# Module 2: Visual Features - Detection, Description and Matching

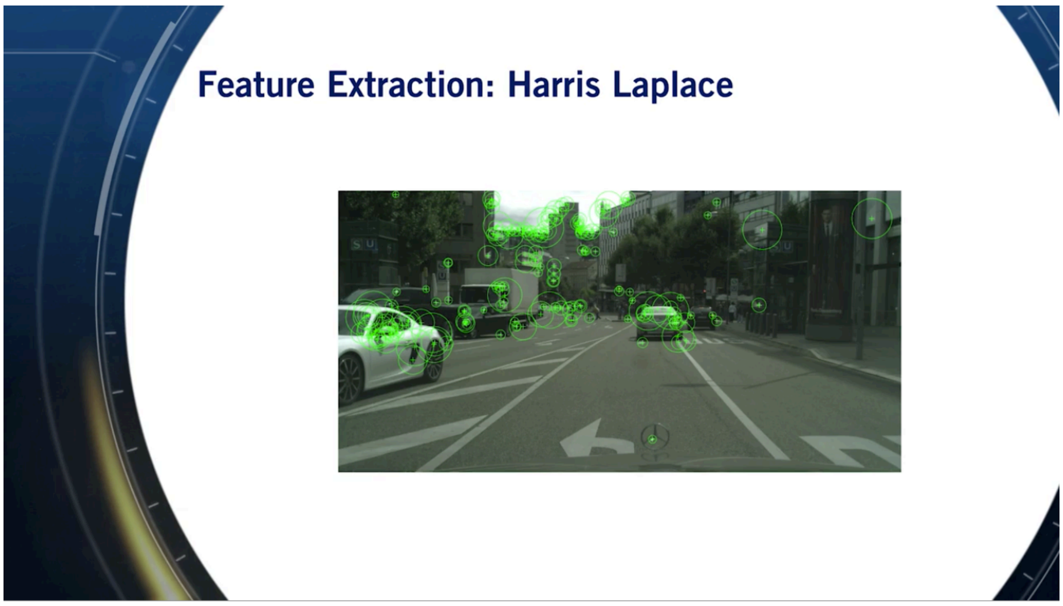
## Lesson 1: Introduction to Image Features and Feature Detectors

単語
- Image stitching: the process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image.

内容
- Feature extraction, the first step of using image features for applications.
- What characteristics make good image features.
- Different algorithms used to extract features in images.

Image Features: A General Process
- Image stitchingの例
  - Second, associate a descriptor for each feature from its neighborhood.
  - Finally, use these descriptors to match features across two or more images.

Feature Detection
- Features are points of interest in an image.
- Points of interest should have the following characteristics:
  - Saliency: distinctive, identifiable, and different from its immediate neighborhood.
  - Repeatability: can be found in multiple images using same operations.
  - Locality: occupies a relatively small subset of image space.
    - The features should not change if an image region far away from the immediate neighborhood changes.
  - Quantity: enough points represented in the image.
    - This is because many applications such as calibration and localization require a minimum number of distinctive points to perform effectively.


Feature Extraction: Harris Laplace

- Efficiency: reasonable computation time.

Feature Extraction（大事）
- Repetitive, texture less patches are challenging to detect consistently.
- Patches with large contrast changes (gradients) are easier to detect (edges).
- Gradients in at least two (significantly) different orientations are the easiest to detect (corners).
  - A corner occurs when the gradients in at least two significantly different directions are large.
- The most famous corner detector is the Harris Corner Detector, which uses image gradient information to identify pixels that have a strong change in intensity in both x and y directions.

Feature Detection: Algorithms
- Harris {corners}: Easy to compute, but not scale invariant.
  - Meaning that the corners can look different depending on the distance the camera is away from the object generating the corner.
- Harris-Laplace {corners}: Same procedure as Harris detector, addition of scale selection based on Laplacian. Scale invariance.
- Features from accelerated segment test (FAST) {corners}: Machine Learning approach for fast corner detection.
  - very high computational efficiency and solid detection performance.
- Laplacian of Gaussian (LOG) detector {blobs}: Uses the concept of scale space in a large neighborhood (blob). Somewhat scale invariant.
- Difference of Gaussian (DOG) detector {blobs}: Approximates LOG but is faster to compute.

