and a corner detection techniques recopilation [10].

CAMShift doesn't need contours, but it needs color distribution. The main problem of this method in this case is that videos contain lot of noise, and the color distribution in the search window changes in every frame considerably.

Unfortunately, all these methods have similar problems, and fail due to the quality of the videos. So the aim now is to find a method that doesn't need contours at all, and also that be sufficiently robust to avoid noise in the image.

5 Selected method : Hyoid Marker

In this section, the technique used to process the image is explained in detail, and finally the results obtained are discussed.

Hyoid Marker is developed using MATLAB R2010a software.

5.1 Application explanation

As exposed in Chapter 4, there are lot of inconveniences to use more standard techniques to solve this problem, basically with the contour extraction techniques, which are normally the main option in case of segmentation.

In 5.1.1, MiFOD technique will be explained in detail. After this, in 5.1.2 it will be described the complete software application developed (Hyoid Marker), that uses MiFOD, but also uses more techniques and restrictions to achieve with the behavior of the hyoid bone detection problem.

5.1.1 MiFOD

MiFOD (Minimum of Function for Object Detection) is a technique developed by the author of this thesis, and is designed to provide object detection and tracking in a grayscale video, using only gray intensity references in the image, to avoid contour extraction techniques, that can fail (as in this case) with noisy or bad quality images.

In Illustration 29 a diagram is shown describing this technique:

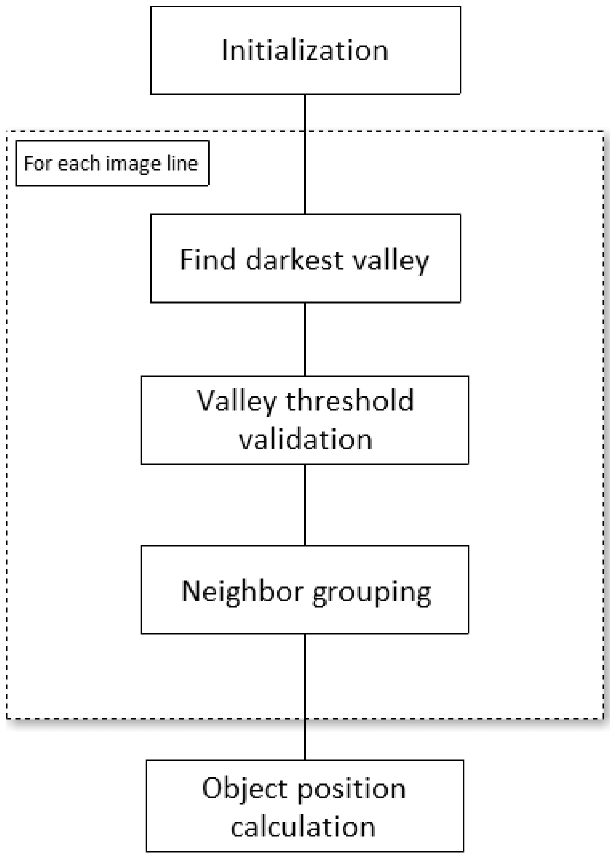


Illustration 29: MiFOD algorithm

The goal of MiFOD is always find the center of the object, and follow it during each frame of the video. To do this, the algorithm starts scanning the object line by line, and obtain the mean point of the object in this line. When all lines are processed, a type of skeleton of the object is obtained, and this will be used to know the center of the object.

Initialization

MiFOD needs at the beginning to select the position of the object to track. With the initial position, it will start to search with a defined window, as shown in the next Illustration 30:

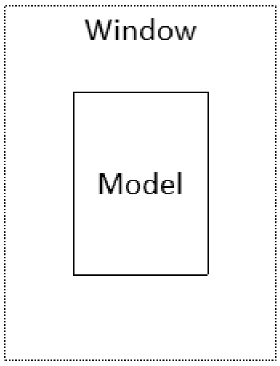


Illustration 30: Model for MiFOD algorithm

The aim of this initialization is to know how is the initial model that it will be tracked during the video, and enlarge the search space doubling the size of this model, to achieve the object movement.

The process starts scanning the selected image in the first frame, only in the Window search space, like in Illustration 31 (where the hyoid is represented), and for each line in the scanning process, a point is computed.

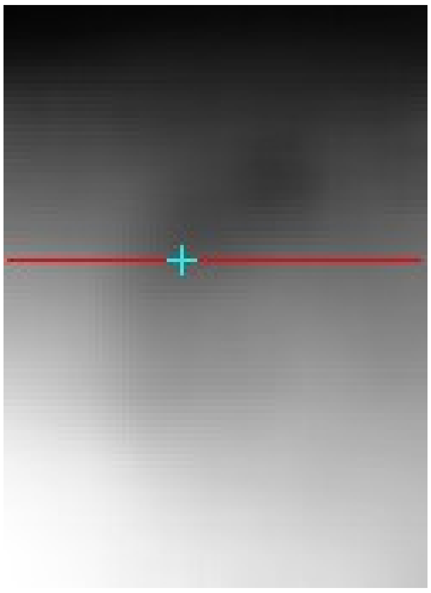


Illustration 31: Object Scanning

Find darkest valley

The algorithm take the current line image as a function, as it can be seen in Illustration 32:

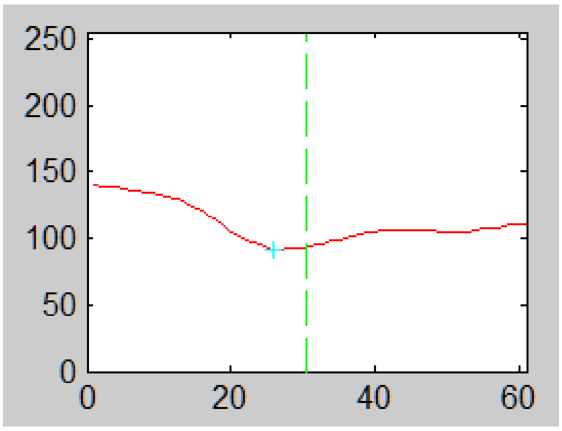


Illustration 32: Line intensity function

The function represents the intensity of each pixel in this line (x-axis is the position of the pixel, and y-axis the intensity through 0 to 255). So now the goal of this part of the algorithm is to search the minimum valley in the line, which represents the darkest pixel (and also, an object representation in this case). MiFOD supposes that an object is represented by a darkest or lightest point (it can be the lightest only inverting the algorithm behavior).

To obtain the valley of the function, we apply the second derivative to find (in this case) positive values (that represents a valley). But more than one valley can be found, so the algorithm takes only the darkest valley (later, this point will be verified, because obviously, it maybe cannot be the object to find).

Valley threshold validation

The algorithm applies three thresholds to the valley. If it passes them, is considered as a valid object point. Those thresholds are user-defined thresholds, like in other techniques, which have to be calculated depending of the goal of the problem. The curve with two thresholds is represented in Illustration 33:

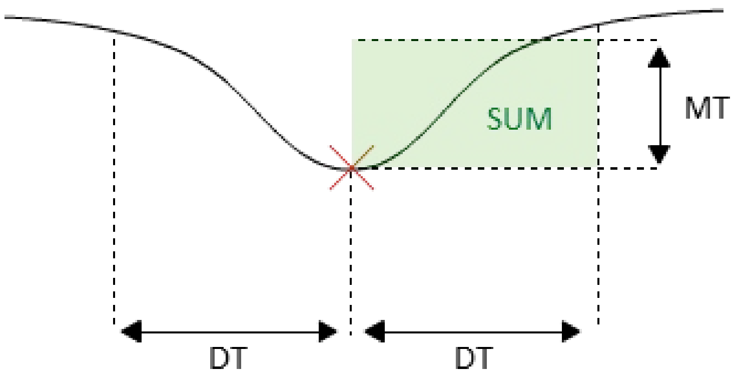


Illustration 33: Valley Threshold

DT is the "Distance Threshold" that defines the amplitude threshold of the valley. The MT is the

"Minimum Threshold" that is the minimum altitude that the curve has to achieve to be a valid point of the object. So, the following formula is calculated:

$$\mathrm{SUM} = \sum_{i=valleycenter}^{DT} pixel_{(i+1)} - pixel_i$$

This is the sum pixel by pixel of the first derivative of the function. The threshold MT has to be achieved in the two sides, so:

$$\mathrm{SUM}_l < -MT \wedge \mathrm{SUM}_r > MT$$

The third threshold is about the color of the point. If there exists a valley that its a minimum, but doesn't coincide with the real color of the object that wants to be detected, the threshold is not passed. This is to avoid confusion about other objects in the search space. Maybe another object has the same curve configuration, but it has not the same intensity.

To apply this threshold, in the first frame (where the algorithm supposes that the object is correctly selected), the color is calculated performing a mean of a window of 10x10 neighbor pixels.

