1. Implementatieplan titel: Week3 Edge detection

1.1. Namen en datum

Martijn van der Struijk & Tim Hasselaar

26-05-2015

1.2. Doel

Het doel van deze opdracht is om van een plaatje de edges te herkennen. Als de edges eenmaal herkent zijn is het de bedoeling dat er ook nog threshholding overheen gedaan wordt. Dit om het plaatje volledig zwart/wit te maken en geen wazig pixels in het plaatje achter te laten.

1.3. Methoden

1. Mean
2. Gaussian
3. Laplacian
4. LoG

1.4. Keuze

Wij hebben gekozen om de methode laplacian te gebruiken. Voor deze opdracht maakt het volgens ons niet heel veel uit dat we ruis overhouden en dan is volgens ons laplacian de snelste manier om edges te bepalen in een afbeelding.

1.5. Implementatie

We gaan alle pixels langs met de laplacian mocht het zo zijn dat de filter buiten het bereik van de pixels valt dan verlengen we de rand pixels van het plaatje tot de filter helemaal gevuld is.

```
IntensityImage * i_image = new IntensityImageStudent(image.getWidth(),
image.getHeight());
   int firstAndThirdThreeKernelLines[] = { 0, 0, 0, 1, 1, 1, 0, 0, 0 };
   int secondThreeKernelLines[] = { 1, 1, 1, -4, -4, -4, 1, 1, 1 };
   for (int image_x = 0; image_x < image.getWidth(); image_x++){
      for (int image_y = 0; image_y < image.getHeight(); image_y++){
         int total_intensity = 0;
         for (int kernel_x = 0; kernel_x < 9; kernel_x++){
            for (int kernel_y = 0; kernel_y < 9; kernel_y++){
               int kernel_on_image_x = image_x + kernel_x - 4;
               int kernel_on_image_y = image_y + kernel_y - 4;

               //First check if the x coordinate isn't outside of the image
               if (kernel_on_image_x < 0){ kernel_on_image_x = 0; }
               else if (kernel_on_image_x >= image.getWidth()){ kernel_on_image_x =
image.getWidth() - 1; }
               else{ kernel_on_image_x = image_x + kernel_x - 4; }


               //Also check if the y coordinate isn't outside of the image
               if (kernel_on_image_y < 0){ kernel_on_image_y = 0; }
```

```
                else if (kernel_on_image_y >= image.getHeight()){ kernel_on_image_y =
image.getHeight() - 1; }
                else{ kernel_on_image_y = image_y + kernel_y - 4; }

                // kernel_y = 0, 1, 2 betekend de bovenste 3 regels van de kernel,
                // kernel_y = 3, 4, 5 betekend de middelste 3 regels van de kernel,
                // en kernely_y = 6, 7, 8 betekend weer de bovenste (gelijk aan de
onderste) 3 regels van de kernel.
                if (kernel_y >= 3 && kernel_y <= 5){
                   total_intensity += (int)image.getPixel(kernel_on_image_x,
kernel_on_image_y) * secondThreeKernelLines[kernel_x];
                }
                else{
                   total_intensity += (int)image.getPixel(kernel_on_image_x,
kernel_on_image_y) * firstAndThirdThreeKernelLines[kernel_x];
                }
                //std::cout << kernel_on_image_x << ", " << kernel_on_image_y << " = " <<
total_intensity << "\n";
             }
          }
          // Het totaal van de waardes in de kernel is 36.
          // Dus door de waarde te delen door 36 zul je altijd een resultaat tussen -255
en 255 krijgen.
          int new_intensity = total_intensity;

          // Uit de kernel komen waardes tussen de -2000 en 2000.
          // Om dit af te vangen verhogen we alles onder de 0 naar 0,
          // En verlagen we alles boven de 255 naar 255.
          if (new_intensity < 0){
             new_intensity = 0;
          }
          if (new_intensity > 255) {
             new_intensity = 255;
          }
          i_image->setPixel(image_x, image_y, new_intensity);
       }
    }
    return i_image;
```

1.6. Evaluatie

We zullen dit gaan uitvoeren voor verschillende soorten plaatjes en nadat het programma gerunt heeft bekijken we de resultaten in en map waar het nieuwe plaatje is komen te staan. Daarnaast zullen we onze resultaten ook vergelijken met de resultaten van het bestaande programma.