Mingam Pierre-yves
Biannic Gilles

ISEN ALL IS DIGITAL! | yncréa

Compte-rendu de projet Robotique

Autonomous Robots Gaming System
Partie réalité augmentée

Sujet :

Utilisation de l'environnement ROS afin d'intégrer un plateau de jeu (Morpion) à la capture vidéo d'un robot (Turtlebot) et de l'afficher dans une interface web.

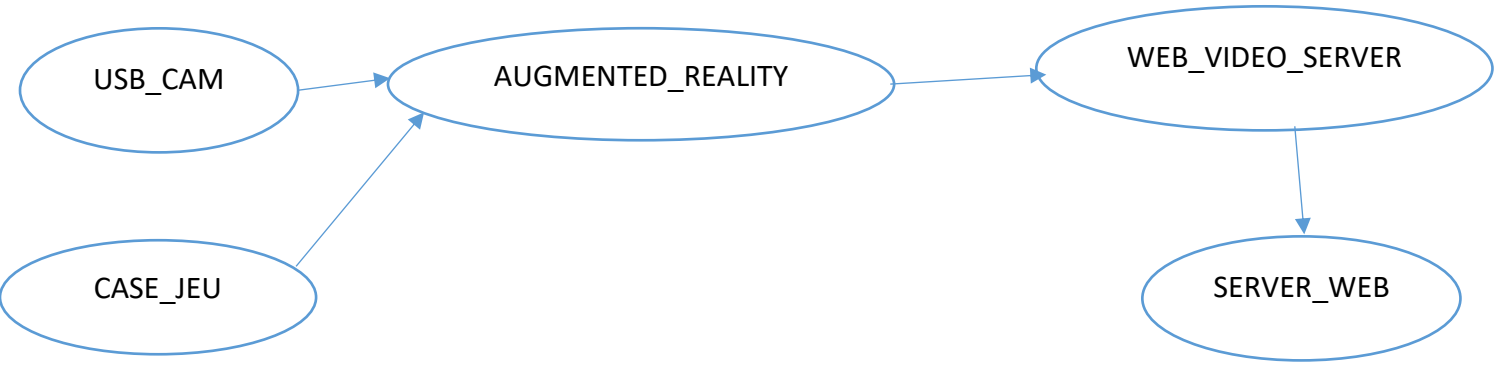
Professeur référent :

Clément Aubry

Table des matières

Architecture

Ci-dessous l'architecture mise en place pour la récupération du flux vidéo de la caméra USB vers le node de réalité augmentée pour enfin arriver vers l'interface web.



Explication :

Le node « USB_CAM » permet de créer un topic pour diffuser le flux vidéo de la caméra. Le node « AUGMENTED_REALITY » souscrit au topic qui est publié par « USB_CAM ». Le node de réalité augmentée va également souscrire au node « CASE_JEU » pour obtenir la grille que l'IA vient de jouer.

Après avoir récupéré les informations, le node « AUGMENTED_REALITY » va publier un topic qui comprend le flux vidéo modifié. On peut y voir le plateau du jeu. Le node « WEB_VIDEO_SERVER » va y souscrire pour ensuite lancer un serveur vidéo et créer une URL utilisable dans l'interface web.

Pour finir le node « SERVER_WEB » créer un serveur web pour créer l'interface web qui va diffuser le flux vidéo récupéré.

Prérequis

Installation :

https://github.com/ISENRobotics/ARGS

Package supplémentaire installé :

https://github.com/RobotWebTools/web_video_server

Réalité augmentée

Le node AUGMENTED REALITY a pour principal rôle d'ajouter le terrain de morpion dans le flux vidéo de la webcam (usb_cam).

Au départ, nous avons cherché à récupérer le flux vidéo envoyé par la kinect. Au fur et à mesure de notre apprentissage de ROS, nous avons appris à utiliser la commande 'roslaunch' pour démarrer des nodes du turtlebot. Ensuite, en utilisant 'rqt_graph' et 'rviz', nous avons trouvé le topic sur lequel le flux vidéo est envoyé :

« /camera/rgb/image_color/compressed » pour le flux de la Kinect,

« /usb_cam/image_raw/compressed» pour le flux de la Webcam.

