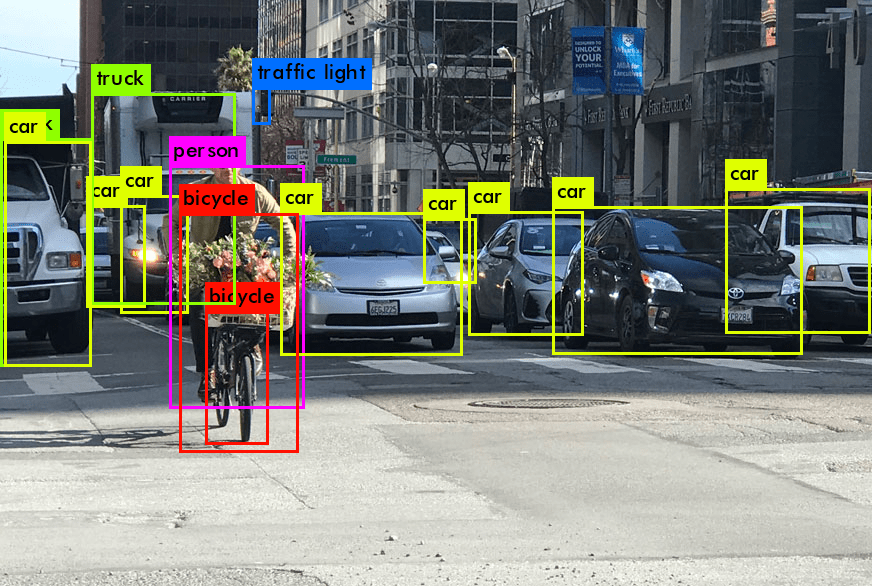
Une image contenant texte

Description générée automatiquement

Projet Vision : Détection et suivi 3D de voitures et/ou de piétons dans des séquences routières



|  |  |  |
| --- | --- | --- |
| Etudiants : | Quentin BERNARD, Lucas ISSARD, Etienne TORZINI | |
|  | |

Date : Janvier 2022 / Février 2022

Résumé

Notre objectif est de réaliser un programme capable de détecter et de suivre des véhicules dans des séquences vidéo issues d’une tête stéréoscopiques embarquée et de localiser ces véhicules en 3D par rapport à la voiture qui embarque la tête stéréo. L’application de cette détection peut se faire pour les véhicules autonomes. En effet, le véhicule doit connaitre son environnement et réagir en fonction de celui-ci. La détection des autres véhicules est donc nécessaire. Notre programme sera réalisé en Python.

Sommaire

[1. Détection de Véhicule 6](#_Toc93996593)

[A. Etude bibliographique 6](#_Toc93996594)

[B. Viola et Jones 6](#_Toc93996595)

[2. Suivi de Véhicule 6](#_Toc93996596)

[A. Etude bibliographique 6](#_Toc93996597)

[B. ? 6](#_Toc93996598)

[3. Mise en correspondance et localisation 3D 6](#_Toc93996599)

[A. Mise en correspondance 6](#_Toc93996600)

[B. Triangulation 8](#_Toc93996601)

[C. Affichage dans un plan 2D 11](#_Toc93996602)

Introduction

# Détection de Véhicule

## Etude bibliographique

## Viola et Jones

# Suivi de Véhicule

## Etude bibliographique

## ?

# Mise en correspondance et localisation 3D

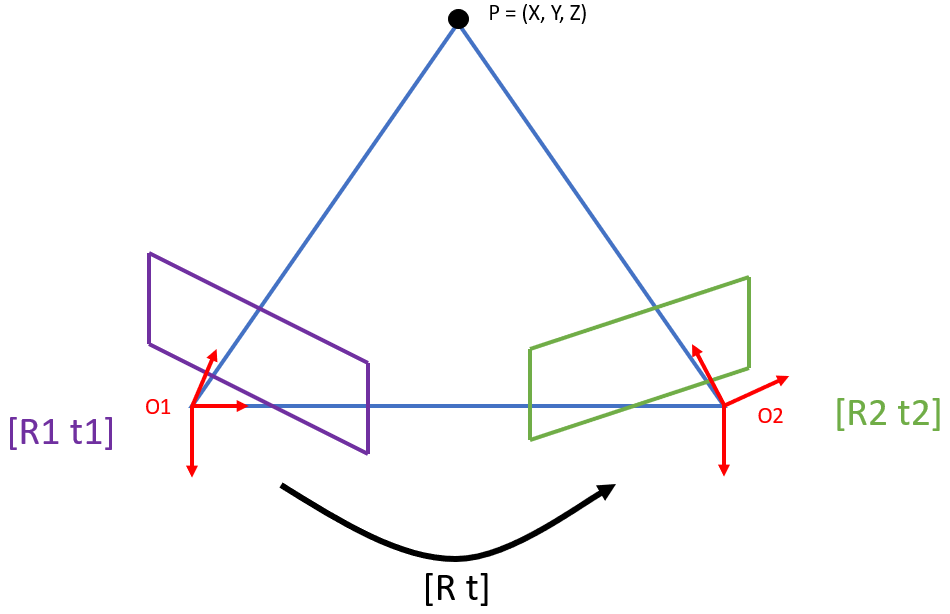
Une seule caméra ne nous permet pas de voir en 3D. Ainsi, pour localiser le véhicule il est nécessaire d’ajouter une ou plusieurs caméras permettant de voir en 3 dimensions. Le premier but est de mettre en correspondance les points sur les images réalisés en même temps mais avec des caméras à des positions différentes. Ensuite, pour déterminer la position du véhicule, on utilise la triangulation.

