# Introduction to Computer Vision

Alvin Lin

August 2018 - December 2018

## Local Features

We want to find patches that are "worth representing" to match from image to image. These will represent either textures or objects. Ideally, they should be covariant to translation, rotation, and scale. If the image is translated, rotated, or scaled, the neighborhoods still preserve their unique properties. They should also be localizable in translation, rotate, and scale, meaning we can estimate the position, orientation and size of the patch.

### Good Features

- Repeatability: the same feature can be found in several images despite geometric and photometric transformations.
- Saliency: each feature has a distinctive description.
- Compactness and efficiency: many fewer features than image pixels.
- Locality: a feature occupies a relatively small area of the image and is robust to clutter and occlusion.

Feature points are often used for image alignment, 3D reconstruction, motion tracking, robot navigation, indexing and database retrieval, and object recognition.

### Finding Corners

A key property of corners is that in the region around a corner, the image gradient has two or more dominant directions. A strategy to find corners is to find centers and estimate the scale from the center. Once we have that, we can estimate the orientation. In a small enough window, a corner has a large gradient change in all directions. Consider shifting a small window $W$ by $(u, v)$:

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

This yields an error comparing each pixel before and after. If we assume a small amount of motion we can use a first order Taylor Series expansion of $I$:

$$\begin{aligned}
I(x+u, y+v) &\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\
&\approx I(x,y) + [I_x \ I_y] \begin{bmatrix} u \\ b \end{bmatrix} \\
E(u,v) &\approx \sum_{(x,y)\in W} \left[ I(x,y) + [I_x \ I_y] \begin{bmatrix} u \\ b \end{bmatrix} - I(x,y) \right]^2 \\
&\approx \sum_{(x,y)\in W} \left[ [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} \right]^2 \\
&= \sum_{(x,y)\in W} [u \ v] \begin{bmatrix} (I_x)^2 & I_x I_y \\ I_y I_x & (I_y)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\
H &= \begin{bmatrix} (I_x)^2 & I_x I_y \\ I_y I_x & (I_y)^2 \end{bmatrix}
\end{aligned}$$

We can define shifts with the smallest and largest change in $E(u, v)$, represented as eigenvalues and eigenvectors of $H$.

- $x_+$: direction of largest increase in $E(u, v)$
- $\lambda_+$: amount of increase in direction $x_+$
- $x_-$: direction of smallest increase in $E(u, v)$
- $\lambda_-$: amount of increase in direction $x_-$

$$\begin{aligned}
Hx_+ &= \lambda_+ x_+ \\
Hx_- &= \lambda_- x_-
\end{aligned}$$

We want $E(u, v)$ to be large for for small shifts in all directions. The minimum of $E(u, v)$ should be large over all unit vectors $[u \ v]$. In general, we can do feature detection by:

1. Computing the gradient at each point in the image.
2. Create the $H$ matrix from the entries in the gradient.
3. Compute the eigenvalues.
4. Find points with the largest response $\lambda > threshold$.
5. Choose those points where $\lambda$ is a local maximum as features.

## Invariance and Covariance

We want features to be invariant to photometric transformations and covariant to geometric transformations. An image is invariant if it is transformed and features do not change. It is covariant if features are detected in the same corresponding locations between two transformed versions of the same image.

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech