FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Computervisie

Bert De Saffel

Master in de Industriële Wetenschappen: Informatica Academiejaar 2018–2019

Gecompileerd op 22 maart 2019

UNIVERSITEIT
GENT

Inhoudsopgave

Hoofdstuk 1

Inleiding

- Verschil tussen computervisie en computergrafiek:
  - Computervisie is de analyse van een beeld of video om zo de structuur en semantiek te achterhalen.
  - Computergrafiek bouwt aan de hand van structuurregels beelden op.
- Verschil tussen detectie en herkenning:
  - Detectie heeft als doel om te achterhalen of een object aanwezig is.
  - Herkenning heeft als doel om dit object te identificeren.

Hoofdstuk 2

Beeldvorming

2.1 Punten

- Elk 2D punt (x, y) kan gerepresenteerd worden via de **homogene coördinaten** $(\lambda x, \lambda y, \lambda)$ voor $\lambda \neq 0$. Homogene coördinaten worden ook **projectieve coördinaten** genoemd, en bevinden zich in het projectieve vlak $\mathcal{P}^2$.
- Het cartesisch punt $(1, 2)$ kan gepresenteerd worden in homogene coördinaten als $(1, 2, 1)$ of zelfs $(2, 4, 2)$.
- Het origineel cartesisch punt kan bekomen worden te delen door λ:

$$(\lambda x, \lambda y, \lambda) = (\frac{\lambda x}{\lambda}, \frac{\lambda y}{\lambda}, \frac{\lambda}{\lambda}) = (x, y)$$

- Een punt kan door oneindig veel homogene coördinaten gerepresenteerd worden.
- Dit kan eenvoudig uitgebreid worden in drie dimensies:

$$(x, y, z, \lambda)$$

2.2 Lijnen

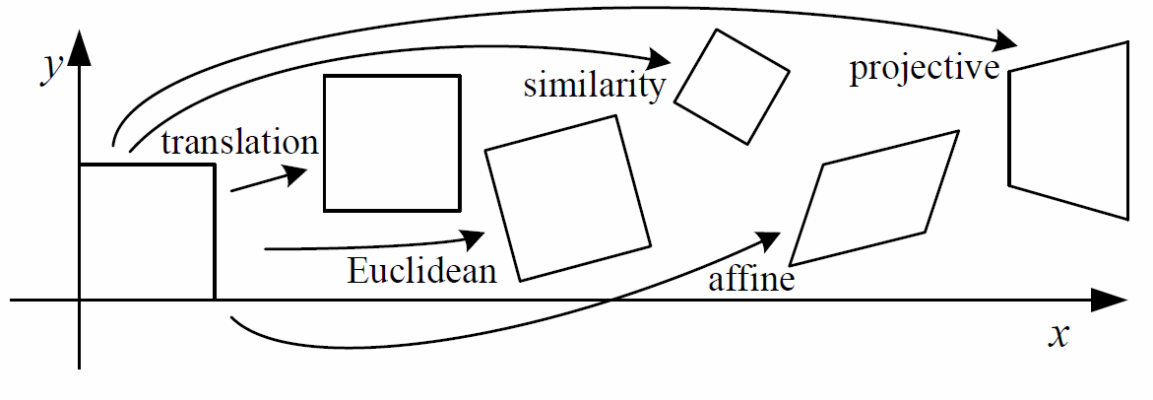
- Een lineaire vergelijking in $\mathcal{R}^2$ kent een aantal problemen:
  - De vergelijking $y = ax + b$ kan geen verticale lijnen voorstellen.
  - De vergelijking $x = ay + b$ kan geen horizontale lijnen voorstellen.
  - De vergelijking $ax + by = 1$ kan geen lijnen door de oorsprong voorstellen.
  - ...
- Een lineaire vergelijking in $\mathcal{P}^2$ heeft de volgende vorm:

$$ax + by + cz = 0$$

  Deze vergelijking kan alle mogelijke lijnstukken voorstellen.
- Deze veralgemening lost ook een aantal geometrische problemen op. In $\mathcal{R}^2$ kunnen twee lijnen ofwel elkaar snijden, ofwel nooit snijden. In $\mathcal{P}^2$ snijden twee lijnstukken altijd, maar kan eventueel in oneindig zijn.

