This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

D2.4 – Integration of new access services in the EasyTV platform

EasyTV Project

H2020. ICT-19-2017 Media and content convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 30 months

Document. ref.: Deliverable 2.4

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	Short name	Country
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES
9	Unione Italiana dei Ciechi e degli Ipovedenti	UICI	IT

PROGRAMME NAME:	H2020. ICT-19-2017 Media and content convergence - IA Innovation action
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	MV
INVOLVED UNITS:	ENG, MV, ARX
DOCUMENT NUMBER:	D2.4
DOCUMENT TITLE:	Integration of new access services in the EasyTV platform
WORK-PACKAGE:	WP2
DELIVERABLE TYPE:	Report
CONTRACTUAL DATE OF DELIVERY:	30-09-2019
LAST UPDATE:	25/09/2019
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
V.0.1	11/06/2019	Draft	Nicolamaria Manes (MV)	Table of Contents definition and document structure
V.0.2	05/07/2019	Draft	Juan Pedro López (UPM) Silvia Uribe (UPM) Alberto Belmonte (UPM)	Added contribution on detection and recognition of textual contend based on OCR techniques.
V.0.3	10/09/2019	Draft	Nicolamaria Manes (MV)	Added contribution on Text-To-Speech Services for automatic voice synthesis of subtitles
V.0.3	12/09/2019	First Release	Nicolamaria Manes (MV)	First Review and formatting whole content
V.0.4	17/09/2019	Second Release	Nicolamaria Manes (MV)	Second Review and fixing content
V.0.5	20/09/2019	Candidate Release 1	Nicolamaria Manes (MV)	Release to be reviewed and approved by peer partners
V.0.6	24/09/2019	Candidate Release 2	Nicolamaria Manes (MV	Integrated comments from peer reviewer UPM and minor fix
V.0.7	25/09/2019	Candidate Release 3	Nicolamaria Manes (MV	Integrated comments from peer reviewer ENG and minor fix

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
TTS	Text To Speech
EBU-TT	European Broadcasting Union – Timed Text
WEB-VTT	Web Video Text Tracks
WCF	Windows Communication Fundation
IIS	Internet Information Service
API	Application Programming Interface
SAPI	Speech Application Programming Interface (Microsoft Speech API)
MSP	Microsoft Speech Platform API
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EAST	Efficient Accurate Scene Text detector
LSTM	Long Short-Term Memory
OCR	Optical Character Recognition
RPN	Region Proposal Network

Table of Contents

List of Figures

List of Tables

Executive Summary

This document describes the process of production of audio subtitles for the EasyTV platform. Broadcasters will benefit from this service by increasing and improving the production of accessible content for blind and visually impaired people.

The document is divided in two main parts: in the first one (chapter 2), we describe the service and the production process of audio subtitles by means of a text to speech service. It describes the architecture of the Web API for converting subtitles, provided as standard subtitle files (i.e. EBU-TT-D or WEB-VTT specifications), from text to audio format. Furthermore, we also provide a short description of the usage of the Web API through a Web GUI interface that implements the production process.

The second part (chapter 3), describes the mechanism of capturing subtitles, available in a graphic format, from video images, converting them into text by means of an OCR and deep learning techniques approach that helped us to define a more general solution.

In chapter 4 we report the tests and results of the OCR techniques adopted for the text detection and recognition related to the mechanism of capturing subtitles in graphic format.

Finally, in chapter 5, we describe the main innovations and advantages provided by the processing techniques adopted.

1. INTRODUCTION

This document describes the processes and technologies adopted to produce audio subtitles for a variety of languages. The resulting service will take advantage of the crowdsourcing platform to access the subtitles provided by broadcasters or by users for a certain piece of multimedia content.

To produce audio subtitles, we considered two different mechanisms: the first one is referred to a web-based text to speech tool developed as a Web API for the Automatic voice synthesis of subtitles. For this mechanism we have the availability of subtitles in a text format (e.g., EBU-TT-D[1] or Web VTT[2]). The subtitle file is parsed and every piece of the content is directly processed by the text to speech engine configured accordingly with the language of subtitles, the speed of the voice used and other parameters that will be explained later in this document.

The second mechanism is related to the subtitles available in a graphic format (e.g., DVB-SUB bitmaps). In this case we developed a new innovative system to extract the text information burned into video images which is based on the detection and recognition of textual content that relies on different deep learning techniques. The output of this phase is the production of a Web VTT file from the information extracted, in order to be automatically processed by the text to speech platform developed for the first mechanism.

The Automatic voice synthesis of subtitles includes a set of functionalities in the form of a Web API and a Web GUI interface.

Web API for the Automatic voice synthesis of subtitles can be also used for translating any text in an audio file in order to be included or mixed with any audio video streaming content (i.e. for audio narratives, HBBTV applications controlled with a remote-control device and so on).

All the services developed for the automatic voice synthesis of subtitles are available in the EasyTV platform through the EasyTV Service Manager Component as reported in the “D1.4. Final release of the EasyTV system architecture”.

[1] https://tech.ebu.ch/publications/tech3380 - The format for the distribution of subtitles over IP

[2] https://www.w3.org/TR/webvtt1/

2. TEXT-TO-SPEECH SERVICES FOR AUTOMATIC VOICE SYNTHESIS OF SUBTITLES

2.1. TTS Service overview and internal architecture

The main tasks of the Text-To-Speech Services for automatic voice synthesis of subtitles can be defined as two different phases: parsing of the standard subtitle file and conversion of texts into audio format and production of the final synchronized audio track based on the timing of each element of the subtitles. To accomplish these two phases, we designed and developed a set of web services that execute a set of tasks and subtasks described later in this document. The system architecture adopted for developing the services is WCF [3](Windows Communication Foundation), which is a unified programming model for building service-oriented applications. It enables developers to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing environments. Using WCF, is possible to send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS (Internet Information Service) or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.

The architecture of the TTS Service relies on the WCF framework and is built using free Text to Speech Engines (Microsoft Speech Platform) and TTS engines available, inside the Microsoft Operating Systems, at the server side of the architecture itself. Furthermore, the architecture of the TTS Service is developed in a way that any Text-To-Speech engine available through the Microsoft Speech API or as a service from the cloud can be easily integrated in the system.

The Figure 1 shows an overview of the high-level architecture with the set of Web API exposed.

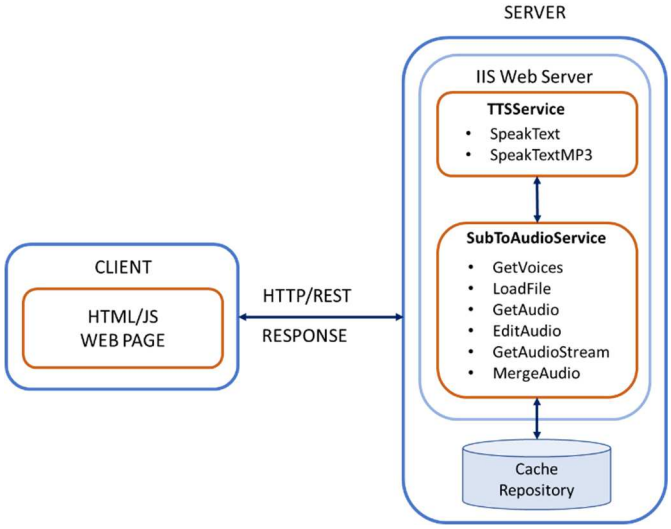


Figure 1: TTS service internal system architecture

As already mentioned, the aim of these API is to load and parse standard subtitle files (i.e. EBU-TT-D or WEB-VTT), convert every element of the subtitles into audio file and merge them in order to be synchronized with the related video content using the timing information contained in the subtitle file.

