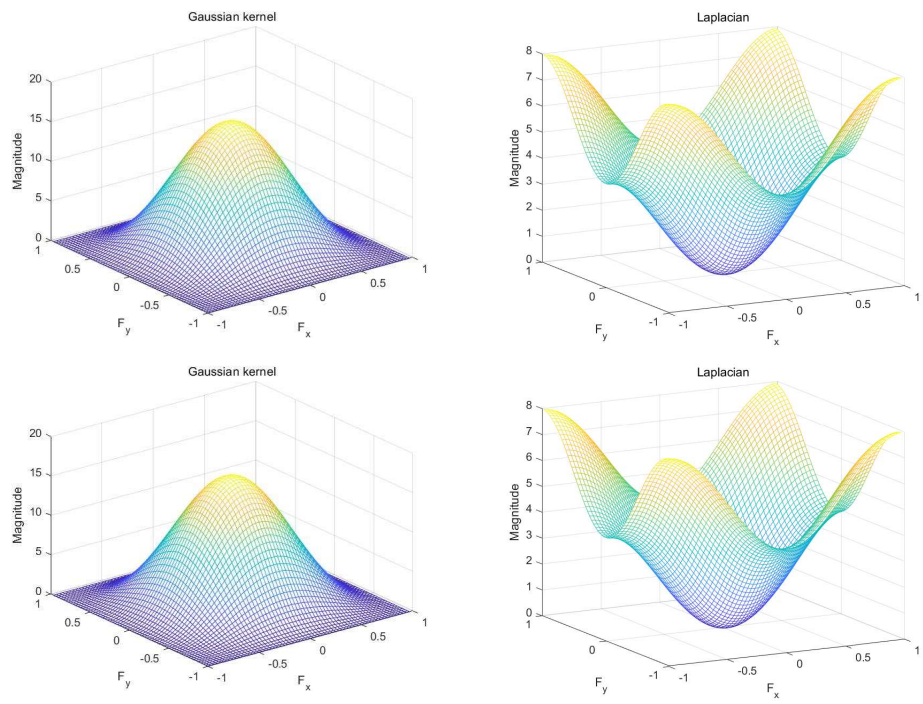
Drucken

Main

4.29.2020

- **Calculation of Imagegradient**

1. The Sinc function is not used for interpolation for image gradient calculation, although it is an ideal low-pass filter in the frequency domain. The reason for this is that the gradient of sinc function spread slowly and infinitely. Therefore the gradient is influenced by several post-values.
2. When the image is folded with a low pass filter such as a Gaussian filter, the filter is normally normalized to get the energy from the image. When folding with the Sobel filter, standardization is not so necessary because the Solber filter is designed to calculate the vertical and horizontal gradient.
3. In addition to the solber filter, there are other highpass filters that are widely used in image processing. For example, the Laplacian filter can as a discrete convolution kernel works as an edge detector used to compute the second derivatives of an image, measuring the rate at which the first derivatives change.
4. the frequency response of different kernels: Gaussian kernel, Sober filter(horizontally and vertically), and Laplacian filter using Matlab. The Gaussian kernel is a lowpass filter in both horizontal and vertical directions and attenuates the high-frequency components, which are the detailed information of the image. Solber edge detectors are high-pass in one direction while low-pass in the other direction, according to the application. Laplacian filter is high-pass in both directions.



- **Harris corner und Edgedetector**

1. Relationship between the Sobel filter and Harris corner and edge detector. The Sobel filter is used to calculate the image gradient in the X and Y direction. This allows the Harris Matrix to be calculated, using its eigenvalues to detect the corner, edge, and surface.
2. Eigenvalues of Harris Matrix: The basic idea of Harris corners and edge detector is to define an image segment and to calculate the change of this image window along different directions using Harris Matrix. If the Harris Matrix has two small eigenvalues, the image segment is a homogeneous surface because there is little change in the image segment along any direction. If the Harris matrix has a large eigenvalue and a small eigenvalue, the edge is detected within this window and the corresponding eigenvector of the small eigenvalue is the direction from the edge because only in this direction there is no change. If the Harris Matrix has two large eigenvalues, this segment contains a corner.
3. When calculating Harris Matrix, a segment is defined and the change of individual pixels is summed up by weight. Usually, we focus more on the central pixel. That is why it weighs more than pixels on the edge.

- Harris-Matrix: $G(x) = \int_W \nabla I(x) \nabla I(x)^\top dx$

$$\nabla I(x) \nabla I(x)^\top = \begin{bmatrix} (\frac{\partial}{\partial x_1} I(x))^2 & \frac{\partial}{\partial x_1} I(x) \frac{\partial}{\partial x_2} I(x) \\ \frac{\partial}{\partial x_2} I(x) \frac{\partial}{\partial x_1} I(x) & (\frac{\partial}{\partial x_2} I(x))^2 \end{bmatrix}$$

5.6.2020

- **Bias-Gain Model Knowledge point**

1. What is the best way to interpret bias and gain visually? Bias brightness, Gain Konstrak.
2. Bias Gain Model: normalized image segment: subtraction of the mean value and subsequent division by the standard deviation.

- **Normierung von Bildsegemt Knowledge point**

1. Standard deviation: N-1 The choice of (n-1) instead of n for the sample is due to the fact that the mean value must have been determined beforehand when calculating the standard deviation sample.
2. Intensity values in a normalized image segment: The standard deviation of the values is 1 and real, normalization does not mean that the values lie in the interval -1 and 1.
3. How is a rotation matrix verified? There are three conditions to meet. The determinant of the matrix is 1, orthogonal columns and norm of the column vector is 1.

- **Euklidische Bewegung tutorial**

Übung2

Euklidische Bewegung

① Rotation eines vektors verändert dessen Länge nicht

$$\|u\|_2^2 \overset{!}{=} \|Ru\|_2^2$$

left multiplication with Rotations matrix

$\Updownarrow$

Coordinate transform-unitary transform

rotations matrix $R^T R = I$

$$\|Ru\|_2^2 = u^T \underbrace{R^T R}_{I} u = u^T u = \|u\|^2$$

② Euklidische Bewegungen mit R and T erhalten den euklidischen Abstand

$$\|a-b\|_2^2 \to \underbrace{\|(Ra+T)-(Rb+T)\|_2^2}$$

$$\downarrow$$

$$\|Ra-Rb\|_2^2$$ rotation erhalten euklidischen Abstand

③ $R = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$ rotations matrix $\begin{cases} R^T R = I \\ \det|R| = 1 \end{cases}$

$$\det(R) = \cos^2\varphi + \sin^2\varphi = 1$$

$$R^T R = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$\to$ R is rotations matrix

5.6.2020-5.14.2020

- **Homework 1: Harris corner und Edgedetector**
- Sober filter convoluted with the image to get the vertical and horizontal gradient. The dimension of the output is the same as the input image.

```
function [Fx, Fy] = sobel_xy(input_image)
    % In this function you have to implement a Sobel filter
    % that calculates the image gradient in x- and y- direction of a grayscale image.
    solberHorizontal = [1 0 -1; 2 0 -2; 1 0 -1];
    solberVertical = [1 2 1; 0 0 0 ; -1 -2 -1];
    Fx = conv2(input_image, solberHorizontal,'same');
    Fy = conv2(input_image, solberVertical,'same');
end
```

