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

Remerciement

J^e remercie.

Table des matières

	Page
Liste des tableaux	vii
Table des figures	ix
1 Introduction	1
1.1 Préambule	1
1.2 Cahier des charges de l'application	2
1.2.1 État des lieux du marché	2
2 OpenCV & Flask	9
2.1 Introduction	9
2.2 Fonctionnement général	9
2.3 Environnement de développement de l'application test	10
3 HBBTV	15
3.1 Fonctionnement général	16
3.1.1 Utilisation, le cas de la Suisse.	16
3.2 Environnement de développement	16
3.2.1 Opera TV Emulator ou Vewd Emulator	16
3.3 Environnement de production	17
3.3.1 The Opera hybrid TV option	17
4 Wowza	21
4.1 Wowza transcoder and overlay	21
4.2 Clamp - Module Streamtoolbox.com	23
5 Produits annexes	29
5.1 HTML to JPEG	29
5.2 Emulateurs Hbbtv	29
6 Conclusion	31

6.1 Discussion	31
6.2 Conclusion	32
A Annexe A	33
Bibliographie	35

Liste des tableaux

TABLE	Page
	vii

Table des figures

FIGURE	Page
1.1 Schémas de fonctionnement applicatif	2
2.1 The easy Flask website MJPEG. - Visualisation Opera	13
2.2 The easy Flask website MJPEG - Visualisation VLC.	13
2.3 Burn text on live stream, display on GUI	14
3.1 HbbTV Broadband vs Broadcast	15
3.2 Opera emulator user interface	18
3.3 Opera emulator installation	19
3.4 Opera emulator devlopper tools	20
4.1 Wowza - Flowchart appareils	25
4.2 Wowza génération du flux d'entrée avec OBS.	26
4.3 Paramétrage du module Wowza Clamp.	26
4.4 Wowza - Affichage non supporté.	27
4.5 Paramétrage du module pour Wowza serveur.	27
4.6 Paramétrage du module pour Wowza serveur.	28

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 coûteux 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.

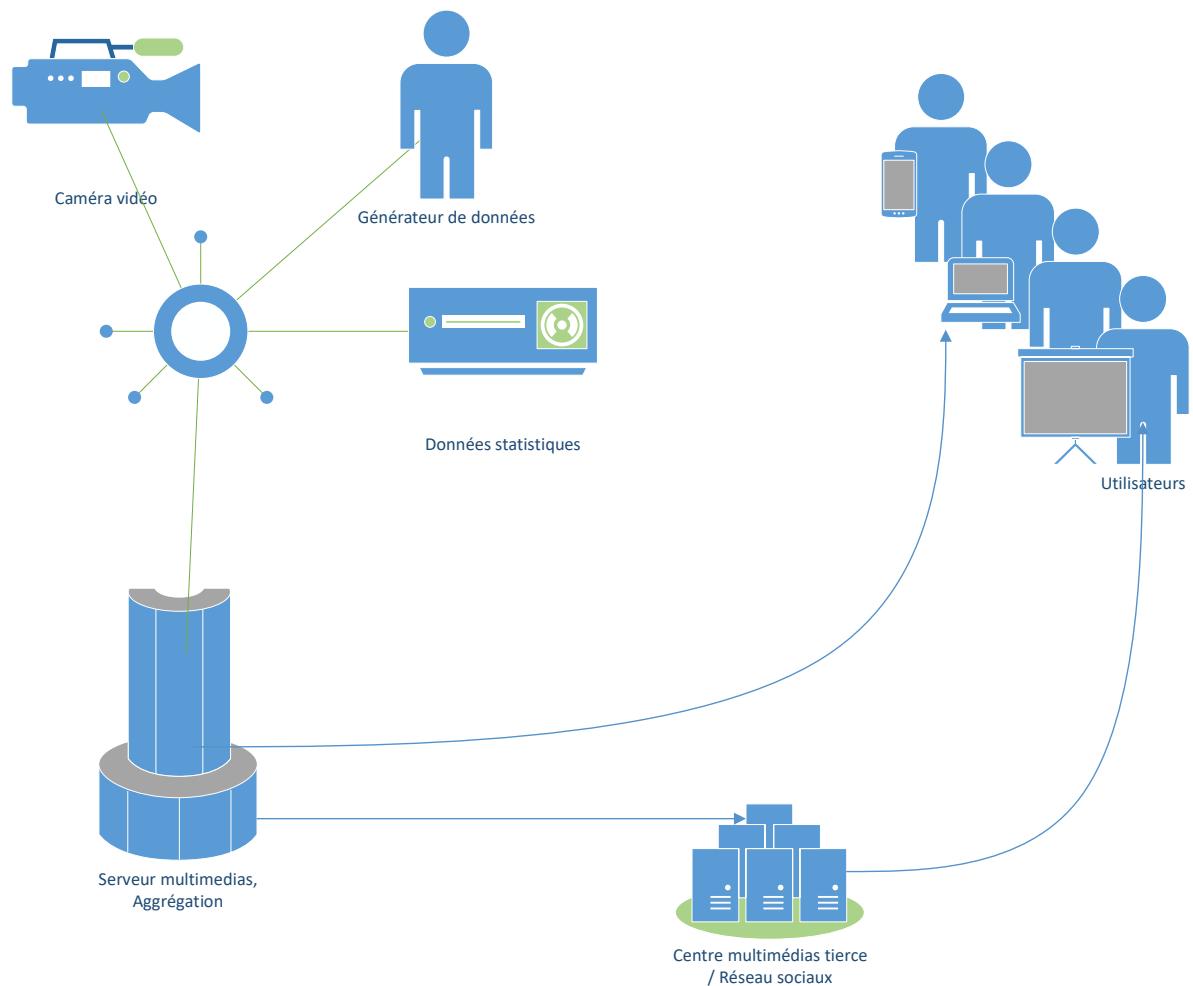


FIGURE 1.1 – Schémas de fonctionnement applicatif.

1.2.1 État des lieux du marché

Application	Description	Difficulté
OpenCV	<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
Wowza	<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
Wowza transcoder	<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
Wowza Clamp	<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
FFmpeg & Libav	<p>Est le programme de référence open-source pour le transcodage. 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.[25] 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.[20] 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 jusqu'en 2020. [22]

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.
Langage : 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).
Moyen

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. [24]
Difficile
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.
[23] Possède une très petite communauté.
Prix 2500\$

Microsoft Selon des représentants de Microsoft, ils pourraient à terme proposer API d'édition de flux vidéo dans un futur proche. A l'heure rien n'existe si ce n'est du transcodage (DirectX - DirectShow)
-

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 CDN¹...]. Le prix pour le serveur de contenu qui est le service le moins contraignant pour le développement se situe entre 65\$ et 95\$ par 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 débugger 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.[18]. 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 codecs 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 [19].

