COMPTE RENDU

TP3 : RECTIFICATION EPIPOLAIRE ET STEREOSCOPIE DENSE

Par
FRANCOIS Rémy

INTRODUCTION

Dans ce TP, nous allons faire la recherche de points similaires dans un système stéréoscopique en se servant de tous les points de l'image et en faisant une recherche avec le voisinage de chaque pixel.

SIMILARITE PAR SSD

Nous disposons des images suivantes :

Images gauche et droite fournies

Dans cette partie, nous allons chercher à déterminer l'indice de similarité des pixels de l'image de gauche avec ceux de l'image de droite.

Pour cela, nous allons avoir besoin des matrices mSSD et mMinSSD. mSSD contient les valeurs de différence (le coût) pour chaque pixel tandis que mMinSSD ne contient que les valeurs minimale des valeurs de disparités.

Pour calculer ce coût nous utiliserons la formule suivante :

$$\mathrm{SSD}(x_l, y, s) = \sum_{i=-w_x}^{w_x} \sum_{j=-w_y}^{w_y} \left(\mathbf{I}_l(x_l + i, y + j) - \mathbf{I}_r(x_l + i - s, y + j) \right)^2$$

Où $I_i(x_i,y_i)$ est le niveau de gris du pixel (x_i,y_i) dans l'image de gauche et $I_r(x_r,y_r)$ est le niveau de gris du pixel (x_r,y_r) dans l'image de droite.

Donc pour chaque pixel nous calculons ce coût, ce qui en terme de code s'implémente avec la ligne suivante pour la formule de coût :

```
*vals += pow(mLeftGray.row(x + i).at<unsigned char>(y+j) - mRightGray.row(x + i).at<unsigned char>(y+j-iShift), 2.0);
```

Pour accéder aux valeurs des pixels dans les images, nous nous servons d'un pointeur. Pour le calculer, nous partons d'une valeur minimale puis on se place au centre de la fenêtre de corrélation.

Nous obtenons l'image suivante pour la disparité de l'image de gauche :

La fonction minMaxLoc permet d'avoir les valeurs minimum et maximum de l'image, cela sert pour la fonction normalize. Cette fonction sert à étaler les valeurs en partant du minimum et du maximum de manière a ce que le minimum ait la valeur 0 et le maximum la valeur 255. Cela permet d'avoir une image nette.

VERIFICATION GAUCHE-DROITE

Désormais il s'agit de calculer la carte des disparités entre l'image de gauche et celle de droite. Mais pour cela, nous allons avoir besoin de la disparité de droite. Le code pour avoir celle-ci est sensiblement proche à celle pour la gauche, cependant, pour le calcul de la disparité, au lieu d'appliquer le décalage sur l'image de droite on le fait sur l'image de gauche et on inverse la façon dont on se décale afin de pouvoir aller vers la gauche.

La disparité de l'image de droite est représentée dans l'image suivante :

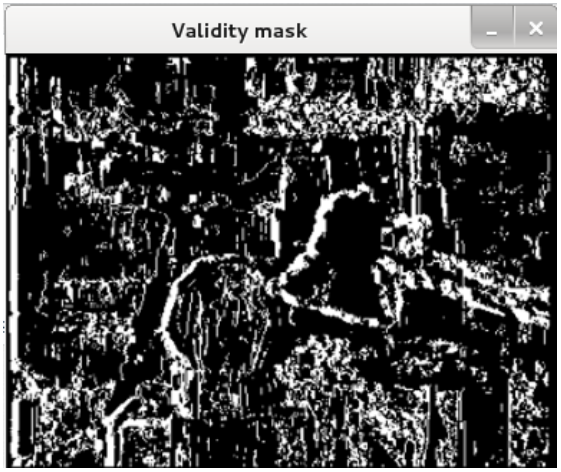
Ensuite pour vérifier les points qui ont un équivalent dans l'autre image, il nous faut effectuer une vérification de cohérence gauche-droite. Voici le code implémentant cette vérification :

```
Mat iviLeftRightConsistency(const Mat& mLeftDisparity,
                    const Mat& mRightDisparity,
                    Mat& mValidityMask) {
Mat mDisparity(mLeftDisparity.size(), CV_8U);
  for (int l = 0 ; l < mLeftDisparity.size().height ; l++){
    for (int c = 0 ; c < mLeftDisparity.size().width ; c++ ){
      double dispariteLeft = (double)mLeftDisparity.at<unsigned char>(l,c);
      double dispariteLeftCorrespondant = (double)mRightDisparity.at<unsigned char>(l,c-dispariteLeft);

      double dispariteRight = (double)mRightDisparity.at<unsigned char>(l,c);
      double dispariteRightCorrespondant = (double)mLeftDisparity.at<unsigned char>(l,c+dispariteRight);

      if (dispariteLeft != dispariteLeftCorrespondant || dispariteRight != dispariteRightCorrespondant){
        mValidityMask.at<unsigned char>(l,c) = 255;
      } else {
        mDisparity.at<unsigned char>(l,c) = dispariteLeft;
        mValidityMask.at<unsigned char>(l,c) = 0;
      }
    }
  }
  return mDisparity;
}
```

Nous obtenons comme disparité l'image suivante :



Cette image représente le masque de validité. Les points blancs sont ceux qui n'ont pas d'homologues, représentant des points occultés ou étant dans une seule image.

CONCLUSION

Dans ce TP, nous avons fait une recherche de points homologues sur tout les pixels de 2 images.