- The image gradients are approximated by applying the Solber filter. Harris matrix of image segment is the weighted summation of individual harris matrixes inside the segment. The weighted summation can be applied through the convolution of the gradient matrix with a low-pass filter. Therefore in this part, symmetric weighting vector w is created as part of a separable filter for the segment W to enhance the influence of central pixels. The filter width is adapted to the segment width. The harris matrix is Symmetrix and the three components are separately stored in three matrixes G11, G12, G22, which corresponds to the weighted squared horizontal gradient, the product of horizontal and vertical gradient and squared vertical gradient of image segment centered at the corresponding pixel.

```
% Approximation of the image gradient
input_image = double(input_image);
[Ix, Iy] = sobel_xy(input_image);
G11_unfilter = Ix.*Ix;
G12_unfilter = Ix.*Iy;
G22_unfilter = Iy.*Iy;
% Weighting
w = fspecial('gaussian',[segment_length 1] ,segment_length/4);
% Harris Matrix G
G11 = conv2(G11_unfilter,w*w','same');
G12 = conv2(G12_unfilter,w*w','same');
G22 = conv2(G22_unfilter,w*w','same');
```

- The feature can be extracted by analyzing the eigenvalues of the harris matrix. A simple criterion is calculating the Harris measurement according to . Eliminate all features which are smaller than the threshold value tau and save the resulting feature position in matrix feature, containing the (x-y pixel coordinate) row and column of the detected corners.

```
%% Feature extraction with the Harris measurement
H = G11.*G22 - G12.^2 - k * (G11 +G22).^2;
%H = H.*zeroBorder(H,ceil(segment_length/2));
corners = H .* (H>tau);
[row, column] = find(corners);
features(1,:) = column;
features(2,:) = row;
```

- plotting the sofar achieved features the result is shown in the following image. Problem: too many noisy features are extracted.

- To prevent extracting various features from one real object the features are not allowed to fall below a certain minimum distance to each other. Furthermore, the image is going to get partitioned into smaller tiles, Within a tile, only the N strongest features are saved.
- Sofar the achieved matric corners is of the same size as the grayscale image but its entries correspond to the values of the Harris measurements H. Measured values below the threshold value have already be set to zero. A border of size min_dist containing only zeros around the matrix corners. In the following part the detected feature will be masked, which implies the harris measurements of the neighboring pixels inside min_dist will be set to zero. To apply mask procedure to all the pixels including the pixels near the edges, the image should be expanded around all fours edges by min_distant pixels.

```
%% Feature preparation
% add zero border
corners_with_border = zeros(size(corners,1)+2*min_dist,size(corners,2)+2*min_dist);
corners_with_border((min_dist+1:min_dist+size(corners,1)),min_dist+1:min_dist+size(corners,2)) = corners;
corners =  corners_with_border;

% sort all elemnts in corners
[sort_array, sort_index_all] = sort(corners(:), 'descend');

sorted_index = sort_index_all(sort_array~=0);
```

- An accumulator array acc_array (one entry per tile) that is going to contain the number of features per tile that have already been found. The parameter tile_size determines the size of the tiles and the dimension of the accumulator array. A vector feature is constructed to store the x and y coordinates of the detected features. Both the maximal number of features per tile as well as the number of preprocessed features in sorted_index may influence the size if the array features.

```
%% Accumulator array
row_acc_array = ceil(size(input_image,1)/tile_size(1));
column_acc_array = ceil(size(input_image,2)/tile_size(2));
acc_array = zeros( row_acc_array, column_acc_array )

maximal_features = row_acc_array * column_acc_array * N ;
num_detected_feature = numel(sorted_index);

features = zeros(2,min(maximal_features,num_detected_feature ));
```

- implement the upgraded Harris detector as the following procedure:

1. pick the first remaining features in sorted_index, check if the Harris measurement is still not zero, that means this feature is still active not masked and should be allocated to one tile. If the measurement of the feature is set to zero, then move to the next feature.
2. get the column and row number of feature from the sorted_index (column-major index of the feature)
3. from the column and row index of the feature determine to which tile this feature should be partitioned.
4. mask the neighboring pixels within the mind_dist threshold, which means set the close features to zero
5. add to to the corresponding entry of accumulator_array, which store the number of detected features
6. add the x and y coordinate of the detected feature to array feature
7. check if the corresponding entry of accumulator array reach N (the maximal number of found features). If so, set all the Harris measurements of the tile to zero, which means mask all the features in the full tile.
8. move to the next feature given by sorted_index

```
%% Feature detection with minimal distance and maximal number of features per tile
num_detected_feature = numel(sorted_index);
Cake = cake(min_dist);
feature_counter = 1 ;

for ind = 1:num_detected_feature
    if corners(sorted_index(ind)) == 0
        continue;
    else
        [row, col] = ind2sub(size(corners),sorted_index(ind));

        tile_row = int16(ceil((row-min_dist)/tile_size(1)));
        tile_col = int16(ceil((col-min_dist)/tile_size(2)));

        corners(row-min_dist:row+min_dist, col-min_dist:col+min_dist) = corners(row-min_dist:row+min_dist, col-min_dist:col+min_dist) .* Cake;
        acc_array(tile_row,tile_col) =  acc_array(tile_row,tile_col) +1 ;
        features(1,feature_counter) = col-min_dist;
        features(2,feature_counter) = row-min_dist;
        feature_counter = feature_counter + 1;

        if  acc_array(tile_row,tile_col) == N
            corners((((tile_row-1)*tile_size(1))+1+min_dist):min(size(corners,1),tile_row*tile_size(1)+min_dist),(((tile_col-1)*tile_size(2))+1+min_di
        end

    end
end
```

- result of Harris-Corner detector

Your Solution

In this assignment, I have implemented the Harris-detector algorithm, which is important for the following corresponding points and the 8-points algorithm. During the assignment, the image-filtering idea has been used to calculate the Harris matrix efficiently. However, the last part "upgraded Harris detector" takes me a lot of time. After discussion with classmates, we have solved this problem successfully.

5.13.2020 tutorial

- **lens model, focus plane, pinhole camera model Knowledge point**

1. All rays originating from a point in front of the lens meet behind the lens in the image plane.
2. Rays parallel in front of the lens meet behind the lens in the focal plane.
3. All rays originating from a point in front of the lens and entering the camera can be assumed to be parallel.
4. For the pinhole camera model, the focal plane is the same as the image plane. Due to the small opening of the camera, it can be assumed that the rays that are incident on the camera are approximately parallel. The focus plane thus corresponds to the image plane.
5. The last component for a vector in the image plane in homogeneous coordinates equal to 1 because the Z component is standardized by the focus length.

