Objekterkennung am Beispiel des Viola-Jones-Objektdetektors

Medieninformatik IL

Andreas Unterweger

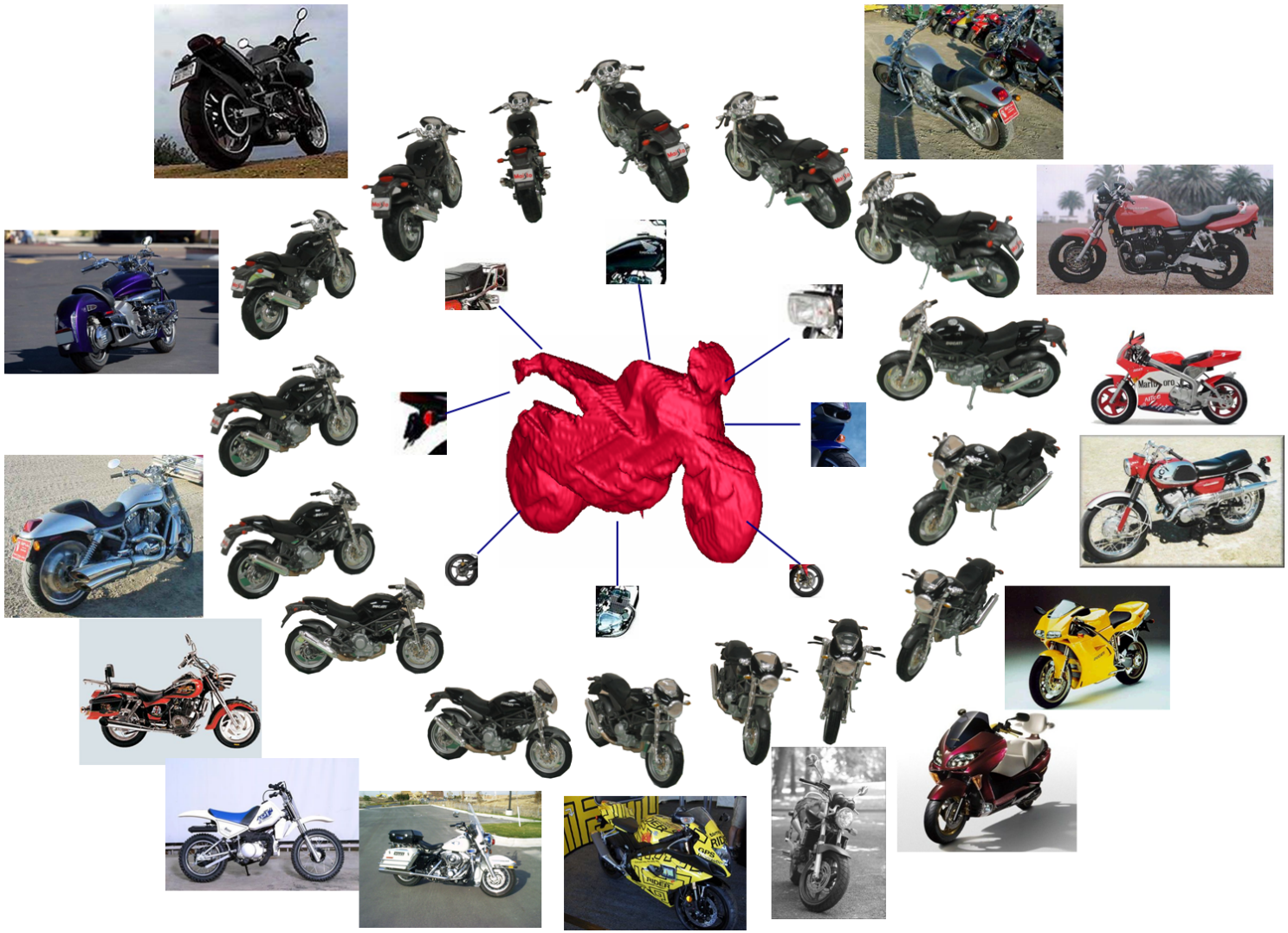
Vertiefung Medieninformatik
Studiengang ITS
FH Salzburg

Wintersemester 2014/15

Objekterkennung (engl. *object detection*)

- Ziel: Finden eines bestimmten Objektes bzw. mehrerer bestimmter Objekte in einem Bild
- Wichtiger Spezialfall: Objekt**klassen**erkennung
  - Klasse: Zusammengehörige Gruppe von Objekten mit gemeinsamen Eigenschaften (z.B. Dreiecke: 3 geschlossen verbundene gerade Linien)
  - Ziel: Erkennung aller Objekte der Klasse
  - Nebenkriterium: Keine Erkennung anderer Objekte
- Menschliche Wahrnehmung ist gut bei Objekt(-klassen-)erkennung
- Herausforderungen
  - Fähigkeiten der menschlichen Wahrnehmung digital nachbilden
  - Eindeutige Objektklassenbeschreibungen finden
  - Blickwinkel und Verzerrungen

Objektklassenbeispiel



Quelle: Yan, P. und Khan, S. M.: 3D Model based Object Class Detection in An Arbitrary View. `http://vision.eecs.ucf.edu/projects/3D_Model_based_Object_Detection/ObjectDetection.html` (28.6.2014), 2007.