2.3 Twee-dimensionale transformaties

Figuur 2.1 toont alle transformaties mogelijk in de twee-dimensionale ruimte. Elke transformatie



Figuur 2.1: Eenvoudige twee-dimensionale transformaties.

maakt gebruik van één of andere matrix.

2.3.1 Similariteit Transformatie

Deze transformatie combineert eigenlijk de rotatie, translatie en schaling met:

- een schalingsfactor s,
- de rotatiehoek θ,
- en deranslatie t_x en t_y.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2.3.2 Affiene transformatie

Deze transformatie laat toe om een figuur scheef te maken terwijl parallellisme bewaart blijft.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2.3.3 Projectieve transformatie

$$\begin{pmatrix} kx' \\ ky' \\ k \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Transformatie	Matrix	Degrees of Freedom	Behoudt
Translatie	$[I\|t]_{3\times4}$	3	Oriëntatie
Rotatie	$[I\|t]_{3\times4}$	6	Lengte
Similariteit	$[sR\|t]_{3\times4}$	6	Hoeken
Affien	$[A]_{3\times4}$	12	Parallellisme
Projectie	$[\widetilde{H}]_{4\times4}$	15	Rechte lijnen

Tabel 2.1: Hiërarchie van drie-dimensionale transformaties.

2.4 Drie-dimensionale transformaties

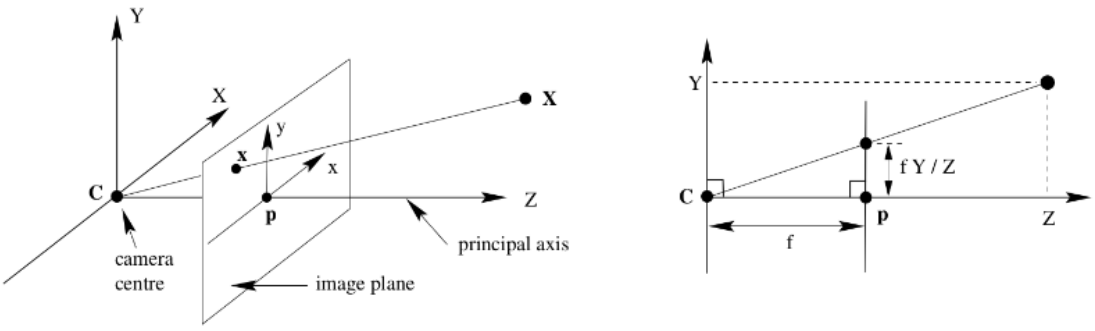
Tabel 2.1 toont de hiërarchie van drie-dimensionale transformaties.

2.4.1 Drie-dimensionale rotatie

- Elke rotatie in drie dimensies is rond een as $\hat{\mathbf{n}}$
- Een verzameling van rotaties rond verschillende assen kan vervangen worden door één rotatie rond één as.
- Rotaties in drie dimensies is niet commutatief.
- Stel:
$$[\hat{\mathbf{n}}]_\times = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix}$$
dan kan elk kruisproduct $\hat{\mathbf{n}} \times \mathbf{v}$ geschreven worden als
$$\hat{\mathbf{n}} \times \mathbf{v} = [\hat{\mathbf{n}}]_\times \mathbf{v}$$
- De rotatiematrix rond een as $\hat{\mathbf{n}}$ en hoek *theta*:
$$R(\hat{\mathbf{n}}, \theta) = \mathbf{I} + \sin\theta[\hat{\mathbf{n}}]_\times + (1 - \cos\theta)[\hat{\mathbf{n}}]_\times^2$$
Dit staat bekend als de **formule van Rodrigues**.
- Elke beweging van een camera kan geschreven worden als de combinatie van een rotatie met rotatiematrix $\mathbf{R}$ en een translatie met translatiematrix T:
$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}$$

2.5 Pinhole model

- Er wordt een oogpunt C verondersteld.
- Op een afstand f bevindt er zicht een projectievlak p. Op een lengte Z en hoogte Y bevindt er zich een punt in drie dimensies (analoog voor het xz-vlak, maar dan met lengte Z en hoogte X).
- Dit driedimensionaal punt kan op het projectievlak geprojecteerd worden door $y = f\frac{Y}{Z}$ en $x = f\frac{X}{Z}$.