Mist server Mist serveur est lui aussi un serveur de distribution et de retranscription multimédia. Il supporte passablement de codec et de transport-stream. Le serveur est très modulaire

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

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.[21]

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'exploitation 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 certain 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 CUDA¹ 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 général².[27] 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. Cet exemple récupère ici le flux "raw" d'une caméra branchée directement à l'ordinateur, par mesure de

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

2. ML module

simplicité. Tout flux vidéo peut être pris comme entrée. Il suffira par exemple de rediriger celui-ci sur 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 [17] 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 MJPEG³ 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-

³. 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.

```

1 # Usage:
2 # 1. Install Python dependencies: cv2, flask. (wish that pip install works like a charm)
3 # 2. Run "python main.py".
4 # 3. Navigate the browser to the local webpage.
5 from flask import Flask, render_template, Response
6 from camera import VideoCamera
7
8 app = Flask(__name__)
9
10 @app.route('/')
11 def index():
12     return render_template('index.html')
13
14 def gen(camera):
15     while True:
16         frame = camera.get_frame()
17         yield (b'--frame\r\n'
18                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')
19
20 @app.route('/video_feed')
21 def video_feed():
22     return Response(gen(VideoCamera()),
23                     mimetype='multipart/x-mixed-replace; boundary=frame')
24
25 if __name__ == '__main__':
26     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 RTSP⁴ 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

⁴. 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.

```

1 #!/bin/python3.6
2 import numpy as np
3 import cv2
4 import easygui
5 import threading
6
7 import transcoder
8 import flask_handler
9 from flask import Flask, render_template, Response
10
11 import sys
12
13 from form import form
14
15 global cap
16 cameraLocation = '/dev/video0'
17 #cameraLocation = 'rtsp://192.168.1.103/live1.sdp'
18 cap = cv2.VideoCapture(0)
19 #cap = cv2.VideoCapture(cameraLocation)
20
21 # Variables
22 text_info = 'Test_of_subtitle'
23
24 def text_box():
25     global text_info
26     while (True):
27         test = easygui.enterbox(text_info, "Title", "Score_1_10")
28
29
30 t1 = form()
31 #t1 = threading.Thread(target=text_box, args=[])
32 t1.start()
33
34 while (True):
35     # Capture frame-by-frame
36     ret, frame = cap.read()
37
38     # Our operations on the frame come here
39     gray = frame # cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
40
41     # Write text
42     font = cv2.FONT_HERSHEY_COMPLEX
43
44     cv2.putText(gray, t1.getText(),
45                 (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH) / 4), int(cap.get(cv2.

```

```

46   CAP_PROP_FRAME_HEIGHT) - 30)), font, 1,
47   (100, 200, 100), 2, cv2.LINE_AA)

48

49  ## Broadcast mjpeg
50  transcoder.setframe(frame)

51

52

53

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

58

59 # When everything done, release the capture
60 cap.release()
61 cv2.destroyAllWindows()
62 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.

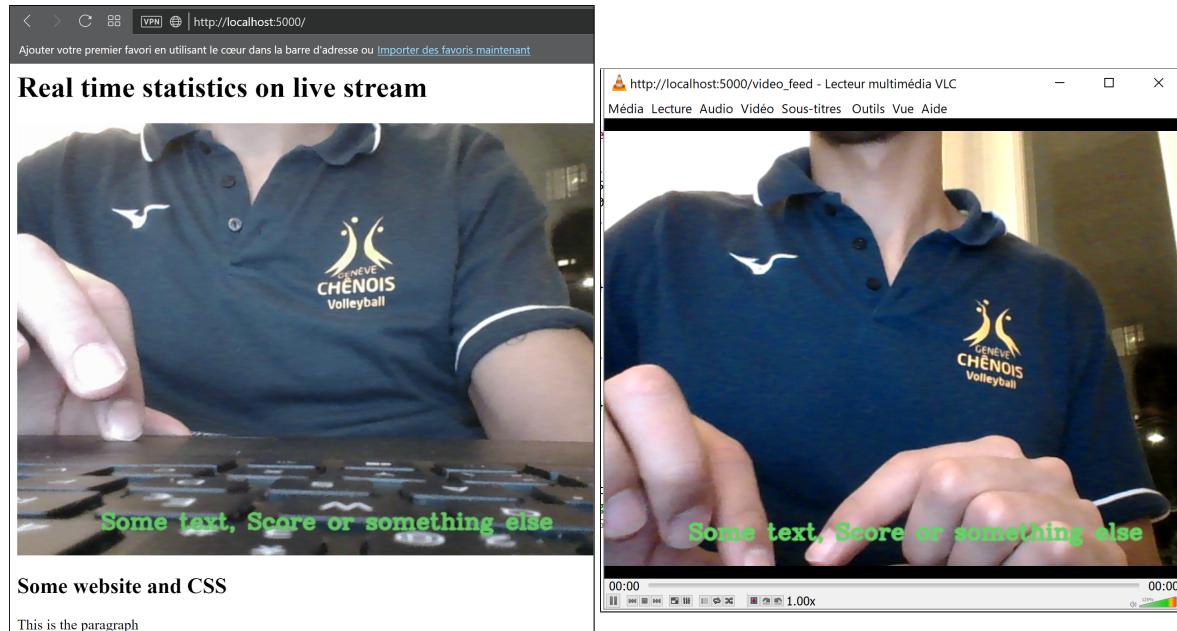


FIGURE 2.1. Affichage du stream avec
Opéra

FIGURE 2.2. Affichage du stream avec VLC

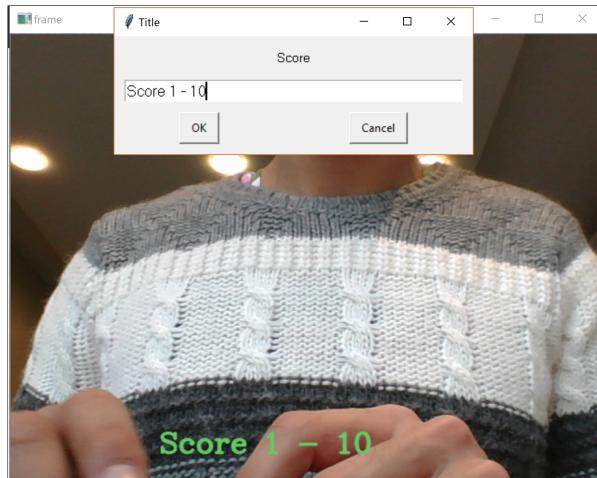


FIGURE 2.3. Intégration texte dans vidéo live, affichage GUI bureau

3 HbbTV

Hybrid 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]