- **Why do we use homogeneous coordinates in computer vision?**

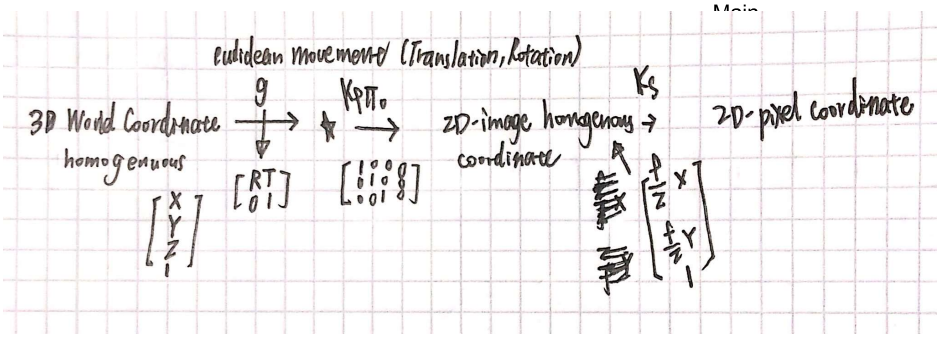
Homogenous coordinates are widely used in projective geometry. They have the advantage that the coordinates of points, including points at infinity, can be represented using finite coordinates. Moreover, Homogenous coordinates allow common vector operations such as translation, rotation, scaling, and perspective projection to be represented as a matrix by which the vector is multiplied, which simplifies the formulation of the euclidean movement.

- **What's the meaning of camera calibration and what does to mean to our course, 3D reconstruction from two different views through the depth information?**

Camera calibration is the process of finding the true parameters of the camera, including focal length, format size, principal point. Given the calibration matrix of the matrix, the pixel coordinate can be back-projected to the calibrated coordinate, which means that the homogenous coordinate is determined by the euclidean matrix (including translation and rotation) and the original coordinate of the 3-D point.

- **perspective projection summary**

The perspective projection takes us from the 3D coordinates to 2D camera image coordinates and pixel coordinates. Inhomogeneous coordinates, we have the transformations:



- **tutorial euclidean movement**

euklidische Bewegung

1) transformation linear $x \to f(x)$ is linear. wenn $f(\alpha x+\beta y) = \alpha f(x) + \beta f(y)$.

$f(z) = Az + zB$

$$f(\alpha x+\beta y) = A(\alpha x+\beta y) + (\alpha x+\beta y)B = \alpha Ax + \beta Ay + \alpha xB + \beta yB$$
$$= \alpha(Ax + xB) + \beta(Ay + yB)$$
$$= \alpha f(x) + \beta f(y) \text{ linear}$$

$g(z) = AzA - B$

$$g(\alpha x+\beta y) = A(\alpha x+\beta y)A - B \neq \alpha(AxA-B) + \beta(AyA-B)$$
nonlinear

2) Dachoperator linear

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \xrightarrow{\hat{x}} \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

$$f(\alpha x+\beta y) = f\left(\begin{bmatrix} \alpha x_1+\beta y_1 \\ \alpha x_2+\beta y_2 \\ \alpha x_3+\beta y_3 \end{bmatrix}\right) = \begin{bmatrix} 0 & -\alpha x_3-\beta y_3 & \alpha x_2+\beta y_2 \\ \alpha x_3+\beta y_3 & 0 & -\alpha x_1-\beta x_1 \\ -\alpha x_2-\beta y_2 & \alpha x_1+\beta x_1 & 0 \end{bmatrix}$$

$\| ? \to$ linear

$$\alpha f(x)+\beta f(y) = \alpha \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} + \beta \begin{bmatrix} 0 & -y_3 & y_2 \\ y_3 & 0 & -y_1 \\ -y_2 & y_1 & 0 \end{bmatrix}$$

5.13.2020 tutorial

- **Image, Coinage, Preimage summary**

1. The dimension of the Coimage of line ist 1.
2. Urbild von Punkt und Cobild von Punkt sind orthogonal.
3. l1 und l2 die Cobilder zweier unterschiedlicher Geraden sind, l1xl2 ist die Schnittgerade der beiden Urbilder. Das Urbild des Schnittpunktes der Bilder beider Geraden.
4. M = XXT convariance matrix besitzt eine orthogonal Eigenwertzerlegung, weil M symmetrisch ist. M =MT
5. A ist eine Matrix, deren Spalten die homegenen Bildkoordinaten einiger Punkte sind. Mann kann durch Rang, Determinante, Singulärwertzerlegung und Eigenwertzerlegung von AAT bestimmen, ob die Punkte auf einer Linie liegen.

image = preimage ∩ image plane , preimage = span(image)

preimage = coimage$^{\perp}$, coimage = preimage$^{\perp}$

⇓

	Image	Preimage	Coimage
Point	span(x) ∩ implane	span(x) ⊂ $\mathbb{R}^3$	span($\hat{x}$) ⊂ $\mathbb{R}^3$
Line	span($\hat{l}$) ∩ im plane	span($\hat{l}$) ⊂ $\mathbb{R}^3$	span(l) ⊂ $\mathbb{R}^3$

plane

- **Essential Matrix knowledge points**

1. property of rotation matrix: square matrix with orthogonal columns normalized to1, determinate of the matrix is 1
2. property of T^: Rank=2, skewsymmetric, and determinant ist 0.
3. Die homogenous coordinate lies in the same equivalence class as the associated space point.
4. The Essential Matrix encodes epipolar geometry. Given a point in one image, multiplying by the essential matrix will tell us the epipolar line in the second view, which can simplify the search of corresponding points from multiple views.

Essential matrix

1) $svd(A) = U\tilde{\Sigma}V^T$. AA^T - A^TA relationship between SVD and EVD

$A = U\tilde{\Sigma}V^T$

$AA^T = U\tilde{\Sigma}V^TV\tilde{\Sigma}U^T = U\tilde{\Sigma}\tilde{\Sigma}U^T = U\left[\begin{array}{c|c}\Sigma^2 & 0\\ 0 & 0\end{array}\right]U^T$ ⇒ U eigenvector of AA^T

$A^TA = V\tilde{\Sigma}^TU^TU\tilde{\Sigma}V^T = V\tilde{\Sigma}\tilde{\Sigma}V^T = V\left[\begin{array}{c|c}\Sigma^2 & 0\\ 0 & 0\end{array}\right]V^T$ V eigenvector of A^TA

singularvalue^2 = eigenvalue of A^TA, AA^T

2) prove $E = \begin{bmatrix}1&0&0\\0&2&0\\0&0&0\end{bmatrix}$ is essential matrix SVD property of essential matrix

2 same singular value + 1 singular value = 0.

singular decomposition of E

$E = \begin{bmatrix}6&&\\&6&\\&&0\end{bmatrix}$ ✓

$E = \begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}\begin{bmatrix}2&0&0\\0&1&0\\0&0&0\end{bmatrix}\begin{bmatrix}0&1&0\\1&0&0\\0&0&1\end{bmatrix}$ ⇒ not essential matrix