Figuur 2.2: Het pinhole model, uitgewerkt voor het yz-vlak.

- Als we nu rekening houden met homogene coördinaten:

$$Z\mathbf{x} = Z\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K_f \mathbf{X}$$

  De deling door Z verdwijnt en de projectie is nu een lineaire transformatie.

2.6 Projectieve transformaties

-

2.7 Lensvertekening

- Lenzen met een grote hoek tonen voorwerpen vaak in gebogen vorm. Er is dus compensatie nodig voor:
  - rechte lijnen te detecteren,
  - circles, rechthoeken, ... te detecteren,
  - beelden te matchen.
- Hoekvertekening kan eenvoudig opgelost worden:

$$x'_c = x_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4)$$
$$y'_c = y_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4)$$

Hoofdstuk 3

Lijndetectie

3.1 Randdetectie

Randdetectie is het detecteren van randen in een beeld. Vaak wordt dit gerealiseerd door hoge verschillen in pixelintensiteit.

3.1.1 Canny randdetector

Het algoritme verloopt als volgt:

1. Er wordt een Gaussische filter met schaal σ gedefinieerd.
   - Dit is nodig omdat elk beeld wel last heeft van ruis.
   - Een Gaussische filter van grootte $(2k+1) \times (2k+1)$ wordt gedefinieerd als:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(- \frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2} \right) \qquad , 1 \leq i, j \leq 2k+1$$

   - Voorbeeld van een 5×5 Gaussische filter met $\sigma = 1.4$:

$$H = \frac{1}{159} \begin{pmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{pmatrix}$$

2. De Sobel operator wordt toegepast om de gradiënt te vinden.
   - Een rand in een beeld kan in meerdere richtingen verschillen. Daarom detecteert het Canny algoritme horizontale, verticale en diagonale randen.
   - De Sobel operator geeft twee matrices met de eerste afgeleide in de horizontale richting (G_x) en verticale richting (G_y).
3. Met deze matrices kan de randsterkte G en richting Θ gevonden worden.

$$G = \sqrt{G_x^2 + G_y^2}$$
$$\Theta = \arctan 2(G_y, G_x)$$

4. Verdun in de richting van de gradiënt.
   - Dit zorgt ervoor dat de interessante lijnen terug fijner worden.
   - Dit is nodig omdat het originele beeld eerst wazig gemaakt wordt in stap 1.
5. Thresholding met T_{strong} en T_{weak}.
   - Pixels met een randsterkte groter dan T_{strong} maken zeker deel uit van een rand.
   - Pixels met een randsterkte lager dan T_{weak} maken zeker geen deel uit van een rand.
   - Pixels met een randsterkte groter dan T_{weak} maar kleiner dan T_{strong} maken deel uit van een rand indien ze een buur hebben die ook deel uit maakt van een rand.
   - Hoe moeten T_{strong} en T_{weak} gekozen worden?
     - De **methode van Otsu** zoekt de T_{strong} die de intra-klasse variantie minimaliseert:

       $$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$$

       met
       - ω_i de kans om tot klasse i te behoren.
       - σ_i de standaarddeviatie van klasse i.
       - t de threshold.
     - T_{weak} wordt dan gelijkgesteld aan de helft van T_{strong}.