Feature Extraction: Harris Laplace
- By using the Laplacian to determine scale, we can detect scale invariant corners that can be more easily matched as a vehicle moves relative to the scene.
- Scale is represented here by the size of the circle around each feature.
- The larger the circle, the larger the principal scale of that feature.

• Good image features need to be salient, repeatable, local, efficient, and numerous.
• Empirical validation is required to choose the best extractor based on application.

# Lesson 2: Feature Descriptors
We need to describe features in a way that it allows for feature comparison to determine the best match between frames.

内容
- What characteristics make a good feature descriptor.
- Different algorithms used to extract feature descriptors from images.

Feature Descriptors
- Feature: point of interest in an image defined by its image pixel coordinates $[u, v]$.
- Descriptor: An N-dimensional vector that provides a summary of the image information around the detected feature.
  - Position: $[u, v]$.
  - Descriptor: $\{f_1 \dots\dots\dots f_N\}$.
- Feature descriptors should have the following characteristics:
  - Repeatability: manifested as robustness and invariance to transform, rotation, scale, and illumination changes.
  - Distinctiveness: should allow us to distinguish between two close by features, very important for matching later on.
  - Compactness & Efficiency: reasonable computation time.

Designing Invariant Descriptors: SIFT（大事）
- Scale Invariant Feature Transform (SIFT) descriptors:
  1. $16 \times 16$ window around detected feature.
  2. Separate into 4 cells, each comprised of $4 \times 4$ patch of pixels.
  3. Compute edge orientation of each pixel in the cell.

4. Suppress weak edges using a predefined threshold.
5. Construct 32 dimensional histogram of orientations for each cell, then concatenate to get 128 dimensional descriptor.
   - つまり1つのpixelのorientationは2dimensional?
   - この32 dimensional histogram of orientationsはどう計算してるの?

Scale Invariant Feature Transform
- SIFT is an example of a very well human engineered feature descriptor, and is used in many state-of-the-art systems. つまりdeep learningの方法はまだ可能性がある。
- The above process is usually compute on rotated and scaled version of the $16 \times 16$ window, allowing for better scale robustness. CNNにもこういう処理があるでしょう。data augmentationという。
- Combines with the DOG feature detector, SIFT descriptors provide a scale, rotation, and illumination invariant detector/descriptor pair.

Other Descriptors
- Speeded-Up Robust Features (SURF).
  - use similar concepts to SIFT while being significantly faster to compute.
- Gradient Location-Orientation Histogram (GLOH).
- Binary Robust Independent Elementary Features (BRIEF).
- Oriented Fast and Rotated Brief (ORB).
- Many more.
- Some like SIFT and SURF are patented and should not be used commercially without approval of the authors.
- しかし、some really good algorithms such as ORB match the performance of SIFT and SURF and are free to use even commercially.

# Lesson 3 Part 1: Feature Matching

内容
- How to match features based on a predefined distance function.
- How to perform brute force matching.
- List faster alternatives to brute force matching, when the number of features to be matched is high.

Feature Matching
- Given a feature and its descriptor in image 1, find the best match in image 2.
- Brute force feature matching might not be fast enough for extremely large amounts of features.
  - cv2.BFMatcher().
- Use a multidimensional search tree, usually a k-d tree to speed the search by constraining it spatially.
  - cv2.FlannBasedMatcher().