## Mise en correspondance

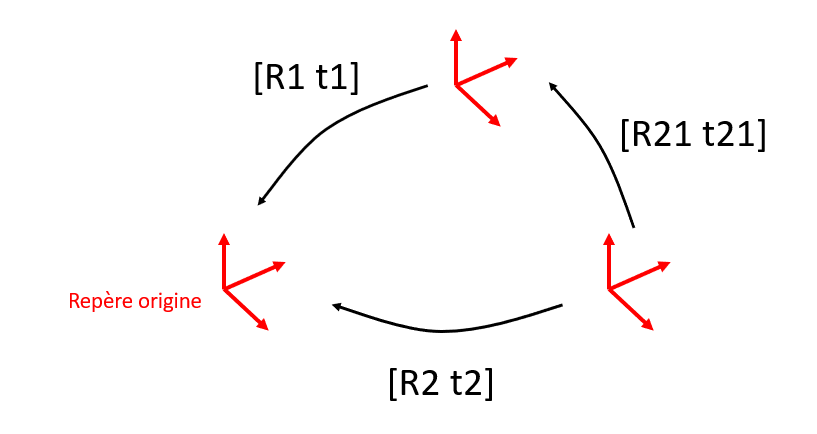
Dans notre cas, nous avons à notre disposition des photos de 2 caméras cote à cote. Nous allons prendre le cas ou l’on utilise notre algorithme de détection de véhicule sur la photo 1 soit la caméra de gauche (repère R1). Notre algorithme nous retourne la position du véhicule en pixel. La première chose à faire de retrouver cette position sur la photo 2 soit la caméra de droite (Repère R2) afin de localiser le véhicule avec la triangulation.

Pour notre projet, on utilise la base de données KITI comme fichier de test. Cette bibliothèque contient un ensemble d’image et de séquence en multi vues avec les différents paramètres de calibrage des caméras (K1 et K2).

Dans notre cas, le schéma ci-dessous représente notre situation. Il y a deux caméras qui prennent les images en même temps. La première à la position O1 et la deuxième à la position O2.



Dans un premier temps il est nécessaire de déterminer la rotation et la translation entre les caméras. Dans la bibliothèque, on a à notre disposition les rotations et les translations entre les caméras et un repère.



Pour la mise en correspondance des points, il est possible d’utiliser la droite épipolaire qui permet de déterminer le point correspondant dans la deuxième image. Or dans ce cas nous avons à notre disposition des images déjà rectifiées. Il n’est donc pas nécessaire de calculer la droite épipolaire. Il suffit de faire une corrélation 2D selon la même ligne. Dans le cas ci-dessous, on souhaite faire la correspondance d’un point entre 2 images. Ce point appartient à la même ligne. Ainsi on peut tracer une droite qui coupe horizontalement l’image. Ensuite, on va parcourir cette droite en faisant une corrélation 2D avec l’image de référence. On pourra choisir la taille du masque qui correspond à la matrice blanche sur le dessin. En parcourant la droite avec la matrice blanche, on va garder en mémoire le maximum du résultat de la corrélation. Lorsque ce nombre est maximal soit 1, cela veut dire que les deux images sont identiques. Ainsi, en jouant sur le seuil de corrélation et le masque, on trouve le point correspondant à l’endroit ou la corrélation est maximale. On pourra donc savoir sa position exacte. Le pixel sera sur la même ligne. La colonne sera située au maximum de la corrélation.

Une image contenant texte, route, bâtiment, extérieur

Description générée automatiquement

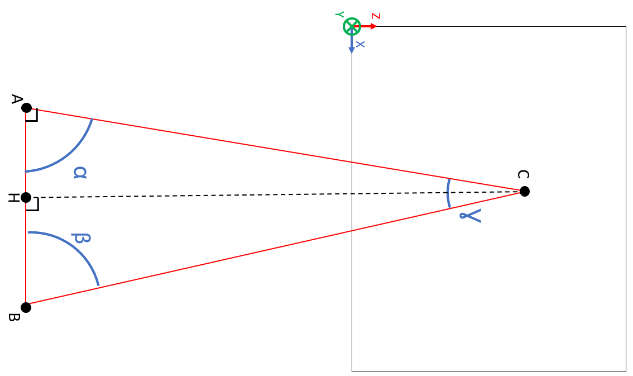
Maintenant que l’on a mis en correspondant les points sur les deux images, il nous reste à calculer la réelle position de la voiture. Dans notre cas nous avons les positions en pixels. On cherche la position du point .

## Triangulation

La triangulation et une méthode pour calculer une position à partir de deux points dont on connait la distance.