3.2 Lijnmodel

De vergelijking van een lijn dat door twee punten (x_1, y_1) en (x_2, y_2) gaat:

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

of

$$(y - y_1)(x_2 - x_1) = (x - x_1)(y_2 - y_1)$$

Een lijn door een punt (p_i, q_i) wordt beschreven als

$$q_i = mp_i + c$$

of

$$c = -p_i m + q_i$$

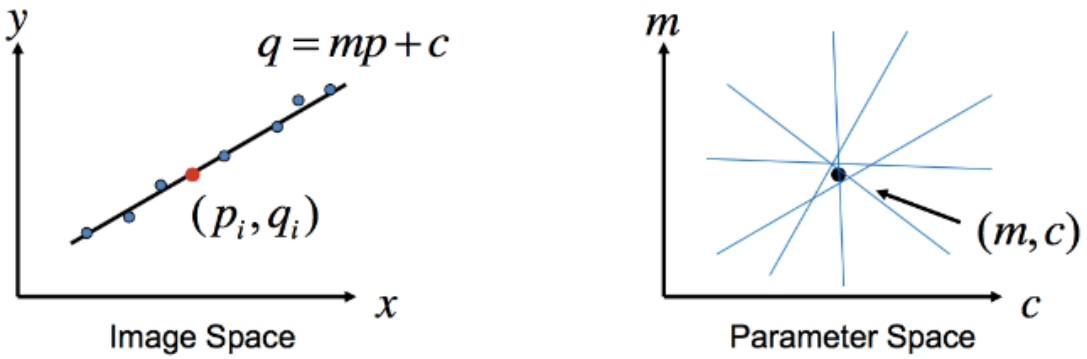
Deze vergelijking definieert een lijn van punten in de parameterruimte. Deze ruimte wordt ook de Hough ruimte genoemd (figuur 3.1).

3.3 Hough transformatie

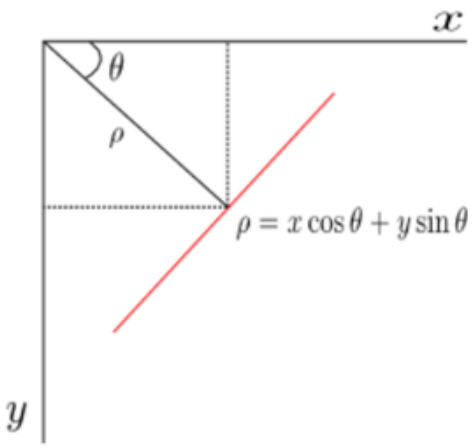
- Een parameterruimte bestaat uit discrete cellen (j, k).
- Elk punt (p_i, q_i) in de beeldruimte draagt een eenheid bij tot de cel (j, k) waarbij

  $$k = \lfloor -p_i \frac{j}{N} + q_i \rfloor$$

  ,$j \in \mathcal{Z}$ en N de resolutie van de parameterruimte.



Figuur 3.1: Punten op een gemeenschappelijke lijn kan vertaald worden naar lijnen door een gemeenschappelijk punt in de Hough ruimte.



Figuur 3.2: Hesse normaalvorm.

- Een cel dat M bijdragen heeft komt overeen met een lijn met M punten.
- ! Probleem met $y = mx + c$:
  - Verticale lijnen kunnen niet voorgesteld worden.
  - De bijdragen worden niet goed gedistribueerd.
- ✓ Normaalvergelijking gebruiken: $\rho = x\cos\theta + y\sin\theta$ (figuur 3.2).
  - $y - mx = c$ kan herschreven worden in normaalvorm door te delen door $\sqrt{1+m^2}$.
  -

3.4 Radon transformatie

Het enige verschil met de Hough transformatie is dat de parameter θ kan gekozen worden.

3.5 Random Sample Consensus (RANSAC)

Het **RANSAC** algoritme wordt gebruikt om inliers te zoeken. Vaak wordt het gevolgd door least-mean-squares fitting.

Het **RANSAC** algoritme werkt als volgt:

1. Kies een willekeurig paar van punten p_1 en p_2.
2. Tel het aantal punten dat binnen een afstand δ van het lijnstuk p_1p_2 ligt.

   De afstand tussen een lijn $ax + by + c = 0$ en een punt (x_0, y_0) wordt gegeven door:

   $$\delta = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$
3. Wanneer het aantal punten een bepaalde threshold overschrijft is de gevonden lijn goed. Anders herstart het algoritme van stap 1.