[3] https://docs.microsoft.com/en-us/dotnet/framework/wcf/

A set of parameters is defined to accomplish this production and synchronization such as audio quality, speed of the TTS engine or the TTS volume.

At the server side we have the IIS Web Server where the WCF service (**SubsToAudioService**), where it is published and running. This service exposes the following API that will be described later in this document:

- GetVoices
- LoadFile
- GetAudio
- EditAudio
- GetAudioStream
- MergeAudio

Beside the **SubsToAudioService** we developed a second service named **TTSService** that accomplishes the task of the conversion of the text into audio which uses the Text-To-Speech Engines through the Microsoft Speech API (SAPI), and Microsoft Speech Platform API (MSP), which are the main Microsoft programming interfaces to integrate TTS and ASR engines in any application.

The **SubsToAudioService** uses a **cache/repository**, where the requests and their data are stored (i.e. audio files, binary data, timing information, subtitle items and so on). The caching service and repository allows a quicker access over the data within the same session. A garbage collection is also performed on older data in order to keep the repository clean.

To use and consume the **SubsToAudioService** and **TTSService** services, a standard web page developed using HTML and Javascript technology can be developed on the client side. Requests to the sever side are executed using http/Restful protocol whose responses are formatted in XML or audio streaming (MP3 or WAV formats).

The **TTSService** is the service developed to manage the TTS engines and converts the audio files from WAV format to MP3 format. It is directly used by the SubsToAudioService GetAudio API to create all the audio files related to each subtitle item.

The **TTSService** scans the server for all TTS engines installed and manage all the languages and voices of the installed engines and it starts and initialize all engines to be ready for the conversion of the text items in audio file.

Since the TTS Engines creates only WAV file format, when a request of a WAV format audio is invoked from the SubsToAudioService the response of this service is the byte array of the audio streaming, while, a request of an MP3 file format is converted first in WAV format from the TTS engine and then a MP3 conversion is executed using the LAME library[4]. Finally, the byte array of the MP3 audio streaming is given back to the requestor.

The TTSService exposes the following API that will be described later in this document:

- SpeakText
- SpeakTextMP3

[4] http://lame.sourceforge.net/

2.2. Audio service Web API

As mentioned above we give now a short description of each Web API of the **SubsToAudioService**:

2.2.1. GetVoices

Description: Gets all voices of installed TTS engines available in the TTS Service.

Signature:

```
[OperationContract]
[WebInvoke(Method = "GET")]
XmlElement GetVoices();
```

Response: this request returns the following information in XML format:

```
<body>
      <voice>
            <name>Microsoft Elsa Desktop - Italian (Italy)</name>
            <lang>it</lang>
      </voice>
      <voice>
            <name>Microsoft Server Speech Text to Speech Voice (ca-ES, Herena)
            </name>
            <lang>ca</lang>
     </voice>
</body>
```

2.2.2. LoadFile

Description: it loads the subtitle standard file (i.e. EBU-TT-D or WEB-VTT) and parse its content. Gives a formatted XML file content with information on each item to be converted in audio format.

Input parameters:

- *fileContent*: byte stream of the subtitles file.
- *format:* subtitle file format (“xml” o “vtt”).

Signature:

```
[OperationContract]
[WebInvoke(Method = "POST", UriTemplate = "LoadFile/{format}")]
XmlElement LoadFile(Stream fileContent, string format);
```

Response: this request returns the following information in XML format:

```
<body>
            <token>3624a506-a12e-44eb-8a01-7dd04ab81b95</token>
            <item>
                  <id>sub0</id>
                  <begin>00:00:00.000</begin>
                  <end>00:00:01.800</end>
                  <duration>1800</duration>
                  <value></value>
            </item>
            <item>
                  <id>sub1</id>
                  <begin>00:00:02.000</begin>
                  <end>00:00:04.000</end>
                  <duration>2000</duration>
                  <value>Tears of Steel (part)</value>
            </item>
</body>
```

The *token* tag is used to define the user session and will be used during the next requests to access data user data in the repository.

The *item* tag contains all data related to a subtitle section:

- *id:* item identifier.
- *begin:* beginning time of the subtitle item.
- *end:* end time of the subtitle item.
- *duration:* duration in milliseconds of the subtitle item.
- *value:* text of the subtitle item.

2.2.3. GetAudio

Description: it creates all the audio files related to each subtitle item contained in the response file of the LoadFile API. It gives synchronization information of each item.

```
https://speakyfacile.eu/easytvproject/tts.ashx? text={text}&voice={voice}&vol={volume}
&rate={rate}&frm={format}&qlt={quality}&key={key}
```

Input parameters:

- *voice*: voice of the TTS engine chosen for the conversion of text to audio file. The Voice parameter is one of the voices listed by the *GetVoices API*).
- *rate:* Speed of the voice to be used (values from -10 to 10).
- *volume*: Volume of the TTS engine (values from 0 to 100).
- *quality:* Quality of the audio file 1 (best quality, more storage) to 8 (worst quality, less storage).
- *token*: identifies the user session. Value from LoadFile API.

- *format:* subtitle file format (“xml” o “vtt”).

Signature:

```
[OperationContract]
[WebInvoke(Method = "POST", UriTemplate =
"GetAudio/{lang}/{rate}/{volume}/{quality}/{token}/{format}")]
XmlElement GetAudio(string lang, string rate, string volume, string quality, Stream
fileContent, string token, string format);
```

Response: this request returns the following information in XML format:

```
<body>
            <item>
                  <id>sub0</id>
                  <begin>00:00:00.000</begin>
                  <end>00:00:01.800</end>
                  <duration>1800</duration>
                  <value></value>
                  <rate>0</rate>
                  <sync>YES</sync>
            </item>
            <item>
                  <id>sub1</id>
                  <begin>00:00:02.000</begin>
                  <end>00:00:04.000</end>
                  <duration>2000</duration>
                  <value>Tears of Steel (part)</value>
                  <rate>0</rate>
                  <sync>OK</sync>
            </item>
            <item>
                  <id>sub2</id>
                  <begin>00:00:18.800</begin>
                  <end>00:00:21.040</end>
                  <duration>2240</duration>
                  <value>Why does he do this?</value>
                  <rate>0</rate>
                  <sync>NO of 139 msec</sync>
            </item>
</body>
```

The content of this XML response is the response of the conversion process from text to audio of each subtitle item. It gives information and alerts about their synchronization according to the timing they are supposed to appear on the screen.

The *item* tag contains all the data of a specific subtitle item including the following three information:

- *rate:* Speed of the voice used to create the audio file.
- *duration:* duration of the audio file in milliseconds.
- *sync:* A value that indicates if the audio is well synchronized according to the timing they are supposed to appear on the screen. Values allowed:
  - YES: The audio file fits well in the begin and end time of the subtitle item. It means it's well synchronized.
  - OK: The audio file is included between the begin time and the begin time of the next audio element, so it doesn't overlap the next audio element.
  - NO of *{num}* msec: The audio file is not synchronized. It overlaps the begin time of the next audio element of a few milliseconds reported in the *sync* value.

