

---

---

# Intégration de statistiques en temps réel sur flux vidéo

*Real time statistics on live stream*

---

par

ZELLER QUENTIN

h e p i a

---

Haute école du paysage, d'ingénierie  
et d'architecture de Genève

Filière communications, multimédia et réseaux  
HAUTE ÉCOLE DU PAYSAGE, D'INGÉNIERIE ET D'ARCHITECTURE DE GENÈVE

Rapport concernant le travail de semestre du Bachelor en  
INGÉNIERIE DES TECHNOLOGIES DE L'INFORMATION dans la  
spécialisation COMMUNICATIONS, MULTIMÉDIA ET RÉSEAUX.

FÉVRIER 2018

Directeurs du travail : El Maliki Tewfiq, Revuelta Andres



# *Abstract*

Ce travail consiste en une étude de marché concernant l'intégration de contenu dans des flux vidéo en direct. Il posera les bases pour l'élaboration du projet en soi. C'est à dire, le développement d'une solution permettant la récupération et l'affichage de données dans un flux vidéo en temps réel. Cette application doit pouvoir être utilisée facilement, sans connaissance ni infrastructures particulières. Elle est en somme, une application tout public. Cette recherche a comme ambition de rendre l'affichage de données sur une vidéo live plus aisés, les solutions actuelles étant restreintes à des cas particuliers.

**Introduction** Une étude du marché succincte où les solutions seront discutées brièvement à propos de leurs pénétrations sur le marché, leurs prix, ainsi que les fonctionnalités qu'elles proposent. (chapitre 1)

**OpenCV & Flask** Discussion d'OpenCV, une librairie pour le traitement multimédia très répandue. Elle permet l'analyse et la retouche des vidéos et est connue en particulier pour ses fonctions de machine learning. (chapitre 2 )

**HBBTV** Discussion de la solution applicative destinée aux postes de télévision européens. HBBTV est une solution permettant l'ajout de contenu et l'interaction directe avec les utilisateurs de la télévision conventionnelle que se soit sur le câble ou via la télévision IP. (chapitre 3)

**Wowza** Discussion à propos du serveur de contenu vidéo Wowza ; Un service permettant la transcription et la distribution de flux vidéo direct ou à la demande. (chapitre 4)

**Produits sur le marché** Discussion des autres solutions disponibles sur le marché. (chapitre 5)



## *Remerciement*

J<sup>e</sup> remercie.



# *Table of Contents*

	<b>Page</b>
<b>Liste des tableaux</b>	<b>vii</b>
<b>Table des figures</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Préambule . . . . .	1
1.2 Cahier des charges de l'application . . . . .	2
1.2.1 État des lieux du marché . . . . .	2
<b>2 OpenCV &amp; Flask</b>	<b>7</b>
2.1 Introduction . . . . .	7
2.2 Fonctionnement général . . . . .	7
2.3 Environnement de développement de l'application test . . . . .	8
<b>3 HBBTV</b>	<b>13</b>
3.1 Fonctionnement général . . . . .	14
3.1.1 Utilisation, le cas de la Suisse. . . . .	14
3.2 Environnement de développement . . . . .	14
3.2.1 Opera TV Emulator . . . . .	14
3.3 Environnement de production . . . . .	15
3.3.1 The Opera hybrid TV option . . . . .	15
<b>4 Wowza</b>	<b>19</b>
4.1 Fonctionnement général . . . . .	19
4.2 Environnement de développement de l'application test . . . . .	19
4.3 Environnement de production . . . . .	20
4.3.1 The Opera hybrid TV option . . . . .	20
<b>5 Produits sur le marché</b>	<b>21</b>
5.1 Fonctionnement général . . . . .	21
5.2 Environnement de développement de l'application test . . . . .	21

---

5.3 Environnement de production . . . . .	22
5.3.1 Linux & FFMPEG . . . . .	22
<b>6 Conclusion</b>	<b>23</b>
<b>A Annexe A</b>	<b>25</b>
<b>Bibliographie</b>	<b>27</b>

## *Liste des tableaux*

<b>TABLE</b>	<b>Page</b>
--------------	-------------



## *Table des figures*

<b>FIGURE</b>	<b>Page</b>
1.1 Schémas de fonctionnement applicatif . . . . .	2
2.1 The easy Flask website MJPEG. - Visualisation Opera . . . . .	11
2.2 The easy Flask website MJPEG - Visualisation VLC. . . . .	11
2.3 Burn text on live stream, display on GUI . . . . .	12
3.1 HbbTV Broadband vs Broadcast . . . . .	13
3.2 Opera emulator user interface . . . . .	16
3.3 Opera emulator installation . . . . .	17
3.4 Opera emulator devlopper tools . . . . .	18



# 1 *Introduction*

## 1.1 Préambule

Ce travail de recherche consiste en la reconnaissance des différents outils logiciels disponibles sur le marché, permettant l'ajout d'informations en temps réel sur un flux vidéo, dans le but d'apporter aux client-consommateurs des données supplémentaires, notamment lors d'évènements sportifs. Nous utiliserons ici l'exemple du volleyball sans pour autant restreindre les recherches à ce seul but. Cette étude de cas concerne bien évidemment un domaine plus vaste que celui du volleyball ou même des jeux.

L'approche pratique de ce travail explorera aussi les différentes possibilités de collecter les informations statistiques via une interface fortement découpée ce qui permettra l'intégration des données de la manière la plus générique possible. Cette modularité permettra d'intégrer d'autres sources de données sans repenser l'intégralité de l'application et permettra une plus grande flexibilité quand au sport/jeux auquel elle s'adresse.

Si ce travail s'avère réalisable au regard des contraintes d'un mémoire de Bachelor, il aboutira en l'implémentation d'une solution fonctionnelle. Il est donc nécessaire de recenser toutes les technologies différentes disponibles sur le marché qui serviront comme outils à ce projet. Dans cette introduction sera présenté un tableau recensant toutes les technologies nécessaires à l'implémentation du sujet. Certain de ses outils seront brièvement testés dans les chapitres suivants afin de démontrer leurs difficultés d'implémentation, leurs qualités ainsi que leurs compatibilités.

De prime abord, il semblerait que deux groupes se distinguent de par les technologies qu'ils emploient. Le groupe télévisuel ordinaire qu'est la "télévision de salon" et un groupe plus orienté ordinateur/téléphone mobile. Nous étudierons s'il est possible de concilier ces deux groupes et choisirons le cas échéant le médium le plus adéquat. Une projection vers le futur est aussi nécessaire, la convergence vers le tout IP ainsi que l'amélioration rapide des Smart-TVs obligent à se projeter vers le futur.

## 1.2 Cahier des charges de l'application

Le but final de ce travail est de trouver un moyen efficace, fiable et peu couteux de diffuser du contenu audio-visuel à valeur ajoutée et en temps réel. Plus concrètement il s'agit dans le cas d'une retransmission sportive de, premièrement récupérer le/les flux vidéos, les sources de données à incorporer aux flux vidéos, puis de les combiner en un seul service multimédia soit par l'injonction directe des données dans le flux vidéo, ce qui reviendrait à "bruler" la vidéo avec les informations ou alors, solution plus modulable, incorporer ces informations sur un flux parallèle que nous discuterons au chapitre 3. Le résultat final est un flux audio-visuel ne nécessitant aucun applicatif supplémentaire pour fonctionner. Il devra être directement utilisable sur une télévision, un smart-phone ou un ordinateur et permettra donc la lecture sur n'importe quel appareil.