Neighbor grouping

When a point in a line is selected as a candidate to represent the object, it needs to be grouped and compared with the other points in other lines. Imagine a normal situation like in Illustration 34 left figure. As said before, other objects can be detected too (unless the object is alone in all the scene), so another object can be similar in valley form and color, and create noise to the detection of the object of interest. To avoid this, a grouping between points is created using a distance threshold.

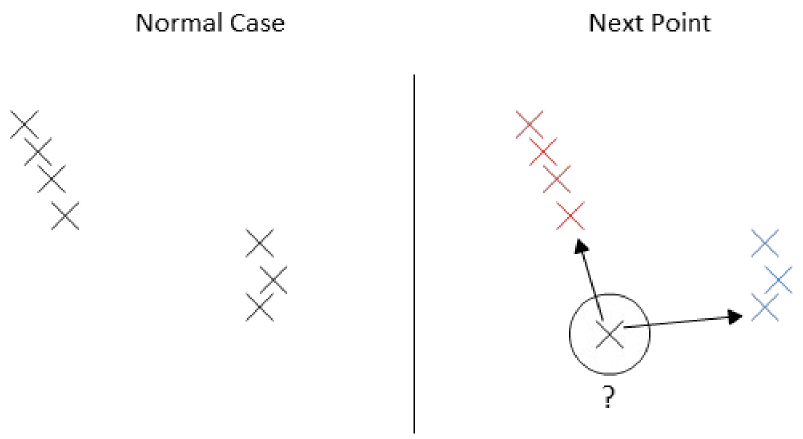


Illustration 34: Normal Situation of point neighborhood

In Illustration 34 right figure, a next point is found. Suppose that red points are the object to find, and the blue ones are noise of another object. The next point can be another object (a point of blue object), or a point of red group (the main object). To decide this, the distance neighbor threshold (NT) is applied. Depending of the restriction of this threshold, the object structure can be more partitioned or not, as shown in Illustration 35. It must be taken into account that the distance is calculated between all the points found, but obviously, the latest point found has always more importance if its near the next point, so in this sense, the algorithm tries to maintain some continuity in the object structure.

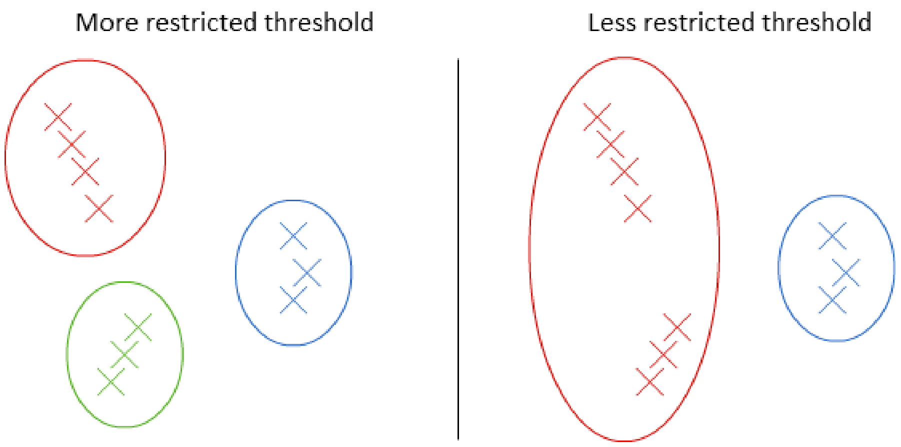


Illustration 35: Neihbor Threshold representation

When all points are found are grouped in all the search window space, cluster centroids are computed for each group, and only one of the centroids is selected, which is the closest centroid to the center of the image. Here the algorithm is supposing that the movement is relatively slow (that is not the case of the hyoid, but it will be seen later that the algorithm uses another techniques in the case of rapid displacement of the object), and this is because it tries to avoid another objects not of interest.

Object position calculation

With one cluster selected, the final structure is represented as in Illustration 36. Final step of the MiFOD algorithm is to move the search window and model of Illustration 29 to the center of the new centroid found, and this is how the tracking is performed.

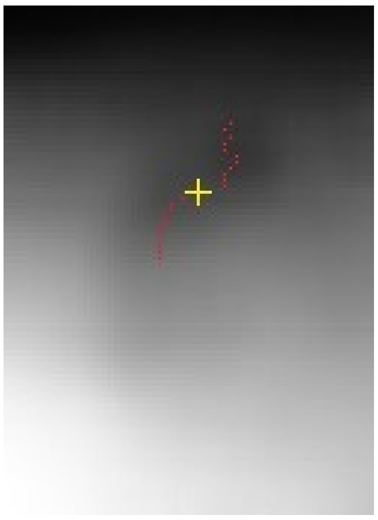


Illustration 36: Final Cluster selection

5.1.2 Hyoid Marker Application

The hyoid marker application is the goal of this work, a software that can mark the hyoid position through videofluoroscopy videos, and create a graph of the hyoid movement. In this section, the behavior of this application is shown, and how it uses the MiFOD algorithm to achieve with the proposed goal.

The complete diagram of the application flow can be seen in the next Illustration 37.

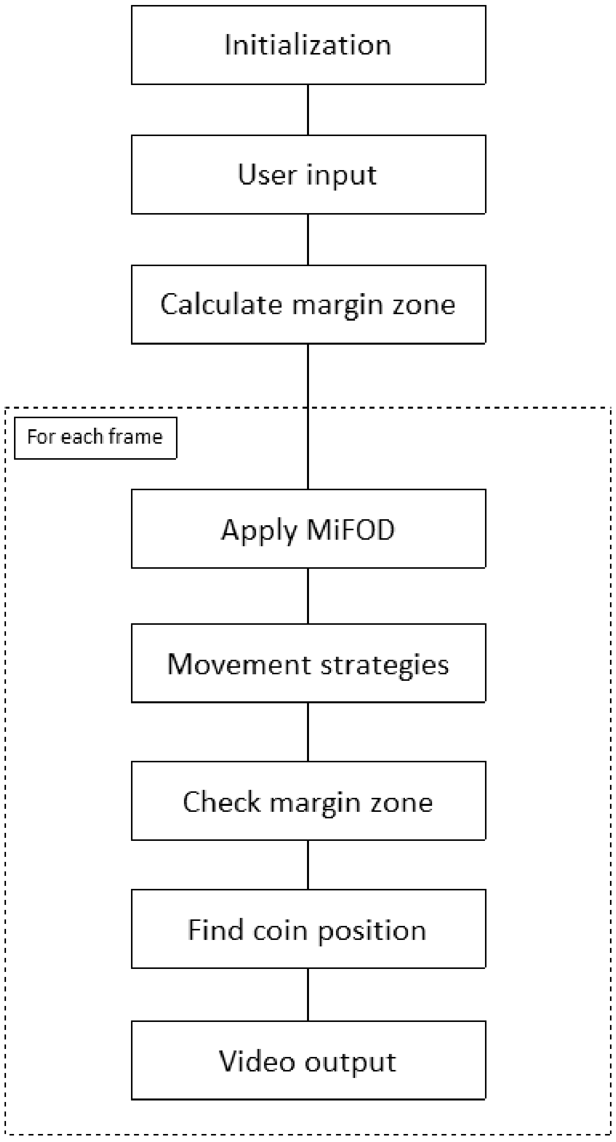


Illustration 37: Hyoid Marker Application

As a general description, Hyoid Marker starts marking in the first frame the hyoid position and a reference (to calculate the trajectory later), and it starts applying MiFOD to detect the Hyoid center. After this, movement strategies and margin limitations are applied to control the trajectory of the Hyoid, and finally, two outputs are generated, one video with the marked hyoid, and another video with the graphical trajectory.

Initialization

First, user selects a video to process, and debug mode can be activated or not (it shows more detailed video output information, like the scanning process of each image in MiFOD algorithm and the curve itself), and also all variables are initialized (algorithm initializations will be discussed in the next results section).