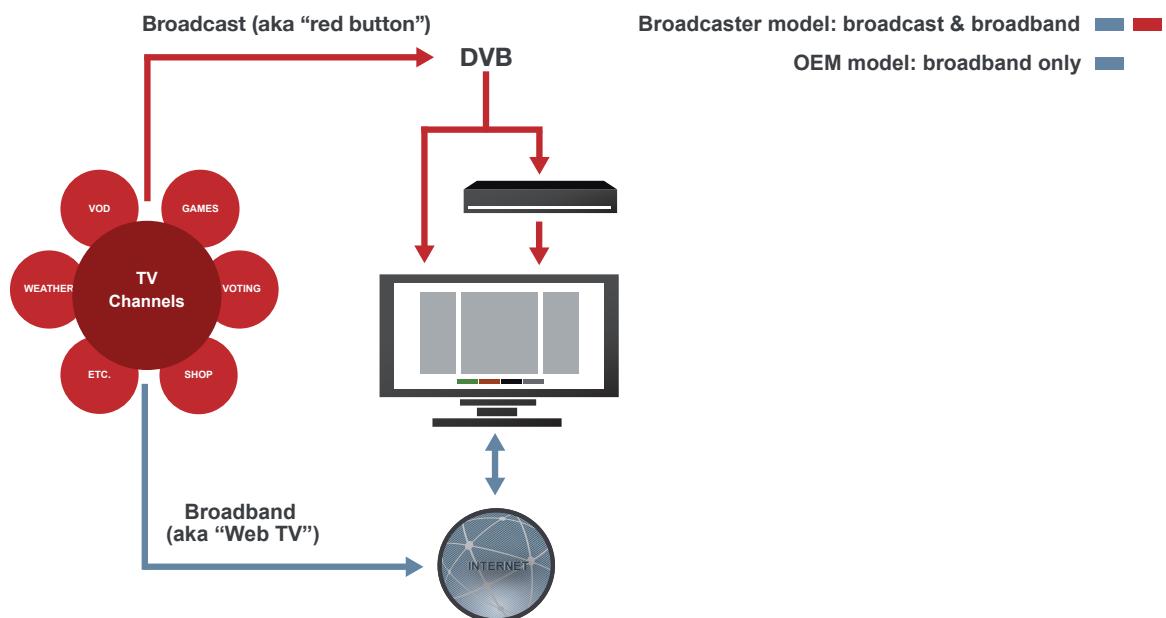


FIGURE 3.1 – HbbTV Broadband vs Broadcast

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.[2] 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] Dans la pratique, il semblerait que ceci ne fonctionnent pas aussi bien. En effet, la fonction HbbTV peut être activée sur les nouvelles box Swisscom, mais celui-ci ne semble pas fonctionner alors que le protocol est activé sur les chaines Suisse uniquement. Ce test a été réalisé avec la dernière Internet-TV box. Les tests sur le cablo-opérateur Genevois Naxoo quant à eux fonctionne. 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

3.2.1 Opera TV Emulator ou Vewd Emulator

L'émulateur Opera TV, récemment renommée Vewd Emulator, rend possible le développement de contenu applicatif en HTML5 et CE-HTML pour différents appareils que sont les Smart TV, lecteurs Blue-ray, Box et aussi les ordinateurs.[3] La position sur le marché du leader Opera rend quasi indispensable cet environnement de développement, du moins pour les tests. L'écosystème Vewd est disponible sur la quasi totalité des télévisions du marché(Samsung, Sony, Verizon, TiVo). Si ce n'est pas au niveau de l'OS, un palliatif logiciel est apporté (LG par exemple). Les box TV comportent aussi cet écosystème, Swisscom TV en faisant partie.

L'environnement de développement-test 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

L’Opera hybrid TV option est une technologie 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.[2] En somme, cette technologie qui s’assure que la HbbTV est compatible au sein de l’environnement Opera. Il prendra notamment en compte les différents canaux de transmissions que sont le broadcast, la broadband ou les deux.

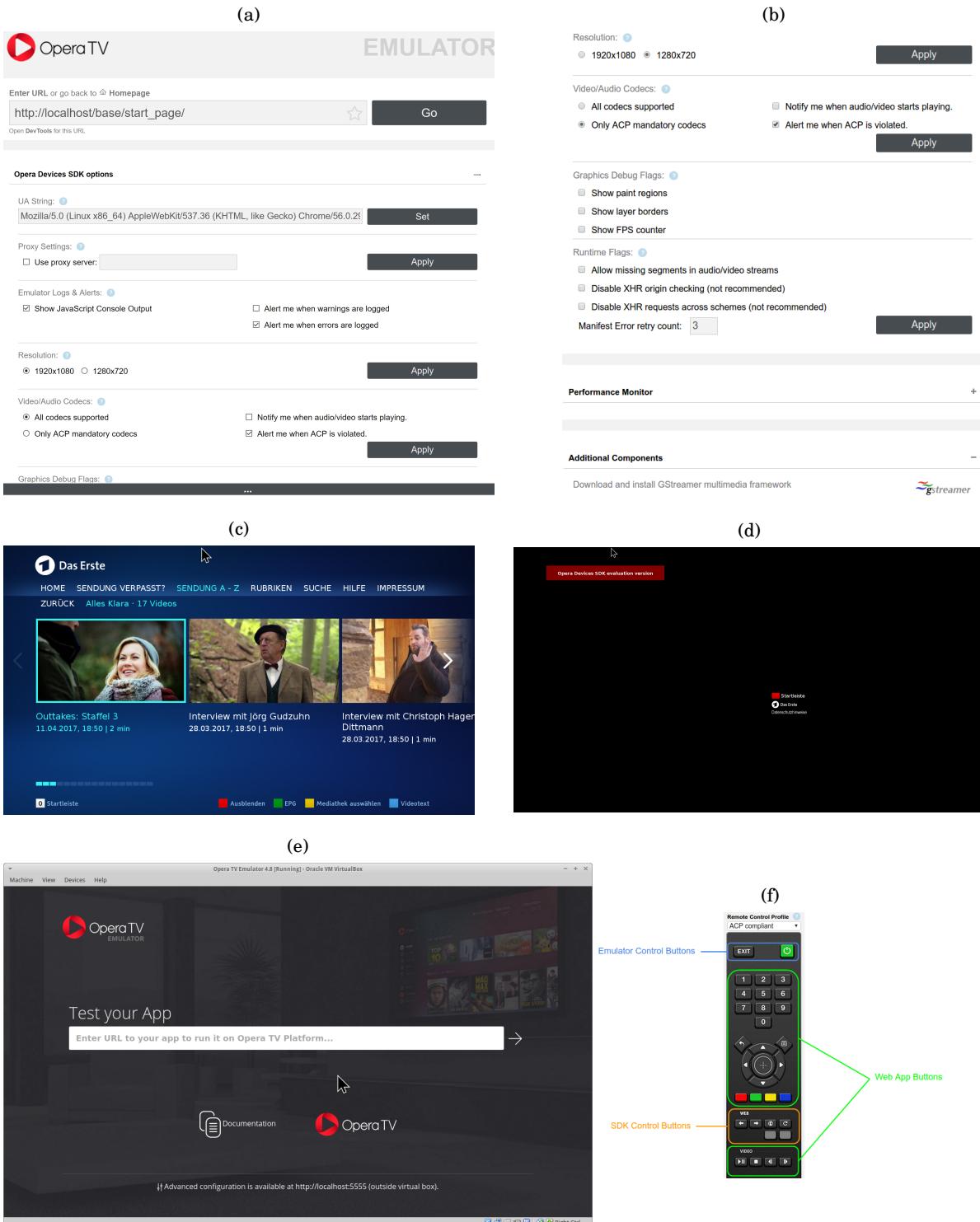
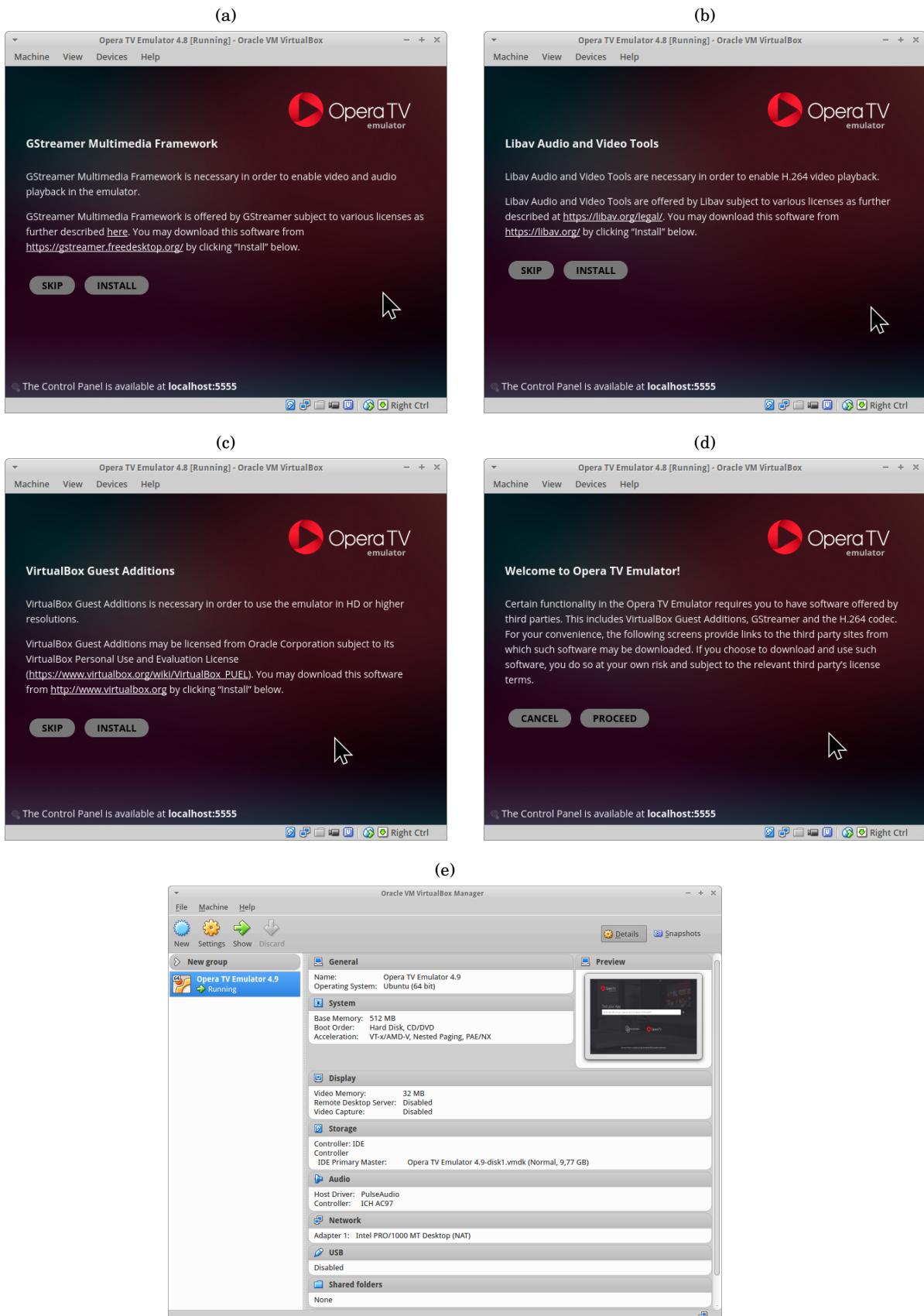


