**Task 2:** Write up

**Geometric Transformations: Perspective Correction and Warping**

Geometric transformations are widely used in image processing to transform the perspective of objects within an image. These transformations allow us to simulate different viewpoints, correct distortions caused by camera angles, or align objects for recognition and analysis. One of the most common types is the perspective transformation, which enables us to warp or skew an image as if viewed from a different angle.

**Algorithms Used to Transform Perspective:**

1. **Homography Matrix**  
   This is a 3×3 matrix used to map points from one plane to another. Given four or more point correspondences between a distorted view and the desired perspective, the homography matrix can project the selected region into a new viewpoint. It is mathematically grounded in projective geometry and forms the basis of many image correction techniques.
2. **OpenCV’s getPerspectiveTransform()**  
   It takes four source points from the original image and four destination points representing the new perspective to calculate the transformation matrix. It solves a system of equations to determine how to warp the image from one view to another.
3. **OpenCV’s warpPerspective()**  
   Once the transformation matrix is obtained, warpPerspective() applies it to the image and produces a warped result. The output simulates the object as seen from the desired new angle.

These techniques are used in:

* Document scanning (flattening curved pages)
* Augmented reality (anchoring digital objects to real world surfaces)
* Object recognition and alignment
* Image rectification in aerial or satellite imagery

**Task 3:** Write up

**Pose Estimation and Human Activity Recognition**

**Pose Estimation**

Pose estimation is a computer vision technique aimed at detecting and localising key points of interest in images or videos, enabling a system to understand the orientation and posture of humans or objects. Its primary goal is to represent spatial configurations in a structured manner, facilitating further analysis like activity recognition, interaction detection, or behavioural analysis.

**Common Pose Estimation Algorithms:**

* **OpenPose**:  
  OpenPose detects multiple body keypoints through deep neural networks and Part Affinity Fields, enabling robust multi person tracking. While accurate, it demands significant computational resources.
* **MediaPipe Pose (BlazePose)**  
  Developed for real time applications, MediaPipe Pose is efficient, providing accurate keypoint detection suitable for mobile and embedded applications, though it might be less robust in complex scenarios compared to heavier models**.**
* **HRNet (High-Resolution Network)**  
  HRNet preserves high resolution representations throughout its architecture, leading to superior accuracy on benchmarks. However, this high accuracy comes at the cost of increased computational complexity.

**Human Activity Recognition (HAR)**

Human Activity Recognition refers to techniques employed by systems to classify and interpret human actions from visual or sensor based input. Activities recognised typically range from simple gestures to complex behaviours such as object interactions or daily tasks. The field is crucial for applications like surveillance, interactive interfaces, health monitoring, and automated inspection.

**Common HAR Algorithms:**

* Support Vector Machines (SVM) with Handcrafted Features  
  Traditional HAR approaches utilise handcrafted visual descriptors like Histogram of Oriented Gradients (HOG) or optical flow features classified by SVMs. This approach is interpretable and computationally efficient but may fail to generalise effectively across varying scenarios.
* **CNN + LSTM Networks**  
  These combine spatial feature extraction using Convolutional Neural Networks with temporal sequence analysis via Long Short Term Memory networks, providing robust recognition for dynamic activities. They require substantial training data and computation resources.
* **I3D (Inflated 3D ConvNet)**  
  I3D applies 3D convolutional layers to simultaneously capture spatial and temporal information in videos. Known for its high accuracy, it is particularly effective but computationally intensive.
* **Temporal Segment Networks (TSN)**TSN divides videos into temporal segments, analysing frames sparsely yet efficiently to classify activities. While less detailed in fine grained motion capture, TSN offers computational efficiency and scalability.

**Relevance of Edge-Finding and Feature-Finding Algorithms**

Edge and feature detection algorithms, including methods like the Canny edge detector and feature descriptors such as SIFT (Scale-Invariant Feature Transform) and ORB (Oriented FAST and Rotated BRIEF), play a foundational role in many traditional computer vision tasks. Specifically, in pose estimation and activity recognition:

* Edge detection techniques identify object boundaries and shapes, crucial for delineating human figures and limbs.
* Feature descriptors provide invariant and robust visual cues necessary for tracking, matching, and recognising object interactions.
* Despite the dominance of deep learning models, these classical algorithms remain significant for tasks such as preprocessing, annotation, and providing interpretable baselines or hybrid solutions.