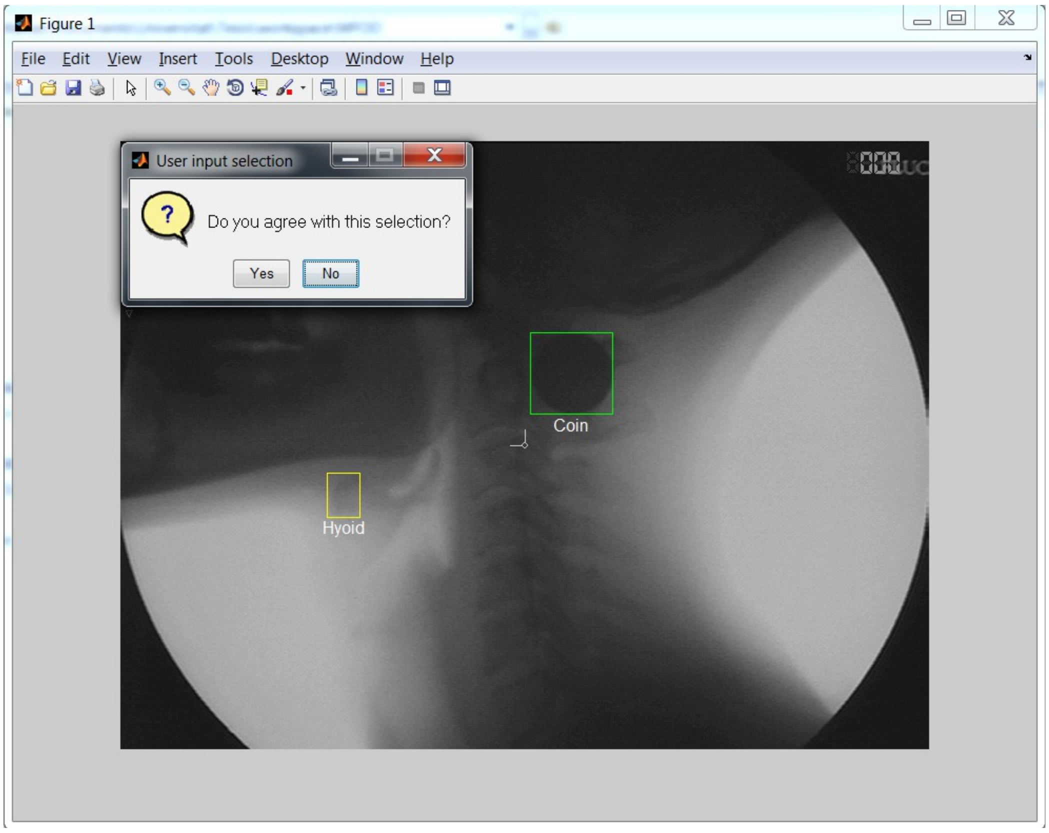
User input

The user is prompted to select, in the first frame, the Hyoid position and the reference position.

Hyoid position is difficult to be detected automatically, and physicians are better marking it, so is more accurate if the physician mark the hyoid in the first frame.

Also, a reference is needed to calculate the real trajectory of the hyoid in a 2D axis. Real means relative to a static position, because in videofluoroscopy videos, the patient can move the head through sagittal plane, and if there are no reference to take into account, the graphic of the trajectory will be incorrect. The selected reference is a coin of 1€ placed by the physician, in the neck of the patient (the black circle that appears in all the videos). This is taken as a reference because of, first, it is a known measure reference (1€ diameter = 23.25 mm), and second, its the easiest shape in the video to identify in all frames (other alternatives are, for instance, the vertebrae, but there are more difficult to detect and follow).

If the selection is correct, user proceeds to accept the selection (Illustration 38), and the video starts to be processed.



Do you agree with this selection?

Illustration 38: User input selection

Calculate margin zone

A margin or security zone is needed if the algorithm fails, and starts searching too far of the hyoid position (normally, when the search window loses the hyoid shape). In this case, the search distance must be limited in a restricted search zone.

Based on the known assumption that hyoid don't move further than mandible horizontal size, the margin is calculated about this area. Mandible can be segmented reducing the number of colors of the image. Because image is always captured with same intensity, the color of the mandible zone its always the same, so its detected and selected, as it can be seen in Illustration 39 left.

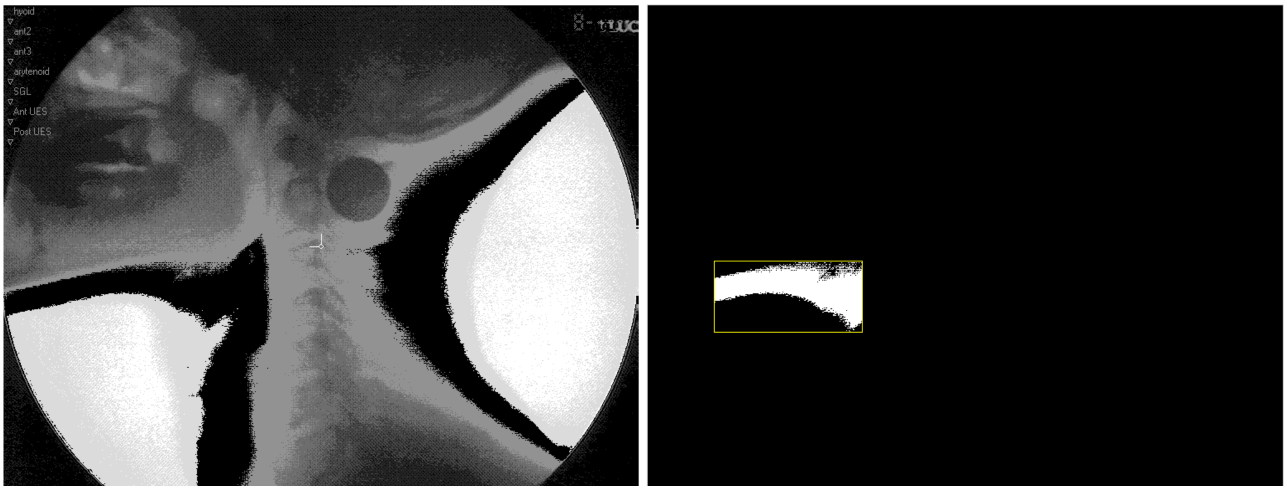


Illustration 39: Margin zone calculation

Later, using the hyoid size, a maximum distance is estimated and the area is fitted using these measures (doubling vertical size of the hyoid and quadrupling the hyoid horizontal size), as shown in Illustration 39 right. During the search of the hyoid, this search will be limited by the area of this zone.

Now, the software knows all the information about the image. The next steps are applied for each frame in the video, that will be processed as follows.

Apply MiFOD

As explained before, the goal of MiFOD algorithm is to find the center of the desired object. In this case, the hyoid is the object to track. See previous section to detailed description of MiFOD algorithm.

Movement strategies

As MiFOD description says, it can be possible that the detected center isn't the correct center of the hyoid (maybe the algorithm can detect more points of another object near to the hyoid, or maybe the algorithm doesn't detect nothing, if the curve derivative doesn't reach the defined thresholds). This can be a catastrophic moment in the marking process, because the search window can move dramatically to another point, and it can be start to search in another direction, losing all references of the tracked object (in this case, the hyoid). To solve this weakness of the

MiFOD algorithm (due to its a local type of search), the Hyoid Marker takes into account the movement of the object from a global point of view using two strategies: Left-search and histogram calculation.

<u>Left-search</u>

Left-search uses the assumption that the hyoid is moving in the left direction, due to a swallowing made by the patient. Normally, due to some rapid movements of the hyoid in the videos in this swallowing step, the hyoid is blurred on the image and its reference can be lost. When this happens, the MiFOD algorithm detects nothing, and in this case, the left-search strategy proposes to start moving to the left, to find again the hyoid. Obviously, maybe is only a low threshold of the MiFOD algorithm, or the image is blurred only during 2 or 3 frames, and the hyoid is not moving at all, so to avoid this situation, the left-search strategy movement is applied only during a limited number of frames. If it passes this number of frames, the position of the hyoid is reset to the latest point, when the strategy is started to apply. But if it works, and left-strategy finds another point similar to the hyoid (MiFOD uses also color comparison), the MiFOD algorithm is applied again in the following frames.

<u>Histogram calculation</u>

If left-search fails, the search position of the hyoid is restored to the latest point, where the left-search strategy is started to apply, and the strategy is switched to histogram calculation strategy. The histogram calculation tries to find the hyoid using its histogram representation. The method its the same idea of HOG (Histogram of Oriented Gradients) but using only the histogram in each cell, not the gradients (due to the discussion of quality of gradients made in the previous chapter).

In the first frame, when the hyoid is selected, the histogram of the object is calculated as a “base histogram”, like Illustration 40:

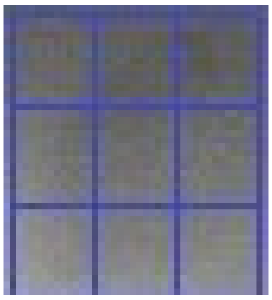


Illustration 40: Histogram cells of the Hyoid

The hyoid is divided in 9 cells (it can be variable, but a 3x3 matrix is a good selection to obtain the better quality/speed relation). For each cell, a histogram is computed, to keep a color distribution representation of each cell in the object.

With the base histogram calculated, when histogram calculation strategy is applied, a histogram is computed for each possible position of the hyoid in the image (with a limitation of the search space). The search window is moved around some pixels in all directions, and tries to find the minimum difference between histograms (between the current one, and the base histogram).

When the histogram calculation is applied, a new center for the object is found, and later MiFOD is applied. If MiFOD finds something again, the algorithm continues normally. If not, the algorithm continues with the histogram calculation strategy until it finds something.