Brute Force Feature Matching
- Define a distance function $d(f_i, f_j)$ that compares the two descriptors.
- Define distance threshold $\delta$.
- For every feature $f_i$ in Image 1:
  - Compute $d(f_i, f_j)$ with all features $f_j$ in Image 2.
  - Find the closest match $f_c$, the match that has minimum distance.
  - Keep this match only if $d(f_i, f_j)$ is below threshold $\delta$.
- 失敗の場合: our feature detector tries to match a feature from Image 1, for which there is no corresponding feature in Image 2.
  - we can solve this problem by setting a distance threshold $\delta$ on the acceptance of matches.
  - 改善後のbrute forceは土色です。

- we usually define $\delta$ empirically, as it depends on the application at hand and the descriptor we are using.

Distance Function

- Sum of Squared Differences (SSD): $d(f_i, f_j) = \sum_{k=1}^{D} (f_{i,k} - f_{j,k})^2$.
  - penalize variations between two descriptors quadratically, making it sensitive to large variations in the descriptor, but insensitive to smaller ones. つまりNorm-2 cost functionです。deep learningの中にもこういう効果が述べられている。
- Other distance functions:
  - Sum of absolute differences (SAD): $d(f_i, f_j) = \sum_{k=1}^{D} |f_{i,k} - f_{j,k}|$.
    - penalize all variations equally.
  - Hamming Distance: $d(f_i, f_j) = \sum_{k=1}^{D} XOR(f_{i,k}, f_{j,k})$.
    - used for binary features, for which all descriptor elements are binary values.

- Feature detection, description, and matching: These three steps are required to use features for various self-driving applications, such as visual odometry and object detection.
- Feature Matching: https://docs.opencv.org/4.0.0/dc/dc3/tutorial_py_matcher.html

# Lesson 3 Part 2: Feature Matching: Handling Ambiguity in Matching

内容
- What consists an ambiguous match.
- How to handle ambiguous matches through the distance ratio.

Brute Force Feature Matching: Case 3
- Ambiguous matches: $SSD(f_1, f_2) = 9$, $SSD(f_1, f_3) = 9$.
- 解決方法：Distance Ratio.

Distance Ratio [Lowe 1999]
1. Compute $d(f_i, f_j)$ for each feature $f_i$, with all features $f_j$ in Image 2.
2. Find the closest match $f_c$.
3. Find the second closest match $f_s$.
4. Find how better the closest match is than the second closest match. This can be done through distance ratio: $0 \leq \frac{d(f_i, f_c)}{d(f_i, f_s)} \leq 1$.
   - If the distance ratio is close to 1, it means that according to our descriptor and distance function, $f_i$ matches both $f_s$ and $f_c$.
   - In this case, we don't want to use this match in the processing later on, as it clearly is not known to our matcher which location in Image 2 corresponds to the feature in Image 1.

Brute Force Feature Matching: Updated
- Define a distance function $d(f_i, f_j)$ that compares the two descriptors.
- Define distance ratio threshold $\rho$.
  - somewhere around 0.5.
- For every feature $f_i$ in Image 1:

1. Compute $d(f_i, f_j)$ with all features $f_j$ in Image 2.
2. Find the closest match $f_c$ and the second closest match $f_s$.
3. Compute the distance ratio $\frac{d(f_i, f_c)}{d(f_i, f_s)}$.
4. Keep matches with distance ratio $< \rho$.

- distanceのthresholdがなくていいの?

- Ambiguous matches are features in the first image that have a similar distance to two or more features in the second image.
- Ambiguous matches can be handled by computing a distance ratio, and making sure this ratio is lower than a predefined threshold for all of our matches.
- Unfortunately, even with the distance ratio formulation, as much as 50% of typical matches can still be wrong when using modern descriptors.
  - This is because repetitive patterns in images and small variations in pixel values, are often sufficient to confuse the matching process.
  - Outliers.

## Lesson 4: Outlier Rejection

Feature matchers are not very robust to outliers.

内容

- The definition of outliers, and how outliers interfere with model estimation that rely on image feature matchers.
- How to handle outliers through the random sample consensus or RANSAC algorithm.