FIGURE 3.2. Opera TV - Interface graphique de test et paramétrage.



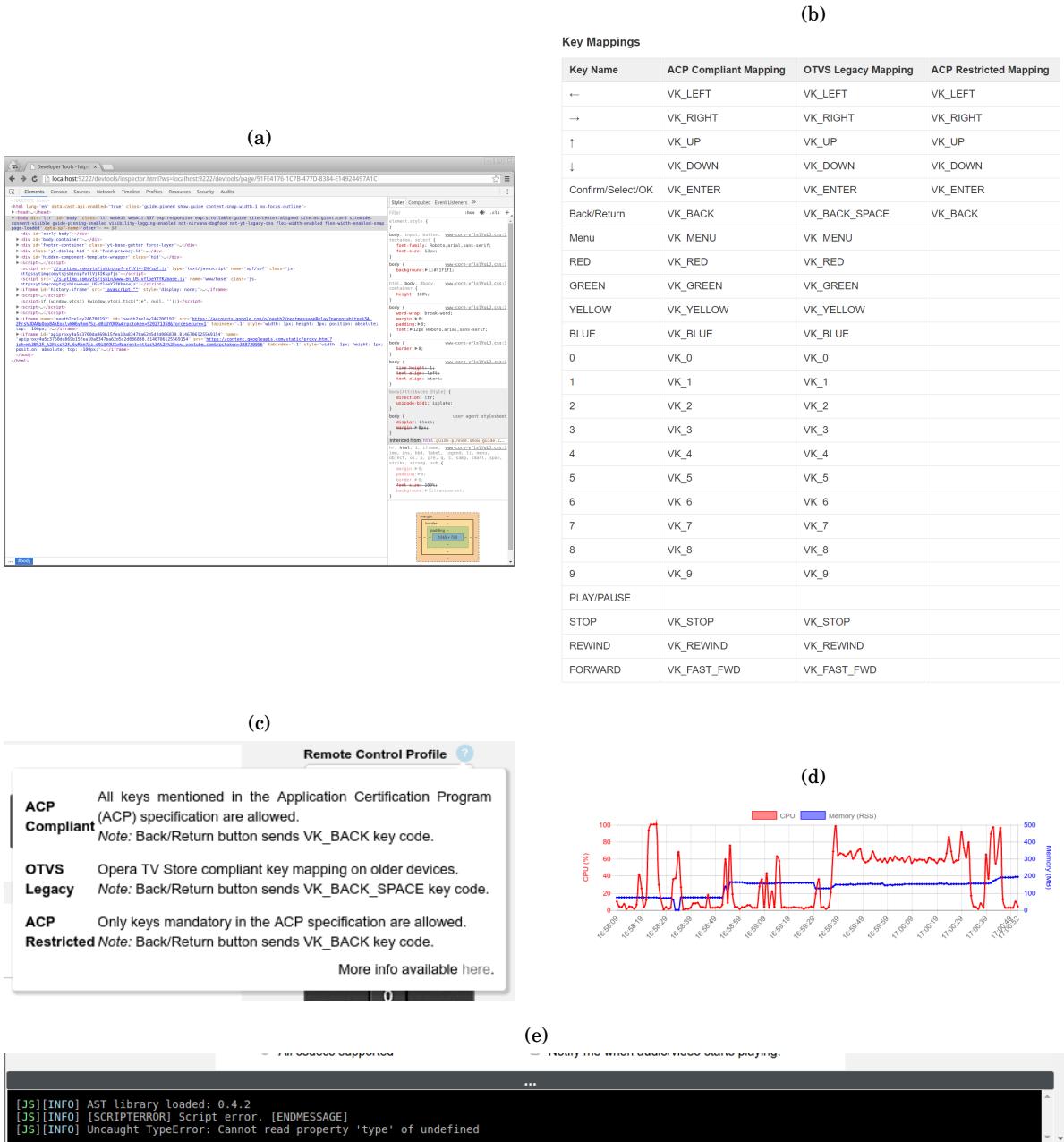


FIGURE 3.4. Outils développeur, dont notamment les key mappings.