2.2.4. **EditAudio**

Description: it allows to edit and modify the audio files according to the new parameters edited by the user of the application.

Input parameters:

- *voice*: voice of the TTS engine chosen for the conversion of the text to audio file. The *voice* parameter is one of the voices listed by the *GetVoices API*).
- *volume*: Volume of the TTS engine (values from 0 to 100).
- *quality:* Quality of the audio file 1 (best quality, more storage) a 8 (worst quality, less storage).
- *token*: identifies the user session. Value from LoadFile API.
- *format:* subtitle file format ("xml" o "vtt").
- *edit*: contains a list of subtitle elements with the corresponding speed value (rate) to apply.

Signature:

```
[OperationContract]
[WebInvoke(Method = "GET", UriTemplate =
"EditAudio/{lang}/{volume}/{quality}/{edit}/{token}")]
XmlElement EditAudio(string lang, string volume, string quality, string edit, string token);
```

Response: The response file of this API is the same of the GetAudio API.

2.2.5. GetAudioStream

Description: Allows listening of a single audio subtitle items previously created from GetAudio API.

Signature

```
[OperationContract]
[WebInvoke(Method = "GET",UriTemplate = "PlayAudio/{id}/{token}")]
Stream GetAudioStream(string id, string token);
```

Input parameters:

- *id:* item identifier.
- *token:* identifies the user session. Value from LoadFile API.

Response: Audio streaming of the audio subtitle item.

2.2.6. MergeAudio

Description: Allows listening of the whole audio subtitle file by merging all previously created audio items created from GetAudio API.

Input parameters:

- *token:* identifies the user session. Value from LoadFile API.
- *format*: format of the audio file (mp3 o wav).

Signature

```
[OperationContract]
[WebInvoke(Method = "GET", UriTemplate = "MergeAudio/{token}/{format}")]
Stream MergeAudio(string token, string format);
```

Response: Audio streaming of the audio subtitle file in the input format.

2.3. TTS Service Web API

As mentioned above we give now a short description of each Web API of the **TTSService**:

2.3.1. SpeakText

Description: Converts the subtitle text item in WAV audio file.

Signature:

```
[OperationContract]
byte[] SpeakText(string text, string voice, int rate, int volume);
```

Input parameters:

- *text*: text of the subtitle item
- *voice*: voice of the TTS engine choosen for the conversion of the text to audio file.
- *rate:* Speed of the voice to be used (values from -10 to 10).
- *volume*: Volume of the TTS engine (values from 0 to 100).

Response: WAV Audio stream of the subtitle item created using the parameters configures as reported in the input parameters section.

2.3.2. SpeakTextMP3

Description: Converts the subtitle text item in MP3 audio file.

Signature:

```
[OperationContract]
byte[] SpeakTextMP3(string text, string voice, int rate, int volume, int quality);
```

Input parameters:

- *text*: text of the subtitle item
- *voice*: voice of the TTS engine choosen for the conversion of the text to audio file.
- *rate:* Speed of the voice to be used (values from -10 to 10).
- *volume*: Volume of the TTS engine (values from 0 to 100).
- *quality:* Quality of the audio file 1 (best quality, more storage) a 8 (worst quality, less storage).

Response: MP3 Audio stream of the subtitle item created using the parameters configured as reported in the input parameters section.

2.4. Usage of the Web API – The Web GUI

In this section we report the usage of the Web API through a web application and its GUI. Furthermore, we also report how some code snipped in Javascript to consume the audio subtitling services.

The following **Figure** 2 Shows the homepage of the web application where to start the process of loading the subtitles file.

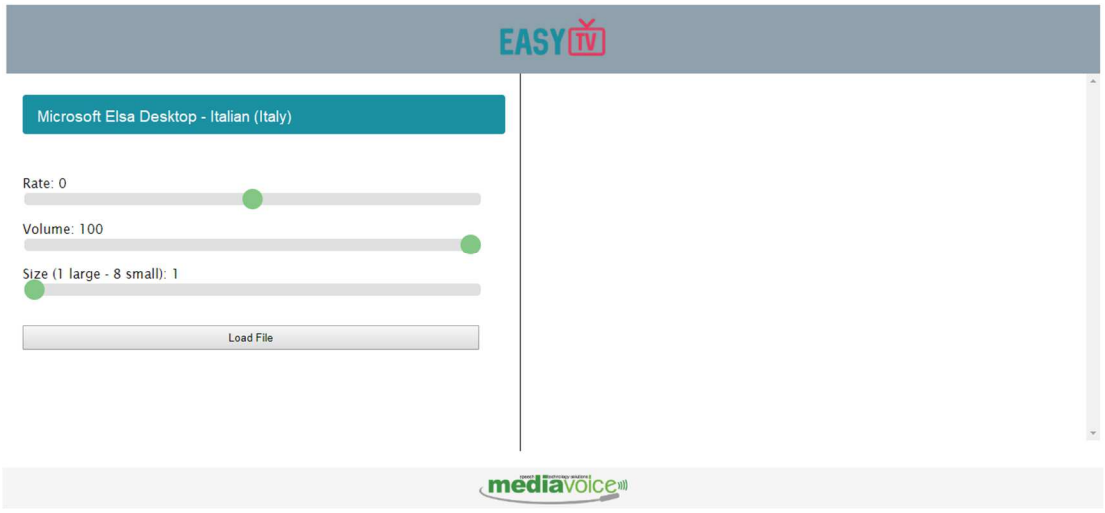


Figure 2: Web GUI - Homepage

The following picture (Figure 3), shows the list of subtitle items parsed and loaded by the web application.

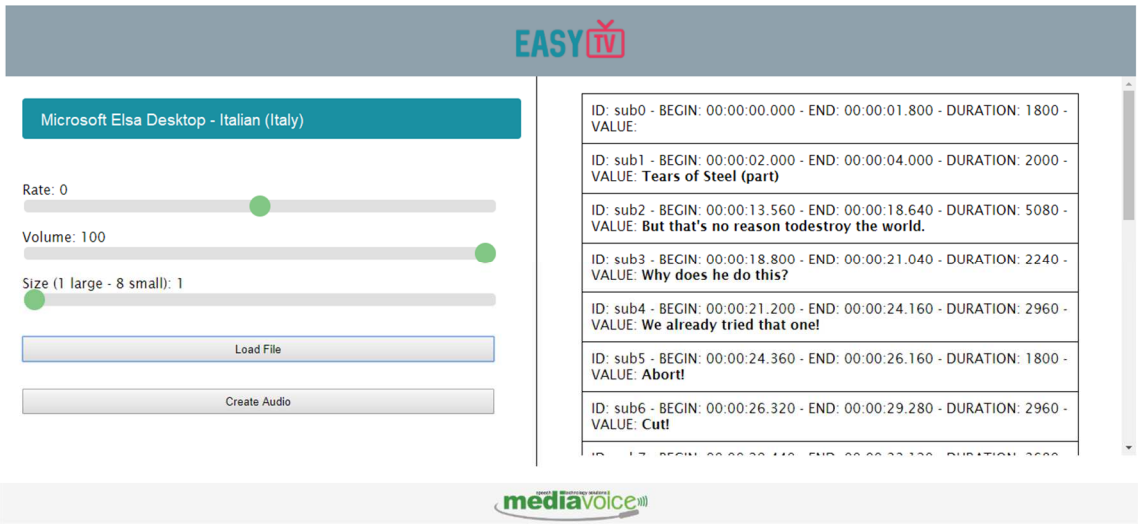


Figure 3: Web GUI - Subtitle items loaded

Finally, the Figure 4, shows the results of the checking phase of the subtitle items related to their begin and end time. Each item has a different color according to its synchronization state, that is:

- Green: The audio file fits well in the begin and end time of the subtitle item. It means it's

well synchronized.