A summarized diagram of the movement strategies applied is shown in the next Illustration 41:

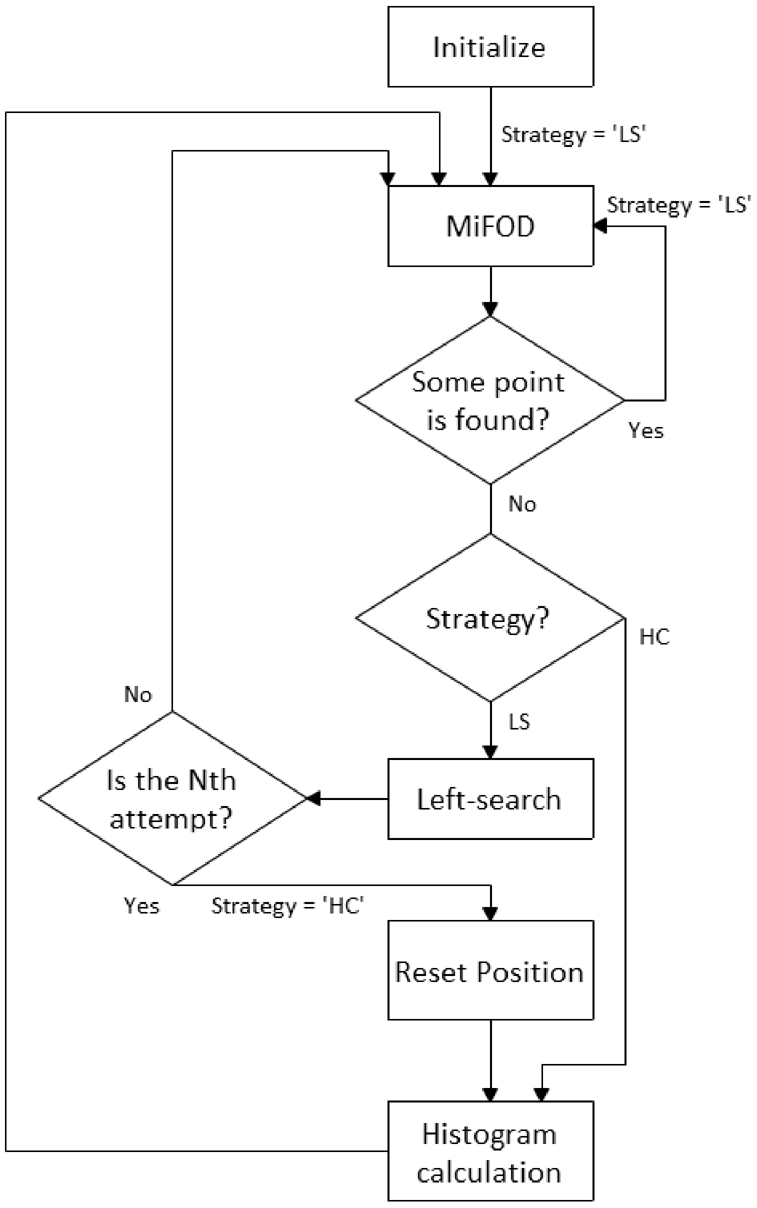


Illustration 41: Movement strategies algorithm

Find coin position

As explained before, a reference is needed to compute later the plot of the hyoid trajectory. To obtain this reference, the coin is detected in each frame to calculate the distance between coin and hyoid, and keep each position to make the plot at the end.

The most easy way to detect the coin is to get the perimeter using an edge detector, and later, fill the region and get the area to calculate position and dimensions. The problem is that the region obtained, has a lot of holes or the perimeter is broken, so its more difficult to fill the figure, to get this property. For this reason, the next fast method is proposed.

To find the coin, first a canny edge detection is applied, using threshold=0.1 for binarization, and sigma=2 for the Gaussian filter. Canny edge is better here because tries to recompose the edges in the coin circle, obtaining a better defined shape. Coin is selected in the first frame, obtaining an approximate size manually. The algorithm starts searching from the center of this selection in each frame, to the borders of the coin (obtained with Canny) horizontally to the right. If the search obtains an edge point, the algorithm stops, and subtracts the coin width to find the other side. If not, it starts searching in the other way (left), and if it find an edge, it adds the width to find the right side. This is made also with the vertical direction. If the search doesn't find any edge, the algorithm uses the latest position found. The representation of this algorithm can be seen in Illustration 42 with the two probable cases:

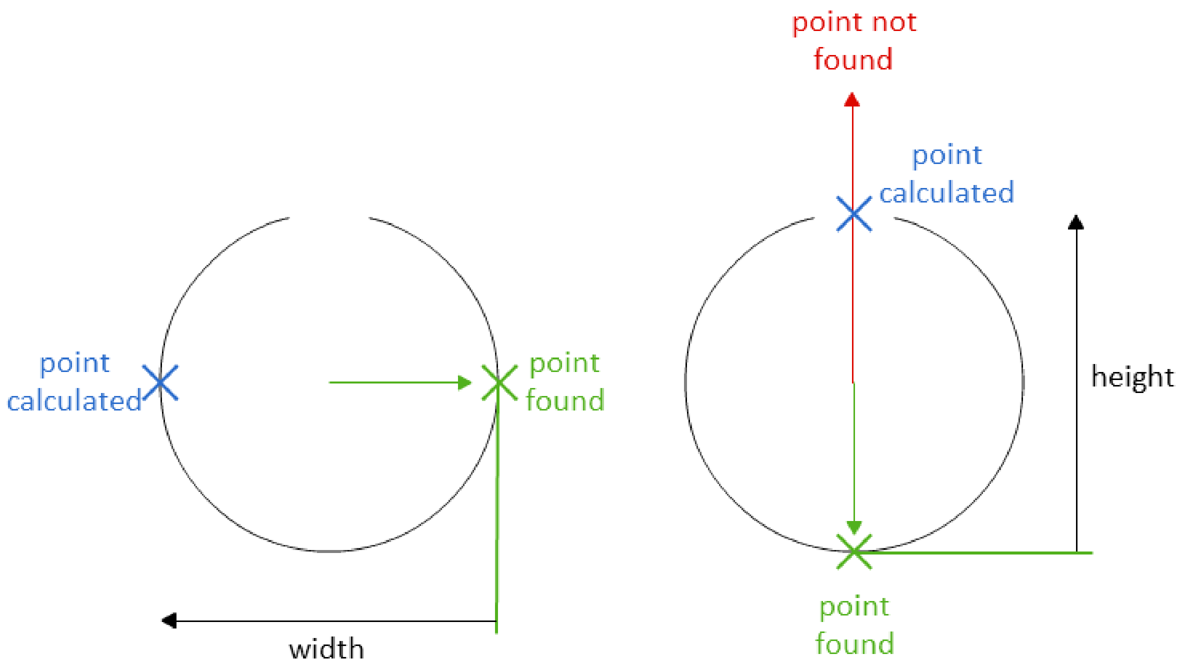


Illustration 42: Coin detection algorithm

In the left figure, the algorithm starts searching to the right and found an edge. Because it found a point, it subtract the width an obtain the calculated point in the left (using the width marked in the first frame by the user). In the right figure, the algorithm starts searching top, but it doesn't found any point (because a hole exists in the perimeter), so it stops when reaches the height coin distance. So it starts searching to the bottom, it find the edge, and it calculates the top point with the height distance. The algorithm is applied in each frame, updating the position of the coin.

Video output

Finally, an output is required by the physician, provided with 2 videos.

The first video contains the tracking itself, with the mark of the hyoid in each frame following its position, and a mark in the coin center, to see the reference position. An image of this video is provided in Illustration 43.

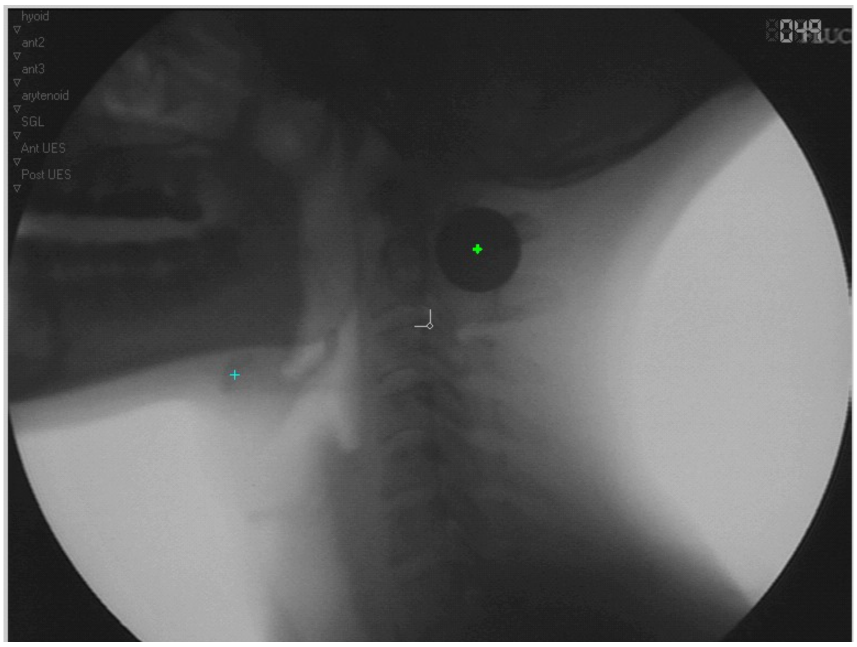


Illustration 43: Marks video output

Also, the plot of the hyoid trajectory is shown in a plot, as it can be seen in Illustration 44. The plot is represented in millimeters distance, using the size of the coin as a real measure reference.