4 Wowza

4.1 Wowza transcoder and overlay

Wowza est un serveur de distribution multimédia. Il permet aussi de transcoder les flux vidéo cependant les utilisateurs du service semblent préférer à l'heure actuelle la librairie FFmpeg pour une bonne partie des transcodage, Wowza ne supportant que l'encodage en H.264, VP8 et VP9. Pour ce qui est du projet, Wowza possède une API de transcodage comportant certaines fonctions d'overlays. Leur documentation propose un module exemple qui ajoute du texte dans un flux transcodé par Wowza. Contrairement à la facilité d'implémentation des overlays OpenCV, ici nous nous retrouvons avec plus d'un millier de lignes de code. Cette implémentation propose deux endroits où ajouter les overlays. La première est lors du décodage. Cette méthode à l'avantage d'être plus performante car les transformations sont ajoutée une seule fois à la lecture du flux. Suivant le format, il se peut que le rendu soit détérioré lors de la mise à l'échelle et l'encodage du flux. La deuxième méthode consiste à modifier la vidéo lors de l'encodage. Elle dispose d'un rendu de meilleure qualité car les overlays ajouté sont de la même résolution que le flux sortant. Il faudra cependant beaucoup plus de ressources pour traiter les vidéos si plusieurs flux de sorties sont paramétrés. Il est possible aussi d'avoir des overlays différents suivant les types ou résolutions de flux de sortie.[26]

Le test du serveur Wowza a été fait avec la version développeur de Wowza, donc la licence est disponible pendant 6 mois.

- Language : Java 8
- OS : Windows 10
- Technologie cartes graphiques : Nvidia Cuda + Intel iGPU
- Architecture : Intel x86, i7-6700
- Version : Wowza 4.7
- Visionneuse : VLC

Le code ci-dessous est un extrait de la classe Java OverlayImage du module exemple, disponible sur le site même de Wowza¹. Cette fonction est appelée sur chaque image présente dans le

1. <https://www.wowza.com/downloads/forums/transcoderoverlayexamplefiles/TranscoderOverlayExampleFiles.zip>

buffer. Nous remarquons à la ligne 11 qu'il a été décidé d'utiliser la classe Graphics2D standard de Java pour retoucher les images. Cette granularité nous permet de retoucher le flux de la même manière qu'avec OpenCV mais cette fois-ci en utilisant les outils Java standard.

Dans la section suivante 4.2, nous profiterons pour expliquer comment installer des modules dans le serveur Wowza.

```

1   /**
2    * Returns a byte[] buffer of the image and all its children
3    * drawn ontop.
4    * @param scaled - the amount to scaled the image by from the
5    * oringal.
6    * @return a byte[] array of the image.
7    */
8   public byte[] GetBuffer(double scaled)
9   {
10
11     byte[] retVal = GetTempBuffer(scaled);
12     BufferedImage bImage=GetBufferedImage(scaled);
13
14     Graphics2D g = bImage.createGraphics();
15     //clear the old image out
16     g.setComposite(AlphaComposite.Clear);
17     g.fillRect(0, 0, GetWidth(scaled), GetHeight(scaled));
18     g.setComposite(AlphaComposite.SrcOver);
19
20
21     if(currentOpacity > 0.0) //don't bother if invisible
22     {
23       g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
24                         RenderingHints.VALUE_INTERPOLATION_BILINEAR);
25       g.setRenderingHint(RenderingHints.KEY_RENDERING,
26                         RenderingHints.VALUE_RENDER_QUALITY);
27       g.setRenderingHint(RenderingHints.
28                         KEY_TEXT_ANTIALIASING, RenderingHints.
29                         VALUE_TEXT_ANTIALIAS_ON);
30
31     if(myImage !=null)
32     {
33       g.drawImage(myImage.imageBuf, (int)(myImage.
34           xOffset*scaled), (int)(myImage.yOffset*scaled),
35           GetWidth(scaled), GetHeight(scaled), null);

```

```

26      }
27      for(OverlayImage img : childrenList)
28      {
29          img.GetBuffer(scaled);
30          float opac = (float) (img.currentOpacity/100.0);
31          AlphaComposite composite = AlphaComposite.
32              getInstance(AlphaComposite.SRC_OVER, opac);
33          g.setComposite(composite);
34          g.drawImage(img.GetBufferedImage(scaled), img.
35              GetxPos(scaled), img.GetyPos(scaled), img.
36              GetWidth(scaled), img.GetHeight(scaled), null);
37      }
38      if(myText !=null)
39      {
40          g.setColor(myText.fontColor);
41          Font newFont = myText.font.deriveFont((float) (
42              myText.font.getSize()*scaled));
43          g.setFont(newFont);
44          g.drawString(myText.text,(int)(myText.xPos*scaled
45              ), (int)(myText.yPos*scaled));
46      }
47      transferImage(bImage,retVal,currentOpacity);
48
49      return retVal;
50  }

```

Listing 4.1 – Extrait du code d'ajout d'overlay officiel de Wowza. Lignes: 216-261[26]

4.2 Clamp - Module Streamtoolbox.com

Clamp est un module transcoder de Wowza du même type que celui montré plus haut. Il permet l'ajout d'overlays sur tout flux vidéo transcodé ar Wowza. Il est développé par Streamtoolbox.com qui est une compagnie développant des outils de retouche vidéo à la volée et de statistiques pour Wowza. Toute leurs solutions sont payantes. La solution qui nous intéresse, Clamp, a un prix unique de 300\$ par serveur.[15] Nous profitons de la présentation de ce module pour montrer comment celui-ci est installé sur le serveur, car elle n'est pas forcément très évidente de prime abord.

Dans la pratique, ce module se place entre le décodage et l'encodage comme le fait le module de la section 4.1. Il instancie un serveur HTTP qui permet non seulement l'utilisation de l'API RESTful via des POST mais aussi celle de l'interface graphique permettant de générer "à la souris" les requêtes REST (voir Figure 4.3). Beaucoup de paramètres sont disponibles dont trois fonctions principales, l'ajout de texte, l'ajout d'image, l'ajout de dates. Cette api possède également des fonctions de transition et d'ombres pour les objets.

Pour streamer le flux vidéo dans le serveur Wowza, nous avons utilisé le logiciel OBS (Open Broadcaster) qui permet de transcoder des flux vidéo de multiples sources vers un serveur multimédia. D'autres méthodes simples sont possibles, dont l'utilisation de VLC.