Ensuite nous avons cherché à créer un node pour récupérer ce flux, le modifier et le publier sur un autre topic. Pour cela nous avons utilisé un script sur conseil de M Aubry :

http://wiki.ros.org/rospy_tutorials/Tutorials/WritingImagePublisherSubscriber

Ce script de M Simon Haller qui utilise la librairie Python opencv, nous a permis de récupérer le flux vidéo, de marquer les points d'intérêts des images par un rond puis de publier sur un topic de sortie.

Nous avons rapidement compris qu'à partir des points d'intérêts il serait difficile de reconnaître l'environnement et de dessiner correctement le terrain. Nous avons pensé à utiliser une feuille de papier avec un damier imprimé dessus pour avoir un point de repère et une échelle dans l'environnement. Dans opencv, il existe une fonction (findChessboardCorners) qui permet de récupérer les positions des coins du damier et ainsi éviter la complexité des points d'intérêts.

Avec la détection des coins du damiers et quelques documentations trouvées sur Internet

(http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_pose/py_pose.html,

https://rdmilligan.wordpress.com/2015/07/02/augmented-reality-using-opencv-and-python/), nous avons été capable de trouver un point d'origine et des vecteurs unitaires dans le plan du damier. Grâce à cela, nous avions un plan dans lequel nous étions capables de dessiner ce que l'on voulait, en l'occurrence : une grille de damier.