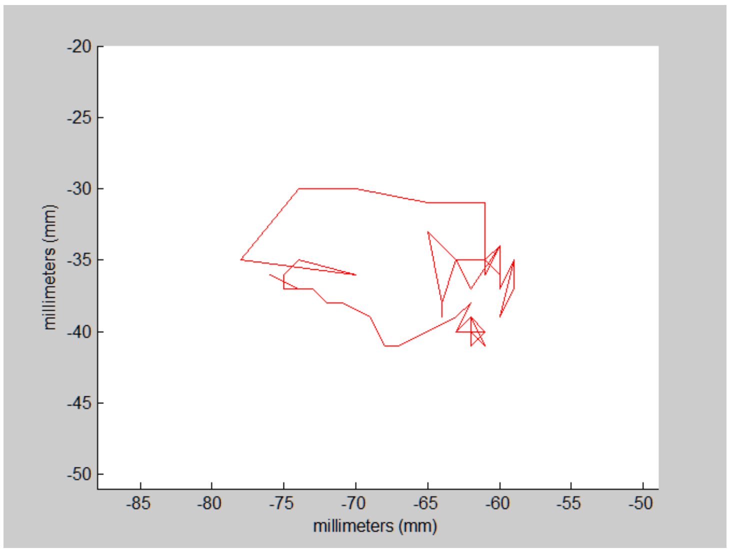


Illustration 44: Plot video output

5.2 Results

Because of the manual selection in the first video frame, multiple tests have to be made for each video, to see the variability that the algorithm can create in the same video. For each of the 4 videos provided by the hospital, 3 tests are computed containing the mark and plot videos.

The videos can be seen in resource folder [R2], with videos numbered with the video number (1, 2, 3 and 4), and a letter for each test ('a', 'b' and 'c'). Test 'a' and 'b' are made with movement strategy debug (it means, the video shows the repetition in movement strategies), and test 'c' is a final video result (without additional information).

5.2.1 Parameters initialization

In this section, the initialization of the parameters to do the tests of the algorithm is discussed.

```
MT = 7;
```

MT is the threshold associated to the color line curve in MiFOD algorithm. The value 7 is calculated doing tests with the debug mode, watching how the curve is detected correctly with different threshold values. Observing to the color difference between hyoid shape and background, the chosen value is the better election to detect correctly the hyoid.

```
DT = HM(3)/2;
```

The Distance Threshold is for the curve distance in MiFOD to sum the derivative in each side. Hyoid size is used to calculate the threshold. Width is the best measure to define this distance, because is not possible that the distance passes this value.

```
NT = HM(3)/2;
```

In the MiFOD algorithm, a threshold is needed to check the distance between each line point in the scan process, to know which points pertain to the same group or not. The hyoid size is used to calculate this measure, because its the minimum distance to take into account to separate hyoid from other objects.

```
CT = 50;
```

Color Threshold is a parameter that detects if the selected object is sufficiently similar to the hyoid color selected in the first frame. If the color of the current object is less than this threshold, the object is selected as the correct one. The threshold 50 is calculated observing the differences of color between hyoid and other objects in the image (mandible, vertebrae, background...). 50 is a good threshold that works good in all cases.

```
WaitRetries = 5;
```

This is the number of retries that left-search movement strategy is used before changing to next strategy. Making experiments with the videos, the hyoid is detected usually in 3 frames, so 5 retries are sufficient.

```
nhcells = 3;
```

For histogram calculation, the number of bins has to be defined. Experiments are made with different sizes, and works bad with lower values than 3, but greater values doesn't improve the results, due to the small size of the hyoid, dividing in more bins only increases the dispersion of the information, and the cost of the computation. So a 3x3 matrix is the better structure to detect the histograms.

```
nhist = 32;
```

Is the number of divisions in each bin histogram in histogram calculation. The image has 64 different colours. The values of nhist are combined with nhcells, to obtain the better combination of the two variables. 64 divisions in the histogram doesn't improve results and only increases computation time, but obviously, less than 32 colours is impossible to use to compare histograms. So nhist is selected with 32 as the better value.

```
disthist = 10;
```

Distance to search with histogram calculation to all directions in pixels. The value 10 is calculated observing the movement of the hyoid and seeing how many pixels it moves in each frame.

5.2.2 Video results

Video results can be viewed in the folder [R2], as commented before. So in this section the algorithm results, using its precision and behavior in each video, are analyzed:

Video 1

The first video obtains one of the better results using this algorithm. As it can be seen, only left-search strategy is used, showing the principal goal of this technique, that is search to the left when the patient does the swallowing movement. In the three tests this strategy is applied, obtaining similar results.

Video 2

This is the worst case, because the definition of the hyoid is so bad. As shown in the three videos, left-search technique is applied because the hyoid contrast is very smooth, and the algorithm doesn't detect anything. When left-search strategy fails (because it goes far away to the left), it always returns to the initial position, and try to apply histogram calculation. In the next two frames, it founds the object, but it loses it soon, repeating the process, until hyoid returns to the initial position, being more contrasted, and detected easily. Anyway, the algorithm follows the object fairly well.

Video 3

This video has two swallowing movements, the second more abrupt than the first one. In this tests only left-search strategy is used too, like in video 1, but something different happens. It stays immobile in some cases detecting the background as the same object, during a considerable number of frames, until the hyoid comes back to initial position. This happens more clearly in tests 'a' and 'c', in the second swallowing movement. This occurs because of the rapid movement of the hyoid , that moves more distance than the search window can see in the next frame, but it detects something that passes the threshold validation in MiFOD algorithm, and it stays there until the hyoid comes back and enter to the search window again.

Video 4

The 4th video only uses MiFOD technique, because hyoid moves slowly (patient doesn't swallow correctly), but also the hyoid doesn't have so much contrast, so it provokes more noise with the background, and sometimes the hyoid center is not detected as well as the other videos.

5.2.3 Computation time

To know the computation time of the algorithm, the time of execution in each video is calculated,

Computation time (seconds)	Test 'a'	Test 'b'	Test 'c'	Average time / Nº video frames
Video 1	42.73	37.03	33.77	37.84 / 90 = **0.42 s/frame**
Video 2	41.74	44.71	36.11	40.85 / 57 = **0.71 s/frame**
Video 3	50.14	48.83	45.90	43.72 / 120 = **0.36 s/frame**
Video 4	42.67	43.91	37.13	41.23 / 105 = **0.39 s/frame**

The second video is the shortest video in frames, but the algorithm has to change between movement strategies, and it wastes more computation time than the other videos, which have a mean of 0.39 s/frame. With this, the computation time of the algorithm can be described as:

- No movement strategy used → 0.39 s/frame
- Movement strategies used → 0.71 s/frame

6 Conclusions and future extensions

The application Hyoid Marker has been developed, and the videos provided by the hospital have been analyzed with good results, so the main goal of the work, which is to provide an application to aid the physicians to mark the videofluoroscopy videos, is acquired.

Another important point is the short time spent by the algorithm analyzing the videos, because the goal is not only to make an application, but an algorithm sufficiently fast to save time to the physicians that analyze the videos, and this is also accomplished, because if the same calculation made in Chapter 3, now is made with the Hyoid Marker times, the time consumed for the 45 videos is now 30 minutes, not 11 hours as before. So the time spent by the physician with this application is very small, compared with the time spent doing it manually.

The algorithm is developed purposely to this concrete problem, and this is an advantage in the sense that it can be adjusted easier and better to mark new videos, that with a more generic algorithm maybe could be more difficult.

Finally, some future extensions for this work will be listed. Not only to the algorithm proposed, but also functionalities of the application that can help more to the physicians:

- The algorithm can be improved to be more robust, combining more techniques to improve for instance the tracking method, or the hyoid detection. Maybe some machine learning techniques could be applied, to define hyoid models using this algorithm.
- Other objects in the image could be detected, to obtain a reference point to calculate the hyoid path, more concretely, the vertebrae. This point is left due to its difficulty to find them in the videos provided, but could be an interesting work to extend.
- Other processes can be included in the software, to obtain a more complete application. The detection of the swallowing phases (explained in Illustration 5) can be marked also automatically, using temporal filters and color comparison. Also the movement and tracking of the bolus, to detect if it goes in the correct direction, or provokes aspiration.
- One of the needs for a better research in these videofluoroscopy experiments, is to ask to the hospital a better quality of the videos, principally with the resolution and contrast parameters.