- **Epilolarequation, Essentialmatrix, 8-points Algorithm relationship**

Eipolar Gleichung Herleitung → E Definition

depth2 depth1

wichtig für 8-Punkte algorithmus

$\lambda_2 x_2 = R\lambda_1 x_1 + T$

$\lambda_2 \hat{T} x_2 = \lambda_1 \hat{T} R x_1 + \underbrace{\hat{T}T}_{0}$

$\lambda = \underbrace{x_2^T \hat{T}^T \hat{T} x_2}_{0} = \lambda_1 x_2^T \hat{T} R x_1$

$x_2^T \hat{T} R x_1 = 0$ ☆ x_1, x_2 correspondy points in calibrated coordinate

fulfill this Epipolar Gleichung

5.20.2020 - 5.27.2020 Homeworks: corresponding points

- **In the previous assignment, we have extracted feature points from two images. In this assignment, we will find the corresponding points by using the correlation matrix as a metric. These corresponding points will be used for the further depth-based algorithm, extract the essential matrix, translation vector, and rotation matrix R.**
- **feature preparation:** The variables Ftp1 and Ftp2 contain the x/y-coordinates of the feature points from the previous assignment. In order to test them with regards to correlation, a window has to defined around the features. If the features lie too close to the image border pixels within the window might be missing. Therefore all features that are located too close to the image border are removed in this part.

```
for i = 1:size(Ftp1,2)
    if Ftp1(1,i)<((1+window_length)/2) |  Ftp1(1,i)> (size(I1,2)-(window_length-1)/2) | Ftp1(2,i)<((1+window_length)/2) | Ftp1(2,i)> (size(I1,1)-(window_
        Ftp1(1,i) = 0;
        Ftp1(2,i) = 0;
    end


end

colsWithZeros = any(Ftp1==0)
Ftp1 = Ftp1(:, ~colsWithZeros)
```

- **build a correlation matrix between two windows centered at filtered feature points:** Correlation between window is more robust compared to signal pixel, feature point. The windows are arranged in vector form as the following picture shows, which will be used to calculate to correlation metric between features points in two images. Each feature window is normalized by subtracting the mean and divide by standard deviation.

Assume there are two features in `Ftp1` resulting in two normalized windows A_n and B_n. The side length `window_length` is assumed to be 3

$$A_n = \begin{bmatrix} a_1 & a_4 & a_7 \\ a_2 & a_5 & a_8 \\ a_3 & a_6 & a_9 \end{bmatrix} \quad B_n = \begin{bmatrix} b_1 & b_4 & b_7 \\ b_2 & b_5 & b_8 \\ b_3 & b_6 & b_9 \end{bmatrix}$$

The resulting matrix `Mat_feat_1` would look as follows:

$$\text{M_feat_1} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \\ \vdots & \vdots \\ a_9 & b_9 \end{bmatrix}$$

```
%% Normalization
Mat_feat_1 = zeros(window_length^2,size(Ftp1,2));
block = zeros(window_length, window_length);
a = (window_length-1)/2;
for i = 1: size(Ftp1,2)
    block = Im1((Ftp1(2,i)-a):(Ftp1(2,i)+a),(Ftp1(1,i)-a):(Ftp1(1,i)+a));
    block = block - mean(block(:));
    block = block/std(double(block),0,'all');
    Mat_feat_1(:,i) = block(:);
end
```

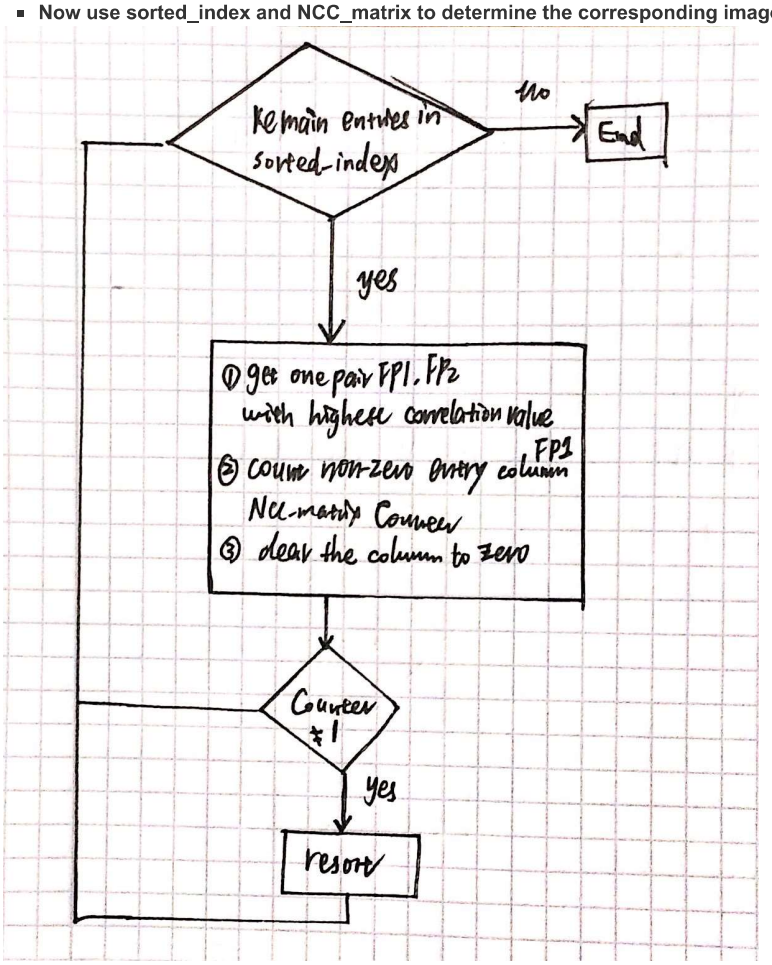
- **Calculation of NCC_matrix:** Now the normalized windows in Mat_feat_1 and Mat_feat_2 are used to calculate the normalized cross-correlation matrix NCC_matrix, dot-product of window vector. All entries in NCC_matrix that are below the threshold value min_corr are set to zero, which means these two feature points are not correlated. Indices of all non-zero entries in NCC_matrix are sorted in decreasing order of their correspondence intensity.

```
%% NCC calculations
NCC_matrix = zeros(no_pts2, no_pts1);
for i = 1: no_pts2
    for j = 1: no_pts1
        NCC_matrix(i,j) = dot(Mat_feat_2(:,i),Mat_feat_1(:,j))/(window_length^2-1);
    end
end

NCC_matrix(NCC_matrix < min_corr) = 0;


[sort_array, sort_index_all] = sort(NCC_matrix(:), 'descend');
sorted_index = sort_index_all(sort_array~=0);
```

- **Now use sorted_index and NCC_matrix to determine the corresponding image points.**



Result: corresponding points in two images

In this assignment, we will find the corresponding points by using the correlation matrix as a metric. The result is quite good and these points can be used for further 3D reconstruction algorithms.