Belangrijke parameters zijn:

- De afstand δ.
- Het aantal punten dat binnen de afstand moeten liggen.
- Het aantal pogingen tot dat het algoritme stopt. Aangezien het een random algoritme is bestaat de kans dat het nooit stopt. Daarom is er een stopcriteria nodig.

3.5.1 Meerdere lijnen

Op figuur 3.3 wordt een afbeelding getoond waarin de randen van een kubus gedetecteerd zijn. Het algoritme kan eenvoudig uitgebreid worden door elke gevonden lijn te verwijderen zodat deze niet opnieuw kan gevonden worden.

Hoeveel paren van punten zijn er gemiddeld nodig om de eerste lijn te vinden op figuur 3.3?

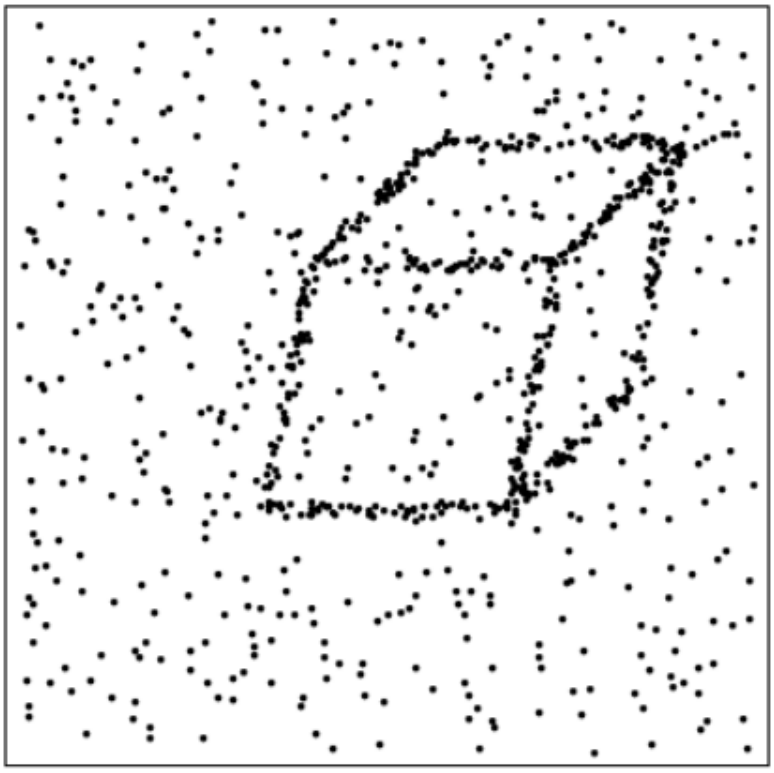
- De kans dat een paar van punten op hetzelfde lijnsegment liggen:

  $$9 \times \frac{50}{950} \times \frac{50}{950} = 0.0249$$
- De kans dat een paar waardeloos is, is dan:

  $$w = 1 - 0.0249$$
- Stel p de kans dat na k pogingen er minstens één goed paar gevonden is. Dan is

  $$1 - p = w^k = (1 - 0.0249)^k$$
- Het logaritme nemen geeft:

  $$k = \frac{\log_2(1 - p)}{\log_2(1 - 0.0249)}$$
- Voor $p = 0.99$ is $k = \pm 182$. Voor $p = 0.999$ is $k = \pm 273$.



Figuur 3.3: Een afbeelding met 9 lijnen waarbij elke lijn ongeveer 50 punten bevat. In het totaal zijn er 500 punten.

3.5.2 Optimalisaties

Voor- en nadelen:

- ✓ In staat om een goede benadering te geven.
- ✓ Weinig geheugen nodig. Enkel de twee gekozen punten moet opgeslagen worden.
- ✓ Kan ook werken voor cirkels, affiene transformaties, ...
- ! De parameters moeten handmatig gekozen worden en is vaak moeilijk te bepalen.
- ! Er is geen bovengrens voor het aantal iteraties dat nodig zijn om uiteindelijk een resultaat te krijgen?
- ! Het algoritme vraagt in de praktijk veel meer tijd dan de theoretische tijd.
- ! Het algoritme is niet-deterministisch. Elke uitvoering van het algoritme levert andere resultaten op.