- Yellow: The audio file is included between the begin time and the begin time of the next audio element, so it doesn't overlap the next audio element.
- Red: The audio file is not synchronized. It overlaps the begin time of the next audio element of a few milliseconds reported in the sync value.

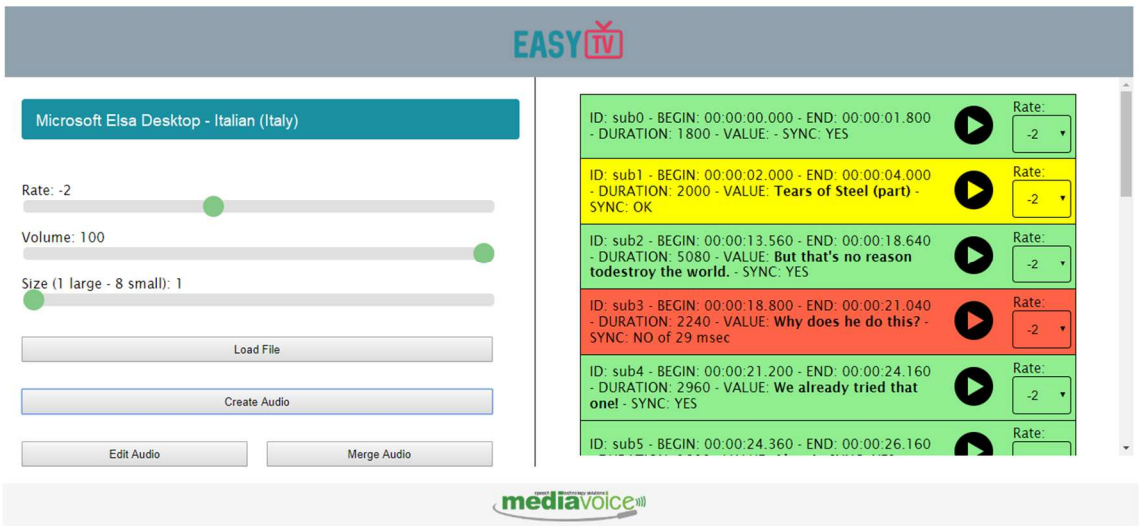


Figure 4: Subtitle items checking and editing

Code snippets

Sample code to consume GetVoices API using GET method.

```
const Http = new XMLHttpRequest();
const url = websiteservice + 'GetVoices';
Http.open("GET", url);
Http.setRequestHeader('Content-Type', 'application/xml')
Http.onreadystatechange = (e) => {
                    .....
}
Http.send();
```

Sample code to consume LoadFile API using POST method.

```
const Http = new XMLHttpRequest();
const url = websiteservice + 'LoadFile/' + format;
Http.open("POST", url);
Http.onreadystatechange = (e) => {
.....
}
Http.send(stream);
```

3. DETECTION AND RECOGNITION OF TEXTUAL CONTENT BASED ON OCR TECHNIQUES

3.1. Character recognition techniques

As it is well known, optical character recognition (OCR) is the main solution for text character recognition for optically processed characters and it aims at converting both hand-written or printed text into data files that can be easily edited and read by any kind of machine. These provides two main advantages mainly related to the increase of the productivity and the efficiency in storing texts. There are other areas that can beneficiate from its application such as multimedia content accessibility, as it is going to be explained during this chapter.

Although there are different techniques to be applied for optical character recognition, the universal system consists of four main phases which are concatenated in order to detect and obtain the desired results [1]:

1. Scanning: this phase is in charge of making the image acquisition.
2. Preprocessing and segmentation: this step is focused on cleaning up and enhancing the quality of the image by applying different method such as noise removal, correction, binarization, dilation, color adjustment and text segmentation.
3. Feature extraction: this phase is in charge of capturing the information from the acquired text image.
4. Recognition or classification: this final step is focused on mapping the previous segmented text to the equivalent textual representation.

Once the main process is defined, the next step when designing a OCR is to consider a set of attributes that may differ from one situation to another such as the text density, its structure, the type of fonts used, the artifacts of the image, the location of the text within the image and so on.

During the following sections we are going to present the main techniques applied in this field. After this, we are going to analyze the possibility of defining a new access service for detecting embedded text in video content based on the application of an OCR model that will be included in the EasyTV platform.

3.1.1. First approaches

Based on the schema mentioned above, there are different method for OCR purposes according to the different feature extraction and recognition techniques that can be applied. In this regard, first approaches were mainly based on the application of different computer vision techniques, as the OCR is one of the earliest addressed tasks in this field, proposing the following schema:

- Apply filters to distinguish between characters and background.
- Apply contour detection to recognize the characters one by one: this is the most challenging phase for generalization purposes, since it requires a lot of manual fine tuning.
- Apply image classification to identify the characters

Without going into details about the most primitive OCRs that were developed during the first decades of the 20th century [3], the 90s where the most prolific in terms of efficient solution definition thanks to the advance in the computation field. In fact, during these years the third and four generation of OCR were developed, and they were mainly focused on solving the poor quality problems for recognizing characters [3], improving the recognition along complex document and finally on using unconstrained handwritten characters [4] and color documents [5].

At that stage, the main purpose for the OCR were achieved but, during the first decade of the 21st century researchers were focused on improving the results in specific situations such as the recognition of cursive handwriting [6], documents with low resolution [7] and the recognition of

additional characters that previously have presented some problems such as the Tamil [8] and Gujarati language [9]. These solutions were mainly based on the application of different classic computer vision techniques during the preprocessing and feature extraction of the image such as shape recognition [6], gradient feature extraction [10], gray distribution.

As can be seen, there were several solutions that provided good results, but they were usually very specific ones for very specific use cases. For this reason, a new approach was needed in order to obtain more general solutions, and that is where the deep learning techniques can make the difference.

3.1.2. Deep learning techniques in the OCR environment

As above-mentioned, deep learning techniques may help to define more general solutions for OCR issues that are still challenging. In this context, although there are some early solutions based on neural networks [11] or on generalized regression neural networks [12], its application has only been generalized lately, at the same time as the real expansion of these methods, so there is still room for improvement.

Considering the attributes of OCR, the application of specialized networks is considered as more convenient than to apply more general solutions. In this way, some approaches are focus on using deep learning in the text detection, while others use these methods also in the text recognition phase. In either case, the advantages of applying deep learning should be taken into account when defining a new OCR approach.

Regarding the current state of the art, there are many examples to be considered according to their own definition. Nevertheless, we present two specific solutions for deep learning text detection and deep learning text detection and text recognition systems that may be taken into account thanks to the results they provide:

- EAST (Efficient accurate scene text detector) [13]: it is defined only for text detection and it has been added to open-CV library to allow its use. It is a version of the U-Net network, that presents good results for detecting features that may vary in size. It finally provides two types of output rotated bounding boxes for text detection.
- SEE [14]: this solution consists of a single deep neural network (DNN) that can learn to detect and recognize text in a semi-supervised way.

