|  |  |  |  |
| --- | --- | --- | --- |
| **06-blobs**  We detected edges by convolving with a filter Convolution with a filter can be viewed as comparing a small "picture" of the desired feature against all local regions in the image.  To detect blob, we need a filter which is shaped like a blob  **Blob detector**  • This Filter is called Laplacian of Gaussian (LoG) • This filter can detect blobs at different scale by changing the radius of the filter    In scale-space analysis Scaled Normalized LoG filter is used(“norm” before G)  **Implementation** Blurring an image with a Gaussian and then taking its Laplacian is equivalent to convolving directly with the LoG filter.    The Laplacian of Gaussian LoG is the second derivative of theGaussian filter. LoG is given by:    **Characteristic scale**  • We can find the characteristic scale of the blob by convolving it with scale-normalized Laplacians at several scales (𝜎) and looking for the maximum response    • We want to extract keypoints with characteristic scale that is covariant with the image transformation  **Difference of Gaussian**  • Difference of two Gaussians (DoG) efficiently approximate LoG: 1. Blur image with σ Gaussian kernel 2. Blur image with kσ Gaussian kernel 3. Subtract 2 from 1    **07-Feature Descriptor**  **Local Feature Extraction**  1. Detection: **Identify** points or regions in the image that are distinctive (**key points** or interest points or feature points) e.g. Harris corner detector, LoG, DoG 2. Description: Compute a **descriptor** for each interest point to capture its local appearance e.g SIFT 3. Matching: Compute distance between feature vectors to **find correspondence** | **Scale-invariant feature transform (SIFT)**  • SIFT is an algorithm that detects and describes local keypoints so that they are to scale, rotation, and illumination changes  • SIFT uses DoG to detect keypoints at multiple scales  **SIFT Descriptor**  1. Find the orientation of the keypoint by computing the gradient magnitude and gradient orientation in a 16x16 window around the keypoint      2. Form weighted histogram (8 bin) for 4x4 regions. Concatenate 16 histograms in one long vector of 128 dimensions    Example    -The size of the circle indicates the scale at which the keypoint was detected  -The orientation of each keypoint is represented by the accompanying line.  -Each keypoint is characterized by a 128 dimensional vector that encodes the gradient orientations in its local neighborhood (16x16) | **SIFT Descriptor Matching**  • Given a feature in I1, how to find the best match in I2?  1. Define distance function that compares two descriptors. 2. Test all the features in I2, find the one with min distance.    **Euclidean Distance function**  • Given two vectors 𝒂 and 𝒃 each of dimension (1x128), the Euclidean distance is defined as:    **08-Feature matching**  • Compute distance between the feature descriptors and match points to the lowest distance • Let’s assume 𝑓1, 𝑓2 are 2 feature vectors and 𝑛 is the length of the vector, then distance between the two vectors can be computed using:    • Problems: Non-distinctive features could have lots of close matches, only one of which is correct  **Nearest Neighbor Distance Ratio**  • 𝑁𝑁1/𝑁𝑁2 where NN1 is the distance to the first nearest neighbor and NN2 is the distance to the second nearest neighbor.  • Sorting by this ratio puts matches in order of confidence.    **Can we refine this further?**  Even after filtering matched points using the Nearest Neighbor Ratio (NNR) test, some incorrect matches msay still remain.  Solution: Model fitting and Alignment  **Image Transformation**  Transformation T is applied to all points in the image, and it changes the coordinates of all points. • For linear transformations, we can represent T as a matrix    **Common Transformations** | **Scaling**  •Scaling a coordinate means multiplying each of its components (X,Y,etc.) by a scalar  • Uniform scaling means this scalar is the same for all components  • Non-uniform scaling: different scalars per component  **Affine Transformations**  Properties of affine transformations: • -Lines map to lines • -Parallel lines remain parallel • -Ratios are preserved • -Closed under composition  **Projective Transformations** **(Homography)**  • Occurs when viewing a three-dimensional scene from a two-dimensional plane • Combination of Affine transformations, and projective warps (represented by 3x3 matrix)    Properties of projective transformations: • Lines map to lines • Parallel lines do not necessarily remain parallel • Ratios are not preserved • Closed under composition • Projective matrix has 8 DOF        **RANSAC conclusions**  Good: Robust to outliers, Effective in handling datasets with a significant amount of noise Bad: Computationally expensive, particularly when dealing with large datasets or complex models, No guarantee of convergence to the optimal solution Common applications: Computing a homography (e.g., image stitching), Estimating fundamental matrix (relating two views) |

|  |  |  |  |
| --- | --- | --- | --- |