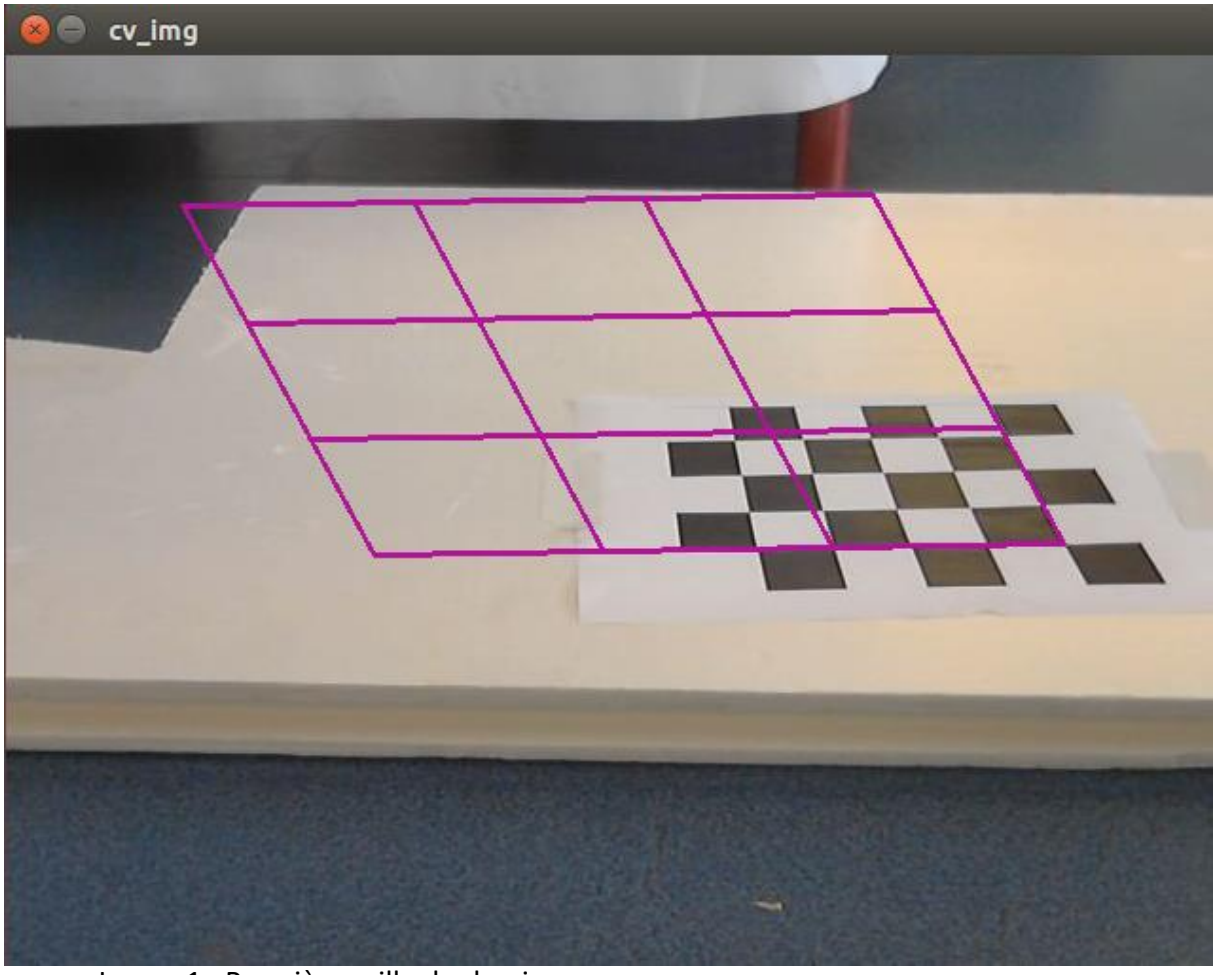


Image 1 : Première grille du damier

En regardant cette image, la grille ne semble pas vraiment dans le plan … Cela est dû au fait que dans la vue de la caméra il y a de la perspective. Comme on l'apprend dans des cours de dessins, deux lignes parallèles dans le plan 2D observé ne sont pas systématiquement parallèles dans l'espace 3D. Elles se dirigent vers un point d'intersection : le point de fuite. Nous avons cherché à retrouver ces points d'intersections (pour les segments horizontaux et verticaux du damier) et à tracer les droites vers ces points d'intersection.