As it can be seen, the deep learning algorithms can be applied to both detecting and recognizing text information, and the selection of one or another only depends on the specific problem to be solved.

3.2. OCR as a starting point for improving the accessibility of multimedia content

During the early ages of the OCR application, it was considered as an aid for blind people, since it may be used as a support tool for accessing textual information but, as long as it provided more and more advantages in the digital age, it developed into a vast field of research to be applied to many other environments.

According to this and thinking about OCR as a text digitalization complex process, it is important to note that it can provide different advantages to be taken into account in the accessibility field. In this regard, through the most common options that OCR allows we can define different applications for improving the access to text information:

- Highlighting words, sentences or paragraphs: this may help the text identification and recognition by users with visual problems.
- Speaking words aloud using text to speech: this allows blind or visual impaired people to access text information in an accessible way. Moreover, this can also provide other possibilities like automated translations of the textual content into a different speak language, helping users that are not able to understand the original text.

- Changing the colors and the size of the text to make easier to read it.

With these ideas in mind, OCR may be very helpful when defining new access services for multimedia content, especially in the TV context. As it is well known, multimedia content consists not only on video and audio information, but also it may also content some additional info as the textual one, that can be presented in an accessible way such as the subtitles that can be digitally managed or not, as in the case of burned text within the image (see Figure 5). This information is usually important data about the content that helps the user to contextualize what he is viewing or listening, but it may get lost for some kind of disability people if there are no additional tools that support its access. For this reason, the application of OCR methods within this scenario may provide important advantages.

Figure 5: Different kind of text within image: subtitles (left) and burned text (right)

Based on this proposal, during the next section we are going to present a new access service focused on detecting text over video that has been defined and deployed within the EasyTV project.

3.3. EasyTV new service based on OCR

3.3.1. Definition, requirements and design of the solution

As it is well known, the main objective of the EasyTV is to make easier the access to the multimedia content to people with different disabilities. In the case of blind and visual impaired people, the main efforts are focused on provide a way to describe what is presented in the video, which is mainly solved by the use of audio guides (although they are a good solution, it is important to say that there are not available for all the contents, due to the related costs). But this tool does not cover all the possibilities in which the information is presented in a multimedia content, especially for the text.

As above-mentioned, some contents include text information that is burned or embedded in the image. This information is not processed during the audio guide creation, so it gets lost for visual impaired people. In some cases, it can be solved by the description of the rest of the info, but in some others, the user loses an important contextual information that it is needed for a better understanding of what they are watching. In a special way, this is the case of informative contents such as news, where all the visual information is completed by additional texts that contextualize and even give more details of what it is shown. For this reason, the new OCR service in the EasyTV project is focused on this type of contents.

Taking as an example the image in Figure 5 right, text information in news are usually presented at the bottom of the image, but sometimes there are other places where it can also appear. Moreover, it is usually a static label, but sometimes a moving label is also included. Finally, the size, font type and color can differ from a program to another, so there are many possibilities to be taken into account.

Considering this, the new OCR service may provide an automated solution for:

- Static text information embedded in the image.

- Dynamic text information (moving) embedded in the image.
- Different locations of the text within the image.
- Different sizes, fonts and colors.

According to these requirements and the need of providing a general solution to be applied to these different scenarios, we have defined a method based on the application of different deep learning techniques to extract the text information burned in video images. Once this information is obtained, next step is to provide it in a way it can be accessible. For doing so, the text information extracted is included in a WEB-VTT file to be processed as a subtitle file, allowing its presentation in an enhanced visual way or even with a text to speech solution.

3.3.2. **Modular development**

Text recognition service based on OCR methods consists of four main steps that have been developed in a modular way, as it can be seen in Figure 6:

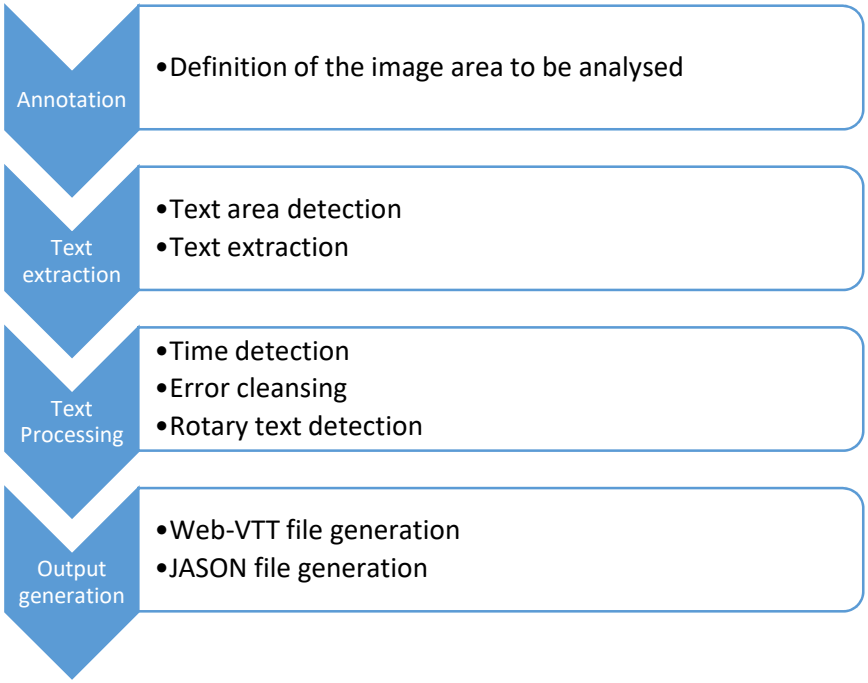


Figure 6: Main steps of the EasyTV text recognition service

Along next section each of these steps are going to be deeply explained.

3.3.2.1 Annotation

As it has been explained, the text information within informative content is usually presented in specific areas of the image, mainly on the bottom. With this is mind, and in order to reduce the processing effort, a first step for defining the area to be analyzed is proposed. In this regard, we have included an additional tool within the content annotation tool (see D2.3 when submitted) to let the administrator select the area to be processed.

Figure 7: Area selection for text detection (the user selects two specific areas on the left, and the user does not select any specific area on the right)

The annotation tool is prepared to provide several areas to process (as can be seen in Figure 7 left). Furthermore, if the user does not define an specific area, then the text detection is applied to the entire image, thus increasing the time processing (Figure 7 right). Moreover, this information is included in a JSON as an output of this step, as it is shown in the following example (each JSON object represents the coordinates of a specific area to be analyzed):

```
[{"x0": 0.05342465753424658, "y0": 0.829683698296837, "x1":
0.9452054794520548, "y1": 0.9878345498783455},
{"x0": 0.026027397260273973, "y0": 0.24574209245742093, "x1":
0.2328767123287671, "y1": 0.5644768856447688}]
```

Figure 8: Annotation tool output example

3.3.2.2 Text extraction

Once the image area to be analyzed is selected, then the second step is focused on extracting the text. This is a complex process that consists of two main methods consecutively applied: definition of the bounding box that comprises the texts and then the extraction of the text inside these bounding boxes.