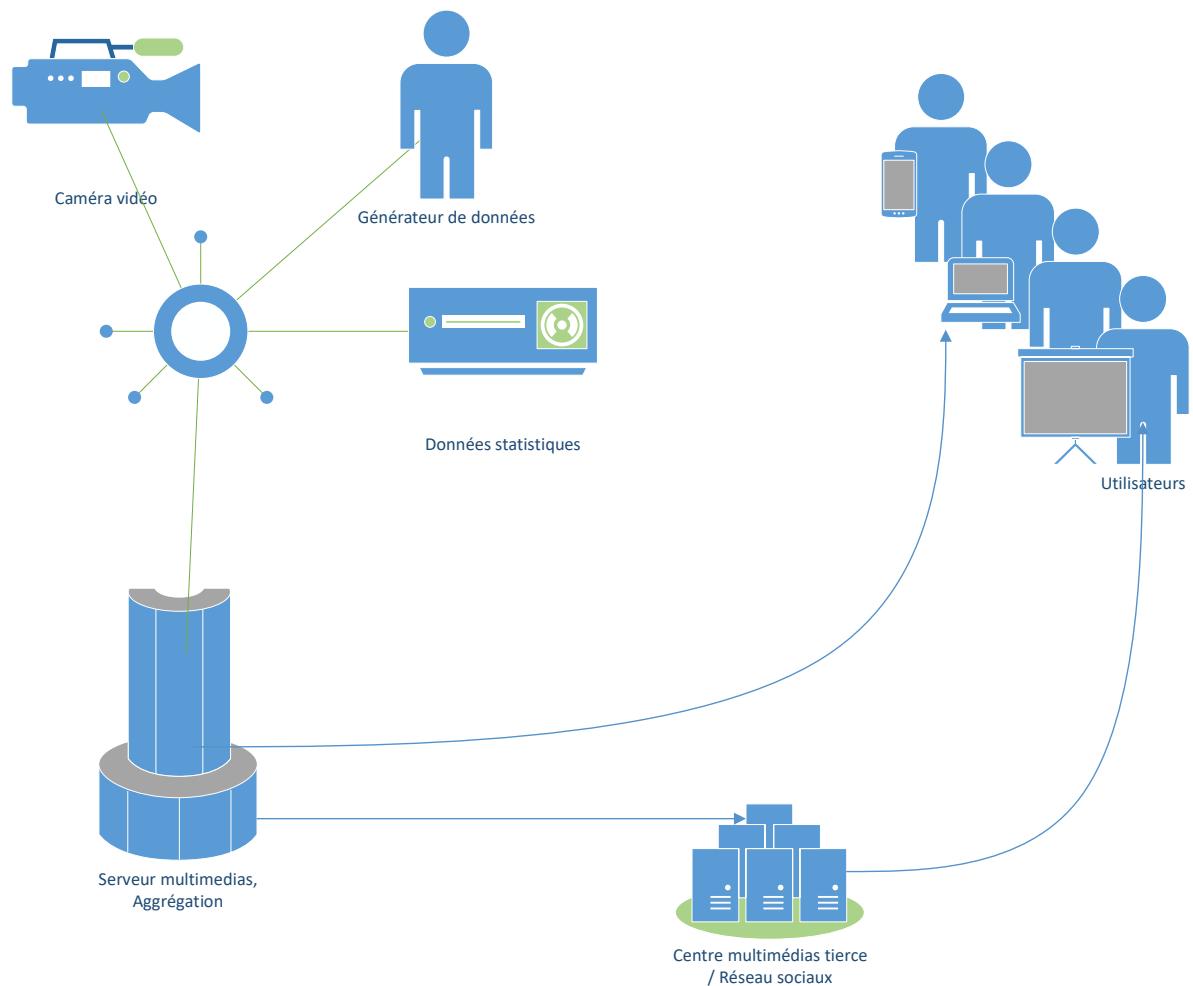


FIGURE 1.1 – Schémas de fonctionnement applicatif.

### 1.2.1 État des lieux du marché

Application	Description	Difficulté
<b>OpenCV</b>	<p>Une librairie sous licence BSD, initialement développée par Intel. C'est une librairie très puissante car elle permet notamment de traiter les vidéos, image par image. Cette api est très complète et propose presque tous les algorithmes utiles pour la "computer vision".</p> <p>Programmation : C, C++, Python, Matlab...</p> <p>Prix : Open-source.</p>	Difficile
<b>Wowza</b>	<p>Wowza est un serveur de contenu multimédia spécialisé dans la distribution de vidéos. La partie transcription est quant à elle principalement déléguée à des librairies tierces telles que FFmpeg.</p> <p>Programmation : XML/fichier de configuration.</p> <p>Prix : 65\$/mo (logiciel), 4500\$ (matérielle)</p>	Moyen
<b>Wowza transcoder</b>	<p>Est un module de Wowza et permet la transcription des vidéos. Ce module permet aussi d'ajouter du contenu lors de la retranscription du flux. Elle utilise une librairie propriétaire propre à Wowza.</p> <p>Programmation : Java</p> <p>Prix : Compris par Wowza server</p>	Difficile
<b>Wowza Clamp</b>	<p>Est plugin RestFull propriétaire. Ce module est probablement une des solutions les plus proches de ce que l'on souhaite. Il est possible d'ajouter des overlay statiques très facilement. Désavantage : restreint à Wowza, à première vue peu maintenu.</p> <p>Programmation : Json (RESTfull, Fichier)</p> <p>Prix : 300\$</p>	Facile
<b>FFmpeg &amp; Libav</b>	<p>Est le programme de référence open-source pour le transcoding. Cependant il est très limité dans l'intégration de contenu visuel. (sous-titre) Écrit en C.</p> <p>Programmation : C++/C, Python : (Avpy, ffmpy), Autres : wrapper</p>	Facile à Moyen

**Red5** Est un serveur de distribution open-source concurrent à Wowza. Il semblerait qu'il soit moins stable que Wowza pour une utilisation professionnelle.[24] Son avantage, outre sa gratuité, est sa modularité. Il est conseillé si l'on veut faire de la programmation Java. Pas de solution d'intégration de contenu connue.

Programmation : Java

Prix : open-source

**Adobe Media Server** Un des pionniers dans la diffusion de contenu vidéo. Actuellement en décrue. Leur format a encore beaucoup d'inertie et est passablement utilisé au niveau de leurs flux de transport. La fin de flash est annoncée pour 2020.[19] Pour ce qui est du serveur, celui-ci ne permet pas l'intégration de contenu type overlay sur les vidéos.