Image Features: Localization

- Problem: Find translation $T = [t_u, t_v]$ in image coordinate system between image 1 and image 2.
  - In practice, we'd also want to solve for the scale and skew due to different viewpoints.
- Matched feature pairs in images 1 and 2:
  - $f_i^{(1)}, f_i^{(2)} \mid i \in [0, \ldots, N]$.
  - $f_i^{(1)} = \left(u_i^{(1)}, v_i^{(1)}\right)$.
- Model:
  - $u_i^{(1)} + t_u = u_i^{(2)}$.
  - $v_i^{(1)} + t_v = v_i^{(2)}$.
- Solve using least squares:
  - $t_u = \frac{1}{N}\sqrt{\sum_i \left(u_i^{(1)} - u_i^{(2)}\right)^2}$.
  - $t_v = \frac{1}{N}\sqrt{\sum_i \left(v_i^{(1)} - v_i^{(2)}\right)^2}$.
- Outliers can comprise a large portion of our feature set, and typically have an out-sized negative effect on our model solution, especially when using least squares estimation.
  - だからidentify these outliers and avoid using them in least squares solution.
- $M = 1$ pair of feature matches.
  - one pair of featureで既に$t_u, t_v$を計算できるから。
- We use a tolerance to determine the inliers, since it is highly unlikely that we satisfy the model with a 100% precision.
  - なかなかscaleが大問題になりそうだと感じている。

Random Sample Consensus (RANSAC)

- Fischler & Bolles 1981.
- RANSAC Algorithm
  - Initialization
    1. Given a Model, find the smallest number of samples, $M$, from which the model can be computed.
  - Main Loop
    2. From your data, randomly select $M$ samples.
    3. Compute model parameters using the selected $M$ samples.
    4. Check how many samples from the rest of your data actually fits the model. We call this number the number of inliers $C$.
    5. If $C >$ inlier ratio threshold or maximum iterations reached, terminate and return the best inlier set. Else, go back to step 2.
    - ここのbest inlier setは多分$C$が一番大きい時の$C$個inliers.
    - つまりこの$C$個inliersからmodel parametersを計算する? 多分$C + M$個だろう。
    - “maximum iterations reached”は多分このmaximum iterations内の一番大きい$C$時のinlier setをbest inlier setとして使うでしょう。
  - Final Step
    6. Recompute model parameters from entire best inlier set.

- Outliers are wrong feature matching outputs, that can occur due to errors in any of the three stages of feature usage.
- RANSAC can be used to efficiently arrive to a good model even when outliers are among the matched features.
- Outlier removal is a key process in improving robustness when using feature matching, and greatly improves the quality of localization results.

# Lesson 5: Visual Odometry

単語
- headlight: A vehicle’s headlights are the large powerful lights at the front.
- Direct linear transformation: (https://en.wikipedia.org/wiki/Direct_linear_transformation) an algorithm which solves a set of variables from a set of similarity relations: $x_k \propto Ay_k$, for $k = 1, \ldots, N$. where $x_k$ and $y_k$ are known vectors, $\propto$ denotes equality up to an unknown scalar multiplication, and $A$ is a matrix (or linear transformation) which contains the unknowns to be solved.

内容
- Why visual odometry is useful for self-driving cars.
- How to perform visual odometry using image features in consecutive frames, along with their 3D position in the world coordinate frame.

Visual Odometry
- Visual Odometry (VO): The process of incrementally estimating the pose of the vehicle by examining the changes that motion induces on the image of its onboard cameras.
  - similar to the concept of wheel odometry, but with cameras instead of encoders.
- VO Pros:
  - Not affected by wheel slip in uneven terrain, rainy/snowy weather, or other adverse conditions.
  - More accurate trajectory estimates compared to wheel odometry.
    - This is because of the larger quantity of information available from an image.
- VO Cons: (大事)
  - Usually need an external sensor to estimate absolute scale. わかる!
    - we usually cannot estimate the absolute scale from a single camera.
      - often a second camera or an inertial measurement unit.