5.27.2020

- **overview of the reconstruction algorithm**
  The 3-D reconstruction algorithm proceeds as follows:

1. recover the essential matrix E from the epipolar constraints associated with a set of corresponding points.
2. Extract the relative translation and rotation from the essential matrix E.

- **Other ways to estimate the essential matrix?**

1. Recover some matrix E from the epipolar constraints and then project it onto the essential space
2. **Optimize the epipolar constraints in the essential space.** The second approach is more accurate because it involves **nonlinear constrained optimization.**

Epipolar Geometry

① $\boxed{l_2 \sim E x_1}$

Epipolarlinie wird über codirtl. identifiziert

$l_i \sim e_i \times x_i$

method 1

$l_2 \sim e_2 \times x_2$

$\lambda_2 x_2 = \lambda_1 R x_1 + T$

$x_2 \sim R x_1 + \frac{1}{\lambda_1} T \Rightarrow e_2 \sim T$

method 2

$P_2 = R P_1 + T$

$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

$P_2 = T$

$\lambda_1 x_2 = T$

$\boxed{e_2 \sim T}$

$l_2 \sim e_2 \times x_2$

$l_2 \sim T \times x_2 = \hat{T} x_2$

$l_2 \sim \hat{T}(R x_1 + \frac{T}{\lambda_1}) = \hat{T} R x_1 + \underset{0}{\underset{\parallel}{\hat{T} T}} \frac{1}{\lambda_1}$

$l_2 \sim \underbrace{\hat{T} R}_{E} x_1$

essential matrix help to find corresponding feature points

② $E = \hat{T} R$

Kameratbewegung in der X,Y- Ebene

$T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \Rightarrow \hat{T} = \begin{bmatrix} 0 & t_3 & t_2 \\ -t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \xrightarrow{t_3 = 0} \begin{bmatrix} 0 & 0 & t_2 \\ 0 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$

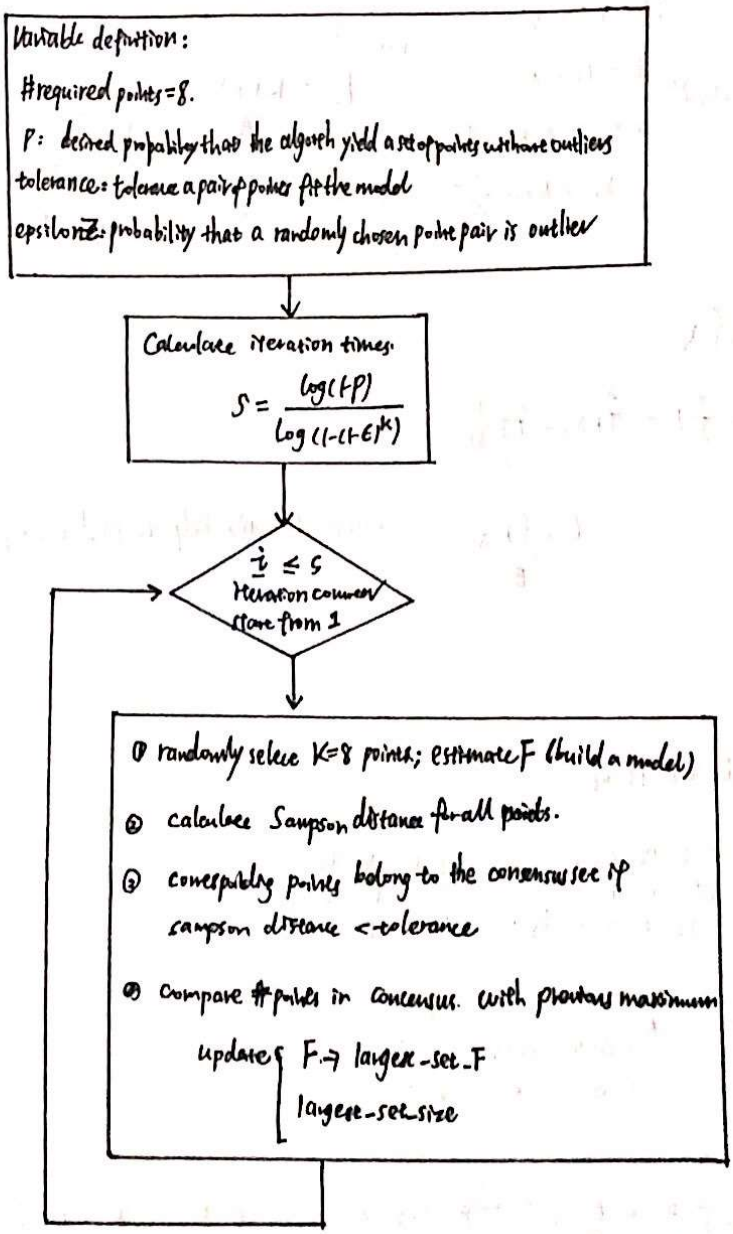
rotation Z achse $R = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$E = \hat{T} R = \begin{bmatrix} 0 & 0 & t_2 \\ 0 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & t_2 \\ 0 & 0 & -t_1 \\ -t_2\cos\varphi + t_1\sin\varphi & t_2\sin\varphi + t_1\cos\varphi & 0 \end{bmatrix} \leftarrow \begin{bmatrix} 0 & a & a \\ 0 & 0 & b \\ c & d & 0 \end{bmatrix}$$

6.3.2020

Ransac Algorithm applied in 8 points Algorithm (Homework 3)

In previous assignments, we have figured out the corresponding points. According to 8 points algorithm, only 8 points are needed to estimate the essential matrix, translation vector, and rotation matrix. There are many combinations of 8 points we can extract from previously obtained corresponding points. Different combinations will give us different models, essential matrix. In this sense, how to select these 8 points plays an important role. In this assignment, the Ransac algorithm is applied to select a set of robust corresponding points. The following chart shows the procedure of the RANSAC algorithm.



How to define the number of iterations is showed below from the statistical perspective.

P: desired probability that the algorithm yield a set of corresponding points where no outliers are included

ε: probability that a randomly chosen pair of corresponding points is an outlier

k: number of points to extract

probability that k points no outlier : $(1-\varepsilon)^k$

probability a set of k points at least one outlier : $1-(1-\varepsilon)^k$

select S sets of k points. each set contain one outlier: $(1-(1-\varepsilon)^k)^S$

select S sets of k points. at least one set no outlier: $1-(1-(1-\varepsilon)^k)^S$

$$P = 1-(1-(1-\varepsilon)^k)^S$$

$$S = \frac{\ln(1-P)}{\ln(1-(1-\varepsilon)^k)}$$

6.10.2020

- **extension of the eight-points algorithm?**

In the case of infinitesimal viewpoint change, one can adapt the eight-point algorithm to the continuous motion case, where the epipolar constraint is replaced by the continuous epipolar constraint, which is dependent on time. Rather than recovering (R; T) we can recover the linear and angular velocity of the camera.