**Microsoft IIS Media Services** Serveur de distribution et de transcodage, ; Il est limité dans l'intégration de contenu visuel. Ce serveur ne doit plus être utilisé comme solution de service multimédia. Son support s'est terminé avec Windows Server 2008 R2. Quand bien même une "extended lifetime" est prévue jusque en 2020. [21]

**Mist server** Serveur de distribution multimédia et de transcription (non live). Très modulaire, rapide et à empreinte CPU-Mémoire faible. Idéal si l'on ne veut pas investir dans une solution hardware car il semble plus performant que ses concurrents de type serveur web.  
Language : C++ (Aussi le programme en lui-même.)  
Prix : Gratuit (OpenGL, sans DRM), Commercial : 2500\$

**Video Logix** Est un boîtier physique permettant de programmer les overlays directement sur un flux live. Plusieurs interfaces sont disponibles pour l'ajout des données dont des interfaces analogiques. L'avantage est un temps de latence très faible. Les désavantages sont l'encombrement physique, la nécessité de le posséder, les flux non numériques (à transcoder).  
 Prix : 1695\$

**GStreamer** Est une librairie spécialisée dans la manipulation de son et d'image. Il permet à l'origine d'afficher ou transcoder du contenu mais possède aussi quelques fonctions d'édition ainsi qu'une multitude de plugins tierces. [23]  
 Language : C  
 Prix : Open-source

**GStreamer** Est un module basé sur Gstreamer qui permet d'ajouter ?  
**QT Overlay** des overlays en temps réel en amont du logiciel Gstreamer. Attention, certains navigateurs web interdisent le site.  
 [22]  
 Prix 2500\$

**OpenCV** : OpenCV est une librairie open source à l'origine développée par Intel. Elle dispose de beaucoup de support de la part de la communauté ainsi que du monde universitaire ; Connue notamment pour ses fonctions de machine learning. Autre avantage ; Elle est disponible sur pratiquement toutes les plateformes même mobiles. Voir chapitre 2 pour plus de détails.

**Wowza** : Streaming Engine est un serveur multimédia permettant la diffusion de contenu vidéo en streaming. Son secteur de marché est principalement la vidéo à la demande ainsi que la vidéo en temps réel. Il dispose de plusieurs outils de retouche, conversion, compression et permet la compatibilité avec de multiples appareils. Des outils de loadbalancing géographique permettent à cet outil d'être utilisable en production à n'importe quelle échelle. Le serveur est construit sur Java ce qui lui donne une flexibilité supplémentaire au niveau du matériel sur lequel il est déployé. Wowza dispose d'un écosystème de plusieurs applications dans le but de cibler les clients auxquels ils sont destinés. Les solutions sont notamment : [ Facebook live streamer, cloud application based on Rest API, Le serveur complet, Un cross platform SDK pour le développement mobile, Un service CDN<sup>1</sup>...]. Le prix pour le serveur de contenu qui est le

1. content delivery network ou réseau de diffusion de contenu en français

service le moins contraignant pour le développement se situe entre 65\$ et 95\$ pas mois. La licence à vie est de 2000\$ ce qui peut être relativement cher si le produit est utilisé de manière accessoire.

**Wowza Transcoder** : Connu sous le nom de ‘Wowza streaming engine’ dans la proposition de leurs produits. C'est une api au noyau de Wowza qui nous permet de programmer son fonctionnement dont potentiellement des retouches images. Nous en discuterons plus en détails dans la partie chapitre 4.

**Wowza Clamp** : Wowza clamp est un plugin Wowza transcoder développé par Streaming-toolbox.com. Ce plugin possède un API RESTful permettant l'ajout d'overlays. Il dispose aussi d'une interface graphique web agissant comme client REST afin d'ajouter des overlays rapidement sans programmation au préalable. Ceci permettant donc de debugger l'application et de se rendre compte du rendu des différents éléments. Une explication de l'installation est disponible dans le chapitre 4. Cependant il semblerait qu'il y ait un soucis de compatibilité avec la nouvelle version de Wowza.

**FFmpeg** FFmpeg est une librairie de transcodage de vidéo très connue. Beaucoup d'applications open-sources utilisent ou tout du moins se basent sur cette librairie. Comme par exemple les très connus Blender, GStreamer, Mplayer (mvp) et VLC. Libav, fork de FFmpeg, est une librairie que l'on retrouvera souvent dans des programmes tierces ainsi que certaines distributions linux telles que la famille RHEL. VLC par exemple préférera la version Libav dans cet environnement par exemple.[17]. Forcément, la question se pose de quid de FFmpeg ou de Libav est plus performant ? En réalité, il n'y a pas de réponse tranchée à la question. Nombreux développements de qualité sont faits sur Libav et leur politique est plus sévère quant à l'intégration de code dans leur librairie. Ils privilégieront la qualité à la fonctionnalité. FFmpeg quant à eux font la course aux fonctionnalités, essayant d'avoir tous les derniers codec supportés, souvent, en lésinant sur la propreté et l'efficacité. Cependant beaucoup privilégieront ce dernier, possédant le plus de fonctionnalité. En effet toutes modifications et implémentations dans la librairie Libav sont copiées dans la librairie FFmpeg. Il y a donc peu de chance de trouver une fonction présente dans Libav et non dans FFmpeg [18].

**Mist server** Mist serveur est lui aussi un serveur de distribution et de retranscription multi-média. Il supporte passablement de codec et de transport-stream. Le serveur est très modulaire et a une empreinte faible ce qui lui permet de tourner sur les plus petits des appareils. Il est particulièrement recommandé si l'on souhaite ajouter ou modifier les fonctionnalités de celui-ci. Un avantage indéniable est qu'il peut tourner sur un Raspberry Pi contrairement à ses concurrents. Son efficacité fait sa force, sa simplicité peut être son défaut. En effet en parcourant des forums officieux, il semblerait que Wowza possède plus de fonctionnalités dans certains cas.[20]

## 2 OpenCV & Flask

### 2.1 Introduction

OpenCV est une librairie multiplate-forme écrite en C/C++ et disponible sur quasiment tous les environnements de développement, que ce soit les bien connus Linux et Windows mais aussi MacOS ainsi que les systèmes embarqués, RaspberryPI et consorts (ARM) ou les smartphones iOS et Android.[10]. C'est une librairie open source prévue pour le machine learning ainsi que pour le traitement d'images (computer vision)[8]. Du fait de sa licence BSD, elle est constamment améliorée par le marché et notamment par les plus grosses entreprises du secteur technologique comme Intel ou Google, ce qui en fait une des librairies leader dans ce domaine. En plus d'être disponible sur la quasi totalité des systèmes d'exploitations elle est également disponible dans multiple-langages que sont C++, C, Python, Java et MATLAB. Nous utiliserons ici la librairie pour Python qui est un langage certes moins optimisé que certains de ses concurrents mais qui à l'instar de Matlab permet de se concentrer sur le cœur du sujet. Il dispose en outre d'une bonne communauté et d'un bon support.

Une autre particularité d'OpenCV et ce qui le rend attractif à l'heure actuelle est sa compatibilité avec les processeurs Nvidia et leurs CUDA<sup>1</sup> Core[9]. Grâce aux avancées des GPUs, principalement dues aux jeux vidéo mais aussi et principalement aux crypto-monnaies les performances sont accrues d'un facteur 30x dans le pire des cas.

OpenCV est la librairie de référence dans le domaine de la "computer vision". Elle comporte plus de 500 fonctions applicables tant à l'imagerie médicale, qu'à la robotique ou à la sécurité. Elle contient aussi un module de machine-learning complet à usage général<sup>2</sup>.[25] Cette popularité en fait un élément incontournable en particulier pour des outils d'analyse de vidéo mais des fonctions d'éditions sont possibles. Le test suivant en fait preuve.