Begriffe

- Positive (Treffer): Erkennung durch Algorithmus
- Negative: Nichterkennung durch Algorithmus
- Korrektheit der Erkennung (boolesch): True/false

	Erkannt	**Nicht erkannt**
Ist Objekt	True Positive	False Negative
Ist kein Objekt	False Positive	True Negative

- Typische Abkürzungen: TP, FP, TN, FN

Metriken I

- Präzision (engl. *precision*): Wahrscheinlichkeit, dass ein Treffer tatsächlich ein Objekt ist

$$p = \frac{TP}{TP + FP}$$

- Sensitivität (engl. *sensitivity*): Wahrscheinlichkeit, dass ein Objekt einen Treffer verursacht

$$sns = \frac{TP}{TP + FN}$$

- Spezifizität (engl. *specificity*): Wahrscheinlichkeit, dass „kein Objekt“ keinen Treffer verursacht

$$spc = \frac{TN}{FP + TN}$$

Metriken II

- Genauigkeit (engl. *accuracy*): Gütemaß für korrekte Trefferzuordnung (Treffer bei Vorhandensein eines Objektes und kein Treffer bei Nichtvorhandensein eines Objektes)

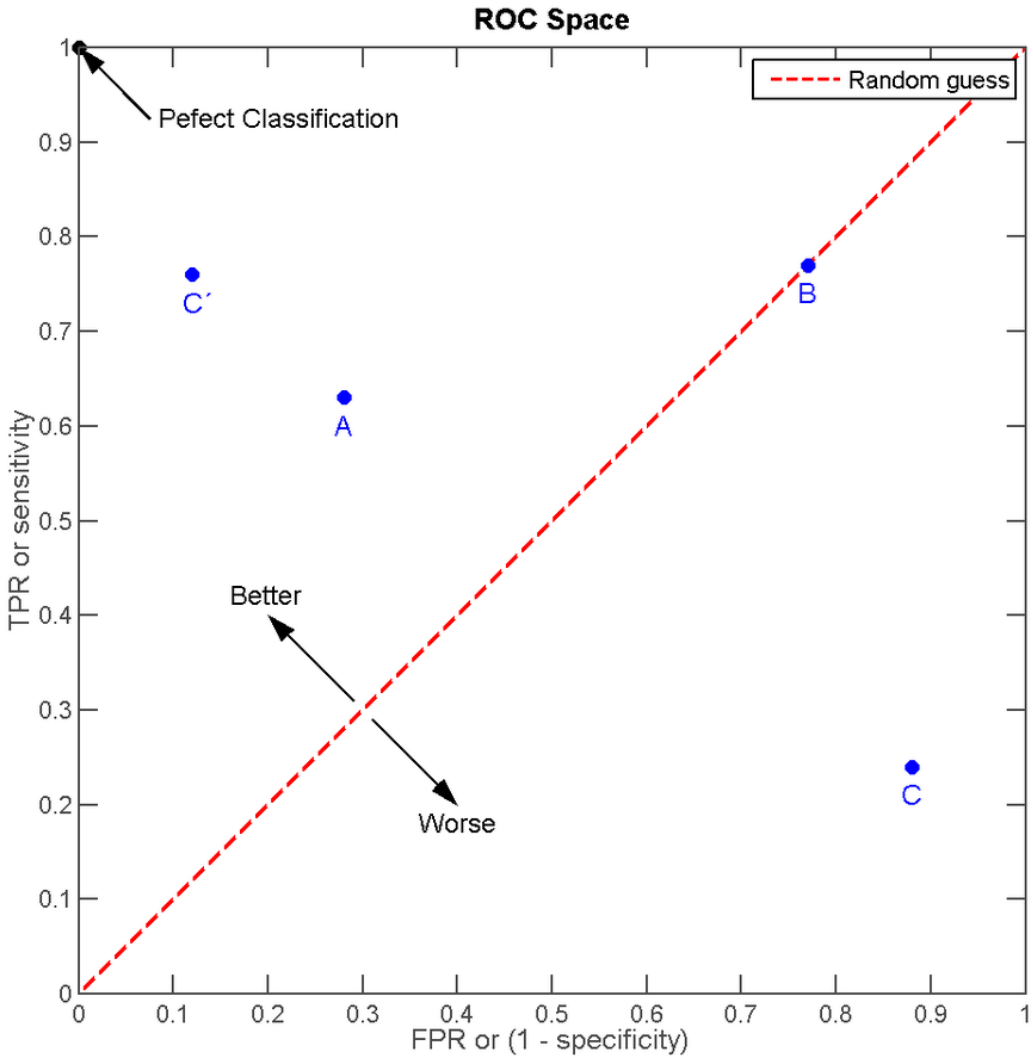
$$a = \frac{TP + TN}{TP + FP + TN + FN}$$

- F_1-Score: Harmonisches Mittel aus Präzision und Sensitivität

$$\begin{aligned} F_1 &= \frac{2}{\frac{1}{p} + \frac{1}{sns}} = \frac{2}{\frac{1}{\frac{TP}{TP+FP}} + \frac{1}{\frac{TP}{TP+FN}}} = \frac{2}{\frac{TP+FP}{TP} + \frac{TP+FN}{TP}} = \\ &= \frac{2}{\frac{2 \cdot TP + FP + FN}{TP}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \end{aligned}$$

