Labo 5 Beeldverwerking 2014
Feature detection

David Van Hamme

12 februari 2014

1 Randdetectie

Randdetectie is een basiscomponent van veel praktische beeldverwerkingssystemen (bv. kwaliteitscontrole, lane departure warning). Het meest gebruikte algoritme hiervoor is de Canny edge detector. De belangrijkste troeven van deze edge detector zijn dat de output lijnen van maar 1 pixel breed zijn, en dat er met hysteresis gewerkt wordt. Dit wil zeggen dat zwakkere randen ook gedetecteerd worden op voorwaarde dat ze verbonden zijn met een sterkere rand.

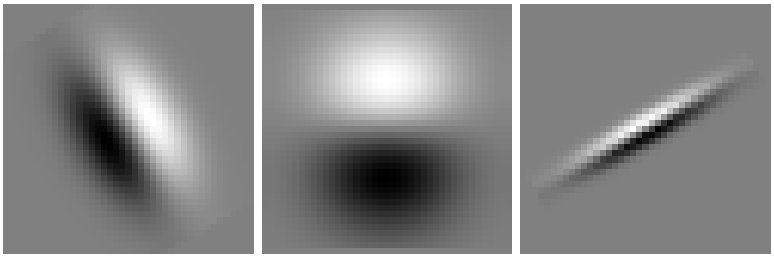
Voor sommige applicaties is het echter interessant om enkel randen in een bepaalde orientatie te detecteren. Hiervoor wordt dan een orientatieselectief randfilter gebruikt, zoals bijvoorbeeld de horizontale Sobel kernel uit labo 2.

2 Opgave 10

Schrijf een programma dat de randen van de gele stroken in **rays.png** extraheert. De naam van de afbeelding is nog steeds een commandolijnparameter. Je kan dit als volgt aanpakken:

- maak een 1D Gaussiaanse kernel met de functie **getGaussianKernel**;
- kopieer deze in de middenste kolom van een vierkante matrix met **Mat::col** en **Mat::copyTo**;
- maak nog een 1D Gaussiaanse kernel met kleinere standaarddeviatie en transponeer deze (**Mat::t**) om een rijmatrix te bekomen;
- filter de vierkante matrix (die de kolomkernel bevat) met deze rijkernel om een 2D Gaussiaan te bekomen;
- leidt deze horizontaal of verticaal af met **Sobel** om een DoG filter (*differential of Gaussian*) te bekomen;
- maak een rotatiematrix voor de juiste hoek van de gele stroken (ca. 75 graden) met **getRotationMatrix2D**;
- roteer je DoG filter met deze rotatiematrix (**warpAffine**);
- zet de afbeelding om naar grijswaarden en filter ze met het geroteerde DoG filter;
- neem de absolute waarde van de filterrespons met **abs**;
- tweak de parameters van de Gaussianen en de kernelgrootte zodat je een goede orientatieselectiviteit verkrijgt;

Tip: om floating point matrices zoals het DoG filter te visualiseren, kan je het minimum en maximum bepalen met **minMaxLoc**, dit schalen naar $[-.5, .5]$ en er .5 bij optellen.



Figuur 1: Voorbeelden van DoG filters.

3 Lijndetectie

Eenmaal de randen in het beeld bepaald zijn, kunnen we er rechte lijnen in gaan zoeken. De meest courante methode is de Hough transformatie. Deze werkt in het tweedimensionale rho-theta parameterdomein. Elk punt in het rho-thetavlak stelt een lijn voor onder hoek theta met de horizontale as, op een afstand rho van de oorsprong. Voor elke combinatie worden de randpixels geteld die zich in de buurt van deze lijn bevinden. Dit principe heet *parameter space voting* en kan ook voor andere parametriseerbare objecten gebruikt worden (bv. cirkels). Elke parametercombinatie die een aantal stemmen hoger dan een drempelwaarde kreeg, wordt als gedetecteerde lijn beschouwd.

4 Opgave 11

Schrijf een programma dat alle lijnen in **rays.png** detecteert en tekent. Nieuwe functies: **Canny, HoughLines, line**.

5 Corner detection

Voor veel applicaties, waaronder de meeste tracking- en herkenningsalgoritmes, is het nodig om kenmerkende punten in het beeld te detecteren, zodat die vergeleken kunnen worden met kenmerkende punten uit een volgende frame of een andere afbeelding. Een kenmerkend punt vanuit beeldverwerkingsperspectief is een punt dat goed gelokaliseerd is. Goed gelokaliseerde punten zijn bv. hoeken van voorwerpen, omdat de omgeving van die punten en de omgeving van punten ernaast sterk verschilt. Slecht gelokaliseerde punten zijn bv. punten op lijnen of in grote vlakken: de omgeving van die punten lijkt zeer sterk op de omgeving van punten er net naast.

Algoritmes die deze punten detecteren, worden *corner detectors* genoemd. De meest courante is de Harris corner detector, maar die is behoorlijk traag. FAST features zijn veel sneller te detecteren, maar iets minder stabiel t.o.v. bijvoorbeeld ruis in de afbeelding of kleine intensiteitsverschillen. Om objecten in verschillende afbeeldingen aan elkaar te relateren worden vaak SIFT of SURF features gebruikt. Deze berekenen een *descriptor* van de omgeving rond het punt die dan vergeleken kan worden met de descriptoren van de andere afbeelding.

6 Opgave 12

Schrijf een programma dat twee afbeeldingen (naam via commandolijn) inleest en in allebei Harris corners detecteert. Test dit op **shot1.png en shot2.png** en kijk of je dezelfde features detecteert. Nieuwe functies: **goodFeaturesToTrack, circle**.

7 Opgave 13

Schrijf een programma dat in elk van de twee afbeeldingen die je via de commandolijn meegeeft tussen de 32 en de 48 FAST features detecteert, er een BRIEF descriptor voor opstelt en de descriptors matcht tussen de twee afbeeldingen. Visualiseer de matches. Nieuwe functies en klassen: **FastFeatureDetector, BriefDescriptorExtractor, BFMatcher, drawMatches**.