### 2.2 Fonctionnement général

L'exemple suivant démontre qu'il est possible de récupérer un flux vidéo temps réel, de le modifier à la volée avec peu de latence puis de l'afficher dans une autre application. Cette exemple récupère ici le flux "raw" d'une caméra branchée directement à l'ordinateur, par mesure

---

1. CUDA : Technologie dite GPGPU (General-Purpose Computing on Graphics Processing Units)

2. ML module

de simplicité. Tout flux vidéo peut être pris comme entrée. Il suffira par exemple de rediriger celui-ci sur un l'interface loopback avec FFmpeg par exemple.

```
# ffmpeg -re -i someInput -map 0 v -f v4l2 /dev/video0
```

Ceci requiert une interface kernel pour fonctionner. Dans cet exemple nous utilisons *v4l2loopback* ( voir [16] pour le dépôt ). Il y aurait bien entendu d'autres manières de faire, comme de récupérer le flux directement au niveau logiciel, mais ce qui intéresse ici est la preuve de concept. Le flux de sortie est visible dans une autre application pour démontrer que ce n'est pas l'affichage qui est modifié mais bien la vidéo. Le lecteur multimédia VLC ou un navigateur Web fera l'affaire. Pour ceci nous utiliserons la librairie Flask qui est l'un des serveurs web les plus utilisés sur Python. Celui-ci s'occupera de mettre les données sous la forme MJPEG<sup>3</sup> pour qu'elles soient compréhensibles par une visionneuse.

## 2.3 Environnement de développement de l'application test

Cette application a été développée en Python sur un environnement linux. Il faut potentiellement faire attention au support de CUDA dont les drivers ne sont pas d'office installés et où l'installation peut être compliquée sur certaines machines trop récentes. Les drivers Nvidia ne sont pas forcément bien supportés sur toutes les plateformes linux, Nvidia ne voulant pas développer de l'Open-Source. Si l'on souhaite des drivers open source, il est à regarder les drivers 'Nouveau' tout aussi performants que les drivers propriétaires à ce jour.

Compte tenu de toutes ces contraintes, l'utilisation du binding OpenCV Python est en faveur de la compatibilité du plus grand nombre. Une autre alternative aurait été Java qui est un langage très propre et académique, en particulier au niveau des structures données qu'il propose. Cependant, Python prend l'avantage dans sa facilité d'implémentation des sources externes grâce à l'implémentation de son fameux outil de gestion de paquet 'pip'. De ce fait les tests ont été faits par le biais du langage python. Il faut noter aussi que ce langage est très utilisé pour les back-end web. Nous tirons parti de ceci aussi grâce à la librairie Flask.

Pour le développement de ce petit exemple sont utilisés :

- IDE : Pycharm
- Language : Python 3.6
- OS : Linux, Fedora 27
- Carte graphique utilisée : Non
- Architecture : Intel x86
- Librairie : Flask ; VideoCamera ; cv2 (OpenCV 2)
- Visionneuse : Navigateur web Opera 53.0 edition developper.

Ci-dessous nous trouvons le code permettant au serveur Flask de construire la page web né-

<sup>3</sup>. MJPEG est un codec vidéo qui consiste en un flux d'images JPEG, d'où l'acronyme pour *Motion JPEG*

cessaire au visionnage du flux vidéo. Le chemin est référencé par les lignes de codes du type : @app.route('path') Nous aurons donc le flux sous le chemin "/video\_feed/" et la page HTML décoratrice à la racine. La fonction video\_feed() s'occupe d'envoyer le flux de bytes au navigateur distant au fur et à mesure que celui-ci est disponible.

```
# Usage:
# 1. Install Python dependencies: cv2, flask. (wish that pip install works like a charm)
# 2. Run "python main.py".
# 3. Navigate the browser to the local webpage.
from flask import Flask, render_template, Response
from camera import VideoCamera

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                  mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Listing 2.1 – Live streaming MJPEG with Flask, inspired from Miguel Grinberg. [11]

Dans la partie qui suit nous montrons comment récupérer un flux en entrée et le modifier. Ici nous récupérons la caméra directement attachée à l'interface loopback de l'ordinateur. Ce chemin correspond à l'architecture d'un système Linux. Pour Windows il faut utiliser la fonction cv2.VideoCapture(Integer) afin de choisir un périphérique vidéo d'entrée. OpenCV accepte également un descripteur de flux réseau RTSP<sup>4</sup> dans cet exemple. Le gros de la fonction correspond en résumé à une boucle while() qui récupère le flux image, les modifie, puis les dépose dans un objet qui sera utilisé par la fonction expliquée précédemment. Les informations sont ajoutées en temps réel grâce à un formulaire texte. Nous voyons par ceci que la modification du flux image par image est celle qui semble la plus simple. De plus, le délai ajouté au flux est quasi inexistant dans le cadre d'un flux "raw" en entrée. Pour un flux réseau cependant, il faut

<sup>4</sup>. Real Time Streaming Protocol

ajouter un petit délai pour prévoir une éventuelle perturbation du réseau, à moins que puissions assurer la qualité de service de celui-ci.

```
#!/bin/python3.6
import numpy as np
import cv2
import easygui
import threading

import transcoder
import flask_handler
from flask import Flask, render_template, Response

import sys

from form import form

global cap
cameraLocation = '/dev/video0'
#cameraLocation = 'rtsp://192.168.1.103/live1.sdp'
cap = cv2.VideoCapture(0)
#cap = cv2.VideoCapture(cameraLocation)

# Variables
text_info = 'Test_of_subtitle'

def text_box():
    global text_info
    while (True):
        test = easygui.enterbox(text_info, "Title", "Score_1_10")

t1 = form()
#t1 = threading.Thread(target=text_box, args=[])
t1.start()

while (True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = frame # cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Write text
    font = cv2.FONT_HERSHEY_COMPLEX

    cv2.putText(gray, t1.getText(),
               (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH) / 4), int(cap.get(cv2.
```

```
CAP_PROP_FRAME_HEIGHT) - 30)), font, 1,
(100, 200, 100), 2, cv2.LINE_AA)
```

```
## Broadcast mjpeg
transcoder.setframe(frame)