Installation :

- Dans les services Windows, après l'installation de Wowza, lancer le service Wowza Streaming Engine ainsi que Wowza Streaming Engine Manager.
- Se rendre sur la page Web port 8088 et configurer ce qui est demandé.
- Dans l'onglet application, créer une nouvelle "live application" et lui donner un nom.
- La configuration du serveur Wowza nécessite l'édition du fichier module C:/ProgramFiles(x86)/WowzaMediaSystems/WowzaStreamingEngine4.7.4/conf/nom-app dans le dossier correspondant au nom de l'application.
- Éditer le fichier de la Figure 4.5 et de la Figure 4.6. Nous noterons l'ajout de l'objet XML Module qui permet d'instancier Clamp dans la live application. Puis son paramétrage en Figure 4.6. "Clamp.caption.source.file" permet de spécifier un fichier au lieu de l'interface REST, nous pouvons également spécifier le port et d'autres paramètres comme la calibration qui ajoute une grille de test par-dessus la vidéo. Le détail de ses paramètres est spécifié dans la documentation Clamp.
- Finalement, il faut ajouter le code du module Clamp sous la forme d'un .jar dans le dossier "/lib/" de Wowza.
- A des fin de debugging il peut être intéressant de lancer le fichier "startup.bat" se trouvant dans le dossier "/bin/" de Wowza. Celui-ci nous permet de voir la sortie standard de l'application et les éventuelles erreurs d'exécution.

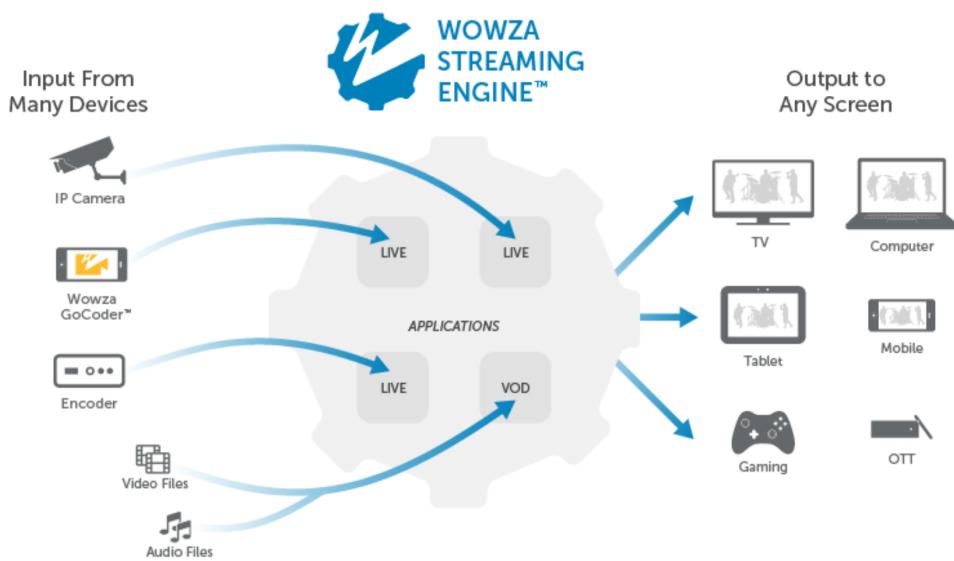
Bien qu'en allant suivit les recommandations du développeur quand a l'installation du module. Lors du visionnage du flux de données, les modifications ne semblent pas être prises en compte. Le problème viendrait du fait que le flux n'est pas transcodé, et que donc les overlays ne peuvent être inséré dans celui-ci. Les lecteurs test inséré dans l'interface graphique de Wowza semblent confirmer cette thèse, ils ne fonctionnent tout simplement pas (voir Figure 4.4). Le module de transcodage Wowza est assez récent et est différent de celui présenté sur le site constructeur de Clamp. Il n'est donc pas possible de suivre leurs recommandations. Dans tous les cas, il n'est pas normal qu'aucun message d'erreur ne soit transmis. Que ce soit au niveau de l'interface graphique de Clamp, de Wowza et même au niveau de la console qui affirme prendre en compte les nouveaux overlays et les ajouter.

Video Workflow Overview

← Previous Next →

Wowza Streaming Engine software receives video from your encoder, camera, or media files and streams out to any device. To do this, you'll create applications in the Wowza Streaming Engine server to do live and video on demand (VOD) streaming.

The following diagram shows a common video workflow:



Other things you can do with Wowza Streaming Engine software include Internet Radio (SHOUTcast) re-streaming, origin/edge configuration, monitoring, performance tuning, and more!

← Previous Next →

FIGURE 4.1. Workflow Wowza - Flowchart

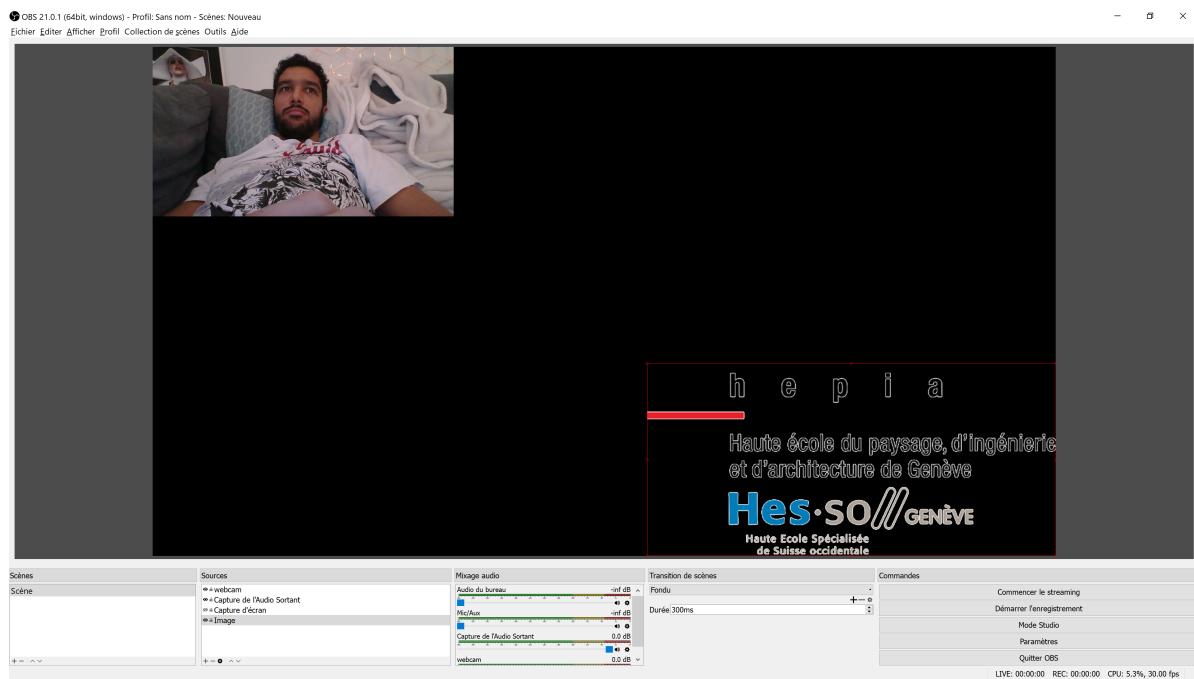


FIGURE 4.2. Génération du flux d'entrée de Wowza avec OBS, Open Broadcaster.