Finally, remark that it has been an intensive and productive work for the author, because it has allowed to him to explore new techniques and possibilities in computer vision area, mixing them with an important and interesting area nowadays, technology applied to the health system.

7 References

7.1 Papers

[1] Laia Rofes, Viridiana Arreola, Jordi Almirall, Mateu Cabré, Lluís Campins, Pilar García-Peris, Renée Speyer, Pere Clavé:

Diagnosis and Management of Oropharyngeal Dysphagia and Its Nutritional and Respiratory Complications in the Elderly

[2] *M.S.H. Aung,* J.Y. Goulermas, S. Stanschus, S. Hamdy, M. Power:

Automated anatomical demarcation using an active shape model for videofluoroscopy analysis in swallowing

[3] *T.F. Cootes, C.J. Taylor, D.H. Cooper, J. Graham*:

Active Shape Models – Their Training and Application

[4] Chunming Li, Chenyang Xu, Changfeng Gui, Martin D. Fox:

Distance Regularized Level Set Evolution and Its Application to Image Segmentation

[5] Gary R. Bradski:

Computer Vision Face Tracking For Use in a Perceptual User Interface

[6] *F. Suard, A. Rakotomamonjy, A. Bensrhair, A. Broggi*:

Pedestrian Detection using Infrared images and Histograms of Oriented Gradients

[7] D.H. Ballard

Generalizing the Hough Transform to Detect Arbitrary Shapes

[8] John Canny

A Computational Approach to Edge Detection

[9] *Mohammad* Awrangjeb

Robust Image Corner Detection based on the Chord-to-Point Distance Accumulation Technique

[10] *Giuseppe* Papari, Nicolai Petkov

Edge and line oriented contour detection: State of the art

7.2 Resources

[R1] ASM Algorithm implementation applied to videofluoroscopy

[R2] Hyoid Marker test video results

7.3 Bibliography

Alberto S. Aguado, Mark S. Nixon, M. Eugenia Montiel

Parameterizing Arbitrary Shapes via Fourier Descriptors for Evidence-Gathering Extraction

M.S.H. Aung, S. Hamdy, M. Power, S. Stanschus, J. Y. Goulermas

Measuring Bolus Transit Times from Videofluoroscopy using Image Profiles and Particle Swarm Optimization

M.S.H. Aung, J. Y. Goulermas, S. Hamdy, M. Power

Spatiotemporal Visualizations for the Measurement of Oropharyngeal Transit Time From Videofluoroscopy

E. Avila Romero, J. Quiroga, A. Forero

Seguimiento de Personas Basado en los Descriptores HOG

M. Ceccarelli, A. Colaprico, L. De Vito, S. Marotta, M. Piscitelli

A Semi-automatic Measurement System for the Swallowing Analysis in Videofluoroscopy

J. Driessen

Object Tracking in a Computer Vision based Autonomous See-and-Avoid System for Unmanned Aerial Vehicles

N. Funk

A Study of the Kalman Filter applied to Visual Tracking

S. Ghosh, Raja S. Alomari, V. Chaudhary, G. Dhillon

Automatic Lumbar Vertebra Segmentation from clinical CT for Wedge Compression Fracture Diagnosis

H. Tao

Object Tracking and Kalman Filtering

S. Huang, S. Lai, C. Novak

A Statistical Learning Approach to Vertebra Detection and Segmentation from Spinal MRI