Metriken III

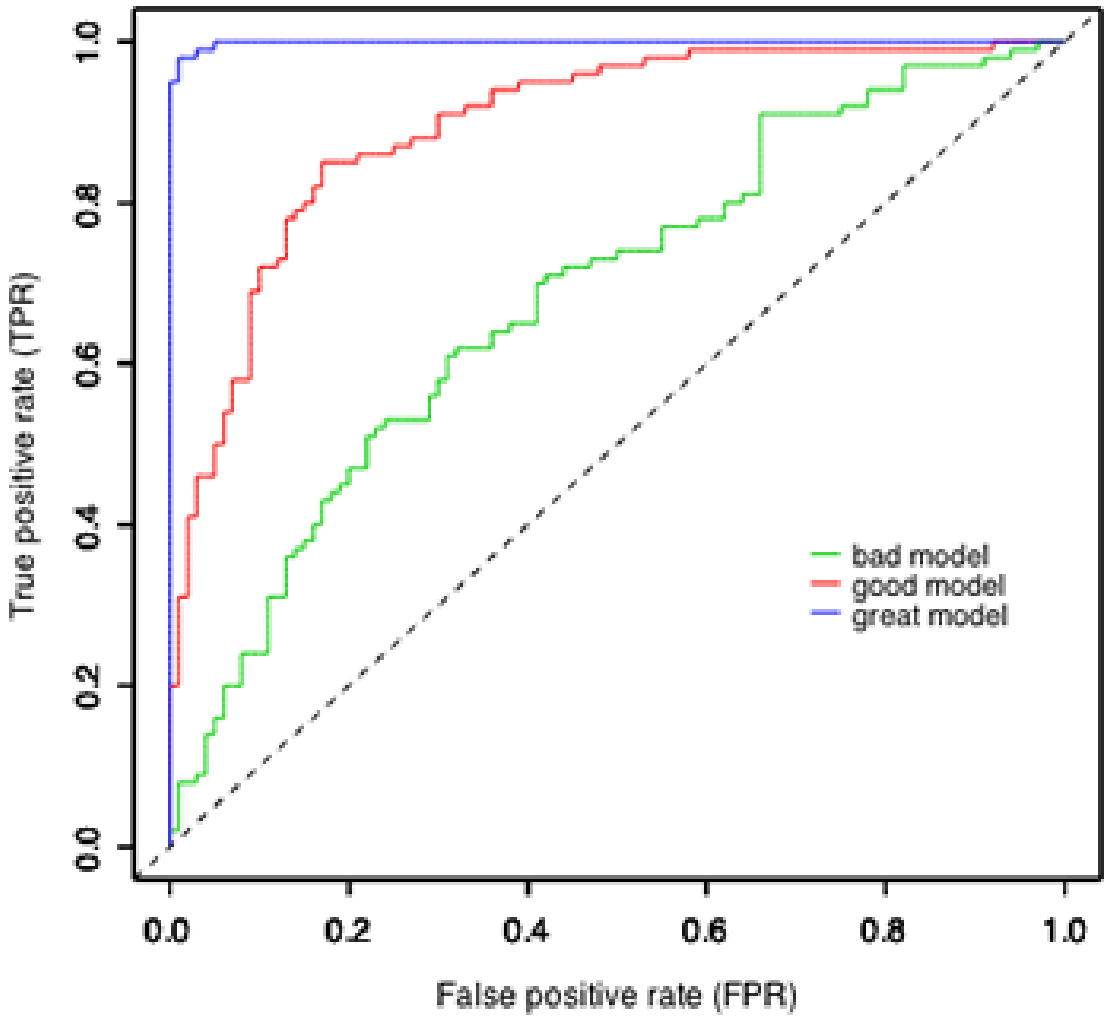
Receiver Operating Characteristic (ROC):



Quelle: `http://commons.wikimedia.org/wiki/File:ROC_space.png`

Metriken IV

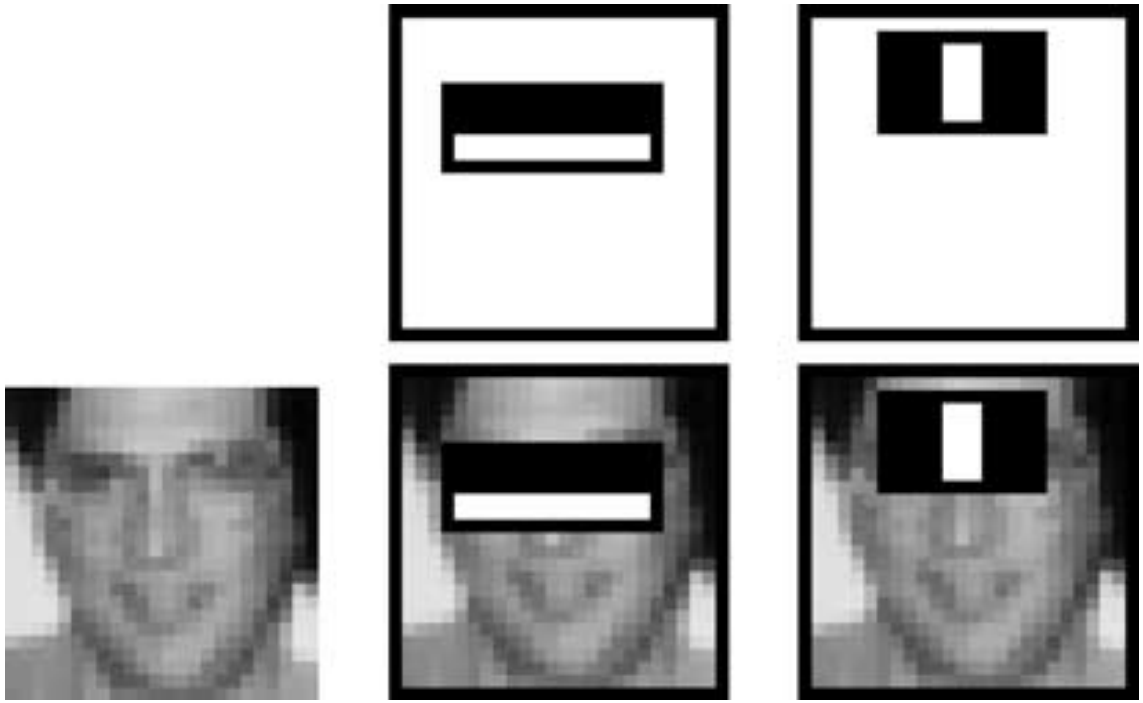
ROC-Kurve: Trefferrate bei variabler Falscherkennungsrate:



Quelle: Weiss, J.: Lecture 22 – Wednesday, November 10, 2010. `http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture22.htm` (28.6.2014), 2010.

Überblick zum Viola-Jones-Objektdetektor

- Merkmale: Abfolgen von Hell-Dunkel-Unterschieden im Objekt
- Beispiel: Gesichtserkennung
  - Vertikal: Augen: Dunkel, Wangen: Hell
  - Horizontal: Augen: Dunkel, Nase: Hell, Augen: Dunkel
  - ...



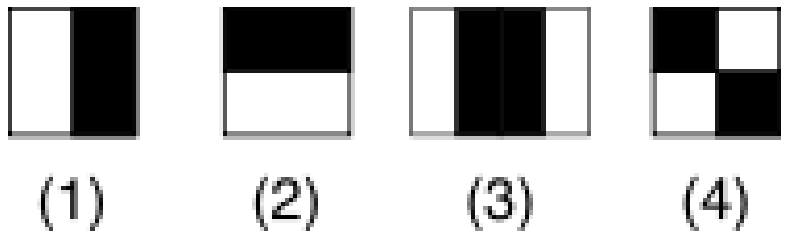
Quelle: Viola, P. and Jones, M. J.: Robust Real-Time Face Detection. International Journal of Computer Vision 57(2), pp. 137–154, 2004.

Merkmale I

- Rechteckige Bereiche
  - Pixel in weiß markierten Bereichen addieren
  - Pixel in schwarz markierten Bereichen subtrahieren

  → Summe ist in gesuchten Objekten hoch, ansonsten niedriger

  → Finden von Objekten über Schwellwert möglich
- Mehrere verschiedene Merkmale:

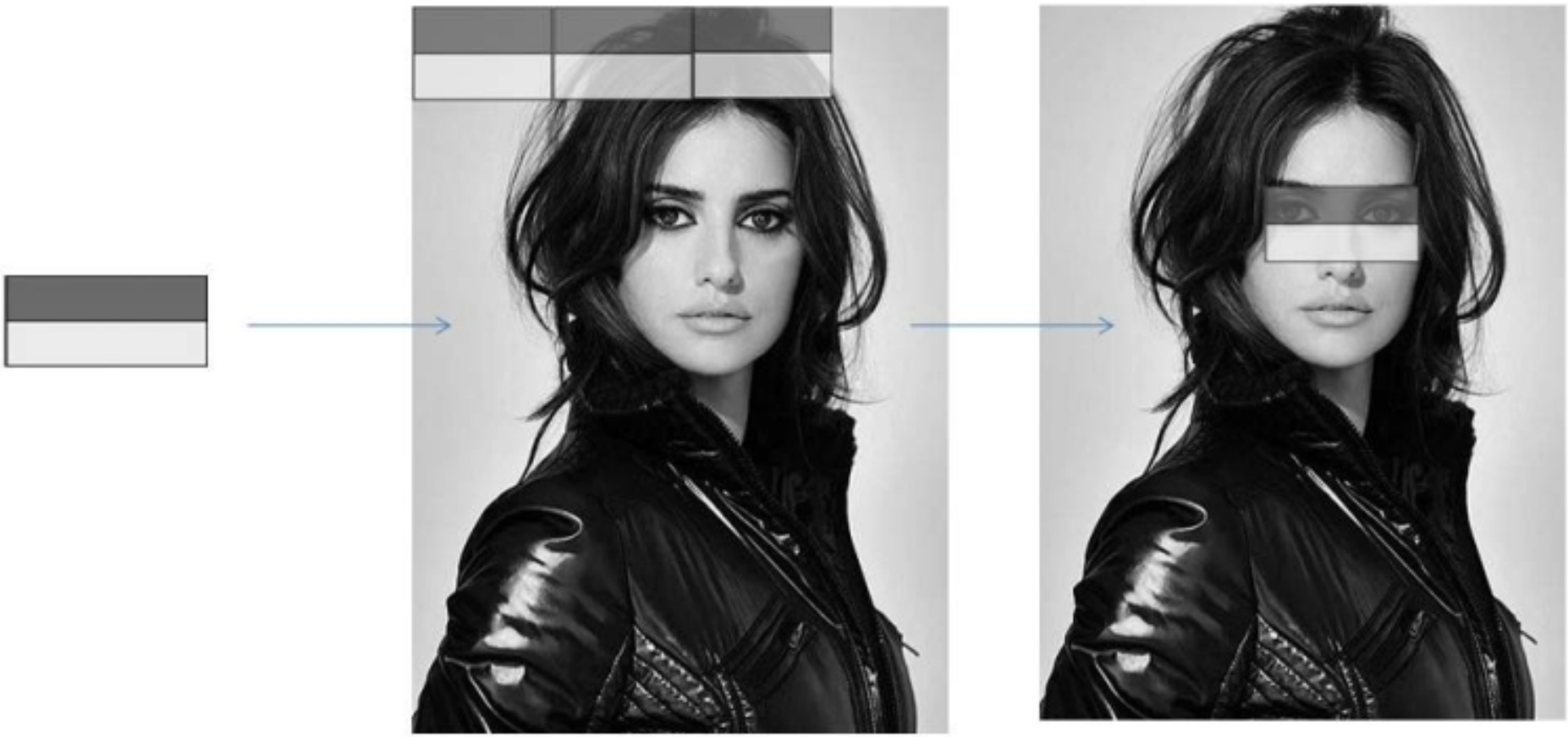


Quelle: `http://commons.wikimedia.org/wiki/File:VJ_featureTypes.svg`

- Suche in einem Bild: Verschieben eines Rechteckfensters an alle möglichen Positionen; dann jeweils Merkmalsanwendung

Merkmale II

- Beispielhafte Rechteckfensterpositionen (vereinfachte Darstellung für ein Merkmal; Animation: `http://vimeo.com/12774628`):



Adaptiert von: Dev, R.: [Virtual Reality: Tutorial #4] Face Detection & Face Recognition. `http://www.durofy.com/virtual-reality-face-recognition/` (28.6.2014), 2012.

Einschub: Integralbilder I

- Häufige Addition/Subtraktion von Pixeln ist zeitaufwändig
- Viele Additionen/Subtraktionen werden mehrfach ausgeführt

→ Vereinfachung durch Integralbilder (engl. *integral images*)

- Integralbild II: Jedes Pixel entspricht der Summe aller darüber- und links davon liegenden Pixel des Ausgangsbildes I mit Auflösung $m \cdot n$:

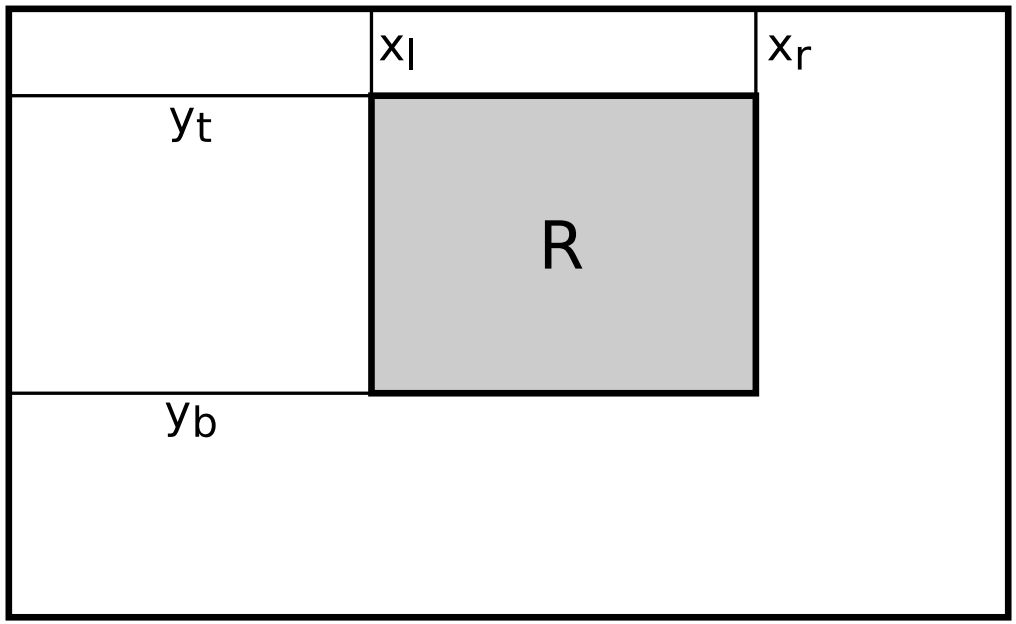
$$II(x,y) = \sum_{x'=0}^{x-1} \sum_{y'=0}^{y-1} I(x',y'), 0 \leq x \leq n, 0 \leq y \leq m$$

- Praktische Nachteile von Integralbildern:
  - Sind um je ein Pixel breiter und höher als ihre jeweiligen Ausgangsbilder
  - Der Wertebereich jeder Integralbildpixel ist deutlich höher (abhängig von der Originalbildgröße!) als der der Ausgangsbildpixel