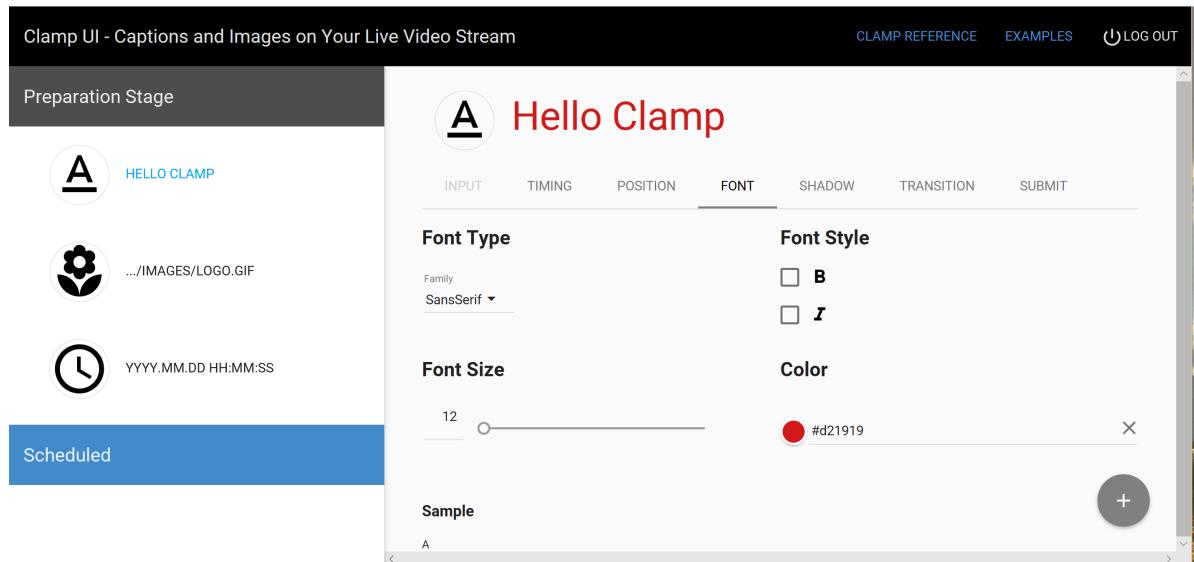


FIGURE 4.3. Paramétrage et test du module Wowza Clamp.

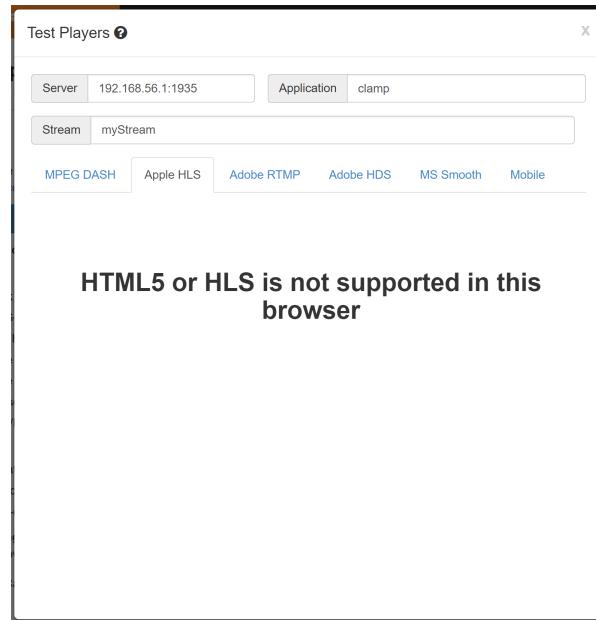


FIGURE 4.4. Affichage non supporté - Interface test.

```
<Modules>
  <Module>
    <Name>Clamp</Name>
    <Description>Dynamic captions on your streams</Description>
    <Class>com.streamtoolbox.Clamp</Class>
  </Module>
```

FIGURE 4.5. Paramétrage du module pour Wowza serveur.

```

<Properties>
  <Property>
    <Name>pushPublishMapPath</Name>
    <Value>${com.wowza.wms.context.VHostConfigHome}/conf/${com.wowza.wms.context.Application}/PushPublishMap.txt</Value>
    <Type>String</Type>
  </Property>
  <Property>
    <Name>clamp.caption.source.file</Name>
    <Value>c:\Users\fires\Documents\GitHub\RealTimeStatisticsOnLiveStream\app\clamp\clamp.json</Value>
  </Property>
  <Property>
    <Name>clamp.ui.enabled</Name>
    <Value>true</Value>
  </Property>
  <Property>
    <Name>clamp.ui.port</Name>
    <Value>8013</Value>
  </Property>
  <Property>
    <Name>clamp.rest.api.authentication</Name>
    <Value>false</Value>
  </Property>
  <Property>
    <Name>clamp.calibrate</Name>
    <Value>true</Value>
  </Property>
  <Property>
    <Name>clamp.encode.source</Name>
    <Value>true</Value>
  </Property>
  <Property>
    <Name>clamp.rest.api.authentication</Name>
    <Value>false</Value>
  </Property>
  <!-- Uncomment this to switch off the calibration grid -->
  <!--Property><Name>clamp.calibrate</Name><Value>false</Value></Property-->
</Properties>

```

FIGURE 4.6. Paramétrage du module pour Wowza serveur.

5 Produits annexes

5.1 HTML to JPEG

Dans l'idée de construire des overlays de qualité, ainsi que d'utiliser ce qui est déjà existant dans le monde du web, il peut être intéressant de considérer la solution HTML pour la création d'overlay. Par exemple une solution populaire pour cette fonction est Imgkit¹ qui est un wrapper de Wkhtmltopdf utilisant le moteur de rendu Webkit, utilisé notamment pour Chrome, Opera et Safari. Beaucoup d'éléments du moteur de rendu sont paramétrables, les dimensions, les polices, etc...

Ces librairies permettent de transformer des pages Web en fichiers images ou PDF. L'énorme avantage étant l'aspect "responsive" des pages web. L'affichage dans différents rapports, 16/9, 4/3, etc... Sont grandement facilité. Il serait également possible d'utiliser cet overlay dans un autre cas de figure et notamment HbbTV.

5.2 Emulateurs Hbbtv

Il existe d'autres emulateurs test pour HbbTV. Certains sont moins lourd et peuvent être pratiques pour des petits tests mais son moins fiable étant donnée qu'ils ne sont que des simulateurs. Notamment : FireHbbTV est un module firefox permettant de tester les applications au sein même du navigateur Firefox. Il utilise cependant le moteur de rendu Firefox qui n'est pas forcément le même que sur les télévisions et en tout cas pas le même qu'opéra qui utilise webkit.

1. <https://pypi.python.org/pypi/imgkit/0.1.1>

6 Conclusion

6.1 Discussion

La liste d'outils et de méthodes disponibles pour l'édition de flux vidéo est très grande. Certains de ces outils ne permettent pas directement de modifier un flux en temps réels et sont donc à exclure dans notre cas. D'autres outils ne semblent pas, ou pas encore bénéficier d'un gros support et il est impossible de dire si ceux-ci seront maintenus par la suite. En pensant particulièrement au module Clamp pour Wowza. Il n'est pas dit non plus que le serveur Wowza continuera à être utilisé dans le futur bien que les fournisseurs multimédias considèrent de plus en plus ce service. Même si cette solution se démocratise, le transcodage peut aussi se faire sur leurs solutions hardware auquel cas les modules de transcoding et overlay logiciel ne sont plus adéquats. La solution de modification du flux à la source est vraisemblablement la solution la plus efficace dans le cas d'ajout d'overlays. Pour commencer, cette solution supprime les temps de latence que peuvent amener le réseau Internet. Il n'est donc pas nécessaire de synchroniser la vidéo avec les flux de données ce qui peut s'avérer une tâche très complexe. Les données sont ajoutées au moment présent de l'action ou de la décision de l'opérateur.