Een aantal optimalisaties:

- **Preventieve :**
- **Lokale optimalisatie:**

- Binnen elke iteratie kunnen lokale optimalisaties uitgevoerd worden op het 'best-so-far' paar.
- Na het vinden van een 'best-so-far' paar, kunnen verschillende strategieën toegepast worden:
  1. Neem alle punten met een afstand kleiner dan $K\theta$ van de huidige lijn en zoek een nieuwe betere lijn. Blijf dit doen tot dat $K = 1$.
  2. Neem nieuwe paren uit de inliers. ToDo: snap ik nie
  3. Combinatie van de vorige twee strategieën.
- Hoeveel keer is een lokale optimalisatie nodig?
  - De kans dat het $k-$de paar het 'best-so-far' is, bedraagt $1/k$. Het gemiddeld aantal 'best-so-far' paren binnen k baren is dan:

$$\sum_{n=1}^{k} \frac{1}{n} \leq \int_{1}^{k} \frac{1}{x}\, dx + 1 = \log_2 k + 1$$

  - Lokale optimalisatie is dan $O(\log_2 k)$ maal nodig.
  - ✓ Door het logaritmisch karakter zal het weinig verschil uitmaken op de uitvoeringstijd van het algoritme.

- **Gerichte sampling:**
  - Er kan met Hough of Radon kandidaten gevonden worden.
  - Met RANSAC worden de kandidaten dan verfijnd.

Hoofdstuk 4

ROC curves

4.1 Inleiding

- ROC = Receiver Operating Characteristic.
- Een eenvoudige manier om classifiers te evalueren.

4.2 Classifiers

- Het toekennen van een klasse aan een object uit een verzameling van voorgedefinieerde klassen.
- Een **Binaire classifier** kent slechts twee klassen.

4.3 Binaire classifiers

- Stel twee klassen α en β. We zijn nu geïnteresseerd of een object in klasse α zit.
- Een **False Positive** komt voor wanneer het object de klasse α krijgt, terwijl hij β is.
- Een **False Negative** komt voor wanneer het object de klasse β krijgt, terwijl hij α is.
- Binaire **confusion matrix**:

Klasse	Voorspelde klasse	
	positief	negatief
positief (#P)	#TP	#FN = #P - #TP
negatief (#N)	#FP	#TN = #N - #FP

- De **true positive rate (TPR)** is:

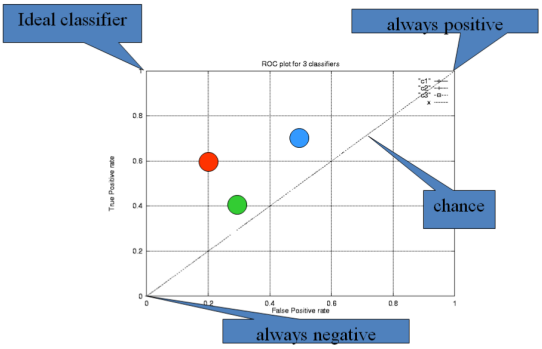
$$TPR = \frac{\#TP}{\#P}$$

- De **false positive rate (FPR)** is:

$$FPR = \frac{\#FP}{\#N}$$

Klasse	Voorspelde klasse		Voorspelde klasse		Voorspelde klasse	
	positief	negatief	positief	negatief	positief	negatief
positief	40	60	70	30	60	40
negatief	30	70	50	50	20	80

Tabel 4.1: Elke kleur stelt een verschillende classifier voor die dezelfde objecten classificeert.



Figuur 4.1: De TPR en FPR van tabel 4.1 gevisualiseerd.

- Welke classifier is het beste?
  - $TPR = 0.4, FPR = 0.3$
  - $TPR = 0.7, FPR = 0.5$
  - $TPR = 0.6, FPR = 0.2$
  - Figuur 4.1 stelt deze waarden visueel voor.
  - De rode classifier is zeker beter dan de groene. De TPR van de rode classifier is niet beter dan de TPR van de blauwe classifier, maar de TPR/FPR verhouding is wel beter.