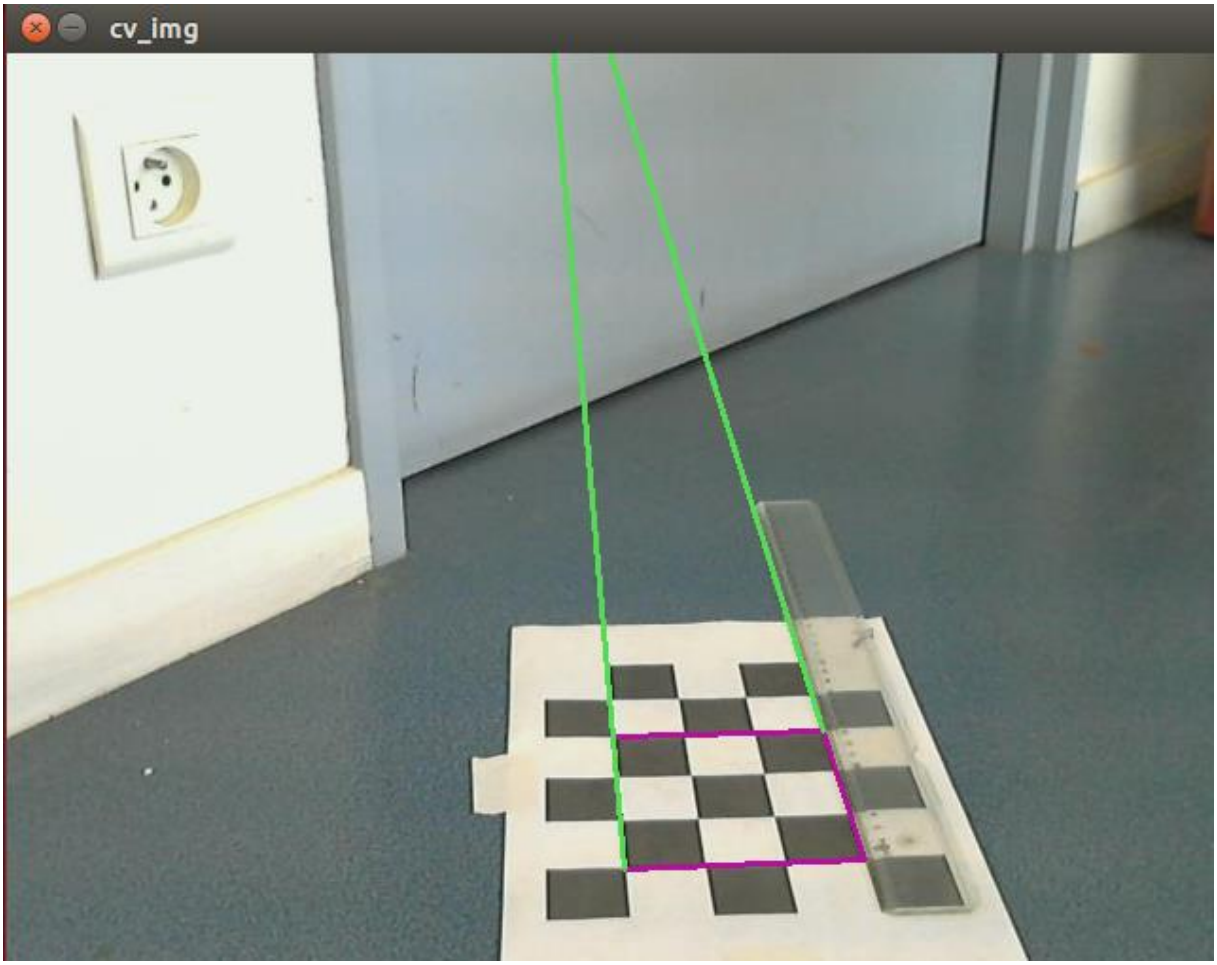
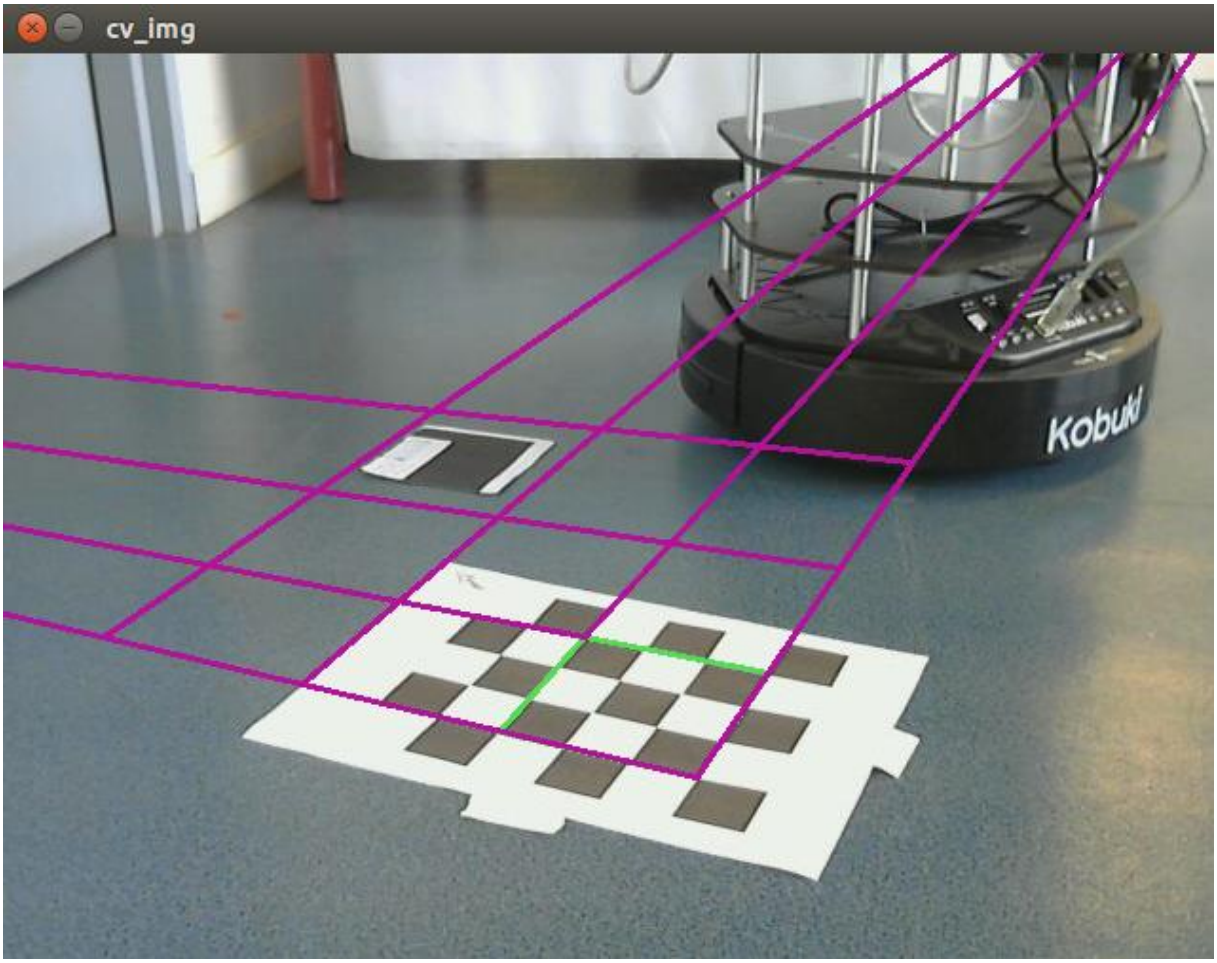