    - What this means is that estimation of motion produced from one camera can be stretched or compressed in the 3D world without affecting the pixel feature locations by adjusting the relative motion estimate between the two images. いい文ですね！
- Camera is a passive sensor, might not be very robust against weather conditions and illumination changes.
    - difficult to perform VO at night and in the presence of headlights and streetlights.
- Any form of odometry (incremental state estimation) drifts over time.
    - For this reason, we often quote VO performance as a percentage error per unit distance travelled.

Problem Formulation

- Estimate the camera motion $T_k$ between consecutive images $I_{k-1}$ and $I_k$:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}.$$

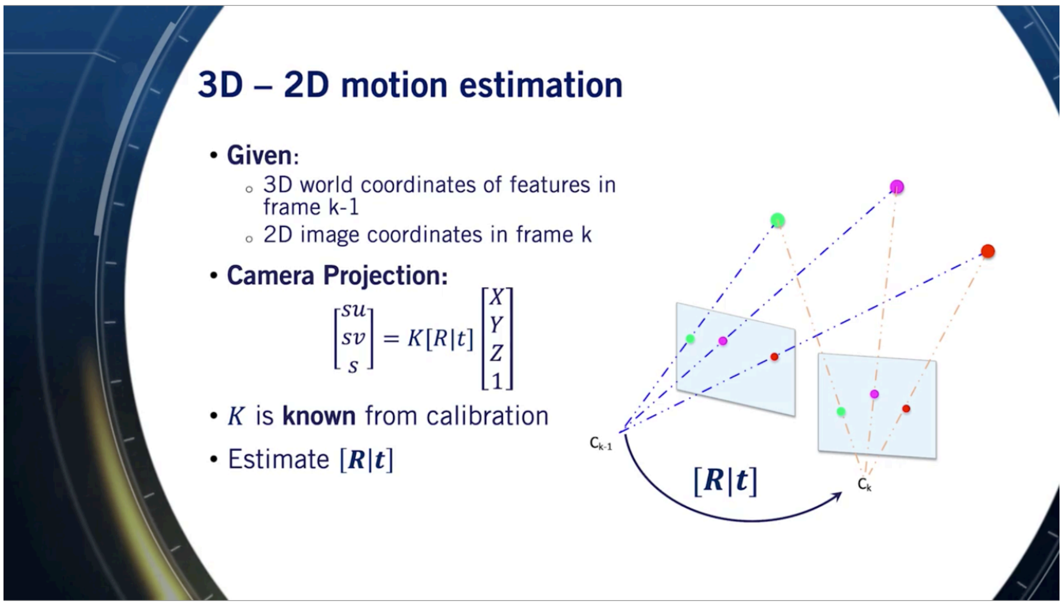
- Concatenating these single movements allows the recovery of the full trajectory of the camera, given frames $C_1, \dots, C_m$.
    - Since the camera is rigidly attached to the autonomous vehicle, this also represents an estimate of the vehicle's trajectory.

Visual Odometry

- Given: two consecutive image frames $I_{k-1}$ and $I_k$.
- Extract and match features $f_{k-1}$ and $f_k$ between two frames $I_{k-1}$ and $I_k$.
- Estimate motion between frames to get $T_k$.

Motion Estimation

- Correspondence types:
    - 2D-2D: both $f_{k-1}$ and $f_k$ are defined in image coordinates.
        - This form of visual odometry is great for tracking objects in the image frame.
        - This is extremely useful for visual tracking and image stabilization in videography. わかる！
    - 3D-3D: both $f_{k-1}$ and $f_k$ are specified in world 3D coordinate frame.
        - requires the ability to locate new image features in 3D space, and is therefore used with depth cameras, stereo cameras, and other multi-camera configurations that can provide depth information.
    - 3D-2D: $f_{k-1}$ is specified in 3D and $f_k$ are their corresponding projection on 2D.