| **02-Image Formation**  Image Formation  An image is matrix of pixels values • Value of a pixel in matrix = how much light • Typical ranges: • [0-255], fits in byte (uint8) • [0-1], floating point (float32)  **Color Image** • 3-dimensional array (3-channel image) • Each pixel in the image is represented by three values corresponding to the intensities of red, green, and blue colors  • Trichromatic theory: Different combinations of signals from these cones allow us to perceive a wide range of colors.  HSV (Hue, Saturation, Value) Color space  - Different model based on perception of light - Hue: what color - Saturation: how much color - Value: how bright - Allows easy image transforms - - Shift the hue - - Increase saturation  To grayscale: 0.30 × 𝑅 + 0.59 × 𝐺 + 0.11 × 𝐵  **Histograms** • Histogram is a graphical representation of the distribution of pixel intensities • How many times does a particular pixel value appear in the image  • The x-axis of the histogram represents the intensity values (bins), and the y- axis represents the frequency or number of pixels at each intensity level.  **03-Filtering**  **Image Filtering** • Slide a fixed-size window (called filter/kernel) over image and perform the same simple computation at each window location • Compute function of local neighborhood at each position • Also called Convolution  **Linear**: (Box filter does averaging)  A screenshot of a diagram  Description automatically generated  Filter(I,f1+f2) = filter(I,f1)+filter(I,f2)  Filter(kI,f) = kfilter(I,f)  Practical details: Dealing with edges • To control the size of the output, we need to use padding • What values should we pad the image with? • Zero pad • Wrap around • Copy edge • Reflect across edge  Note: Filtering vs. Convolution • In classical signal processing terminology, convolution is filtering with a flipped kernel, and filtering with an upright kernel is known as cross-correlation. We will call everything convolution. | **Gaussian Filtering**: (good for zero-sum noise) •What’s wrong with box filter? Excessively smooth edges •What’s the solution? • Weighted average to the neighboring pixels, giving more emphasis to the central pixels and less to the ones farther away  A math equation with a number  Description automatically generated with medium confidence  **Median Filtering** (good for salt and pepper):  A median filter operates over a window by selecting the median intensity in the window  **04-Edges**  **Edge Detection** •An edge is a place of rapid change in the image intensity function  •Why Edges matter? Intuitively, edges carry most of the semantic and shape information in the image  A white background with black text  Description automatically generated  A grid of squares with numbers and symbols  Description automatically generated  A diagram of a mathematical equation  Description automatically generated  A math equations and formulas  Description automatically generated  **Canny Edge Detector** **Algorithm**: 1. Smooth image (only want “real” edges, not noise) 2. Calculate gradient direction and magnitude 3. Non-maximum suppression perpendicular to edge  - Want single pixel edges, not thick blurry lines - Need to check nearby pixels - See if response is highest 4. Threshold into strong, weak, no edge  - Still some noise - Only want strong edges - 2 thresholds T and t, 3 cases - R > T: strong edge (always edges) - R < T but R > t: weak edge (edge iff connected to strong edge) - R < t: no edge 5. Connect together components | **05-corners**  Why extract keypoints? •Motivation: image alignment • We have two images – how do we combine them?  Step 1: extract keypoints Step 2: match keypoint features  Step 3: align images  • Sky: bad • - Very little variation • - Could match any other sky • Edge: ok • - Variation in one direction • - Could match other patches along same edge  • Corners: good! • - Only one alignment matches  **Characteristics of Corners**  •Compactness and efficiency • - Many fewer corners than image pixels •Saliency • - Corners are distinctive •Locality • - A corner occupies a small area of the image; robust to occlusion •Repeatability • - The corner can be found in despite geometric and photometric transformations  **Applications** • Corners are useful for: • Image alignment • 3D reconstruction • Motion tracking • Robot navigation • Database indexing and retrieval  **Corners** •Basic idea: A small window is placed over the image. If shifting it in any direction causes a significant intensity change, a corner is present.  A collage of different images  Description automatically generated  A diagram of a function  Description automatically generated  A diagram of a circle with a red line  Description automatically generated  A whiteboard with text and symbols  Description automatically generated | A math equations on a white background  Description automatically generated    **Behavior w.r.t. image transformations** •To be useful for image matching, “the same” corner features need to show up despite geometric and photometric transformations •We need to analyze how the corner response function and the corner locations change in response to various transformations  A diagram of a graph  Description automatically generated  • Harris corner detector is **invariant to additive changes in intensity**, i.e. changes in overall “Brightness” • It is not **invariant to scaling of intensity**, i.e. Changes in “Contrast”  **• How do the detected corner locations change if the image pattern is translated?** • - Harris matrix is **shift-invariant**, and so is the corner response function • - However, the locations of the corners are **equivariant (or covariant) w.r.t. shifts**  **• How do the detected corner locations change if the image pattern is rotated?** • - the rotation changes but the eigenvalues stay the same, so the response function is **invariant** • - The locations of the corners are **equivariant (or covariant) w.r.t. rotations**  **• How do the detected corner locations change if the image pattern is scaled?**  • - Assuming fixed-size neighborhoods for calculating the Harris matrix, the corner response function is **not invariant** and the corner locations are **not equivariant w.r.t. scaling**  **Behaviour of harris corner w.r.t Image scaling**  Harris corner detector is **not scale invariant** • We cannot use the same window to detect keypoints with different scale. • To detect larger corners we need larger windows. • IDEA: Multi scale-space filtering |