Image 2 : Affichage des lignes de fuite



Le résultat visuel se trouve nettement amélioré, mais il reste un dernier détail perturbant : la taille des cases semblent augmentées au fur et à mesure qu'elle s'éloignent. Pour corriger cela, nous avons essayé de trouver un facteur de diminution de la largeur des cases. Ce facteur a été calculé en calculant un ratio entre un segment proche et un autre plus éloigné. On obtient ainsi un damier plus réaliste :

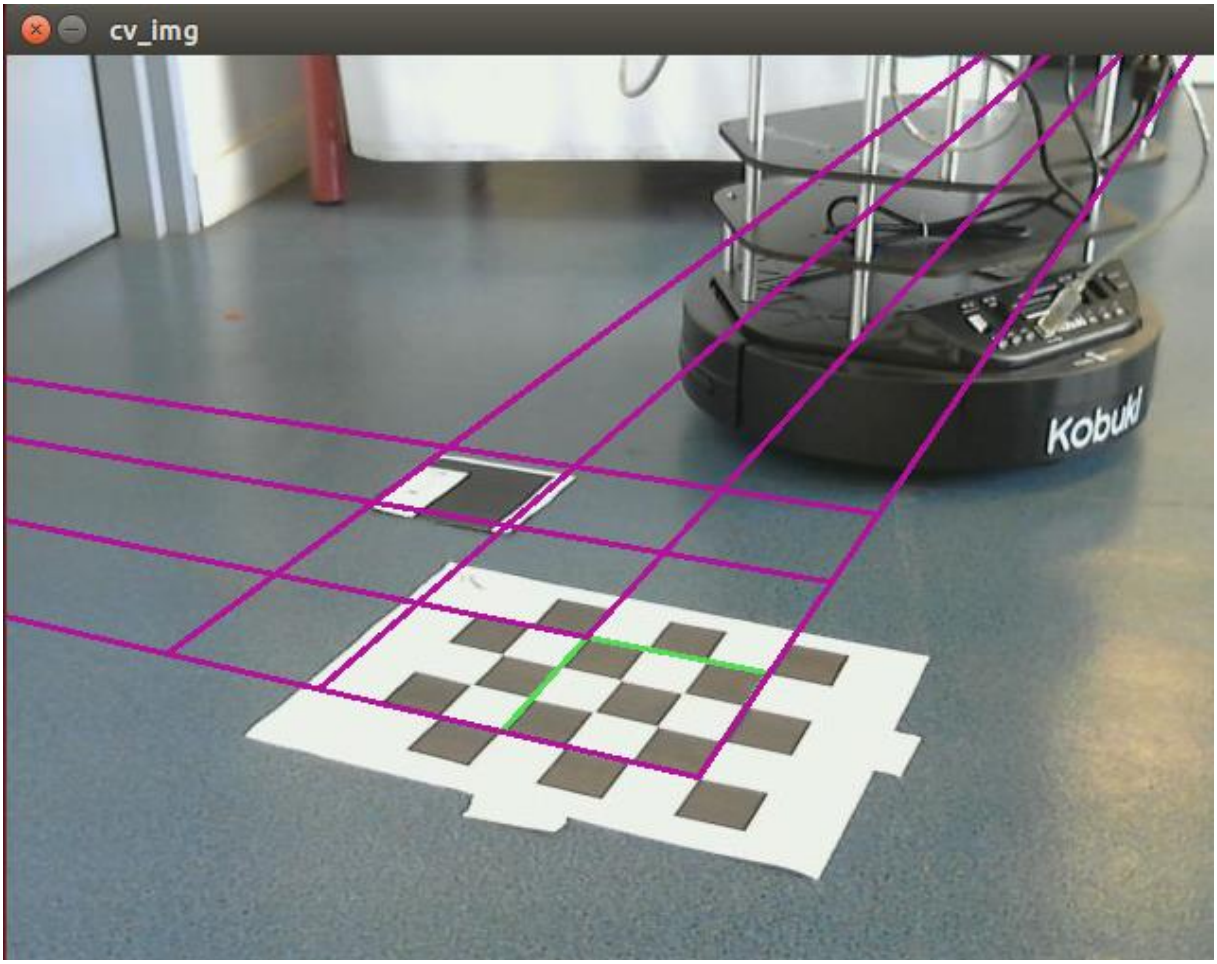


Image 4 : Image avec largeur des cases variables

Ensuite nous avons calculé tous les points d'intersections entre les droites tracées, et grâce à ces points nous avons pu tracer des croix aux emplacement joués (récupérés sur le topic /Case_jeu).