# Display the resulting frame
cv2.imshow('frame', gray)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
sys.exit(1)
```

Listing 2.2 – Live streaming MJPEG with Flask, récupération et modification des frames. [11]

La Figure 2.1 et Figure 2.2 ci-dessous nous montre la sortie graphique de l'application visionnée sur un navigateur web. La Figure 2.3 quant à elle nous montre le formulaire qui nous permet d'insérer le texte.

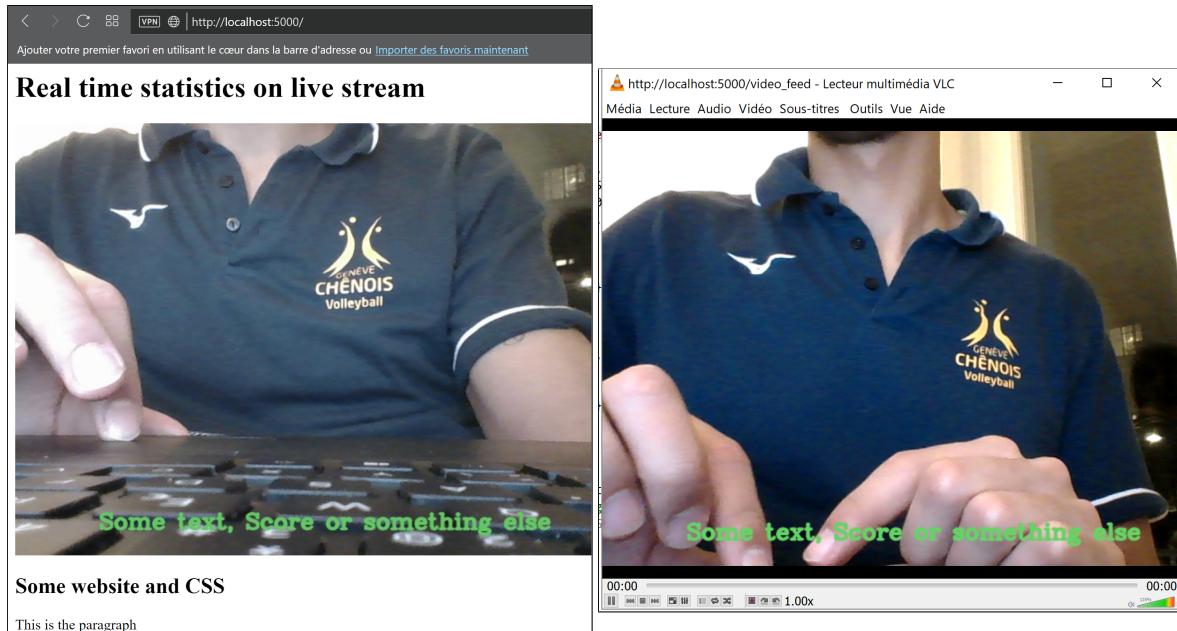


FIGURE 2.1. Affichage du stream avec Opéra

FIGURE 2.2. Affichage du stream avec VLC

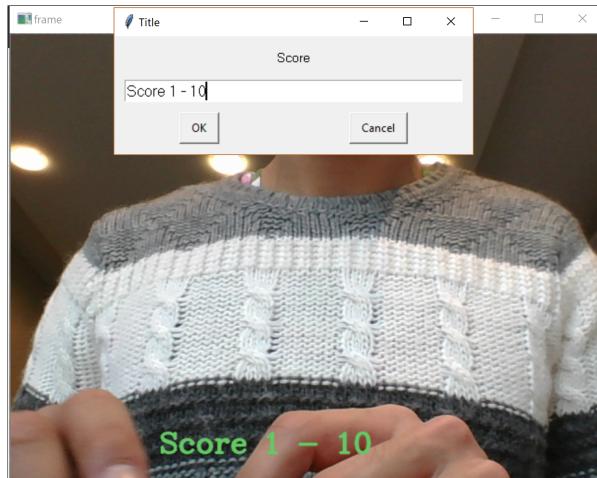


FIGURE 2.3. Intégration texte dans vidéo live, affichage GUI bureau

### 3 HbbTV

**H**ybrid Broadcast Broadband TV, plus connu sous l'acronyme HbbTV est un standard européen permettant le partage d'informations et de services en complément à un flux multimédia destiné à l'utilisateur final. Il a été inventé en France en 2006.

Il fait partie des outils ou protocoles TV OTT, acronyme signifiant télévision Over The Top ou services par contournement en français et qui définissent les contenus ne passant pas par le bouquet proposé par l'opérateur internet / télévision. C'est donc l'antithèse de la télévision linéaire, mode de consommation traditionnel. [1]

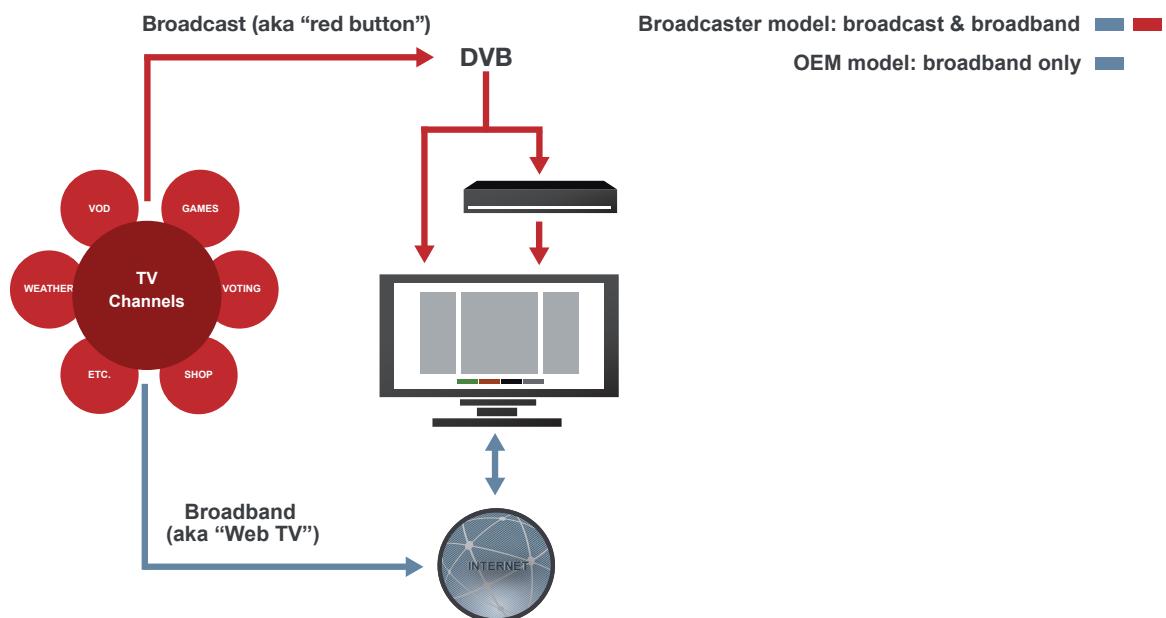


FIGURE 3.1 – HbbTV Broadband vs Broadcast

L'objectif du consortium est de créer un seul standard permettant d'acheminer du contenu broadcast et broadband à travers une seule interface de type web. Le but est donc d'empêcher l'émergence de multiples standards propriétaires ainsi que les désavantages qui en incombent.[3] Les services proposés peuvent être infinis, passant de la vidéo à la demande, se dissociant donc du contenu vidéo de base jusqu'au quiz interactif lors d'une émission télévisée.

## 3.1 Fonctionnement général

### 3.1.1 Utilisation, le cas de la Suisse.

Dès 2011, les suisses ont à disposition la HBBTV. Cependant, elle n'est pas disponible sur tous les médiums d'information. En effet, elle est encore indisponible chez certains cablo-operateurs, sur la TNT ainsi que sur Swisscom TV et ceci représente une grande majorité des utilisateurs.