- **Why do we need the four-points algorithm?**

The eight-point algorithm only provides unique solutions (up to a scalar factor) if all 3D points are in a "general position". However under certain degenerate configurations, for example, all points lie on a plane or spherical surface, these points fulfill certain constraints. We can exploit these additional constraints and integrate them into our 3D_reconstruction algorithm, to reduce the number of required corresponding points.

The derivation of the four-point algorithm is similar to that of the eight-points algorithm. The main idea is constructing a linear equation system, which the corresponding points in multiple views have to fulfill. The essential matrix or tomography matrix is related to the translation vector T and rotation matrix and they are the solution of the linear equation system. Considering that the actual extracted corresponding points are corrupted by noise, the linear equation system may only have an exact solution. Under this condition, we use the least_square solution through singular value decomposition to approximate the solution, the desired essential matrix, or the homography matrix. In the four-point algorithm, the homography matrix is decomposed into R, N, and T=d.

6.17.2020 Computer vision challenge introduction

The key point of this project is foreground and background separation. Following ideas have been analyzed:

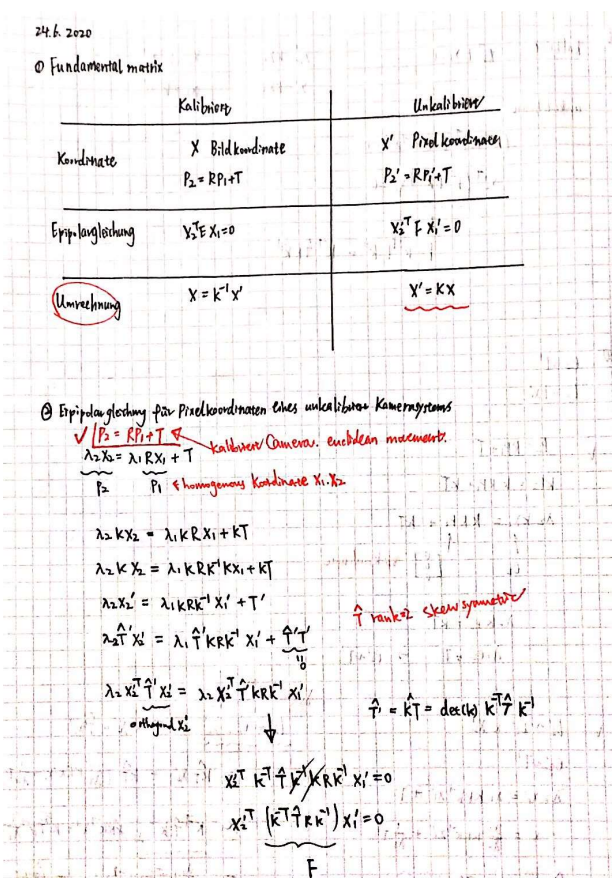
(1) use Depth information: Firstly calculate the depth information of all pixels from the current frame and previous frame. Then compare the depth of all pixels in these two consecutive frames and determine the depth-change of each pixel. The pixels whose depth change exceeds a certain threshold are assigned to 1 as the dynamic foreground. This method needs two cameras from different views to recover the depth of certain pixels. However, as far as what we have learned this semester we can only calculate the depth of extracted feature points. In order to get the depth of all pixels, the K matrix is needed. With K we can project the image from pixel coordinate to calibrated camera coordinate and get the depth of all pixels. How ever, the intrinsic parameters of the camera are not provided and it is hard to estimate the K matrix without further information.

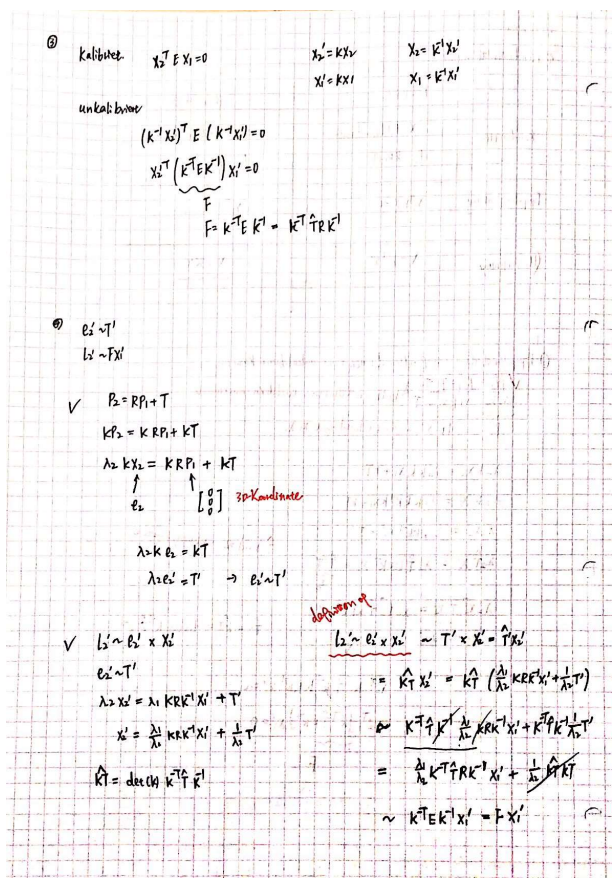
(2) block process, motion estimation detection between two frames: This method is similar to the depth comparison from two consecutive frames. However, this frame difference method requires only one camera frame. Considering the camera is static, this method may have good results. There are several ways to detect the motion from two frames. The image is split into 8x8 blocks, each block is compared with blocks of the same location in the previous block. The comparison criterion can be SSD(sum of squared error), SAD（sum of absolute error），SVD（sigular value decomposition, and compare the change of singular values). We have tested the SVD and get certain results, however, the computation complexity is quite high.

(3) block process, singular value decomposition： split the frames into 8*8 blocks and compare the change of singular values of two corresponding block from two consecutive frames, high computational complexity

(4) deep learning, CNN: quite a powerful method, only require single image input and returns the outline of people. However, this method requires large effort for labeling the training data. Using a pretrained network as a black block is not in line with the original intention of this project.

6.24.2020





Progress of CV Challenge G18:

In this week we have further discussed the ideas how to do the foreground and background separation and finally, we have decided to apply adaptive background subtraction, which is a quite efficient method with less computation complexity compared with Blocking matching, especially exhaustive searching, Singular value decomposition. Furthermore, it requires less training samples compared with the convolutional neural network.

The main idea of our adaptive background subtraction is as follows, Background subtraction method makes different operations between the current image frame and the background image to get the gray image of the moving area of the target. The background can be either provided or learned from certain frames. In this CV challenge, we have used the mod operation the calculate the background using 100 frames. The idea behind this is that the background is static and has almost constant grayscale value in the video sequence. However, the dynamic foreground doesn't have this property. The background is updated every 100 frames. We have tried several combinations of the number of frames needed to calculate the background and after how many frames the background needs to be updated. The combination we will apply in other final project is 100 and 100, which brings sufficient performance. The gray image is thresholded to extract the moving area. In order to avoid the influence of environmental illumination, the background image. The following flowchart shows the structure of our algorithm.