Voici un schéma qui explique la technique de triangulation

Les point A et B sont des positions que l’on connaît et le point C est la position que nous cherchons à déterminer.



Comme nous connaissons les positions des points A et B nous pouvons calculer la distance AB.

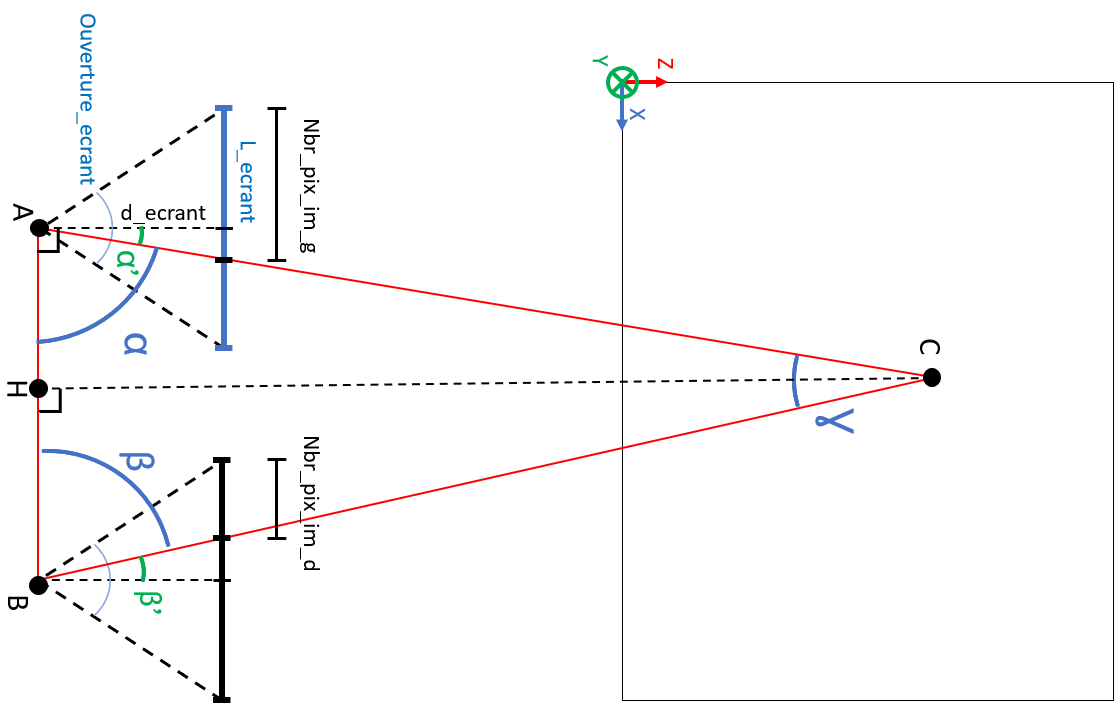
Et en mesurant les angle α et β nous pouvons calculer Ɣ le troisième angle du triangle ABC

Avec :

Ainsi avec la loi des sinus :

Et nous pouvons calculer la distance HC et BH avec sinus et cosinus de BC :

Nous pourrons donc calculer la position du point C.

Pour calculer les angle α nous allons nous passer dans ce schéma :

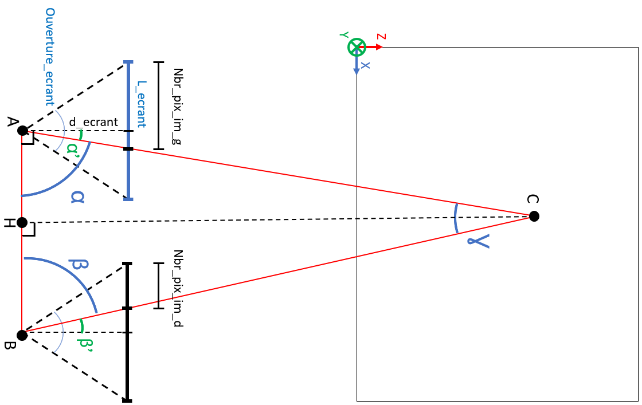
Les seules données que nous avons sont : L’angle d’ouverture (ouverture\_ecrant) de la caméra le nombre de pixel en longueur de l’écran (L\_ecrant) et le nombre de pixels séparant le bord de l’image et le point C sur cette image (Nbr\_pix\_im\_g).

Calculons déjà la d\_ecrant  (distance en pixel):

Puis nous pouvons calculer l’angle α’ avec :

Donc nous nous pouvons calculer l’ange  avec :

On fait de même avec l’angle et nous pouvons calculer les coordonnées de C comme vue précédemment.



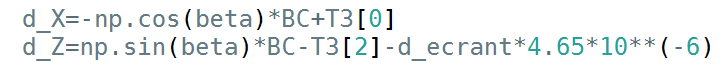
## Affichage dans un plan 2D

Pour afficher les positions des voitures calculer par rapport au véhicule il faut ajuster les distances calculées.

L’écran est une distance fictive il faut donc la soustraire et il faut soustraire la position de la caméra 3 (point B) par rapport au point d’origine.

Une image contenant texte

Description générée automatiquement



Conclusion