La dernière amélioration que nous avons apportée, c'est d'imprimer le damier sur une feuille A3 au lieu de la A4. Ci-dessous, le damier avec la dernière version du node et la première version (avec damier) en filmant un damier A4, puis les mêmes versions du script sur une feuille A3. On constate que la détection est largement plus performante :

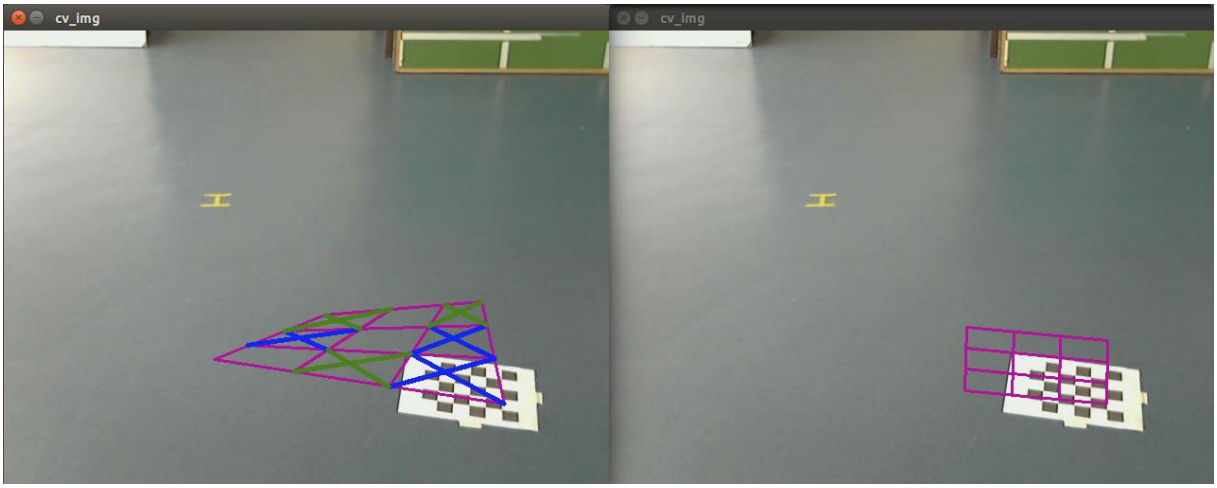


Image 5 : Résultat avec damier en format A4

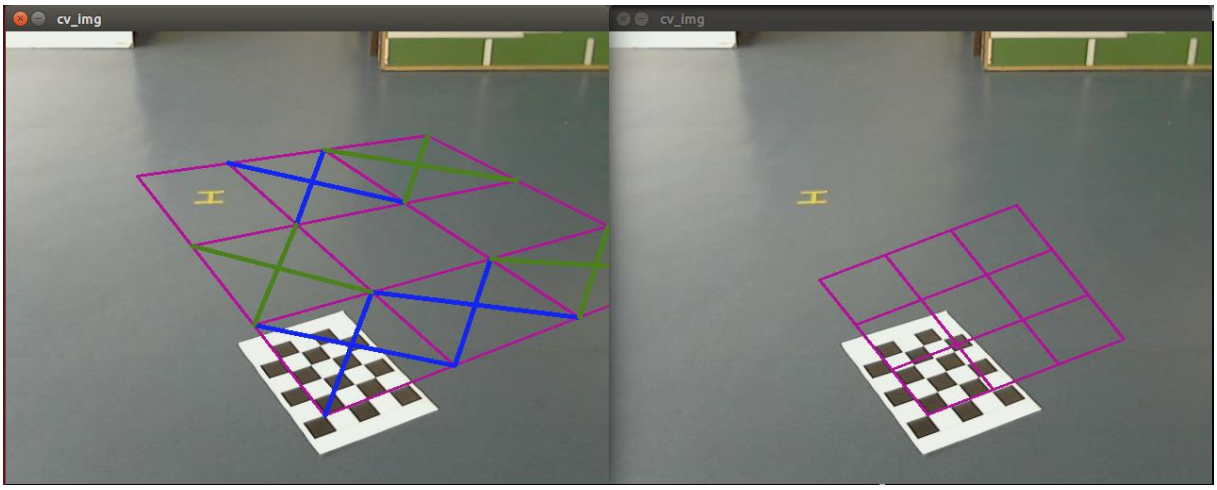


Image 6 : Résultat avec damier en format A3

Interface web

Fonctionnement interface web :