[12]. En 2017, la régie publicitaire Admeira, filiale de la SSR Ringier et Swisscom lance un projet pilote pour incorporer des fonctionnalités publicitaires couplées au programme regardé ainsi qu'aux publicités proposées.[13] En parcourant ce qu'il se fait au niveau international, il semblerait que les régies publicitaires ainsi que les constructeurs soient les premiers intéressés. Ils permettent ainsi gentiment de démocratiser cette technologie. Des constructeurs de télévision parviennent même à proposer des contenus ciblés en analysant le contenu vidéo de l'utilisateur.

## 3.2 Environnement de développement

Begins a section.

### 3.2.1 Opera TV Emulator

L'émulateur Opera TV rend possible le développement de contenu HTML5 et CE-HTML pour différents appareils que sont les Smart TV, lecteurs Blue-ray, Box et aussi les ordinateurs.[?] La position sur le marché du leader Opera rend quasi indispensable cet environnement de développement, du moins pour les tests.

L'environnement de développement se caractérise dans une machine virtuelle tournant sur le bien connu, VirtualBox. Il permet de s'abstraire de l'accès physique à une machine/TV ainsi que de rendre plus prédictif les protocoles de tests. Il dispose de deux interfaces graphiques, la première étant le flux vidéo de la sortie standard qui propose une interface "TV-like" et dont la sortie sera celle de notre application développée. Sur la figure [3.2(e)] ci-après nous voyons la "landingpage" de l'émulateur opera. Celui-ci nous permet d'entrer le lien de l'application HbbTV que nous souhaitons grâce à un clavier virtuel. Cependant cette interface n'est pas très ergonomique et gère mal l'interaction clavier-souris.

Une autre interface beaucoup plus efficace nous est proposée. Il s'agit de l'interface web, disponible par défaut sur le port 5555. Elle dispose notamment de fonctions utiles au debugging (fps counter, paint regions, terminal event, javascript event, performance monitor...) [3.3], et un certain nombre de paramètres de fonctionnement de l'émulateur [3.2(a) 3.2(b)] et du SDK ainsi que les différentes compatibilités assurées. Cette interface propose aussi une télécommande virtuelle [3.2(f)] nous permettant de tester les différentes réactions aux événements boutons.

