6.y Vaststellen van het Verkeer

Het vaststellen van ingaand en uitgaand verkeer lijkt vrij simpel te zijn. Het is echter helemaal niet zo. Dat komt doordat functies zoals segmentatie en mens-detectie soms foute resultaten opleveren (en dat kan geen systeem vermijden). Het ontstaat wanneer er naar de verzamelde data gekeken wordt en het is niet duidelijk welke metingen goed zijn en welke foutief.

Het is dus noodzakelijk om eerst die fouten te proberen vinden en fixen, en pas daarna de data verder analyseren. In paragraaf 6.y.1 wordt uitgelegd hoe data verzameld wordt, in 6.y.2 hoe fouten geïsoleerd worden en gefixed. Tot slot wordt in paragraaf 6.y.3 de berekening van ingaand en uitgaand verkeer besproken.

## 6.y.1 Verzamelen van Data

Data wordt verzameld in ‘rounds’. Een round begint wanneer de liftdeuren open zijn en eindigt wanneer ze beginnen dicht te gaan. Het heeft geen zin om buiten die interval te werken omdat mensen toch de lift niet in- of uitgaan wanneer de deuren dicht zijn.

Verder wordt data per frame verzameld. Dat houdt in dat per frame de volgende operaties worden gevoerd:

1. Functies die in 6.x besproken zijn worden geroepen om objecten te associëren met het juiste aantal mense die ze bevatten.
2. Objecten ondergaan een paar tests. Die tests checken of ze binnen of buiten de lift liggen.
3. Het totaal aantal objecten binnen en buiten de lift wordt geteld en opgeslaan.

Het is vrij simpel om te check of een object binnen of buiten de lift ligt: Als het boundingbox van het object helemaal binne die van de lift past, dan wordt het meegeteld als ‘in’, zo nee dan als ‘uit’. Het aantal mensen die het object bevat worden opgeteld bij de juiste teller (de in- of uit-teller). De twee tellers worden dan opgeslaan in een variable genaamd *history.*

## 6.y.2 Error-detectie en oplossing

Het variable *history* bevat nu (aan het eind van een ronde) een matrix van per-frame-data. Laten we de lijst van waarden die per-frame de ‘in’-teller representeren de *inStream* noemen. En de lijst van waarden die per-frame de ‘uit’-teller representeren de *outStream.*

De *inStream* en *outStream* kunnen fouten bevatten. Er volgt een voorbeeld: Neem aan dat het volgende de ‘ideale’, foutloze beschrijving is van *history*:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Frame # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| In | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| out | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

Helaas in het praktijk ziet het meer zo uit:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Frame # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| In | 2 | 2 | 0 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| out | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

De error ligt in frame 3, waar de segmentatie en/of labeling per ongeluk 0 opleverden. Deze fout is makelijk te herkennen omdat het een onlogische gat in de reeks vormt. Het is vanzelfsprekend wat de juiste waarde moest zijn.

Dit is hoe het algoritme werkt. Hij scant de *history* en probeert de bovengenoemde gaten te vinden en fixen. Gaten worden gedetecteerd door kettingen van de eenzelfde getal te vinden die omringd zijn door langere kettingen van een andere getal(len).

Het gat wordt vervolgens vervangen een van zijn buren, of een gemiddelde daarvan.

## 6.y.3 Verkeer Berekening

De *inStream* en *outStream* zijn het fout-detectie ondergaan en geven nu een mooie benadering van de werkelijkheid. Het is nu tijd om daaruit de ingaand en uitgaand verkeer af te leiden.

Dit wordt gedaan door afgeleiden. Laten we de afgeleide van de *inStream* ‘*din’* noemen en de afgeleide van de *outStream* ‘*dout’*. We gaan door met het voorbeeld van net:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Frame # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| In | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 |
| Out | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Din | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 |
| Dout | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Opmerkelijk is het feit dat wanneer een persoon uit de lift stapt (frame 6-7) vinden we een complement-duo in *din* en *dout*.

Dus op die manier worden transities gedetecteerd. *Din* wordt doorgelopen totdat een non-zero getal opgenomen wordt. Vervolgens wordt de respectievlijke omgeving in *dout* gecheckt. Als daar een complement te vinden is, dan vond een transitie plaats (uitgaand of ingaand).

Alle gedetecteerde transities worden bij elkaar opgeteld en naar de GUI gestuurd. Daar worden ze zichtbaar voor de gebruiker.