Exercise: Image and Video Processing

Sheet 2 – Image Registration

Exercise 1 – Class CameraModel

Design a class *CameraModel* which implements the following methods. The first two functions transform the position (*srcX/srcY*) of a pixel to a new position (*destX/destY*).

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

  The function uses the cylindrical camera model to transform a pixel. Assume a focal length of *1.0*. The image coordinates (*srcX/srcY*) should be normalized to the interval [-0.5, 0.5] before the transformation. *destY* is normalized by multiplying the transformed value with the image height and adding ½ of the image height. *destX* is multiplied with the image width and ½ of the width is added.

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

  Normalization is not required in case of the 8 parameter model. Declare the 8 parameters in the class. What are suitable default values?

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

  Implement a function to transform image *src* to image *dest* based on the cylindrical camera model. Load image *test1*, transform it, and store it. What are the differences?

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

  Implement a function to transform image *src* to image *dest* based on the 8 parameter model. Load image *test1* and transform it with the following parameters:

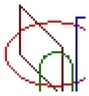
  ```
  a)    a11=0.985,  a22=0.985,  a12=0.174, a21=-0.174
  b)    b1= 2 * 10e-4
  c)    tx=100, ty=100,  a11=-1,  a22=-1
  ```

  How does image *dest* look like?

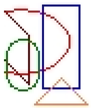
Exercise 2 – Class CameraModel

Add the following function to the class *CameraModel*, to calculate the optimal translation between two images (*tx* and *ty* describe the translation).

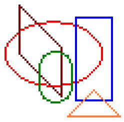
- `void getTranslation (Image &img1, Image &img2, int &tx, int &ty);`
- Implement a **full search algorithm**, and shift the second image *img2* to all positions of the first image. Calculate the sum of absolute difference as similarity metric for all overlapping pixels in both images. Store the translation with minimum difference in (*tx* ,*ty*).
- Test the function with the following images (*test5a*, *test5b*):



test5a



test5b



merged image

- Calculate the values (*tx* ,*ty*). Create a destination image with a suitable size, merge the images into the destination image and store it. How many pixels are compared?
- The images *test6a*, *test6b* and *test6c* have been transformed by using the 8 parameter model. The color of undefined image regions is yellow (RGB: 255,255,128). Create a panoramic image by using `getTranslation()`.Calculate the translation and merge the images by ignoring all yellow pixels.
- What is the run time of your program? Why? Run your program with the images *test6a_small*, *test6b_small* and *test6c_small*. How could you improve your program?

Exercise 3 – Parameters of the camera model

- What information do you get by analyzing the motion in videos?
- Name 3 techniques to model camera motion. What are the differences (advantages / disadvantages)?
- Assume the following 4 motion vectors. Calculate the 8 parameters of the camera model (hint: you can ignore perspective transformations). How many vectors do you need to calculate the model? What kind of camera motion do the vectors represent?
  - (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)