Le but de cette partie est de récupérer le flux vidéo modifié provenant du node de réalité augmentée. Pour réaliser cela deux nodes supplémentaire sont utilisés.

web_video_server :

Le problème qui se posait était que le flux vidéo modifié (celui comprenant la grille de jeu du morpion) était disponible sur un topic, or pour l'afficher dans une interface web il était nécessaire d'avoir une URL. Comme si le flux provenait d'une caméra IP.

Après plusieurs essais, le package « web_video_server » a été sélectionné. D'après la documentation disponible sur ROS, ce package est le plus récent pour cette fonctionnalité. Ce package regroupe deux anciens package.

Pour le bon fonctionnement, il était nécessaire de modifier les sources du package. Cette modification permet de modifier l'adresse IP ainsi que le port que va générer ce package. Ici nous avons optés pour l'adresse :

http://192.168.1.62:8182

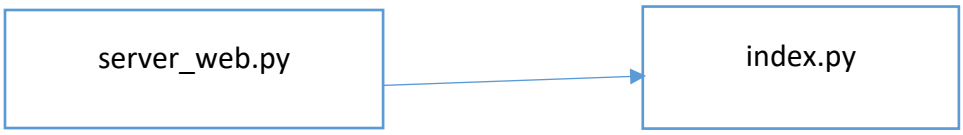
Le choix de souscriptions à un topic précis est réalisé dans l'appel de cette URL en lui passant différents paramètres.

server_web :

Ce node est le node qui créer l'interface graphique récupérant le flux vidéo comprenant la grille de jeu.

Python a été choisi pour développer ce node car ce langage était plus simple pour la création d'un node étant donné que nous avions toujours utilisé le Python pour la création de node. Le deuxième argument pour le choix de cette technologie est qu'il est simple de créer un serveur web. Donc le Python a été sélectionné.

Après avoir mis en place le serveur web, nous obtenons l'architecture suivante :



Le fichier server_web.py contient la création du node, ainsi que la création du serveur web. Celui- ci est créé sous l'adresse :

Localhost :8888

Néanmoins le serveur web est créé seulement si le node « server_web » arrive à souscrire au topic généré par le node « usb_cam ». Cela permet d'éviter le lancement du serveur si la caméra USB ne diffuse aucunes images.

Ensuite le serveur web est paramétré pour rechercher le fichier « index.py » se trouvant dans le répertorie courant du server_web.py.

Le fichier index.py contient le code HTML (qui est une base de template bootstrap Knight Theme). Le template a été épuré pour accueillir une iframe pour afficher la vidéo, ainsi qu'un formulaire pour pouvoir jouer (partie non fonctionnelle dans l'application générale).

Pour utiliser l'URL créer par le node « web_video_server », l'appel dans le iframe est fait de cette façon :