Einschub: Integralbilder II

- Erlauben Berechnung der Summe S_R aller Bildpixel in einem rechteckigen Bereich R in konstanter Zeit:

$$S_R = II(x_r, y_b) - II(x_l, y_b) - II(x_r, y_t) + II(x_l, y_t)$$

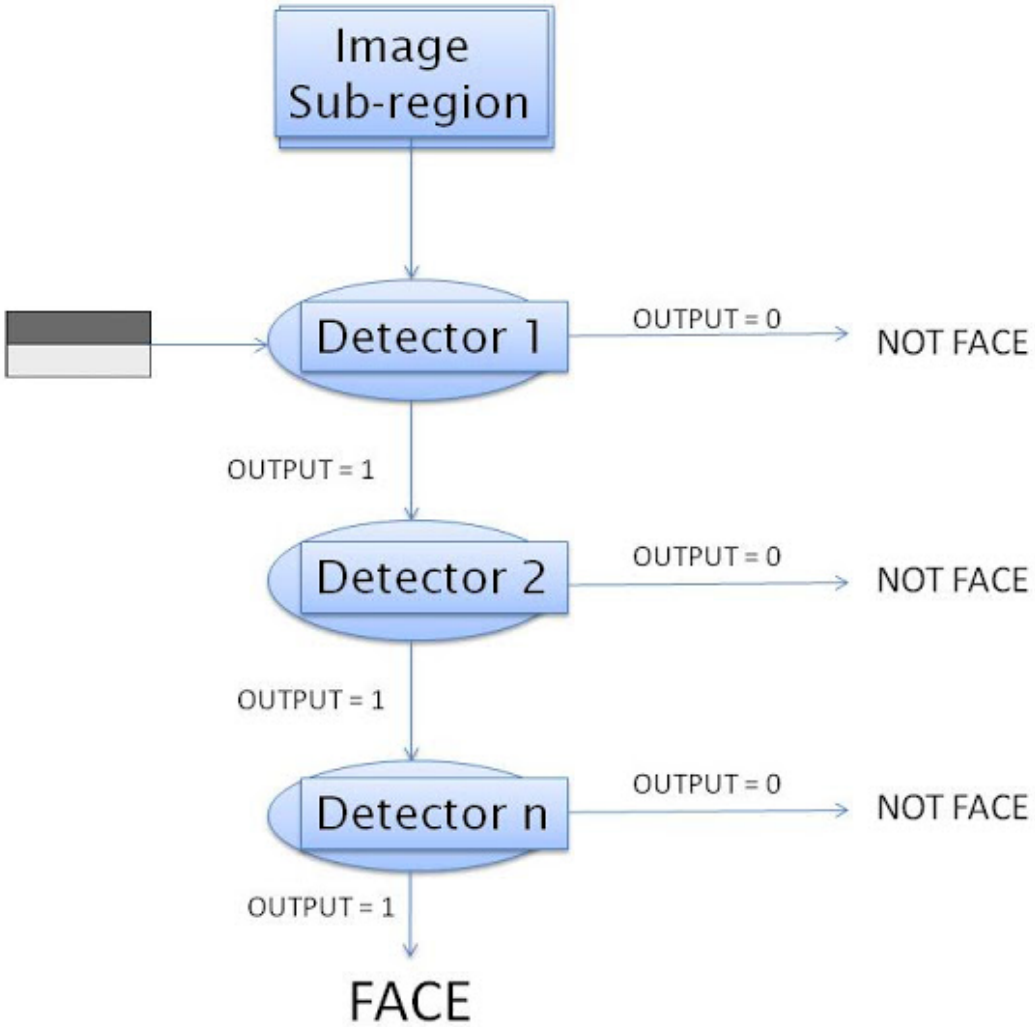


Adaptiert von: Crow, F. C.: Summed-area tables for texture mapping. In Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '84, pp. 207–212, New York, NY, USA. 1984.

Detektorkaskadierung I

- Ein Detektor pro Feature
- Kaskadierung (Hintereinanderschaltung) von Detektoren:
  - Liefert ein Detektor 0 („kein Objekt gefunden“) zurück $\rightarrow$ Abbruch
  - Liefern alle Detektoren 1 („Objekt gefunden“) $\rightarrow$ Objekt erkannt
- Entwurfsprinzipien:
  - Hohe Sensitivität pro Detektor
  - Weniger aufwändige Detektoren weiter vorne in Kaskade reihen
- Vorteile:
  - Sehr schnell
  - FP-Rate jedes Detektors kann relativ hoch sein (Spezifizität niedrig)
- Nachteile
  - TP-Rate jedes Detektors muss relativ hoch sein
  - FN-Rate jedes Detektors muss relativ niedrig sein
  - Objektähnliche Muster bedürfen mehr Zeitaufwand durch Kaskade

