# ECEn 631 Motion Field

## Objectives:

- Learn optical flow and its limitations.
- Learn feature detection and feature tracking.
- Learn to track features across multiple frames to get correspondences for long baseline applications.
- Learn to compute the fundamental matrix from images taken by one camera from multiple views.

## Instructions:

- Generate a PDF file that includes your video links for all three tasks.
- Submit your PDF file and source code file(s) in one zip file without the folder or directory.
- Use your first name and last name (e.g., justinsmith.zip) as the file name.
- Login to myBYU and submit your work through BYU Learning Suite online submission.
- Download the Motion Field video from BYU Learning Suite.
- Convert the image frame to 8-bit single channel using cvtConvert() with the CV_RGB2GRAY flag before processing.
- Undistortion of the images before processing is not necessary.
- Use goodFeatureToTrack() function to detect initial features to track.

## Task 1: Optical Flow 30 points

- Use the pyramid LK method to obtain a **spares** motion field of ~500 features (corners) for each frame pair.
- Input only the previous frame and current frame and do not create or use the pyramids.
- Use goodFeaturesToTrack() to detect the best ~500 features from the previous frame and enter these points as the previous points to the calcOpticalFlowPyrLK() function.
- Set the pyramid level to 0 and obtain motion field sequences that are calculated between Frames *n* and *n*+1 (*n* = 1, 2, …N-1), Frames *n* and *n*+2 (skip one frame for *n* = 1, 2, …N-2), Frames *n* and *n*+3 (skip two frames for *n* = 1, 2, … N-2), and so on until the number of found matches falls below 50% of the number of feature points in the previous frame.
- Show motion vectors in each frame by drawing a green dot at the location of the previous point and a red line to show where it moves to in the next frame.
- Combine the first motion field sequence (no skipping frames) and the last motion field sequence (only ~50% of matches are found) into one video and include your YouTube video link in your PDF file.
- Repeat the same process above and use the same parameters (especially the search size) but with a pyramid level of 2 or higher and include your video link in your PDF file.
- Include your observation and what you learn from this task in your PDF file.
- Submit your code.

## Task 2: Feature Matching 30 points

- Use a template matching method (SSD or NCC) to obtain a sparse motion field of ~500 features (corners).
- Choose your own template size and search window size (can vary for different baseline but large enough to get a match).
- Obtain motion field sequences that are calculated between Frames *n* and *n*+1 (*n* = 1, 2, …N-1), Frames *n* and *n*+2 (skip one frame for *n* = 1, 2, …N-2), Frames *n* and *n*+3 (skip two frames for *n* = 1, 2, … N-2), and so on until the number of skipped frames reaches the same number as that in Task 1 (number of matches drops below 50%).
- Show motion vectors in each frame by drawing a green dot at the location of the previous point and a red line to show where it moves to in the next frame.
- Combine the first motion field sequence (no skipping frames) and the last motion field sequence into one video and include your video link in your PDF file.
- Include your observation and what you learn from this task in your PDF file.
- Submit your code.

## Task 3: Multi-Frame Feature Tracking 40 points

Finding corresponding feature points is an important task for 3D vision applications. For a calibrated stereovision system, image pairs can be rectified, and the corresponding feature pairs can be located along the horizontal lines. Even without rectification, epipolar geometry (constraint) can be used to assist finding the corresponding feature pairs. For structure from motion (SFM) applications, the camera motion (R & T) between any two frames is unknown and the corresponding features must be tracked across multiple frames. Camera motion and 3D object structure can be calculated using these matching features from multiple frames.

- Write a program to
  - Detect ~500 good features from the first frame using goodFeaturesToTrack().
  - Modify the feature matching function you developed for Task 2 to detect and track the matching features across multiple frames.
  - Use the matching features between two consecutive frames and findFundamentalMat() function (with CV_FM_RANSAC flag) to detect outliers. The findFundamentalMat() function returns a status vector that shows which feature points are not good (incorrect motion vector).

  - Remove these outliers from the list of the matching feature points.
  - Use the remaining feature points (outliers removed) found in the current frame to find their matching features in the next frame.
  - Some matching feature points in the previous frames will disappear or become outliers in the subsequent frames.
  - The difference between this task and Task 2 is that the intermediate frames can now be used to assist feature matching and tracking (no skipping frames).
  - For example, you can first match features in Frames *n* and *n*+1 and then match the matched features in Frame *n*+1 to features in Frame *n*+2 and so on. The end result is a list of matched features between the beginning frame *n* and the ending frame *n*+*m*.
  - *m* is the number of frames when the number of matched features drops to below 50% in Task 1.
- Show motion vectors in each frame *n* by drawing a green dot at the location of the previous point and a red line to show where it moves to in the *n*+*m* th frame.
- Include your observation and what you learn from this task in your PDF file.
- Submit your code.