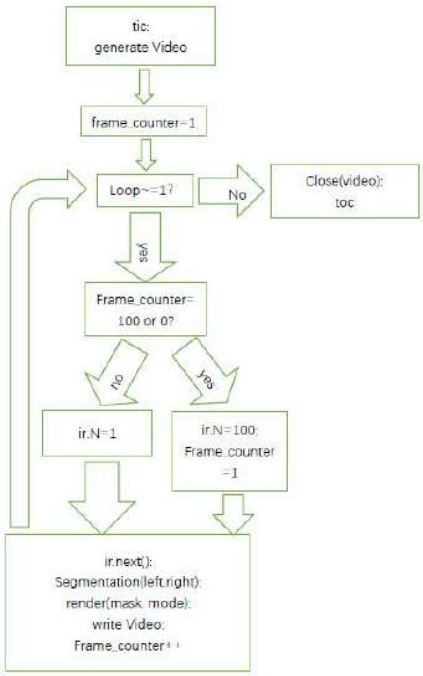


Figure 3: Challenge

7.1.2020

planare Epipolargleichung

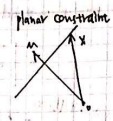
① $P_2 = RP_1 + T$ euclidische Bewegung

$P_2 = RP_1 + T\frac{1}{d}n^T P_1$

$= (R + T\frac{1}{d}n^T)P_1$

$\lambda_2 x_2 = (R + T\frac{1}{d}n^T)\lambda_1 x_1$

$x_2 \sim \underbrace{(R + T\frac{1}{d}n^T)}_{H} x_1$ Homographie matrix



$n^T x = d$

$\frac{1}{d} n^T x = 1$

② $H = R + T\frac{1}{d}n^T$ SVD

$H = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$

③ $\hat{x}_2 H x_1 = 0$

$\hat{x}_2 [h_1\ h_2\ h_3] \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = 0$

$[\hat{x}_2 h_1\ \hat{x}_2 h_2\ \hat{x}_2 h_3] \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = 0$

$x_1 \hat{x}_2 h_1 + y_1 \hat{x}_2 h_2 + z_1 \hat{x}_2 h_3 = 0$

$B = x_1 \otimes \hat{x}_2 = \begin{bmatrix} x_1 \hat{x}_2 \\ y_1 \hat{x}_2 \\ z_1 \hat{x}_2 \end{bmatrix} \in \mathbb{R}^{9\times 3}$

$\text{rang}(\hat{x}_2) = 2 \rightarrow \text{rang}(B) = 2$

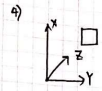
$\hat{x}_2 H x_1 = 0.$

$B^T H^S = 0 \quad [x_1\hat{x}_2\ y_1\hat{x}_2\ z_1\hat{x}_2] \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = 0$

$\rightarrow$ 2 unabhängige Gleichungen

H^S bis auf Skalierung schätzen $\rightarrow$ 8 Gleichung $\rightarrow$ 9 corresponding pointpairs.

④



rotation um z achse

$R = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Bewegung in X-Y-Ebene $T = \begin{bmatrix} t_1 \\ t_2 \\ 0 \end{bmatrix}$

normal vector/-zu Ebene $n = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}$

$x_2 = Hx_1$

$\underbrace{\lambda_2 x_2}_{P_2} = H\underbrace{(\lambda_1 x_1)}_{P_1}$ same Tiefe.

✓ wenn $\lambda_1 = \lambda_2 = \lambda \rightarrow x_2 = Hx_1$

$\Rightarrow H = R + \frac{1}{d} T n^T$

$= [R] + \frac{1}{d}\begin{bmatrix} t_1 \\ t_2 \\ 0 \end{bmatrix} [n_1\ n_2\ n_3]$

$= R + \frac{1}{d}\begin{bmatrix} n_1 t_1 & n_2 t_1 & n_3 t_1 \\ n_1 t_2 & n_2 t_2 & n_3 t_2 \\ 0 & 0 & 0 \end{bmatrix}$

$= \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} + \frac{1}{d}\begin{bmatrix} n_1 t_1 & n_2 t_1 & n_3 t_1 \\ n_1 t_2 & n_2 t_2 & n_3 t_2 \\ 0 & 0 & 0 \end{bmatrix}$

$\Rightarrow P_2 = HP_1$ gleiche Tiefe.

$z_1 = z_2 = z.$

$\frac{1}{z} P_2 = H \frac{1}{z} P_1$

$x_2 = \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = Hx_1$

Problem with two-view reconstruction based on the fundamental matrix:

The before discussed 8 points algorithm is based on the calibrated camera, which means we analyze the whole problem in the calibrated camera coordinate. If the camera parameters K cannot be estimated in a calibration procedure beforehand, we have to deal with reconstruction from uncalibrated views.

While it is straightforward to extend the eight-point algorithm, such that we can attract a fundamental matrix from a set of corresponding points, there may remain several problems.

Firstly, compared with the fundamental matrix, the essential matrix has a relatively more special structure, rank 2, and 2 identical singular values. This structure constant can be used for constraint in our 8 point algorithm. However, we cannot impose a strong constraint on the specific structure of the fundamental matrix (apart from the fact that the last singular value is zero). In fact, any matrix of rank 2 can be a fundamental matrix. This kind of degree of freedom may cause problems during the procedure.

Secondly, for the essential matrix, there are four possible solutions, combinations of the translation vector and rotation matrix. We can further decide which one is the right answer by checking the depth information. However, for a given fundamental matrix F, there does not exist a finite number of decompositions into extrinsic parameters R; T and intrinsic parameters K, which means that we cannot extract the right E and R form a given essential matrix.

7.8.2020

Kalibrierung

Schachbrett im Ursprung liegt und die Z-Achse senkrecht zum Schachbrett ist

↓

Z komponente aller Punkte gleich 0.

$$X' \sim K\Pi_0 \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \overbrace{K[r_1\ r_2\ T]}^{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad 3\times 4$$

homogenen Pixelkoordinate

homogene Schachbrett Koordinate

$$K\Pi_0 \left[R \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix} + T \right]$$

$$[r_1\ r_2\ r_3] \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix} + T$$

scalar

$\hat{X}' H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = 0$ → schätzen H bis auf Skalierung mittels 4-Punkte-Algorithmus

Ansichten des Schachbretts → (2n×6)-Gleichungssystem $Vb = 0$ zu lösen?

$$B = K^{-T}K^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}$$

6 unbekannte Parameter

$$h_1^T B h_2 = 0$$

jeder Anblick eine Homographie, 2 Gleichungen

↑ 3 Ansichten

During the online session, Professor Diepold had said that there is a similar lecture held by professor Cremers in Informatik TUM, where the reconstruction problem is solved by a convex optimization way. In last semester I have learned some optimization methods, such as Gradient descent, Gaussian-Newton method, and so on. And I am interested in how these methods are applied in the reconstruction problem from multiple views.