3.3.2.2.1 Bounding box definition

The bounding box detection in the image is mainly based on a combination of one deep learning object detector and a recurrent unit that connect sequences in time. Actually, the state-of-the-art object detector Faster R-CNN [15] is achieving very high performance between detections and accuracy. A Region Proposal Network (RPN) is proposed to generate high-quality class-agnostic object proposals directly from convolutional feature maps. Then the RPN proposals are fed into a Fast R-CNN [16] model for further classification and refinement, leading to the state-of-the-art performance on generic object detection. However, it is difficult to apply these general object detection systems directly to scene text detection, which generally requires a higher localization accuracy. In generic object detection, each object has a well-defined closed boundary, while such a well-defined boundary may not exist in text, since a text line of word is composed of a number of separate characters or strokes.

To deal with these problems CTPN was proposed [17]. In this work, this gap is filled by extending the RPN architecture to accurate text line localization. An in-network recurrent mechanism that allows the model to detect text sequence directly in the convolutional maps is presented, avoiding further post-processing by an additional costly CNN detection model. VGG [18] network is used as backbone for feature extraction. Then, a RPN is used to detect potential regions of interest in the image. The Fully connected Fast-RCNN architecture part is replaced by a bidirectional recurrent neural network that takes the feature maps sequentially and finally a fully connected layer performs a regression obtaining the final bounding box parameters and scores as in Fast-RCNN.

To train the network two dataset have been used. The ICDAR 2015 (Incidental Scene Text - Challenge 4) [19] includes 1,500 images which were collected by using the Google Glass. The training set has 1,000 images, and the remained 500 images are used for test. This dataset is challenging due to the fact that includes arbitrary orientation, very small-scale and low-resolution text. The second dataset is the multilingual scene text dataset [20]. It contains 248 images for training and 239 for testing. The images include multi-languages text, and the ground truth is labeled in text line level.

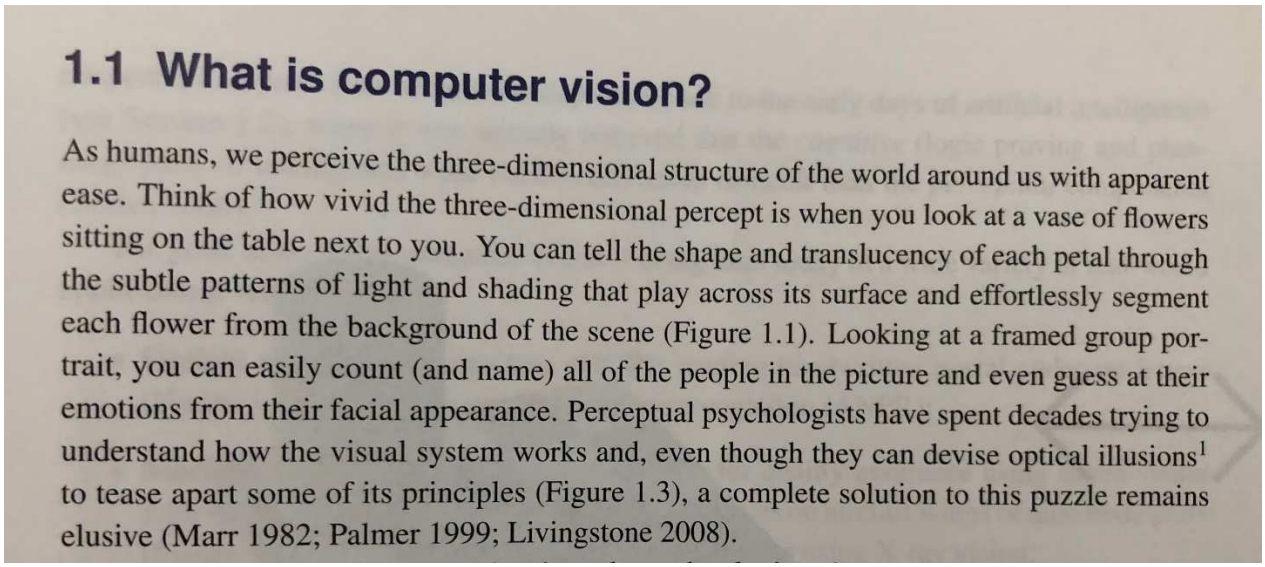
The training hyperparameters have been fixed to the proposed in the CTPN paper. VGG16 model pre-trained on the ImageNet data is used. The new layers have been initialized by using random weights with Gaussian distribution of 0 mean and 0.01 standard deviation. 0.9 momentum and 0.0005 weight decay with Adam optimizer is included in training. The learning rate was set to 0.001 in the first 16K iterations, followed by another 4K iterations with 0.0001 learning rate. As output example Figure 9 shows some final bounding box detections over the test set of ICDAR and Multilingual scene text datasets.

Figure 9: Examples of text detection. Top left shows a simple text in a panel, top right shows the detection over a chinesse text and bottom image one text from internet

3.3.2.2.2 Text extraction

For text extraction Tesseract [21] development was used. Tesseract was developed as a proprietary software by Hewlett Packard Labs. In 2005, it was open sourced by HP in collaboration with the University of Nevada, Las Vegas. Since 2006 it has been actively developed by Google and many open source contributors. Tesseract acquired maturity with version 3.x when it started supporting many image formats and gradually added a large number of languages. Tesseract 3.x is based on traditional computer vision algorithms. In the past few years, Deep Learning based methods have surpassed traditional machine learning techniques by a huge margin in terms of accuracy in many areas of Computer Vision. In version 4, Tesseract has implemented a Long Short-Term Memory (LSTM) based recognition engine using the concept of recurrent neural networks.

In this work, Tesseract is used with the most recent Deep Learning version. The language to detect is selected manually. Finally, page segmentation mode is set to lines due to the bounding boxes detected by the previous detector are text lines. Two example of text detections are presented below in Figure 10 and Figure 11.



1.1 What is computer vision?

As humans, we perceive the three-dimensional structure of the world around us with apparent ease. Think of how vivid the three-dimensional percept is when you look at a vase of flowers sitting on the table next to you. You can tell the shape and translucency of each petal through the subtle patterns of light and shading that play across its surface and effortlessly segment each flower from the background of the scene (Figure 1.1). Looking at a framed group portrait, you can easily count (and name) all of the people in the picture and even guess at their emotions from their facial appearance. Perceptual psychologists have spent decades trying to understand how the visual system works and, even though they can devise optical illusions[1] to tease apart some of its principles (Figure 1.3), a complete solution to this puzzle remains elusive (Marr 1982; Palmer 1999; Livingstone 2008).

Algorithm Output

1.1 What is computer vision? As humans, we perceive the three-dimensional structure of the world around us with apparent ease. Think of how vivid the three-dimensional percept is when you look at a vase of flowers sitting on the table next to you. You can tell the shape and translucency of each petal through the subtle patterns of light and Shading that play across its surface and effortlessly segment each flower from the background of the scene (Figure 1.1). Looking at a framed group por- trait, you can easily count (and name) all of the people in the picture and even guess at their emotions from their facial appearance. Perceptual psychologists have spent decades trying to understand how the visual system works and, even though they can devise optical illusions! to tease apart some of its principles (Figure 1.3), a complete solution to this puzzle remains elusive (Marr 1982; Palmer 1999; Livingstone 2008).

