Report on Vanishing Point Estimation in Computer Vision.

Code Overview

The code is structured in several parts, each addressing a specific aspect of image processing:

1. Image Loading and Display
2. Image Blurring
3. Edge Detection
4. Line Detection
5. Line Drawing
6. Vanishing Point Detection
7. RANSAC Algorithm

Image Loading and Display

Function: load_images(filename)

This function loads a picture from the given file location and changes its colour space from BGR to RGB.
Reads the picture using OpenCV's cv2.imread function and converts the colour space using cv2.cvtColor. This conversion is required because Matplotlib requires RGB format for proper picture presentation, while OpenCV reads images in BGR format by default. The program terminates, and an error message is produced if the file cannot be read or is not an image.

Function: show_images(image)

Utilising Matplotlib, the picture is shown.

To provide a clear view of the picture, plt.imshow is used to render the image and plt.axis('off') to conceal axis labels.

Image Blurring

apply blur(image)

Gives the picture a Gaussian blur.

By smoothing the picture and applying a Gaussian kernel to it, OpenCV's cv2.getGaussianKernel and cv2.filter2D reduce noise and enhance detail in the image.

The degree of blurring is controlled by using a kernel size of 70 and a standard deviation of 4.

Edge Detection

Function: apply_canny(blur_image)

Uses Canny edge detection to enhance the blurry image.

cv2.Canny locates gradients in the picture to recognise edges. Strong and weak edge detection is aided by the thresholds (40 and 70), while edge detection is improved by the aperture Size and L2gradient parameters.

Output: An picture with the identified edges highlighted is the result.

Line Detection

Function: detect_lines(edge_image)

Applying the Hough Transform, detects lines in the edge-detected picture.

Using cv2.HoughLines, lines are detected according to their parameterised representation (rho and theta). To prevent noise, only lines with precise orientations—roughly vertical or horizontal—are retained.

For theta, the technique employs a resolution of 1 pixel and 1/120 radians; for line detection, it uses a threshold of 55.

Line Drawing

Function: show_lines(image, lines)

Represents identified lines on the source picture.
Using cv2.line, lines are drawn on the picture after the line parameters (rho and theta) are converted to Cartesian coordinates. Show_images is utilised to showcase the outcome.
Output: Red lines are used to indicate the detections that were produced in the original image.

Vanishing Point Detection

Function: find_intersection_point(line1, line2)

Determines the location at which two lines meet.
Determines the intersection by solving a system of linear equations created by the line

equations. In case the lines are not parallel, it returns the location of junction.

Function: find_dist_to_line(point, line)

Determines the separation between a point and a line.
Determines the perpendicular distance between a point and a line using geometric formulae, which aids RANSAC in evaluating how well lines suit a certain model.

Function: RANSAC(lines, ransac_iterations, ransac_threshold, ransac_ratio)

Uses the RANSAC algorithm to determine the vanishing point.
Line pairs are sampled iteratively, their intersections are computed, and the inliers (lines near the junction) are identified. The inlier ratio is used to refine the vanishing point. RANSAC enhances robustness and manages outliers.

Function: show_point(image, point)

Shows the picture's vanishing point.
Annotates the image with the vanishing point marked, and marks the point with a green circle. Additionally prints the vanishing point's coordinates.

Outcomes

- **picture loading:** The picture loads and is shown successfully.
- **Blurring:** This technique lessens noise in the image and gets it ready for edge recognition.
- **Edge Detection:** The foundation for line detection, this technique highlights edges in the picture.
- **Line Detection:** This technique locates and illustrates important lines in a picture.
- **Vanishing Point Detection:** This technique, which discerns the vanishing point from observed lines, is helpful in figuring out perspective in pictures.

Options and Things to Think About

- **Blurring Alternatives**: Depending on the properties of the picture and the intended smoothing effect, other filter types, such as bilateral or median filters, might be utilised.
- **Alternatives for Edge Detection**: For distinct edge detection properties, Sobel or Laplacian edge detectors may be utilised in place of Canny.
The probabilistic Hough transform (cv2.HoughLinesP) presents an alternative for more effective line identification in certain situations.
- **Alternatives to Vanishing Points:** Depending on the requirements of the application, different algorithms like the Direct Linear Transform (DLT) or robust statistical techniques may be employed.