The bundle adjustment method is a nonlinear method for reconstruction problem. It solves the following problem. X1 and X2 are the extracted corresponding points and the 3D coordinate of the point is Xj. In this formula, we sum up the projection error of the projection points in two views(image1 and image2) of N corresponding point pairs. The variable, the optimized variable is Translation vector T, Rotation matrix R, and the 3D coordinate of the projected point, which means we can achieve T, R, and 3D coordinate all together, while in previous 8 points or 4 points algorithms, we first achieve R and T, them we calculate the depth information and get the corresponding 3D coordinate. The Bundle adjustment has converted the reconstruction problem into an optimization problem, which can be solved using Gradient descent, Gaussian-Newton, LM method, and other powerful optimization problem.

$$E(R, T, \boldsymbol{X}_1, \ldots, \boldsymbol{X}_N) = \sum_{j=1}^{N} \left|\tilde{\boldsymbol{x}}_1^j - \pi(\boldsymbol{X}_j)\right|^2 + \left|\tilde{\boldsymbol{x}}_2^j - \pi(R, T, \boldsymbol{X}_j)\right|^2$$

In addition to the retrieval sequence of T, R, and 3D coordinate, there is a second difference I have spotted, which may not be correct. During the Bundle adjustment, the formulation of the convex optimization problem, the certain Gaussian measurement noise has been included. The idea comes to form another lecture I have taken this semester, Statistical Signal Processing from professor Utschick. It can be shown that the optimization problem is derived from the maximum likelihood estimation based on a linear model with Gaussian noise. However, in 8 point algorithm, we have not introduced the measurement noise in our algorithm, expect when extracting the essential matrix using its special structure. The introduction of noise during the algorithm may bring us some kind of robustness.

How to combine 8 point algorithm and Bindle adjustment together?

After comparing the two algorithms, I come up with the idea to combine them together. It is known that the initialization of the optimization variable plays an important role in convex optimization. If the variables are initialized properly, it will no take a long time to converge the optimal solution. Therefore, I think that we can use the result of the eight-points algorithm as initialization, cause we can assume that this solution not quite far away from the optimal solution. Then apply bundle adjustment, we can get more accurate results.

7.15.2020

This week we have started to review the CV Challenge Project from other groups. We have briefly discussed the main ideas other teams have implemented. In total, we have viewed 5 groups. What surprised me is, all of them use kinds of image or video processing methods and none of them applied the depth information. The reason behind this is that, without the calibration matrix of the camera, we cannot integrate the depth information into the foreground and background segmentation. This makes me a little bit frustrated because what we have learned this semester, 3D reconstruction from two views cannot be implemented in the final project. From my point of view, it would

be better if we implement what we actually have learned this semester not only in MATLAB homework but also into CV Challange.

For the final project, we have used sequences of images to calculate a background and implemented background subtraction between the current image and the available calculated background. We construct the background from a statistical perspective. We regard each pixel as a random temporal process and the pixel value of each frame is the realization of this random process. For background pixels, the variance of the process is small, which the variances of the foreground are bigger. Therefore we have set the mode of each pixel of a sequence of frames as the background.

After viewing the project of all 5 groups, the algorithm that Group 28 surprised me. In their final project Gaussian mixture model has been applied to model the background. They have used MATLAB Computer Vision Toolbox ForegroundDetector to computer a mask of the foreground of the image. This algorithm requires several training sets to learn a model. In this technique, it is assumed that every pixel's intensity values in the video can be modeled using a Gaussian Mixture Model, the mixture of Gaussian models with different mean, variance, and weights. For separation of foreground and background, this method can output probability for each pixel, whether it is foreground or background in the current frame. Although this method is embedded into MATLAB CV Toolbox and the implementation is not quite complicated, I still learn a lot from this method, Background modeling through GMM.

7.22.2020

Overview of Computer Vision 2020:

In this course, I have learned how to achieve the 3D reconstruction from two static views, the object is taken pictures from two different views, which is related through the translation vector and rotation matrix. Firstly we can extract features from two different images separately. We have computed the gradient and applied the Harris detector to extract the corner points. Then the selected feature points can be matched to possible corresponding feature points through correlation computation. These corresponding feature pairs played an important row in the reconstruction algorithm: 8 point algorithm. The 8 point algorithm builds a linear equation system by using the euclidean motion and the corresponding feature points. The solution of the constructed linear equation system is known as the essential matrix, which has a strong structure constrain on singular values and is related to the translation vector and rotation matrix. From the essential matrix, we can get 4 possible solution combinations of T and R. Then we can find the right combination by viewing the depth information of each feature point. After successfully extracting the translation vector and rotation matrix, which means the motion of the camera is known, we can reconstruct the 3D object.

My favorite part of this lecture is practical MATLAB homework, which serves as a good supplementary of the lecture. Not only the topics in lecture video have been discussed in MATLAB homework, but also some extended topics, including the correlation comparison by finding the corresponding feature pair, RANSAC algorithm by applying the 8 point algorithm in order the enhance the robustness of the algorithm. However, the degradation of the 8 point algorithm planar 4 point algorithm is not included in MATLAB homework, which is also an interesting topic. I think it would be better to extend the MATLAB homework about this topic.

The CV challenge is also an interesting project related to video processing. Foreground and background separation can be solved many methods, including background subtraction, frame difference, motion estimation, convex optimization, convolutional neural network, and so on. After discussion, our team has decided to apply adaptive background subtraction and have achieved certain performance.

This is my first time to write Laborbuch. I think this is a good method to track the learning procedure. The summary of lecture content, exercise, practical MATLAB homework, individual question, and comment about the algorithms are included in this Laborbuch.

7.29.2020

I am quite happy to see the feedback on our CV challenge project from other teams. Although the criteria of each team may differ a little bit, from most of them we have received positive feedback. However, there are several problems spotted by other teams, which I think is quite reasonable and some suggestions may be quite useful for future projects.

1. One team has made several suggestions on our documentation. Firstly, the functions mostly flow charts were used but the charts weren't explained in the text. I totally agree that it would be better to have a short-written explanation of the flow charts to help the reviewer understand them better. Secondly, the result images are only shown without a short description or illustration. I think this is reasonable and I think we should have added the running time for each selected folder, which can provide the review with an intuition about the speed of our algorithm. Thirdly, in our final report, there is a short part illustrating the small problems of our algorithm. It is suggested to include some screenshots of these problems and describe them in the result section. What I have learned from these comments is that for a project document it is important to combine text descriptions and screenshots.

2. Some problems with the GUI have been spotted. Firstly, the GUI should be named as start_gui.m. However, we have saved it as GUI.m. This problem should have been avoided. Secondly, it has been spotted that the stereo input and stereo image. Thirdly, it has been suggested that the selected working folder should be shown, which I think is quite reasonable and can show the user which folder the GUI is working on. Moreover, during the viewing the other teams' projects, I have seen that some of them have additionally included the number of proceeded images and the fraction of the total number of images in the selected folder. Some of them have also included the processing time. I think these points can increase the user experience and I would like to include them when writing GUI in the future.