Detektorkaskadierung II

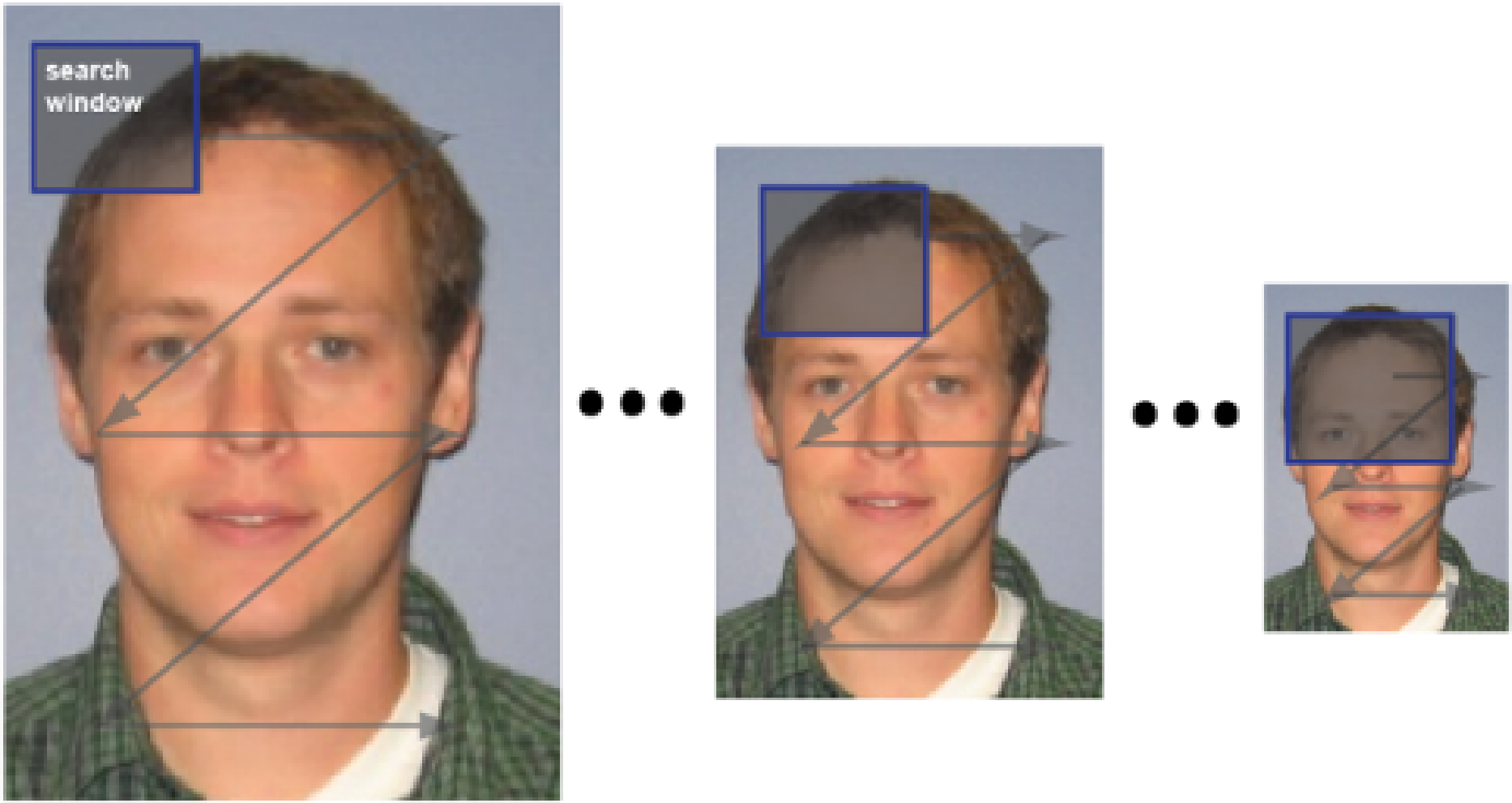


Adaptiert von: Dev, R.: [Virtual Reality: Tutorial #4] Face Detection & Face Recognition. `http://www.durofy.com/virtual-reality-face-recognition/` (28.6.2014), 2012.

Skalierung I

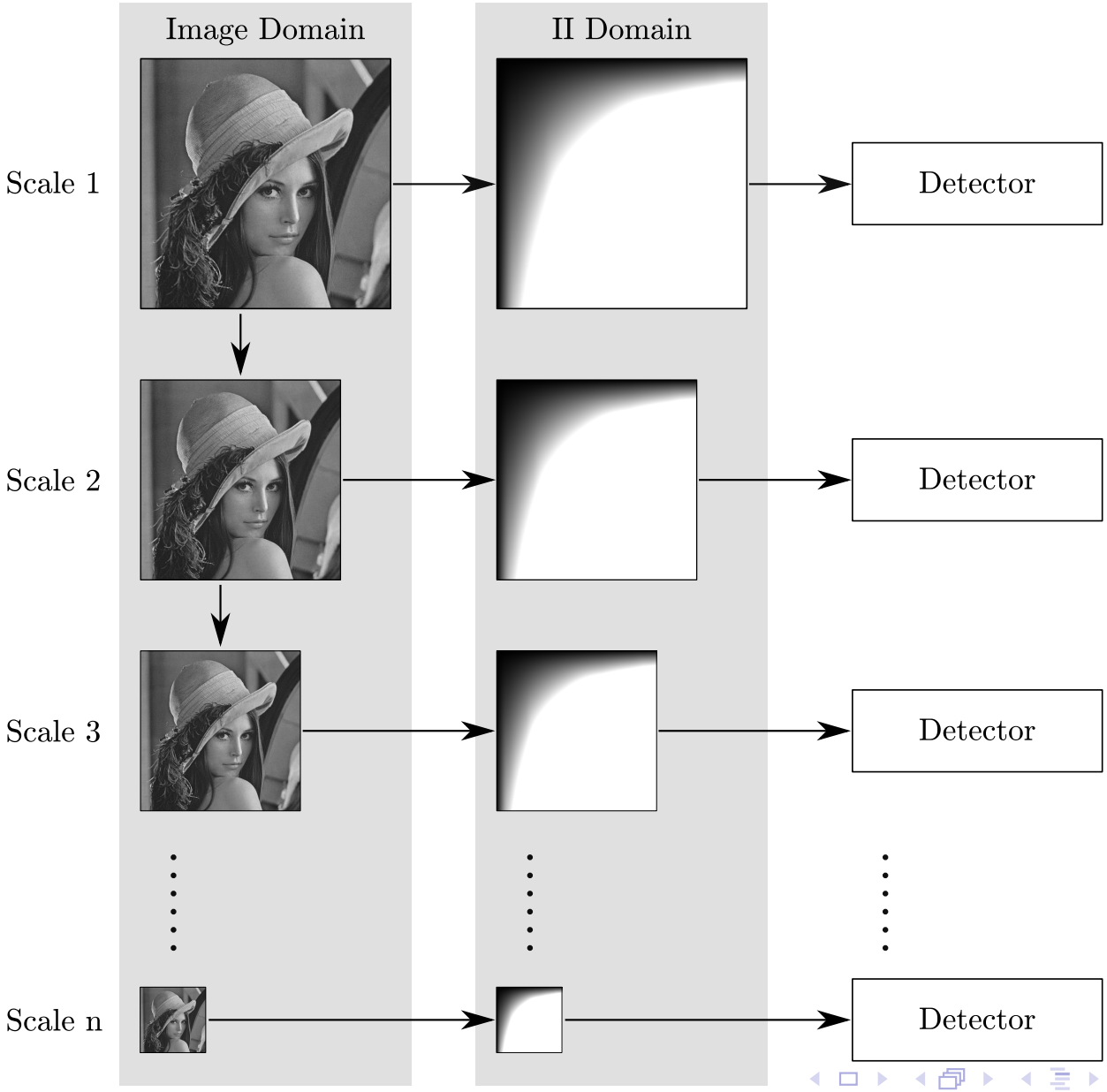
- Objekte in Bildern haben unterschiedliche Größen $\rightarrow$ Erkennung in mehreren Auflösungen (engl. *Multi-scale detection*); Prinzip:
  - Beginne in Originalauflösung
  - Berechne Integralbild
  - Erkenne Objekte mittels Detektorkaskade
  - Reduziere Auflösung um einen konstanten Faktor und wiederhole (Abbruchkriterium ist typischerweise Bildauflösung = Merkmalgröße)
- Praktische Alternative: Features skalieren (oft weniger aufwändig)
- Parameter:
  - Skalierungsfaktor pro Schritt (beeinflusst Gesamtgeschwindigkeit)
  - Toleranz zum Zusammenfassen erkannter Objekte in verschiedenen Auflösungen (auch bei Rechteckfenstern innerhalb einer Auflösung)