### 3.3 Environnement de production

#### 3.3.1 The Opera hybrid TV option

The Opera hybrid TV option est un basée sur leur propre Opera Devices SDK (Standard Development Kit) qui est une technologie déjà bien répandue sur le marché avec plusieurs millions de set-up box dans le monde. Ce kit de développement implémente plusieurs modules de compatibilité tels que (OIPF, CE-HTML, CEA-2014...) et de ce fait supporte toutes les applications HbbTV. Ils se targuent aussi d'avoir une technologie éprouvée permettant des performances et intégration supérieures à ce que propose le marché.[3]

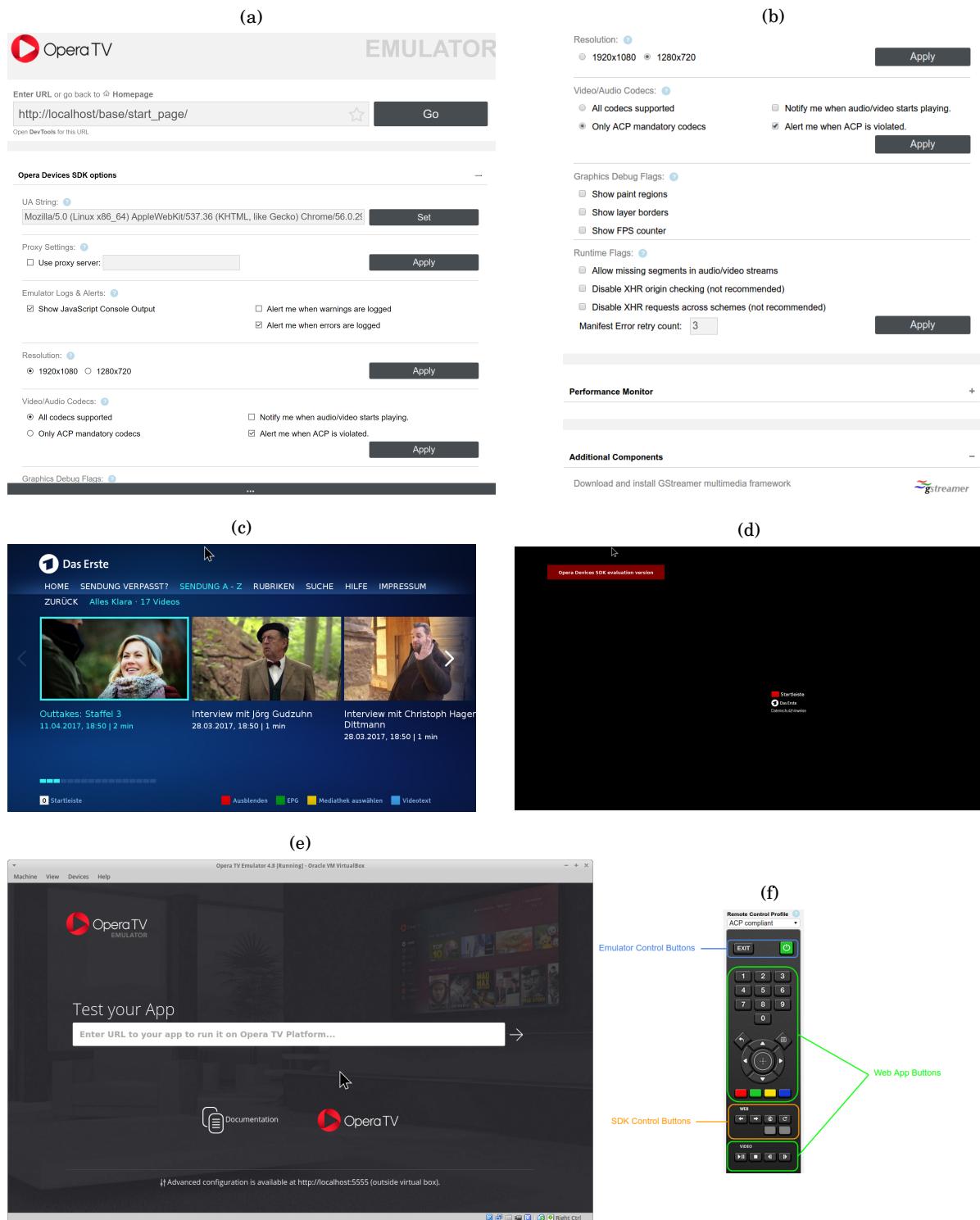
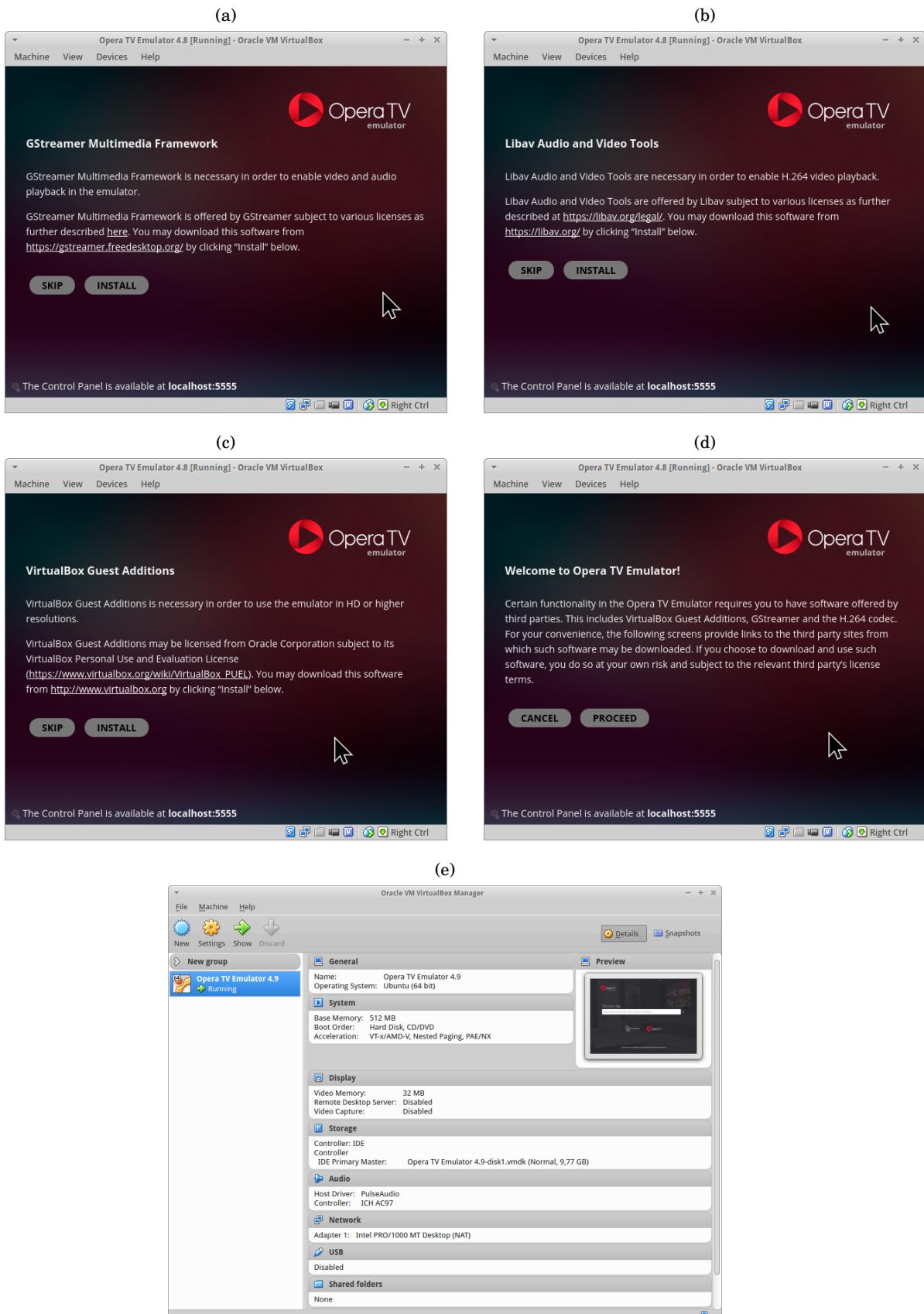


FIGURE 3.2. Interface utilisateur.



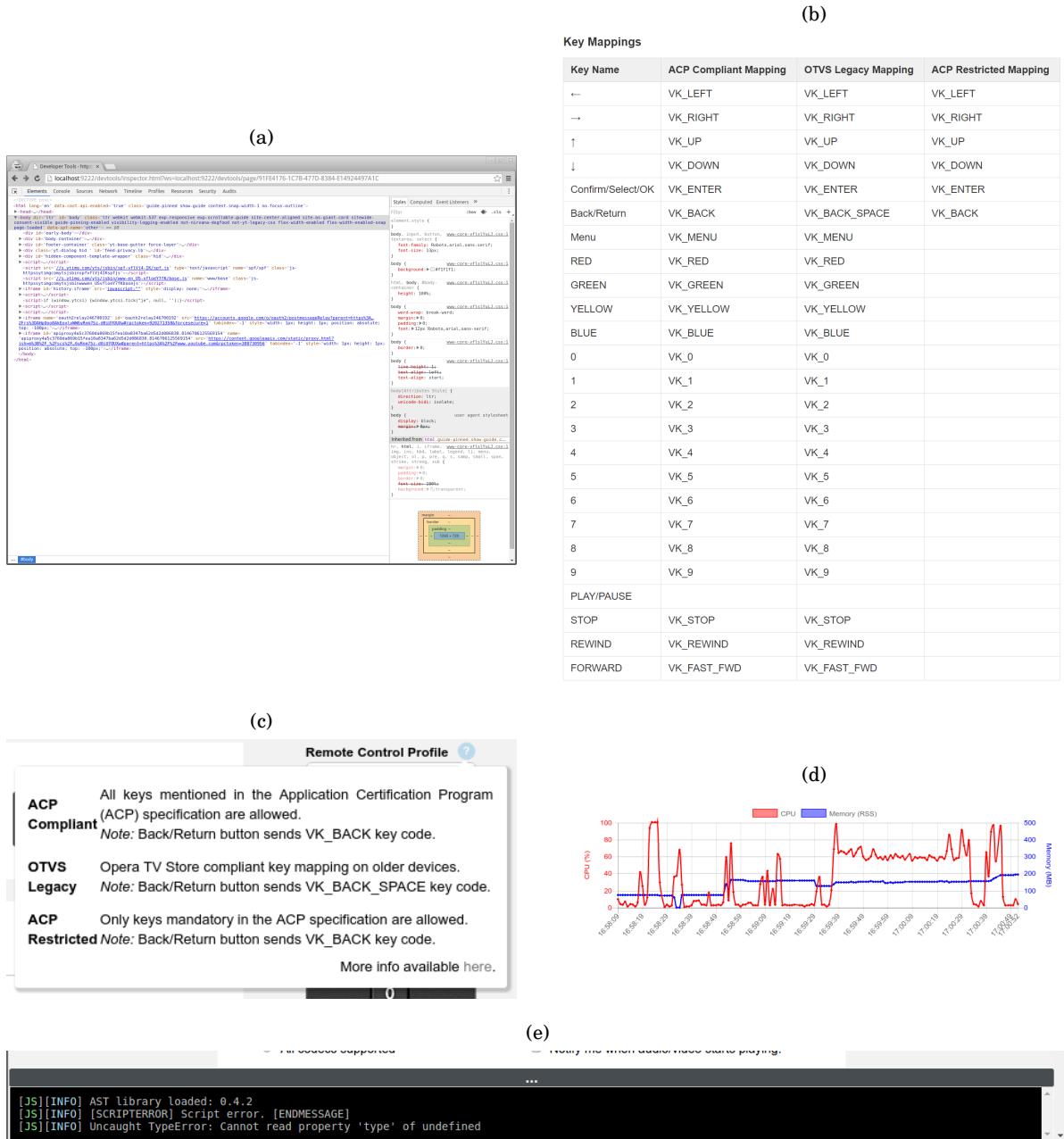


FIGURE 3.4. Outils développeur.

## 4 Wowza

OpenCV est une librairie multiplate-forme<sup>1</sup> et open source prévue pour le machine learning ainsi que pour le traitement d'image (computer vision)[8]. Du fait de sa licence BSD, cette librairie est constamment améliorée par le marché et notamment par les plus grosses entreprises du secteur technologique, ce qui en fait une des librairies leader dans ce domaine. En plus d'être disponible sur la quasi totalité des systèmes d'exploitations du marché elle est également disponible dans multiple- langages que sont C++, C, Python, Java and MATLAB à ce jour et sans compter évidemment les différents binding. Nous utiliserons ici la librairie pour Python qui est un langage certes moins optimisé que certains de ses concurrents mais qui à l'instar de Matlab permet de se concentrer sur le cœur du sujet. Il dispose en outre d'une bonne communauté et d'un bon support.

Une autre particularité d'OpenCV et ce qui le rend attractif à l'heure actuelle est sa compatibilité avec les processeurs Nvidia et leurs CUDA<sup>2</sup> Core[9]. Grâce aux avancées des GPUs à l'heure actuelle, principalement due au jeux vidéo mais aussi et principalement au crypto-monnaie les performances sont accrue d'un facteur 30x dans le pire des cas.

### 4.1 Fonctionnement général

OPENCV

### 4.2 Environnement de développement de l'application test

pycharm opencv, livrairies, flask...

