COMPTE RENDU
Projet Compression - Traitement d'image
Sujet 4 : RGB-D

CR 1 : Organisation et premières recherches

Pour notre première séance sur notre sujet RGB-D on a décidé de faire des recherches sur internet et de rassembler ce qu'on a trouvé pour pouvoir établir un plan de travail.

Nous avons vu des définitions et des explications intéressantes :
-Explication théorique très théorique et mathématiques
https://www.mdpi.com/1424-8220/12/2/1437

-Explication du concept et visuels par une entreprise qui vend des caméra :
https://www.e-consystems.com/blog/camera/technology/what-are-rgbd-cameras-why-rgbd-cameras-are-preferred-in-some-embedded-vision-applications/

Mais nous avons aussi vu des applications concrètes:

-Application possible de ces camera RGBD - Generation de scène 3D a partir de video d'environnement : https://www.youtube.com/watch?v=i3iSeZNg3w8

-Des façons de gérer le rgb d en python :
http://www.open3d.org/docs/latest/tutorial/Basic/rgbd_image.html

Et enfin, nous avons regardé une bonne vidéo de vulgarisation sur le sujet :
Vulgarisation sur le RGBD : https://www.youtube.com/watch?v=bRkUGqsz6SI

Voici le résumé que nous avons dégagé de cette vidéo :
—----------

Pour avoir une image RGB D ils ont deux caméra RGB calibrées qui simule oeil droit et oeil gauche. Avec ça il suffit de checker l'image d'une caméra (par exemple la gauche) et comparer le décalage qu'il y'a entre chaque pixel sur l'image de l'autre caméra (toujours dans l'exemple la droite).

C'est cet écart a chaque pixel qui donne une notion de profondeur par triangularisation. Si l'écart entre les deux pixel est faible alors le pixel est loin et s'il est grand le pixel est proche. Ça marche bien pour des objets texturés (avec du relief) parce qu'il est facile d'identifier et de retrouver le pixel de droite représentant le pixel de gauche. Mais avec des images lisse (exemple un mur tout blanc) retrouver le pixel c'est pas facile (voir pas possible) ce qui fait un bruit noir de "pixel pas retrouvé dans la deuxieme camera".

COMPTE RENDU
Projet Compression - Traitement d'image
Sujet 4 : RGB-D

Pour y remédier, il y a une caméra infrarouge qui envoie des structures de point (une forme connue du device) et pour retrouver la bijection entre les pixel caméra gauche et caméra droite on utilise l'ancienne méthode + l'identification du motif sur les deux images. Comme le motif est connu on peut facilement trouver ce qu'ils donnent sur les deux caméras et alors avoir une différence et donc une profondeur.

—----------

Nous en avons bien discuté et nous avons décidé de convenir d'un plan comme celui ci :

1 - Se documenter sur le RGBD (pourquoi? comment? et chercher ce qui se fait de mieux pour en générer synthétiquemenT) C'est ce qu'on est en train de faire actuellement

2 - Faire une version "peu performante" de génération de la couche D (peut être sans apprentissage)

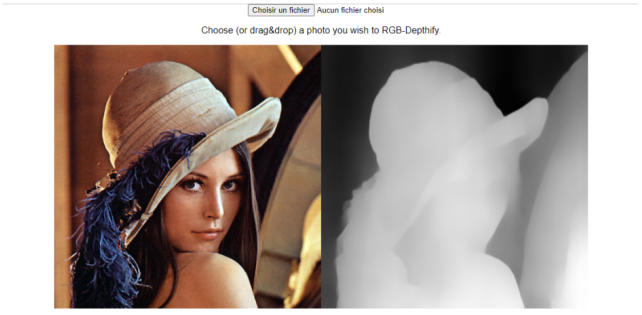
3 - Faire quelques traitement d'image et quelques compression avec cette couche D "peu performante" (pour faire un RGBD vs RGB) Et en tirer des conclusions (PSNR ou taux de compression)

4 - Revoir l'implémentation de la couche D (implémenter si possible de l'apprentissage) et montrer la construction step by step de cette construction de la couche depth

5 - Refaire l'étape 4 (pour faire RGBDold vs RGB vs RGBDnew)

6 - Faire une application (web ou logiciel) qui permet de : Créer la Depth a partir d'une image RGB Faire de la détection de contour (via couche D) Faire une compression (via couche D) etc..

Pour finir, nous avons regardé ce qu'il se fait en python et en javascript pour générer à l'heure actuelle des images RGB D a partir d'image RGB et voici un résultat qu'on à pu obtenir :



COMPTE RENDU
Projet Compression - Traitement d'image
Sujet 4 : RGB-D

CR 2 : Poursuite des recherches, et comment générer une D à partir de RGB:

Nous avons vu cette semaine comment une composante D est generée dans la plupart des cas sur les codes trouvés en ligne.

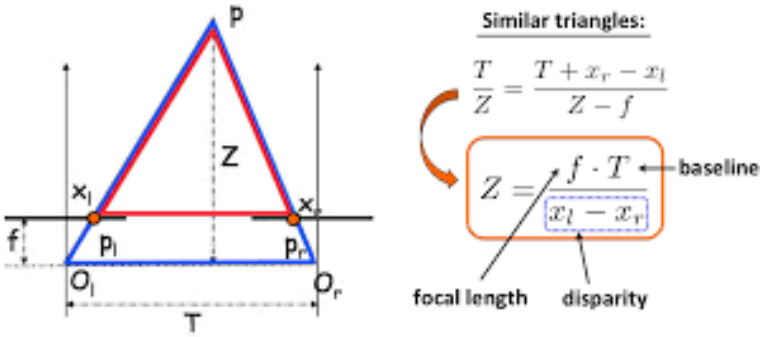
Deux façon de faire :

Entrainement / modèle via dataset
Cela implique d'avoir un dataset assez conséquent.
Il y en a plusieurs en ligne :

- Redwood
- SUN
- NYU
- TUM

L'objectif est d'apprendre à un modèle pleins d'image RGB D, de telle sorte à ce qu'il saura par la suite déduire via son apprentissage la dimension Depth depuis une simple image RGB.
Avantages : besoin seulement d'une image RGB et c'est le plus rapide/moins coûteux une fois le modèle entraîné.
Inconvénient : moins intéressant pour nous (modèle = boîte noire), on a juste a récupérer un dataset et le tour est joué.

Generation via image stereo calibrés
Cela consiste à utiliser une image pour chaque œil.
Elles représentent le même objet avec deux points de vue différents.
C'est en analysant les décalage (xl - xr) entre l'image de gauche et l'image de droite que l'on va avoir une **carte de disparité**. Et via cette disparité et aidé de la distance T (Baseline) et la distance focale f on peut déterminer Z la profondeur.



On a décidé après discussion avec notre encadrante de projet de partir sur la génération via image stereo calibrés. Et pour se faire, nous avons cherché des dataset d'images stéréo avec la distance focale et la baseline fournie: ICI

COMPTE RENDU
Projet Compression - Traitement d'image
Sujet 4 : RGB-D

CR 3 : Première implémentation de notre D synthétique

Cette semaine, nous avons codé l'algorithme permettant à partir d'une image stéréo d'obtenir une image appelée carte de disparité.

TODO fournir l'algo (en schéma) et un exemple d'image input et output

A partir de cette image, nous pouvons via cette formule

TODO image formule

calculer la profondeur et avoir une carte des profondeur aussi appelé Depth, c'est notre composante D du RGBD.

TODO donner l'image RGB, la carte disparité et la composante D.

Faire des test si on a le temps