J. Illingworth, J. Kittler

A Survey of the Hough Transform

L. Rofes, V. Arreola, M. Romea, E. Palomera, J. Almirall, M. Cabré, M. Serra-prat, P. Clavé

Pathopysiology of oropharyngeal dysphagea in the frail elderly

B. Sirmacek, P. Reinartz

Kalman Filter Based Feature Analysis for Tracking People from Airborne Images

C. Stauffer, W. Eric, L. Grimson

Learning patterns of activity using real-time tracking

G. Welch, G. Bishop

An Introduction to the Kalman Filter

S. Wong, K. Wong, W. Wong, C. Leong, D. Luk

Tracking Lumbar Vertebrae in Digital Videofluoroscopic Video Automatically

Y. Zheng, M. Nixon, R. Allen

Automated Segmentation of Lumbar Vertebrae in Digital Videofluoroscopic Images

8 Annex I : Source Code

8.1 Hyoid Marker

Here the source code of Hyoid Marker is shown:

```
%% Hyoid Marker
%  Master Thesis in Artificial Intelligence (UPC)
%  Automatic Demarcation for Videofluoroscopic Swallowing Study
%  Author: Matias Lizana
%  ------------------------------------------------------------
%  This program marks the hyoid bone of a videofluoroscopic swallowing
%  study, using MiFOD technique (Minimum of Function for Object Detection)
%  and histogram comparation as an auxiliar technique. The hyoid position
%  is calculated using the coin added to the patient as a reference.

try
%----------------------
% 1: INITIALIZATION --
%----------------------

%% 1.1 User input and open video stream
clear
video = input('What video do you want to process? ','s');
VIDEO_DEBUG = str2double(input('Do you want to debug the video? (1:yes/0:no) ',
's'));
STRAT_DEBUG = str2double(input('Do you want to debug the movement strategies?
(1:yes/0:no) ', 's'));
if(VIDEO_DEBUG~=0 && VIDEO_DEBUG~=1)
    VIDEO_DEBUG=0;
end
if(STRAT_DEBUG~=0 && STRAT_DEBUG~=1)
    STRAT_DEBUG=0;
end
readerobj = mmreader(['Videos/' video '.avi']);
vidFrames = read(readerobj);
numFrames = get(readerobj, 'numberOfFrames');
if(VIDEO_DEBUG==0)
    outputvideo = avifile(['Output/' video '-marks.avi'],'Compression','None');
else
    outputvideo = avifile(['Output/' video '-debug.avi'],'Compression','None');
end

%% 1.2 Initialize variables

% MiFOD parameters
MT = 7;                         % Minimum Threshold calculation (sum of
derivative)
CSD = 5;                        % Color Search Distance
CT = 50;                        % Color Threshold
h = fspecial('average', 10);    % This filter is used to difumine the image

% Hyoid structure
hyoid = struct;                 % Hyoid position x-y (relative to window)
hyoid.x = [];
hyoid.y = [];
hyoid.group = [];               % Clustering of different groups by relative
distance
hyoid.centroidsx = [];          % Centroids x-y of clustering groups
hyoid.centroidsy = [];
hyoid.numGroups = 0;              % Total number of groups
travel = [];                      % Points storing the hyoid travel
auxtravel = [];                   % Aux points stored for left-strategy

% Video, user input and plotting
initFrame = 1;                    % Frame to start computing the video
resp = 0;                         % Response to the user selection
plotaxis = [Inf -Inf Inf -Inf]; % Axis to plot the hyoid travel
ET = 0;

% Strategy
SearchStrategy = 0;               % 0: Left-search to follow hyoid
                                  % 1: Histogram comparation
                                  % When 0 is used and fail, rewind and apply
strategy 1
                                  % If 1 is used and works good, change to
strategy 0
FindWait = 0;                     % Wait to find a good result
WaitRetries = 5;                  % Retries to apply the same strategy
WaitFrame = 1;                    % The frame to keep and return if strategy fails
HMAux = [];                       % Hyoid model to keep and restore if strategy
fails
VideoAux = 0;                     % Says if its needed to keep the frames in a
buffer
VideoFrames = [];                 % Frame buffer to keep temporal frames

% Histogram computation
nhcells = 3;                      % Number of cells in the window
(nhcells*nhcells)
nhist = 32;                       % Number of levels in the histogram calculation
disthist = 10;                    % Maximum distance to search with the window

%% 1.3 First frame user input
I=vidFrames(:,:,:,initFrame);
I=rgb2gray(I);
fig = figure;

% Do while the user is not agree with the selection
while(resp==0)

    clf(fig);
    imshow(I);
    hold on;

    % Get hyoid position
    HM = getrect;
    rectangle('Position',HM,'EdgeColor','y');

text(double(HM(1)+HM(3)/2),double(HM(2)+HM(4)+10),'Hyoid','Color','w','Horizonta
lAlignment','center');

    DT = HM(3)/2;        % Calculate Distance Threshold (curvature)
    NT = HM(4)/2;        % Calculate Neighbour Threshold (distance to group hyoid
points)
    hyoid.color = mean(mean(I(uint32(HM(1)+HM(3)/2-
CSD):uint32(HM(1)+HM(3)/2+CSD),uint32(HM(2)+HM(4)/2-
CSD):uint32(HM(2)+HM(4)/2+CSD)))); % Hyoid color (using neighbour points)

    % Get coin position
    c = getrect;
    cc = [c(1)+c(3)/2 c(2)+c(4)/2 c(3) c(4)];
    rectangle('Position',[c(1) c(2) c(3) c(4)],'EdgeColor','g');

text(double(cc(1)),double(cc(2)+c(3)/2+10),'Coin','Color','w','HorizontalAlignme
nt','center');

    % Answer if the selection is correct
    choice = questdlg('Do you agree with this selection?', ...
    'User input selection', ...
    'Yes','No','No');
    switch choice
        case 'Yes'
            resp = 1;
        case 'No'
            resp = 0;
    end
    hold off;
end

close(fig);

%% 1.4 Calculate hyoid margin zone
% 182 is the color of histogram that pertains to the hyoid zone (all videos)
% Discretize in 8 level colors
M=histeq(I,8);
% Map to a binary image
M(M(:,:)~=182)=0;
M(M(:,:)==182)=1;
% Cut distant areas
M(:,uint32(HM(1)+HM(3)*2):size(M,2))=0;
M(:,1:uint32(HM(1)-HM(3)*4))=0;
M(uint32(HM(2)+HM(4)*2):size(M,1),:)=0;
M(1:uint32(HM(2)-HM(4)*2),:)=0;
% Get the bounding box and show if debug
margin = regionprops(M, 'BoundingBox');
if(VIDEO_DEBUG==1)
    fmargin = figure;
    imshow(M,[0 1]);
    rectangle('Position',margin.BoundingBox,'EdgeColor','y');
end

fig = figure;

%% 1.5 Find Base Histogram
HMbasehist = FindHist(I,nhcells,nhist,HM,0,0,'');

% For each frame in the video
f = initFrame;
while f<numFrames

    tic;

    %-----------------------
    % 2: MiFOD technique --
    %-----------------------

    %% 2.1 Initizalization
    % Get the current image
    I=vidFrames(:,:,:,f);
    I=rgb2gray(I);
    HSW = [int32(HM(1)-HM(3)/2) int32(HM(2)-HM(4)/2) HM(3)*2 HM(4)*2];  % Hyoid
Search Window
    HSWI = I(HSW(2):HSW(2)+HSW(4),HSW(1):HSW(1)+HSW(3));                 % Hyoid
Search Window Image
    % Equalize window (not all image) to emphasize colour changes
    HSWI = histeq(HSWI);
    % Heavy average filter to smooth detection curve
    HSWI = imfilter(HSWI,h,'replicate');

    % Initialize Hyoid structure
    hyoid.x = [];
    hyoid.y = [];
    hyoid.group = [];
    hyoid.centroidsx = [];
    hyoid.centroidsy = [];
    hyoid.numGroups = 0;

    % Scanning
    for i=1:size(HSWI,1)

        %% 2.2 Find the darkest pixel in the line that is a valley
        DC = diff(HSWI(i,:),2);
        min = Inf;
        for m=1:size(DC,2)
            if(DC(m)>0)
                if(HSWI(i,m) < min)
                    min = HSWI(i,m);
                    iMin = m;
                end
            end
        end

        %% 2.3 Sum the first derivative curve
        HF = 0;                                 % Hyoid flag (flag check if it's
relevant)
        DCF = diff(int32(HSWI(i,:)),1);      % First derivative to calculate
descent

        % Left sum (down)
        suml = 0;
        ST = int32(iMin-DT);
        if(ST < 1)
            ST = 1;
        end
        for k=ST:iMin
            suml = suml + DCF(k);
        end

        % Right sum (up)
        sumr = 0;
        ST = int32(iMin+DT);
        if(ST > size(DCF,2))
            ST = size(DCF,2);
        end
        for k=iMin:ST
            sumr = sumr + DCF(k);
        end

        % Sum derivative reaches threshold?
        if(suml < -MT && sumr > MT)
            % Check if its a valid point
            % Color reaches threshold?
            if(abs(int32(hyoid.color - HSWI(i,iMin))) < CT)
                HF = 1;                    % Is a valid point

                %% 2.4 Grouping point
                % Check distance to grouping
                mindist = Inf;
                for p=1:size(hyoid.x,2)
                    dist = pdist([iMin,i;hyoid.x(p),hyoid.y(p)]);
                    if(dist < mindist)
                        mindist = dist;
                        closest = p;
                    end
                end

                % Assign new point
                hyoid.x = [hyoid.x iMin];
                hyoid.y = [hyoid.y i];
                if(mindist < NT)
                    % Assign the same group as the closest point
                    hyoid.group = [hyoid.group hyoid.group(p)];
                else
                    % New group
                    hyoid.numGroups = hyoid.numGroups+1;
                    hyoid.group = [hyoid.group hyoid.numGroups];
                end
            end
        end

        % Draw scan line and center if debug
        if(VIDEO_DEBUG==1)
            subplot(2,2,1);
            imshow(HSWI);
            hold on;
            line([1 size(HSWI,2)],[i i],'Color','b');
            if(HF==0)
                plot(iMin,i,'r+');
            else
                plot(iMin,i,'y+');
            end
            subplot(2,2,3);
            if(HF==0)
                plot(1:size(HSWI,2),HSWI(i,:),'b',iMin,HSWI(i,iMin),'r+');
            else
                plot(1:size(HSWI,2),HSWI(i,:),'b',iMin,HSWI(i,iMin),'y+');
            end
            hold on;
            line([size(HSWI,2)/2 size(HSWI,2)/2],[0
255],'Color','g','LineStyle','--');
            axis([0 size(HSWI,2) 0 255]);
            hold off;
            outputvideo = addframe(outputvideo,getframe(fig));
        end
    end

    %% 2.5 Object position calculation
    if(~isempty(hyoid.x))

        % Calculate centroids by groups
        hyoid.centroidsx = zeros(1,hyoid.numGroups);
        hyoid.centroidsy = zeros(1,hyoid.numGroups);
        for p=1:size(hyoid.x,2)
            hyoid.centroidsx(hyoid.group(p)) = hyoid.centroidsx(hyoid.group(p))
+ hyoid.x(p);
            hyoid.centroidsy(hyoid.group(p)) = hyoid.centroidsy(hyoid.group(p))
+ hyoid.y(p);
        end
        H = hist(hyoid.group,hyoid.numGroups);
        mindist = Inf;
        % Minimum Movement Threshold (minimum distance between clusters
discovered)
        MMT = (HM(3)+HM(4)/2)/2;
        for p=1:size(hyoid.centroidsx,2)
            hyoid.centroidsx(p) = hyoid.centroidsx(p) / H(p);
            hyoid.centroidsy(p) = hyoid.centroidsy(p) / H(p);
            % Check distance to the centroid
            dist =
pdist([size(HSWI,2)/2,size(HSWI,1)/2;hyoid.centroidsx(p),hyoid.centroidsy(p)]);
            if(dist < mindist)
                mindist = dist;
                % Check minimum distance to be a neighbour
                if(dist < MMT)
                    HC = [hyoid.centroidsx(p) hyoid.centroidsy(p)];
                end
            end
        end
        % Calculate the new position
        HM = [ (int32(HSW(1)) + int32(HC(1)) - int32(HM(3)/2)) (int32(HSW(2)) +
int32(HC(2)) - int32(HM(4)/2)) HM(3) HM(4) ];
        % There exists a new position, so strategy is reset

    %--------------------------
    % 3: Movement Strategies --
    %--------------------------
    %% 3.1 Reset Strategies

        if(SearchStrategy==0 && FindWait>0)
            FindWait = 0;
            % Copy auxiliar frames
            if(STRAT_DEBUG==0)
                for auxf=1:size(VideoFrames,2)
                    outputvideo = addframe(outputvideo,VideoFrames(auxf));
                end
            end
            travel = [travel auxtravel];
            VideoFrames = [];
            auxtravel = [];
        elseif(SearchStrategy==1)
            SearchStrategy=0;
        end

    else
        if(SearchStrategy==0)
        %% 3.2 Left-search strategy (swallowing movement)
            % If is the first time, keep the current information
            if(FindWait==0)
                HMAux = HM;
                WaitFrame = f;
            end
            % If there are too many retries, change strategy
            if(FindWait > WaitRetries)
                HM = HMAux;
                FindWait = 0;
                SearchStrategy = ~SearchStrategy;
                f = WaitFrame;
                VideoFrames = [];
                % Back to the loop start
                continue
            end
            % Calculate new position to the left
            HM = [ HM(1)-HM(3)/2 HM(2)-HM(3)/16 HM(3) HM(4) ];
            FindWait = FindWait + 1;

        elseif(SearchStrategy==1)
        %% 3.3 Histogram calculation strategy (lost movement)
            % Delete auxiliar video frames
            minh = Inf;
            for y=-disthist:disthist
                for x=-disthist:disthist
                    HMhist = FindHist(I,nhcells,nhist,HM,x,y,HMbasehist);
                    dif = 0;
                    for i=1:nhcells*nhcells
                        dif = dif + abs(HMhist(i).difhist -
HMbasehist(i).difhist);
                    end
                    if(dif < minh)
                        minh = dif;
                        x0 = HM(1)+x;
                        y0 = HM(2)+y;
                    end
                end
            end
            HM = [ x0 y0 HM(3) HM(4) ];
        end
    end

    %-----------------------
    % 4: Margin position --
    %-----------------------

    %% 4.1 Calculate margin position
    % Reach left-margin
    if(HM(1)+HM(3)/2 < margin.BoundingBox(1))
        HM(1) = HM(1) + HM(3)/2;
    end
    % Reach right-margin
    if(HM(1)+HM(3)/2 > margin.BoundingBox(1) + margin.BoundingBox(3))
        HM(1) = HM(1) - HM(3)/2;
    end
    % Reach top-margin
    if(HM(2)+HM(4)/2 < margin.BoundingBox(2))
        HM(2) = HM(2) + HM(4)/2;
    end
    % Reach bottom-margin
    if(HM(2)+HM(4)/2 > margin.BoundingBox(2) + margin.BoundingBox(4))
        HM(2) = HM(2) - HM(4)/2;
    end

    %---------------------
    % 5: Coin position --
    %---------------------

    %% 5.1 Finds the coin position
    cc = FindCoin(I,cc);

    %-----------------------
    % 6: Plotting Output --
    %-----------------------

    %% 6.1 Marks video
    clf(fig);
    if(VIDEO_DEBUG==1)
        subplot(2,2,2);
        imshow(HSWI);
        hold on;
        colors = ['r','b','g','c','m','y'];
        for p=1:size(hyoid.x,2)
            plot(hyoid.x(p),hyoid.y(p),colors(mod(hyoid.group(p),6)));
        end
        plot(HC(1),HC(2),'y+');
        hold off;
        subplot(2,2,4);
        imshow(I);
        hold on;
        plot(HSW(1)+HC(1),HSW(2)+HC(2),'y+');
        text(double(HSW(1)+HC(1)),double(HSW(2)+HC(2)+15),[num2str(HM(1)) ','
num2str(HM(2))],'Color','w');

rectangle('Position',margin.BoundingBox,'EdgeColor','w','LineStyle',':');
        hold off;
    else
        image = imshow(I);
        hold on;
        plot(uint32(HSW(1)+HC(1)),uint32(HSW(2)+HC(2)),'y+');

%rectangle('Position',margin.BoundingBox,'EdgeColor','w','LineStyle',':');
        plot(cc(1),cc(2),'g+','linewidth',2);
        %text(cc(1),cc(2)+15,[num2str(int32(cc(1))) ','
num2str(int32(cc(2)))],'VerticalAlignment','middle','Color',[1 1 1])
        hold off;
    end

    % Calculate relative position to plot
    rpos = [-(cc(1)-(HM(1)+HC(1))) cc(2)-(HM(2)+HC(2))];

    if(STRAT_DEBUG==1)
        outputvideo = addframe(outputvideo,getframe(fig));
    end
    if(SearchStrategy==0 && FindWait>0)
       % Saves it to the buffer (because auxiliar strategy can fail)
        VideoFrames = [VideoFrames getframe(fig)];
        auxtravel = [auxtravel rpos(1) rpos(2)];
    else
        % Write current frame to the video
        if(STRAT_DEBUG==0)
            outputvideo = addframe(outputvideo,getframe(fig));
        end
        travel = [travel rpos(1) rpos(2)];
    end
    ET = ET + toc;
    f = f + 1;
end

% Close marks stream
```

```
outputvideo = close(outputvideo);
close(fig);
% Show elapsed time
disp(['Elapsed time: ' num2str(ET) ' (' num2str(ET/numFrames) ' for each
frame)'])

%% 6.2 Plot video
videoplot = avifile(['Output/' video '-plot.avi'],'compression','None');
pfig = figure;
hold on;

%% 6.3 Calculate axis with coin as (0,0)
% Calculate pixel/mm relation (1€ diameter equal to 23.25 mm)
coinrel = 23.25 / c(3);
for i=1:2:size(travel,2)-1
    travel(i) = travel(i)*coinrel;
    travel(i+1) = travel(i+1)*coinrel;
    if(travel(i) < plotaxis(1))
        plotaxis(1) = travel(i);
    end
    if(travel(i) > plotaxis(2))
        plotaxis(2) = travel(i);
    end
    if(travel(i+1) < plotaxis(3))
        plotaxis(3) = travel(i+1);
    end
    if(travel(i+1) > plotaxis(4))
        plotaxis(4) = travel(i+1);
    end
end

%% 6.4 Plot axis
axis([int32(plotaxis(1)-10) int32(plotaxis(2)+10) int32(plotaxis(3)-10)
int32(plotaxis(4))+10]);
xlabel('millimeters (mm)');
ylabel('millimeters (mm)');
for i=4:2:size(travel,2)
    plot([travel(i-3) travel(i-1)],[travel(i-2) travel(i)]);
    videoplot = addframe(videoplot,getframe(pfig));
end

% Close plot stream
close(pfig);
videoplot = close(videoplot);

% Catch the errors
catch e
    e
    outputvideo = close(outputvideo);
end
```