Figure 10: Text extraction using tesseract over book text document

Figure 11: Tesseract applied over detected bounding box by CTPN network

3.3.2.3 Text processing

This step takes advantage of the fact that we are working with video, not just images. This means that the text processing can use the temporary redundancy along different frames to correct the detection, to clean errors and to detect how many time the same text information is presented, that is, to detect scene texts. Moreover, this step is also in charge of distinguish between static and rotary text information within the video content.

For moving text detection is important to note that this kind of information complies with the following characteristics: the number of the detected characters along different frames is not always the same, the bounding box that contains the text is usually big (in relation with the image size) and, finally, the text information stays during a period of time in the image. Moreover, the detection process defined for this service consists of a set of specific steps that are explained bellow:

1) First of all, a 25 frame sampling is done in order to reduce the number of frames to process. By doing so, the system only processes the text information in each 25 frames.
2) Once the sampling is done, the sentence that has been detected is split into words.
3) In order to avoid the detection of incomplete words, first and last detected words are deleted.
4) A set of adjacent characters is detected in each sampled frame (for this service we detect 100 different characters, but this parameter can be modified according to the scenario) and then this string is sought along the different frames in order to find when it appears again (this take place when the matching is higher than a limit -90% in this case-).
5) When this is done, the individual moving information is what has been detected between these two points.

After this point the moving text is processed as the static one. Once a string is detected, the temporal redundant information is used to clean some errors and to select correct sentence that has been detected. To sum up, the steps to make this clean process are the following:

1) To apply the NLTK[5] python library for character error cleaning.
2) The detected text must appear a minimum time at least (50 frames) in order to confirm that

[5] https://www.nltk.org/

the detection is correct.

3) Then, the most repeated sentence along the detection is selected as the correct one (this may be improved by the application of other different natural language processing methods, but it is not necessary since the results are good enough to not increase the process time with additional steps). This helps also to establish the starting and ending point of each text in the image.

3.3.2.4 Output generation

Once this is done, all the text information in the video is completely extracted and recognized. This output is provided in a JSON file that includes the extracted sentences and the frames where they are shown, as it is shown below:

```
[["Causa especial 20907  2 . Senyal sala. PREMI D HONOR DE LES LLETRES
CATALANES. ULTIMA HORA. HORA LA POETESSA I  ASSAGISTA MARTA PESSARRODONA  NOU
PREMI D HONOR DE LES LLETRES CATALANES  ES LA SISENA DONA  EN 51 EDICIONS
QUE GUANYA LA SISENA DONA  EN 51 EDICIONS  QUE GUANYA LA DISTINCIO D OMNIUM
CULTURA. ", [80, 228]], ["Toni Punti. ", [820, 933]], ["LA GUANYADORA ES L
ESCRIPTORA MARTA PESSARRODONA. ", [939, 1133]], ["MARTA PESSARRODONA VA
NEIXER A TERRASSA EL 1941. ", [1143, 1899]], ["HA CONREAT POESIA  PROSA
ASSAIG  CRITICA  BIOGRAFIA. ", [4415, 4564]], ["HA ESTUDIAT A FONS LA
LITERATURA DE DONES. ", [4574, 4676]]]
```

Figure 12: Example of the text processing output in a JSON file

Finally, a WEB-VTT file is also created to be used by the text to speech system created in the project. An example of this file can be seen in Figure 13:

```
00:00:03.000 --> 00:00:09.006
Causa especial 20907  2 . Senyal sala. PREMI D HONOR DE LES LLETRES
CATALANES. ULTIMA HORA. HORA LA POETESSA I  ASSAGISTA MARTA
PESSARRODONA  NOU PREMI D HONOR DE LES LLETRES CATALANES  ES LA SISENA
DONA  EN 51 EDICIONS  QUE GUANYA LA SISENA DONA  EN 51 EDICIONS  QUE
GUANYA LA DISTINCIO D OMNIUM CULTURA.

00:00:32.000 --> 00:00:37.005
Toni Punti.

00:00:37.000 --> 00:00:45.008
LA GUANYADORA ES L ESCRIPTORA MARTA PESSARRODONA.

00:00:45.000 --> 00:01:15.030
MARTA PESSARRODONA VA NEIXER A TERRASSA EL 1941.

00:02:56.000 --> 00:03:02.006
HA CONREAT POESIA  PROSA  ASSAIG  CRITICA  BIOGRAFIA.

00:03:02.000 --> 00:03:07.005
HA ESTUDIAT A FONS LA LITERATURA DE DONES.
```

Figure 13: Example of the text processing output in a WEB-VTT file

4. TESTS AND RESULTS

To start the discussion of the results the first step is to collect all the information related to the initial stage in the complete algorithm chain. The first part is the bounding box detection. The algorithm proposed (CTPN) have been trained using both commented datasets mixed between then to have not only a big quantity of data, but also some data in several different languages. The training process takes about 0.35 seconds per iteration. So, it will take about 2 hours to finish the 20k iterations. After some tests, we decided to increase the number of iterations to get a better result due to the network loss continued decreasing. In this regard, using 30k iterations the results are better resulting in better accuracy but in more training time. All the processing methods have been applied on a PC with the following characteristics: 32GB RAM, Intel i7 5930K and a Titan RTX 24GB graphic card. The results for the bounding boxes predicted using the test set provided by the datasets are shown in the following table:

Table 1. Metrics for the bounding box detection

Learning rate/Iterations	Precision	Recall	F1-Score
16k with 0.001 and 4k with 0.0001	71%	50%	59%
20k with 0.001 and 10k with 0.0001	75%	52%	62%

The results are promising, providing a slightly better performance than in the original paper due to the combination of both presented datasets. Some detection results related to the proposed use of the algorithm are presented below:

Figure 14: Example of the bounding box detection (subtitles detection)

Figure 15: Example of the bounding box detection (additional information)

These two examples contain different cases of detections. The first image presents an example of subtitles detection. The second is a news program with a lot of parts with different types of texts (titular, text on screens, tv logo, text in video objects…).

According to the obtained results, we can claim that CTPN has a strong capability of detecting extremely challenging text and very small-scale ones, some of which are even difficult for human. Moreover, the detector performs favorably against the multilingual texts.

Having said that, the main problem of this algorithm is the processing time. It has been tested with different image sizes to evaluate performance and the results are presented in the next table:

Table 2. Bounding box detection processing time

Input Image Size (pixels)	Algorithm Processing Time (seconds)
1920x1080	0.81
1280x960	0.66
980x480	0.54
1800x100	0.32

As can be seen, it is interesting to analyze the results. The first three input sizes are common video shapes used in video recordings. The time processing is quite high for each entire image -more than a half of a second for a 980*480 pixel image. For the last case, we proposed an alternative way to process the video: instead of using the entire image, we only analyze the area where the texts are burned. This allows a significant reduction of the time processing. Having this in mind, we can conclude that the detector is very efficient in accuracy, but the total time is dependent on the duration of the videos.

To solve or reduce this problem, video resolution can be decreased, thus losing accuracy in both bounding box detection and text detections. Another solution can be to avoid applying the detector every frame, but employ it every specific number of frames due to the high redundancy of the text information.

Finally Figure 16 shows an example of the extracted text from Figure 15, including the bounding boxes vertices.