Nous avons vu que dans l'exemple avec OpenCV, la latence dans l'ajout d'overlay était quasi inexistante. Ensuite, cette solution nous permet un peu plus de modularité dans le projet. En effet, il serait toujours possible de changer de serveur de streaming. L'application serait donc compatible avec tout serveur de diffusion multimédia. Dans le cas de streaming pour un réseau social par exemple, le temps de latence serait réduit la connections étant directe et sans transcoding. Dans le cas où nous voudrions changer d'appareil par exemple, il serait possible d'utiliser une Raspberry Pi (ARM) avec le serveur "Mist" par exemple. Il est donc intéressant de garder l'aspect serveur multimédia et édition de vidéo découpé. L'effet indésirable pourrait être les ressources nécessaires sur le lieu de l'enregistrement de la vidéo. Cependant l'édition de vidéo image par image n'est pas extrêmement gourmande en calcul. L'exemple d'OpenCV nous montre qu'avec un affichage HD et un processeur de 2015, les ressources utilisées sont de moins de 10%. Il faudrait également ajouter une étape de transcoding pour envoyer le flux compressé sur le réseau mais de toute façon ceci doit être fait peu importe que l'on ajoute un overlay ou non. Une autre solution consisterait à envoyer directement depuis la caméra le flux sur un serveur externe, nous avons vu qu'il était extrêmement simple de récupérer un flux RTSP avec la bibliothèque d'OpenCV. Ainsi

tout semble converger vers l'utilisation de la solution OpenCV avec les avantages suivant :

- Rapidité d'implémentation, car beaucoup de fonctions.
- Compatibilité de la librairie sur la majorité des plateformes.
- Compatibilité de la librairie sur la majorité des architectures.
- Grand soutien de la part du marché et des grandes entreprises, en constant développement.
- Beaucoup de littérature.
- Possibilité d'utiliser les fonctions de machine learning.
- Gratuit - Open source.

Les outils de transcodage que nous pourrions utiliser avec la librairie OpenCV sont plus ou moins équivalent. Et permettent tous de transcoder la vidéo. Cependant, la librairie FFmpeg ou Libav est présente sur tout les repositories Linux avec de beaucoup de support car une multitude d'applications utilisent cette librairie. Nous avons vu que des bindings existent entre la librairie et les différents langages de programmation, l'implémentation du transcodage n'est donc pas une tâche très ardue.

La seconde solution intéressante est l'utilisation d'HbbTV. La construction de cette application nécessite des connaissances en développement web et est complètement différente de la solution que nous avons proposée plus haut. Celle-ci ne nécessite pas de transcodage et utilise tout simplement les fonctions disponibles sur les téléviseurs pour afficher des informations en dessus de l'image. Cette solution est très voisine au développement d'application Vewd pour téléviseur. Les mêmes technologies étant utilisées. C'est simplement la mise en forme et le support de l'application qui est sensiblement différentes. Il serait possible donc ici de faire une pierre deux coups et de construire une application Vewd compatible HbbTV. La question de la pénétration du marché reste cependant encore en suspens. HbbTV doit être utilisé sur un canal Télévisé, en ne prenant donc pas en compte les flux de télévision IP. L'infrastructure pour l'ajout de statistiques ou d'overlay sur les canaux de télévision standard sont déjà très présent, en pensant notamment au Tennis. Peu d'utilisateur encore à ce jour utilisent la fonction HbbTV quand celle-ci leur est disponible. L'ajout récemment de publicité via cette fonction ou encore le fait qu'elle est désactivée d'office sur beaucoup de téléviseur ne facilitent pas la chose. En voyant grand nous pourrions imaginer développer une application Vewd compatible HbbTV. Ceci allant dans le sens de la vidéo à la demande qui se démocratise de plus en plus, en particulier vers le public plus jeune. Les flux vidéo pouvant passer via internet, une plus grande flexibilité d'accès au contenu proposé serait possible. Couplé à cette solution nous pourrions imaginer un flux vidéo plus primitif, consistant en l'impression de rendu HTML - CSS directement sur la vidéo. Et donc en utilisant les fonctions d'overlays discuté dans ce document, pour les médias ne supportant pas HbbTV.

6.2 Conclusion

A Annexe A

A^{nnexe}

Bibliographie

- [1] WIKIPEDIA, *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, 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/, Article, 14.04
- [15] STREAMTOOLBOX.COM OFFICIAL, *lamp for Wowza Streaming Engine*, <https://streamtoolbox.com/clamp/>, Site, -
- [16] [GITHUB] JANTEKB , *Exemple de code et de configuration pour les outils streamingtoolbox.com.*, <https://github.com/jantekb/streamtoolbox-examples/>, Code, Nov 7 2015
- [17] [GITHUB] UMLAEUTE , *v4l2loopback - a kernel module to create V4L2 loopback devices.*, <https://github.com/umlaeute/v4l2loopback>, Code, Fev. 26 2018
- [18] LIBAV OFFICIAL , *wiki : Using libav**, https://trac.ffmpeg.org/wiki/Using%20libav*, API, 10 nov. 2017
- [19] [GITHUB] NIKLAS HAAS , *FFmpeg versus Libav*, <https://github.com/haasn/mpvhq-old/wiki/FFmpeg-versus-Libav>, Article, Mar 22 2015
- [20] ADOBE OFFICIAL , *Flash & The Future of Interactive Content*, <https://theblog.adobe.com/adobe-flash-update/>, Article, 07-25-2017
- [21] MIST OFFICIAL , *Comparison from Mist website (bias)*, <https://mistserver.org/comparison>, Article, December 2017.
- [22] WINDOWS OFFICIAL , *Search product lifecycle*, <https://support.microsoft.com/en-us/lifecycle/>, Site, -
- [23] RIDGERUN.COM , *Gstreamer QT Overlay*, https://developer.ridgerun.com/wiki/index.php?title=Gstreamer_QT_Overlay#Network_streaming, Site, 23 November 2017

- [24] GSTREAMER OFFICIAL , *Gststreamer QT Overlay*, <https://gststreamer.freedesktop.org/>,
code : git ::/anongit.freedesktop.org/gststreamer/gststreamer, Site, -
- [25] OODLESTECHNOLOGIES , *Red5 Vs Wowza Head To Head* , <http://www.oodlestechnologies.com/blogs/Red5-Vs-Wowza-Head-To-Head>, Site, 12 déc. 2017
- [26] WOWZA OFFICIAL , *How to add graphic overlays to live streams with Wowza Transcoder* , <https://www.wowza.com/docs/how-to-add-graphic-overlays-to-live-streams-with-wowza-transcoder>, Site,
11.13.2017
- [27] ADRIAN KAEHLER & GARY BRADSKI , *Learning OpenCV 3* , O'Realy, Book, 2017

