Compte rendu premier rdv

membres présent : tous
date : le 15/09/20 de 8h30 à 12h30

Objectifs

1. Elaborer les taches qui seront réalisé au premier sprint
2. Installation des softwares
3. Mise en place d'un début d'env. de developpement.
4. Topo rapide sur le mode agile

Réalisation

1. la comprehension de l'algo

toutes les taches seront évalués à l'issue du sprint selont leur difficulté envisagé (de 0 à 10/10 pour le plus dur)

les points à surveiller sont les suivants :

- fonctionnement du découpage en macrobloc 8x8
- transformation en cosinus durect (DCT)
- zig zag scanning (parcours de l'image selon la diagonale)
- run lenght level encoding (RLL)
- codage de huffman.

découpage en macrobloc :

pour une résolution en 720p, avec blocs de 16x16, il y a 45x80 blocs ça suppose une transformation flux-video ==> bitestream bibliothèque pour capture video : open-cv *pip install opencv-python*
0915 : on a un début de code qui permet de prendre des screenshots (dossier webcam capture) en appuyant sur espace.

1ere étape du sprint 1 : transformer un screenshot en bitstream, connexion socket entre 2 clients. décodage du bitstream pour afficher une image

norme MPEG-1: bloc de 16x16, encodage PAL 25i/s, 352x288px => 22x18blocs.

blocs

- **blocs I : intracodé (indépendant)**
- blocs P : prédictif (décrite par différence avec images précédentes)
- blocs B : prédicitif + images suivantes

- blocs DC : moyennes par bloc.

MPEG1 : sépration partie video et partie audio. on utilisera une bibliotèque pour conversion flux audio en MP3.

codage de Huffman

on a compris la base de l'encodage (avec l'arbre binaire), reste quelques difficultés au niveau du décodage (perte d'information ?) 1100 : prototype permettant de générer la chaine d'analyse frequentielle necessaire à la construction de l'arbre binaire, sous forme de dictionnaire. 1130: arbre binaire construit, reste l'encodeur. on a un encodeur ASCII a titre de comparaison de taille (pour evaluer la pertinence de la compression pour les livrables)

On constate une compression de l'ordre de 30 à 60 % pour rapport au codage ASCII (transformation d'un caractère ascii en sa représentation binaire)

sprint 1 : fabrication d'encodeur/décodeur de Huffman.

transformée DCT

Bibliothèque python : scipy (scipy.fftpack.dct). *pip install scipy* transformation pour passer d'une description en couleur par pixel a chaque frame à une description frequentiel.

2. installation des softwares

client git : gitkraken repository sur git-hub : https://github.com/EVEEX-Project/
IDE python : pycharm avec python 3.8
il y aura un fichier requirements pour l'installation des bibliothèques
webcam : modèle 720p intégré a nos ordis portables. serveur discord : https://discord.gg/7Ne6ZMw Documentation : présente sur le serveur discord dans le chat "documentation" et dans le dossier documentation du projet Trello reprenant les étapes des sprints => https://trello.com/b/tQPCDirN/projet-eveex

3. mise en place de l'environement de dev

Structure du programme (prototype python pour commencer)
La structure sera ammené à changer à l'avenir, on separera les repository en 2 avec un git consacré au projet en lui même, et l'autre consacré à l'aspect scolaire du projet (rapports/comptes-rendus/livrables)

4. topo mode agile

récap sprint 1:

- finir l'encodage/decodage Huffman => Hugo

- passage screenshot vers bitstream => Alexandre, Guillaume
- connexion entre 2 clients par socket => Jean-Noël
- décodage bitstream vers screenshot => Guillaume, Alexandre
- rapport interview à rediger avant le 22 => Guillaume
- diagramme de Gantt => Alexandre
- environement de dev fonctionnel pour tout les memebres => tous

TRES IMPORTANT : à l'issue de ce sprint, les participants devront évaluer la difficulté des taches qui leur sont été affectés, avec le ressenti. Il y aura également une reflexion mené avec les encadrants sur la quantité de travail à fournir à chaque sprint (nottament en fonction des cours)