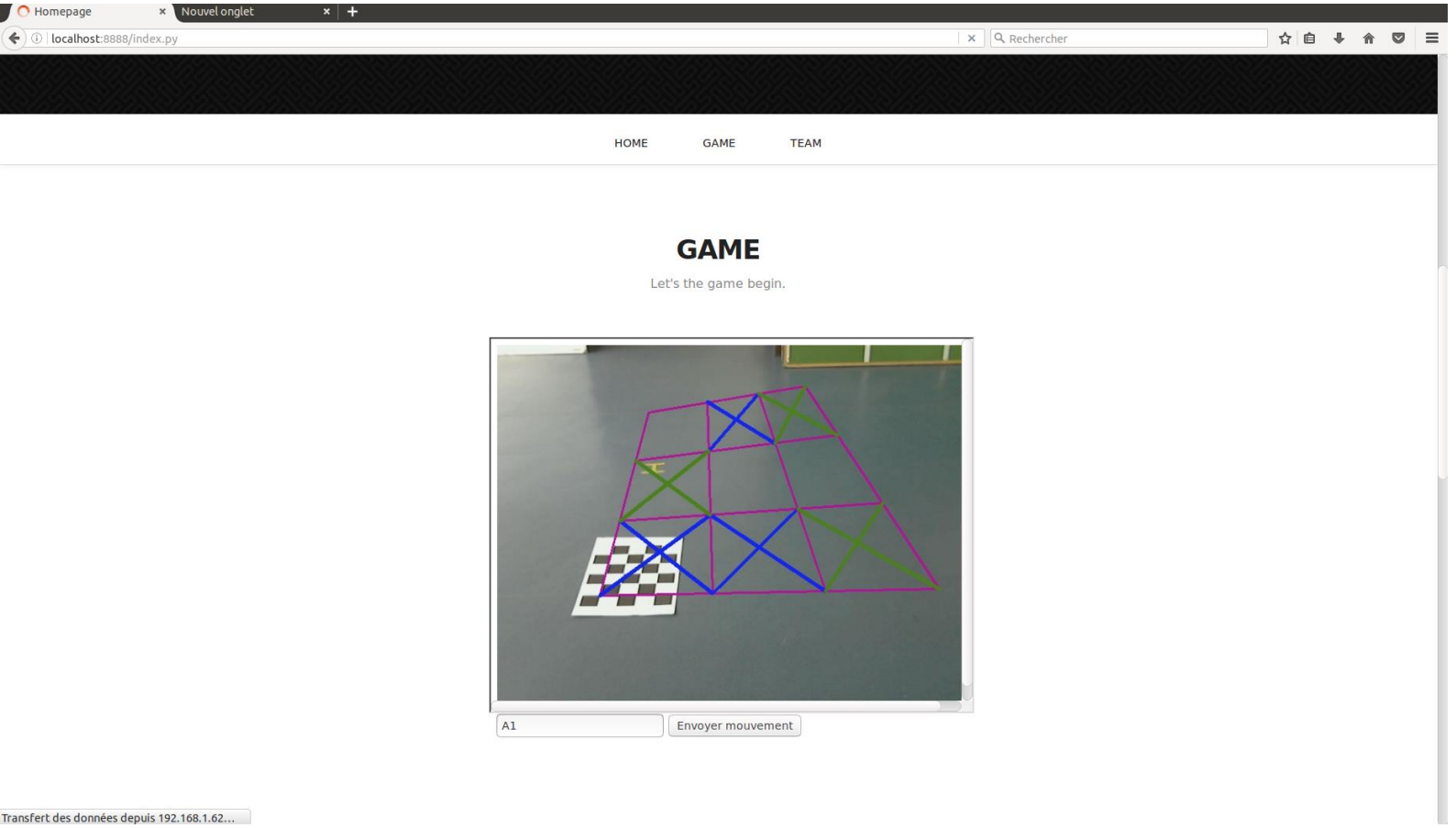
http://192.168.1.62:8182/stream?topic=/augmented_reality_output/image_raw&type=ros_compressed

Explication de l'URL :

- 192.168.1.62 :8182 : Cette adresse est l'adresse générée par le node « web_video_server »
- /stream : Ce paramètre permet de signifier au node « web_video_server » que l'on cherche à recevoir un flux vidéo (snapshot peut être utilisé pour recevoir des captures d'écran)
- topic= : Ce paramètre permet d'indiquer à quel topic nous voulons souscrire
- /augmented_reality_output/image_raw : Ce paramètre est le nom du topic qui ici nous intéresse pour récupérer le flux vidéo modifié
- type= : Ce paramètre permet d'indiquer dans quel format est le flux vidéo ou les images entrantes
- ros_compressed : ros_compressed est le format sélectionné pour la diffusion de la vidéo

Résultat sur l'interface web :

Le résultat obtenu dans l'onglet GAME de l'interface est le suivant :



Problèmes

Un fichier .launch est initié. Celui-ci regroupe le node de réalité augmentée ainsi que le node lançant le serveur web. Néanmoins des problèmes persistants n'ont pas permis d'avoir un fichier launch fonctionnel.

Le node du serveur web se lance en spécifiant python ou avec la commande rosrun dans un terminal. Mais le comportement est différent lors de l'exécution avec le roslaunch.

Pour la réalité augmentée, la perspective était un des problèmes majeurs de cette partie, il reste des imperfections sur celle-ci, on peut le voir sur la grille au-dessus.

Annexes

Interface web