```
# Usage:  
# 1. Install Python dependencies: cv2, flask. (wish that pip install works like a charm)  
# 2. Run "python main.py".  
# 3. Navigate the browser to the local webpage.  
from flask import Flask, render_template, Response  
from camera import VideoCamera  
  
app = Flask(__name__)
```

---

1. MacOS, IOS, Android, Windows, Linuxes,...  
2. CUDA : Technologie dite GPGPU (General-Purpose Computing on Graphics Processing Units)

```
@app.route('/')
def index():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                  mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```

Listing 4.1 – Live streaming MJPEG with Flask, inspired from Miguel Grinberg. [11]

## 4.3 Environnement de production

### 4.3.1 The Opera hybrid TV option

# 5 *Produits sur le marché*

## 5.1 Fonctionnement général

OPENCV

## 5.2 Environnement de développement de l'application test

pycharm opencv, livrairies, flask...

```
# Usage:  
# 1. Install Python dependencies: cv2, flask. (wish that pip install works like a charm)  
# 2. Run "python main.py".  
# 3. Navigate the browser to the local webpage.  
from flask import Flask, render_template, Response  
from camera import VideoCamera  
  
app = Flask(__name__)  
  
@app.route('/')  
def index():  
    return render_template('index.html')  
  
def gen(camera):  
    while True:  
        frame = camera.get_frame()  
        yield (b'--frame\r\n'+  
              b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')  
  
@app.route('/video_feed')  
def video_feed():  
    return Response(gen(VideoCamera()),  
                  mimetype='multipart/x-mixed-replace; boundary=frame')  
  
if __name__ == '__main__':
```

```
app.run(host='0.0.0.0', debug=True)
```

Listing 5.1 – Live streaming MJPEG with Flask, inspired from Miguel Grinberg. [11]

## 5.3 Environnement de production

### 5.3.1 Linux & FFMPEG

## *6 Conclusion*

P our conlure,



## *A Annexe A*

**A**<sup>nnexe</sup>



## Bibliographie

- [1] WIKIPEDIA, *Over-the-top media services*, [https://en.wikipedia.org/wiki/Over-the-top\\_media\\_services](https://en.wikipedia.org/wiki/Over-the-top_media_services), 28 January 2018, at 13 :23.
- [2] OPERA PRESS, *Deliver seamless entertainment experiences*, The Opera hybrid TV option, 1 February 2018.,[www.opera.com/media/b2b/tv/Opera-TV-Emulator.pdf](http://www.opera.com/media/b2b/tv/Opera-TV-Emulator.pdf), pp. 1.
- [3] OPERA PRESS, *Opera TV Emulator*, The Opera TV emulator introduction, 2 February 2018., pp. 1.
- [4] s  
J. D. MASUCCI AND J. W. SCHIEFELBEIN, *The rhd6 mutation of arabidopsis thaliana alters root-hair initiation through an auxin- and ethylene-associated process*, Plant. Physiol., 106 (1994), pp. 1335–1346.
- [5] R. PAYNE AND C. GRIERSON, *A theoretical model for rop localisation by auxin in arabidopsis root hair cells*, PLoS ONE, 4 (2009), p. e8337.  
doi :10.1371/journal.pone.0008337.
- [6] S. RIGAS, G. DEBROSSES, K. HARALAMPIDIS, F. VICENTE-ANGULO, K. A. FELDMAN, A. GRABOV, L. DOLAN, AND P. HATZPOULOS, *Trh1 encodes a potassium transporter required for tip growth in arabidopsis root hairs*, The Plant Cell, 13 (2001), pp. 139–151.
- [7] HBBTV APPLICATION DAS ERSTE ,*Example of application with video content* , <http://hbbtv.daserste.de/index.php>, Main application, 30 January 2018.
- [8] OPENCV OFFICIAL ,*About OpenCV, Support Question* , <https://opencv.org/about.html>, <https://opencv.org/platforms/>, Open Source Computer Vision Library, 4 February 2018.
- [9] OPENCV OFFICIAL ,*About OpenCV, CUDA support* , <https://opencv.org/platforms/cuda.html>, Performances, 4 February 2018.
- [10] OPENCV PLATFORMS ,*Platforms* , <https://opencv.org/platforms/>, Cuda - Android - iOS, 01.03.2018.

- [11] MIGUEL GRINBERG ,*Video Streaming with Flask* , <https://blog.miguelgrinberg.com/post/video-streaming-with-flask>, Code, October 20 2014.
- [12] RTS INFO, *Un nouveau service de TV interactive est lancé en Suisse*, <https://www.rts.ch/info/suisse/4710599-un-nouveau-service-de-tv-interactive-est-lance-en-suisse.html>, Article, 05 mars 2013
- [13] SWISSCOM CHRONIQUE, *HBBTV : Le nouveau teletext inaugure la TV de demain..*, <https://www.swisscom.ch/fr/chroniques/technologie/hbbtv-television-interactive.html#T=3c4e5434-dbb8-4243-84d9-8fa965326573&TS=0TC9cL8VsfTsieADGzM6CBZFFw4xoVlCooYGR5LYSkk>, Article, 28 septembre 2017
- [14] COMMUNICATION D'ENTREPRISE SSR, *Swisscom propose dès aujourd'hui ce successeur multimédia du télétexte avec de nombreuses fonctionnalités intéressantes.*, [https://rtsr.ch/a\\_la\\_une/swisscom-tv-2-0-propose-la-hbbtv-sur-les-chaines-de-la-ssr-srg/](https://rtsr.ch/a_la_une/swisscom-tv-2-0-propose-la-hbbtv-sur-les-chaines-de-la-ssr-srg/), Article, 14.04
- [15] [GITHUB] JANTEKB , *Exemple de code et de configuration pour les outils streamingtoolbox.com.*, <https://github.com/jantekb/streamtoolbox-examples/>, Code, Nov 7 2015
- [16] [GITHUB] UMLAEUTE , *v4l2loopback - a kernel module to create V4L2 loopback devices.*, <https://github.com/umlaeute/v4l2loopback>, Code, Fev. 26 2018
- [17] LIBAV OFFICIAL , *wiki : Using libav\**, [https://trac.ffmpeg.org/wiki/Using%20libav\\*](https://trac.ffmpeg.org/wiki/Using%20libav*), API, 10 nov. 2017
- [18] [GITHUB] NIKLAS HAAS , *FFmpeg versus Libav*, <https://github.com/haasn/mpvhq-old/wiki/FFmpeg-versus-Libav>, Article, Mar 22 2015
- [19] ADOBE OFFICIAL , *Flash & The Future of Interactive Content*, <https://theblog.adobe.com/adobe-flash-update/>, Article, 07-25-2017
- [20] MIST OFFICIAL , *Comparison from Mist website (bias)*, <https://mistserver.org/comparison>, Article, December 2017.
- [21] WINDOWS OFFICIAL , *Search product lifecycle*, <https://support.microsoft.com/en-us/lifecycle/>, Site, -
- [22] RIDGERUN.COM , *Gstreamer QT Overlay*, [https://developer.ridgerun.com/wiki/index.php?title=Gstreamer\\_QT\\_Overlay#Network\\_streaming](https://developer.ridgerun.com/wiki/index.php?title=Gstreamer_QT_Overlay#Network_streaming), Site, 23 November 2017
- [23] GSTREAMER OFFICIAL , *Gstreamer QT Overlay*, <https://gstreamer.freedesktop.org/>, code : git://anongit.freedesktop.org/gstreamer/gstreamer, Site, -

[24] OODLESTECHNOLOGIES , *Red5 Vs Wowza Head To Head* , <http://www.oodlestechnologies.com/blogs/Red5-Vs-Wowza-Head-To-Head>, Site, 12 déc. 2017

[25] ADRIAN KAEHLER & GARY BRADSKI , *Learning OpenCV 3* , O'Realy, Book, 2017