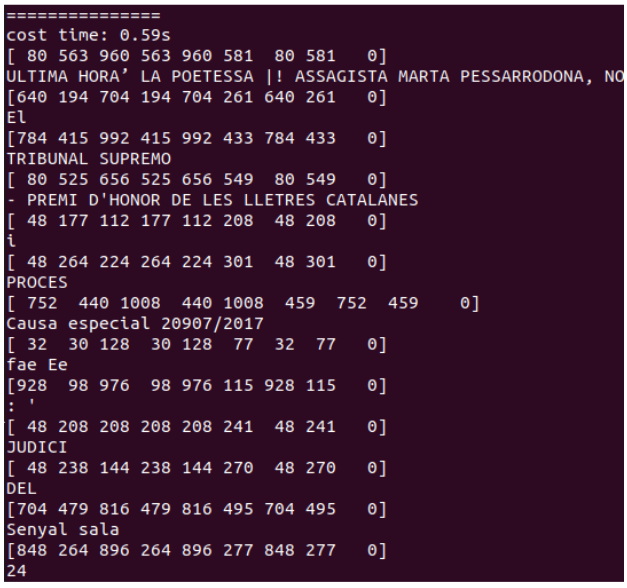


Figure 16: Example of extracted text

Text extraction method works properly but there are some issues to take into account. In this regard, and as it can be seen in the example, if the entire image is processed, then some extracted texts may not be the ones desired, such as the tv logo. Moreover, these extractions can also contain strange characters that have to be removed in a next cleaning step.

5. INNOVATIONS

The main innovations of this service are related to the detection and recognition of content based on OCR techniques. This service brings the following innovations as listed here below:

- Combination of some existing specific deep learning techniques for OCR purposes in order to provide a general solution.
- Ability to detect both static and moving information in an automatic way.
- Usage of the time redundancy detection for error correction along scene text recognition.

Furthermore, we have also many general advantages using these techniques, such as:

- It helps access the text information burned in the video content that is not usually accessible.
- It allows the system administration to select the area for text detection. This helps to reduce the process timing.
- It helps to detect moving text information along with video content.
- It provides a WEB-VTT output to be used by the text to speech platform.
- The text provides a JSON output improving the presentation in the Companion Screen Application by changing the presentation parameters (i.e. color and size).

The following highlights other innovations regarding the process of converting subtitles from text to audio format:

- Automation of the production of audio subtitles.
- Increase the availability of more accessibility content for blind and visual impaired people.
- TTS parameters customization for improving audio quality of subtitles.
- Production of audio subtitles in different languages thanks to the integration and use of speech synthesis engines available in various languages and voices.

6. CONCLUSIONS

In this document we described the process of production of audio subtitles to be consumed by blind and visually impaired people in the EasyTV platform.

We described the two phases (text to audio conversion and text detection and recognition) of the above production process, based on the format in which subtitles are provided and reported the results of the detection and recognition phase for subtitles provided in a graphic format.

We implemented a Web API for converting text into audio that can be expanded including any TTS engine (Text To Speech) in its back-end architecture, both local and cloud-based TTS engines.

Furthermore, all the components provided by the service of production of audio subtitles have been implemented as separate modules that can be used based on the needs of the broadcasters.

Finally, the services available from the two mechanisms described in this document will also be available in the Service Manager component of the EasyTV ecosystem so that the broadcasters can use them for their own implementation of the production process workflow.

7. REFERENCES

[1] Priyanto Hidayatullah, Nurjannah Syakrani, Ida Suhartini, Wildan Muhlis, "Optical Character Recognition Improvement for License Plate Recognition in Indonesia", *IEEE symposium UKSim-AMSS 6th European Modelling Symposium*, 2012.

[2] Gustav Tauschek. Reading machine. U. S. Patent 2026329, ht tp:// www. google. com/patentsvid=USPAT2026329, December 1935FLEXChip Signal Processor (MC68175/D), Motorola, 1996

[3] S. Mori, C. Y. Suen, K. Yamamoto, "Historical review of OCR research and development", Proc. IEEE 80 (1992) 1029-1058.

[4] Tanaka H, Nakajima K, Ishiqaki K, Akiyama K, Nakagawa M, "Hybrid pen-input character recognition system based on integration of online-offline recognition", Proceedings of the fifth International Conference on Document Analysis and Recognition (ICDAR-1999) 209-212. "

[5] J. Mant as, "An overview of character recognition methodologies", Pattern Recognition 19 (1986) 425-430.

[6] Arica N, Yarman-Vural F T., "Optical character recognition of cursive handwriting", IEEE T ransactions on P att ern Anal. And Mach. Intell. Volume 24, Issue 6, (2002) 801-813.

[7] Chunmei Liu, Chunheng Wang, Ruwei Dai, " Low resolution character recognition by Image quality evaluation", Proceedings of the Eighteenth International Conference on Pattern Recognition (ICPR 2006) 864-867.

[8] Kannan R J, P rabhakar R, Suresh RM, " Off-line cursive handwritten Tamil character recognition", Int ernational Conference on Securit y technology (2008) 159-164.

[9] P rasad J R, Kulkarni U V, P rasad R S, " Offline handwritten character recognition of Gujarati script using pattern matching", 3Rd International Conference on Ant i-counterfeit ing Security and Ident ification in communication, 2009, 611-614.

[10] Hailong Liu, Xiaoging Ding, "Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes", P roceedings of the Eighth Int ernational Conference on Document Analysis and Recognition (2005), (vol. 1) 19-23.

[11] Ohhira T., Pecharanin N., Taguchi A., Iijima, N.; Akima, Y.; Sone, M., " Chinese character recognition by the auto recognition system ", IEEE International Conference on Neural Networks, Volume: 5, (1995) 2222-2225.

[12] Wang Yut ao, Qin Tingting, Tian Ruixia, Yang Gang, "Recognition of license plate character based on wavelet transform and generalized regression neural network", Control and Decision Conference (CCDC), 2012 24th Chinese, 1881-1885.

[13] Zhou, Xinyu & Yao, Cong & Wen, He & Wang, Yuzhi & Zhou, Shuchang & He, Weiran & Liang, Jiajun. (2017). EAST: An Efficient and Accurate Scene Text Detector. 2642-2651. 10.1109/CVPR.2017.283.

[14] Bartz, C., Yang, Haojin, Meinel,. C. “SEE: Towards semi-supervised end-to-end scene text recognition”. On 32nd AAAI Conference on Artificial Intelligence, 2018.

[15] Ren, Shaoqing et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*39 (2015): 1137-1149.

[16] Girshick, Ross B.. “Fast R-CNN.” *2015 IEEE International Conference on Computer Vision (ICCV)* (2015): 1440-1448.

[17] Tian, Zhi et al. “Detecting Text in Natural Image with Connectionist Text Proposal Network.” *ArXiv* abs/1609.03605 (2016): n. pag.

[18] Simonyan, Karen and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” *CoRR* abs/1409.1556 (2015)

[19] Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S., Bagdanov, A., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., Shafait, F., Uchida, S., 16 Z. Tian, W. Huang, T. He, P. He and Y. Qiao Valveny, E.: Icdar 2015 competition on robust reading (2015), in International Conference on Document Analysis and Recognition (ICDAR).

[20] Pan, Y., Hou, X., Liu, C.: Hybrid approach to detect and localize texts in natural scene images. IEEE Trans. Image Processing (TIP) 20, 800–813 (2011)

[21] Smith, Rory. “An Overview of the Tesseract OCR Engine.” *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* 2 (2007): 629-633.