## 3D – 2D motion estimation

- **Given**:
    - 3D world coordinates of features in frame k-1
    - 2D image coordinates in frame k
- **Camera Projection:**

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = K[R|t]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- $K$ is **known** from calibration
- Estimate $[\boldsymbol{R}|\boldsymbol{t}]$

3D-2D motion estimation

- ORB-SLAMで見たことがあるが、まだ完全に分かっていない。逆に難しいのは3Dの$f_{k-1}$と2Dの$f_k$はどうやってお互いにcorrespondしているの?
- Note that since we cannot recover the scale for a monocular visual odometry directly, we include a scaling parameter $s$ when forming the homogeneous feature vector from the image coordinates.
  - “forming the homogeneous feature vector from the image coordinates”は$\begin{bmatrix} su \\ sv \\ s \end{bmatrix}$です。
  - 多分メインの課題は$s$でしょう?

Perspective N Point (PNP)

- Camera Projection: $\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = K[R\,|\,t]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \forall f_i.$
- PnP
  - Solve for initial guess of $[R\,|\,t]$ using Direct Linear Transform (DLT)
    - Forms a linear model and solves for $[R\,|\,t]$, with methods such as singular value decomposition (SVD).
    - However, the equations we have are nonlinear in the parameters of $R$ and $t$.
  - Improve solution using Levenberg-Marquardt algorithm (LM).
    - an iterative nonlinear optimization technique.
  - Need at least 3 points to solve (P3P), 4 if we don’t want ambiguous solutions.
    - when only 3 features, 4 possible solutions results.
  - RANSAC can be incorporate into PnP by assuming that the transformation generated by PnP on four points is our model.（大事）
    - We then choose a subset of all feature matches to evaluate this model and check the percentage of inliers that result to confirm the validity of the point matches selected.
  - cv2.solvePnP(): Solves for camera position given 3D points in frame k-1, their projection in frame k, and the camera intrinsic calibration matrix.
  - cv2.solvePnPRansac(): Same as above, but uses RANSAC to handle outliers.

• Visual odometry can be used to provide accurate trajectory estimate for a self-driving car without suffering from wheel slipping effects due to adverse weather conditions.
• Visual odometry can be performed using 2D-3D feature correspondences and the PnP algorithm.

学んだこと

1.　同じ写真で、ORBでdetect and computeすると、$500 \times 32$のdescriptorsが算出された。

つまりfeature (keypoint)の数は500、descriptorのdimensionは32。

SIFTだと、$1508 \times 128$のdescriptorsが算出された。ちょっと多すぎる?

SURFのdefault hessianThresholdは100。100だと、$3431 \times 64$のdescriptorsが算出された。
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html
によると、In actual cases, it is better to have a value 300-500.

400にすると、$1580 \times 64$は算出された。

FAST detector (StarDetector)とBRIEF extractorを一緒に使うと、$384 \times 32$のdescriptorsが算出されている。

2. FLANN based matching: https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html
Lowe’s ratioも入っている。

この例はSURFを使っている。多分SURFが一番いい?
違う。ORBはbinary descriptor。下記の資料を読み終わったら理解できる。https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html
まだ読み終わっていないけど。。。とりあえずORBではなくSURFを使っている。
3. matchの構造体はDMatchクラス。data memberはdistance, imgIdx, queryIdx, trainIdx。Lowe’s ratioを適用するときはdistanceを使う。
4. ORB-SLAMをやった時も既に知ってたが、また忘れた。cv2.recoverPoseから貰ったrmat, tvecは、cameraの軌跡を計算する時、1フレーム目座標系にしないといけないので、rmat, tvecのinverseを掛け算する。
$T = T@np.row\_stack((np.column\_stack((rmat.T, -rmat.T@tvec)), [0,0,0,1]))$
にした。$T$の初期値は1 frame目のカメラ座標。1 frameを基準座標系としたら、$T$の初期値は
$np.eye(4)$です。
$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix}.$$