Lab 4 –
Image mosaics from feature matching

16.02.2017

Dagens

- Eksperimentere med features
- Estimere homografier
- Lage mosaikk

Features i OpenCV

- Mange metoder
  - Noen detektorer
  - Noen deskriptorer
  - Noen detektorer *og* deskriptorer

- Modulen *features2d*

- Modulen *xfeatures2d*

Features i OpenCV

- Konstruksjon

```
cv::Ptr<cv::Feature2D> detector = cv::xfeatures2d::SURF::create();
cv::Ptr<cv::Feature2D> desc_extractor = cv::xfeatures2d::LATCH::create();
cv::BFMatcher matcher{desc_extractor->defaultNorm()};
```

- Bruk

```
// Detect keypoints
std::vector<cv::KeyPoint> frame_keypoints;
detector->detect(gray_frame, frame_keypoints);
cv::KeyPointsFilter::retainBest(frame_keypoints, 500);

// Extract descriptors.
cv::Mat frame_descriptors;
desc_extractor->compute(gray_frame, frame_keypoints, frame_descriptors);

// Match descriptors.
std::vector<std::vector<cv::DMatch>> matches;
matcher.knnMatch(frame_descriptors, base_descriptors, matches, 2);
std::vector<cv::DMatch> good_matches = extract_good_ratio_matches(matches, 0.8);
```

Steg 1: Eksperimenter med features

- Test forskjellige detektorer
- Still på parametere
- Hva inneholder `cv::KeyPoint`?

Steg 2: Feature matching

- Implementer

```
std::vector<cv::DMatch> extract_good_ratio_matches(
  const std::vector<std::vector<cv::DMatch>>& matches, double max_ratio)
```

- Eksperimenter med matching
  - Forskjellige detektorer
  - Forskjellige deskriptorer
  - Ulike parametere

- Beveg kameraet på ulike måter, se på ulike scener

- Hvordan kan du forbedre matchingen?

Steg 3: Beregn reprojeksjonsfeilen

```
inline float HomographyEstimator::reprojectionError(const cv::Point2f& pt1, const cv::Point2f& pt2,
                                                    const Eigen::Matrix3f& H, const Eigen::Matrix3f& H_inv)
{
  // Step 3: Compute reprojection error.
  Eigen::Vector2f vec_1(pt1.x, pt1.y);
  Eigen::Vector2f vec_1_in_2 = vec_1; //?

  Eigen::Vector2f vec_2(pt2.x, pt2.y);
  Eigen::Vector2f vec_2_in_1 = vec_2; //?

  return 0.f;
}
```

$$\varepsilon_i = d(H\boldsymbol{u}_i, \boldsymbol{u'}_i) + d(\boldsymbol{u}_i, H^{-1}\boldsymbol{u'}_i) \quad \textbf{(Reprojection error)}$$

Notation	
Euclidean distance	$d(\cdot,\cdot)$
Inhomogenous $H\tilde{\boldsymbol{u}}_i$	$H\boldsymbol{u}_i$
Inhomogeneous $H^{-1}\tilde{\boldsymbol{u}}'_i$	$H^{-1}\boldsymbol{u}'_i$

Steg 4: Mosaikking

- Hva gjør S?

```
cv::Matx33d S{
0.5, 0.0, 0.25 * frame_cols,
0.0, 0.5, 0.25 * frame_rows,
0.0, 0.0, 1.0};
```

- Flytt og skaler begge bildene med similaritetstransformen *S*, og samregistrer *frame* med *base_img* ved å anvende homografien *H*

```
cv::Mat mosaic;
cv::warpPerspective(base_img, mosaic, S, frame.size());
cv::warpPerspective(frame, frame_warp, /* ? */, frame.size());
```

- Sett bildene sammen til et mosaikk

```
cv::Mat mask = frame_warp > 0;
frame_warp.copyTo(mosaic, mask);
```

- Hvordan unngå effekter på kanten? (cv::erode()?)

Steg 4: Mosaikking

Transformation of $\mathbb{P}^2$	Matrix	#DoF	Preserves	Visualization
Translation	$\begin{bmatrix} I & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$	2	Orientation + all below	
Euclidean	$\begin{bmatrix} R & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$	3	Lengths + all below	
Similarity	$\begin{bmatrix} sR & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$	4	Angles + all below	
Affine	$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$	6	Parallelism, line at infinity + all below	
Homography /projective	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$	8	Straight lines	

Videre

- Kombiner flere detektorer
- Les gjennom homografiestimatoren, og prøv å forstå stegene.
- Prøv med cv::findHomography(…) i stedet for vår metode
- Lag mosaikk med blanding i stedet for ren overskriving
  - Alfablending
  - Laplaceblending
- Utvid programmet til å sette sammen mosaikker av flere bilder