Skalierung II



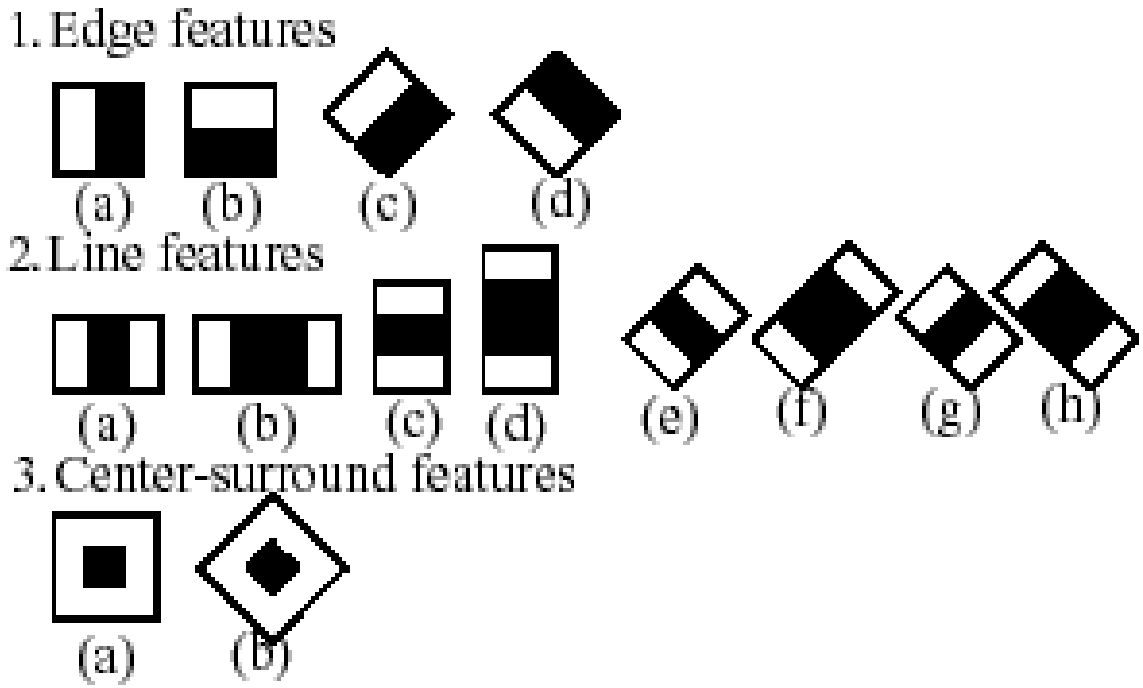
Quelle: MathWorks: vision.CascadeObjectDetector System object.
http://www.mathworks.de/de/help/vision/ref/vision.cascadeobjectdetector-class.html (28.6.2014), 2014.

Skalierung III



Erweiterungen

- Erweiterung: Diagonale Features (nach Lienhart und Maydt):



Quelle: opencv dev team: vision.CascadeObjectDetector System object. `http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html` (28.6.2014), 2014.

- Realisierung mittels diagonaler Integralbilder (ohne Details)

Danke für die Aufmerksamkeit!

Fragen?