8.2 FindHist

Now, the function to find the histogram of the object is described:

```
%% FindHist
%  Master Thesis in Artificial Intelligence (UPC)
%  Automatic Demarcation for Videofluoroscopic Swallowing Study
%  Author: Matias Lizana
%  ------------------------------------------------------------
%  This function compares two histograms, used to search for the best
%  similar position in the image
%  Parameters:
%       I : Image to process
%       n : Size of the window
%       nhist : Precision in levels for histogram
%       rect : Coordinates of the window
%       xinc : x-displacement for the search window
%       yinc : y-displacement for the search window
%       basehist : Base histogram to compare with

function [ hist ] = FindHist(I,n,nhist,rect,xinc,yinc,basehist)

%% Initialization
hist = struct;
% Mapping variables
x = rect(1) + xinc;
y = rect(2) + yinc;
width = rect(3);
height = rect(4);
% Size of block in window
x_w = (width/n);
y_w = (height/n);
%% Compute histogram
for kx=0:n-1
    for ky=0:n-1
        % Index position of current block
        bi = kx*n + ky + 1;
        x_pos = x + x_w*kx;
        y_pos = y + y_w*ky;

        % Current window on image
        SI = I(uint32(y_pos):uint32(y_pos+y_w),uint32(x_pos):uint32(x_pos+x_w));

        % Calculation of parameters
        hist(bi).area = [x_pos, y_pos, x_w, y_w];
        hist(bi).hist = imhist(SI,nhist);
        hist(bi).difhist = 0;

        % Calculate diference with base histogram (if it exists)
        if(~isempty(basehist))
            for h=1:nhist
                hist(bi).difhist = hist(bi).difhist + abs(hist(bi).hist(h) -
basehist(bi).hist(h));
            end
        end
    end
end
end
```

8.3 FindCoin

This is the function to find the coin reference:

```
%% FindCoin
%  Master Thesis in Artificial Intelligence (UPC)
%  Automatic Demarcation for Videofluoroscopic Swallowing Study
%  Author: Matias Lizana
%  ------------------------------------------------------------
%  This function searches the coin shape center to take as reference in the
%  videofluoroscopy study
%  Parameters:
%       I: Image to search the coin
%       cc: coin center (1,2) and size (3,4)
%       <- coin: the final coin center position

function [coin] = FindCoin(I,cc)

coin = [0 0];

%% Edge extraction
CI = edge(I,'canny',0.1,2);

%% X-search
i=uint32(cc(1));
j=uint32(cc(2));
while(CI(j,i)~=1 && i < size(CI,2))
    i=i+1;
end
% Its inside right margin
if(i<cc(1)+cc(3))
    coin(1) = i-cc(3)/2;
else
    i=uint32(cc(1));
    while(CI(j,i)~=1 && i > 1)
        i=i-1;
    end
    % Its inside left margin
    if(i>cc(1)-cc(3))
        coin(1) = i + cc(3)/2;
    % If not, gets auxiliar point
    else
        coin(1) = cc(1);
    end
end

%% Y-search
i=uint32(coin(1));
while(CI(j,i)~=1 && j < size(CI,1))
    j=j+1;
end
% Its inside the bottom margin
if(j<cc(2)+cc(4))
    coin(2) = j - cc(4)/2;
else
    j=uint32(cc(2));
    while(CI(j,i)~=1 && j > 1)
        j=j-1;
    end
    % Its inside the top margin
    if(j>cc(2)-cc(4))
        coin(2) = j + cc(3)/2;
    % If not, gets auxiliar point
    else
        coin(2) = cc(1);
    end
end

coin(3) = cc(3);
coin(4) = cc(4);

% If the point is outside the margins, gets the auxiliar point
if(coin(1) > cc(1) + cc(3) || coin(2) > cc(2) + cc(4))
    coin = [cc(1) cc(2) cc(3) cc(4)];
end

end
```