Übung zur Vorlesung Videoanalyse

Blatt 2 – Analyse der Kamerabewegung

Aufgabe 1 – Klasse CameraModel

Entwerfen Sie eine Klasse *CameraModel*, die folgende Funktionen zur Verfügung stellt. Die ersten beiden Funktionen transformieren die Position (*srcX/srcY*) eines Pixels in eine neue Position (*destX/destY*).

- **`void transformPointCylindric (double srcX, double srcY, double &destX, double &destY);`**

  Die erste Funktion bildet das zylindrische Kameramodell ab. Nehmen Sie dabei eine Brennweite von 1.0 an. Die Bildkoordinaten (*srcX/srcY*) sollen vor der Umrechnung auf das Intervall [-0.5, 0.5] normalisiert werden. *destY* wird normalisiert, indem der transformierte Wert mit der Bildhöhe gewichtet und ½ der Bildhöhe hinzuaddiert wird. *destX* wird mit der Bildbreite gewichtet und ½ der Bildbreite hinzuaddiert.

- **`void transformPointEightParameter (double srcX, double srcY, double &destX, double &destY);`**

  Beim 8-Parameter-Modell ist eine Normalisierung der Koordinaten nicht notwendig. Definieren Sie die acht Parameter innerhalb der Klasse. Welche Anfangswerte sollten für die 8 Parameter gewählt werden?

- **`void transformCylindric (Image &src, Image &dest);`**

  Entwerfen Sie eine Funktion, um ein Bild *src* mit dem zylindrischen Kameramodells in das Bild *dest* zu transformieren. Laden Sie anschließend das Bild *test1*, transformieren Sie es und speichern es. Wie hat sich das Bild verändert?

- **`void transformEightParameter (Image &src, Image &dest);`**

  Entwerfen Sie eine Funktion, um ein Bild *src* mit dem 8-Parameter-Modell in das Bild *dest* zu transformieren. Laden Sie das Bild *test1* und transformieren Sie es mit folgenden Parametern:

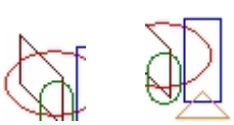
  ```
  a)   a11=0.985,  a22=0.985,  a12=0.174, a21=-0.174
  b)   b1= 2 * 10e-4
  c)   tx=100, ty=100,  a11=-1,  a22=-1
  ```

  Wie sehen die Ergebnisbilder aus?

Aufgabe 2 – Klasse CameraModel

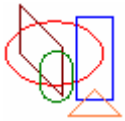
Erweitern Sie die Klasse CameraModel, um die Verschiebung zweier Bilder zu berechnen (*tx* und *ty* beim 8-Parameter Modell). Entwickeln Sie dazu folgende Funktion:

- `void getTranslation (Image &img1, Image &img2, int &tx, int &ty);`
- Implementieren Sie eine **vollständige Suche**, bei der Sie das zweite Bild *img2* über das gesamte erste Bild verschieben. Nehmen Sie als Differenzmaß die Summe der absoluten Differenzen für alle überlappenden Pixel beider Bilder. Speichern Sie die Verschiebung mit der minimalen Differenz in (*tx*,*ty*).
- Testen Sie die Funktion mit folgenden beiden Bildern (*test5a, test5b*):



test5a

test5b



zusammengesetztes Bild

- Berechnen Sie die Werte (*tx*,*ty*) für die beiden Bilder. Erzeugen Sie ein Ausgabebild in passender Größe, setzen Sie die Bilder automatisch zusammen und speichern Sie das Hintergrundbild. Wie viele Pixel werden bei der vollständigen Suche miteinander verglichen?
- Die Bilder *test6a*, *test6b* und *test6c* wurden mit dem 8-Parameter-Modell passend transformiert. Unbekannte Bildbereiche wurden dabei durch gelbe Pixel ersetzt (RGB: 255,255,128). Erzeugen Sie mit Hilfe der Klasse *CameraModel* ein Panoramabild aus den drei Bildern. Gelbe Pixel sollen bei der Berechnung der Verschiebung und beim Zusammensetzen des Hintergrundbildes nicht berücksichtigt werden.
- Wie lange benötigt das Programm, um das Hintergrundbild zu erzeugen? Warum? Schlagen Sie Optimierungen vor.

Aufgabe 3 – Berechnung der Kamerabewegung

- Warum werden Bewegungen in Videos analysiert?
- Nennen Sie drei Verfahren zur Modellierung der Kamerabewegung? Wo liegen die wesentlichen Unterschiede? Nennen Sie Vor- und Nachteile der Ansätze?
- Gegeben sind jeweils vier Bewegungsvektoren. Berechnen Sie für die Vektoren das entsprechende Kameramodell. Sie können zur einfacheren Berechnung annehmen, dass ***keine*** perspektivische Verzerrung vorliegt. Welche Kamerabewegung liegt vor?
  - (a) (10/10)→(40/10), (90/10)→(40/90), (10/40)→(10/10), (90/40)→(10/90)
  - (b) (10/10)→(20/15), (90/10)→(80/15), (10/40)→(20/35), (90/40)→(80/35)