**Summary:**  
  
Pose estimation and human activity recognition represent two pivotal areas within computer vision, contributing to understanding human motion and interactions through visual data analysis. The algorithms used in each field differ significantly in terms of their accuracy, efficiency, complexity, and computational requirements. Furthermore, edge and feature detection techniques remain relevant, providing essential support to both pose estimation and HAR tasks by supplying a strong foundational information.

**Task 4:** Write up

**Research, Identify, and Explain a Relevant Technique for Identifying the Human Hand**

The identification of human hands in computer vision is linked to recognition, hand hygiene monitoring, and human computer interaction. In the context of this project identifying when a customer uses their hand to handle unpackaged bread is important to enhance hygiene and trigger appropriate alerts. Among several approaches available one efficient method is MediaPipe.

**MediaPipe Hands**

MediaPipe Hands is a cross platform machine learning framework that performs real time hand tracking by detecting 21 hand keypoints per hand using a two stage pipeline. It first uses a palm detection model to localize the hand region, followed by a hand landmark model that predicts 21 3D coordinates for each detected hand. It combines fast inference with accurate and detailed information making it suitable for detecting hand object interaction scenarios like grabbing bread from a shelf.

**How It Works**

1. **Palm Detection:** A lightweight single shot detector (SSD) model locates the hand region, optimized to detect palms rather than the whole hand to reduce ambiguity and improve efficiency.
2. **Hand Landmark Model:** Once a palm is detected, a second model estimates 21 hand landmarks. These correspond to key joints on the fingers and palm, providing highly granular detail for pose estimation or interaction analysis.
3. **Post Processing:** The landmarks are further smoothed and can be used for gesture recognition, depth estimation, or to determine whether a hand is interacting with an object.

**Why MediaPipe Hands?**

* **Accuracy:** It provides precise detection of both hands and finger joints even in complex poses or partial occlusions.
* **Speed:** The system runs in real-time on both CPU and GPU, making it ideal for low-latency applications.
* **No Training Required:** Unlike deep learning models like YOLO that require annotated datasets, MediaPipe Hands works out-of-the-box.
* **Ease of Integration:** It supports Python and can be integrated with OpenCV for processing frames from video or webcam feeds.

**References**

[1] OpenCV Documentation, “Geometric Image Transformations,” OpenCV.org. [Online]. Available:<https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html>

[2] A. Kaehler and G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, 1st ed., Sebastopol, CA, USA: O'Reilly Media, 2016.

[3] A. Rosebrock, “4 Point OpenCV getPerspective Transform Example,” PyImageSearch, Aug. 25, 2014. [Online]. Available: <https://pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/>

[4] OpenCV Documentation, “Geometric Transformations of Images,” OpenCV.org. [Online]. Available: <https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations.html>

[5] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172-186, Jan. 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/8765346>

[6] Google AI Blog, "BlazePose: On-device Real-time Body Pose Tracking," Google AI, 2020. [Online]. Available: <https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html>

[7] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep High-Resolution Representation Learning for Human Pose Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349-3364, Oct. 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/9052469>

[8] J. Donahue et al., "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677-691, Apr. 2017. [Online]. Available: <https://ieeexplore.ieee.org/document/7558228>

[9] J. Carreira and A. Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [Online]. Available: <https://ora.ox.ac.uk/objects/uuid:c8881c3e-c91d-444c-b142-75f9ff6377af>

[10] OpenCV Documentation, "Feature Detection and Description," OpenCV.org. [Online]. Available: <https://docs.opencv.org/4.x/db/d27/tutorial_py_table_of_contents_feature2d.html>

[11] OpenCV Documentation, "Canny Edge Detection," OpenCV.org. [Online]. Available: <https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html>

[12] F. Zhang, V. Bazarevsky, and M. Grundmann, “MediaPipe Hands: On-device Real-time Hand Tracking,” *Google AI Blog*, 2020. [Online]. Available: <https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html>  
  
[13] MediaPipe, “MediaPipe Hands - ML Pipeline for High-fidelity Hand and Finger Tracking,” *Google Developers*, 2024. [Online]. Available: <https://developers.google.com/mediapipe/solutions/vision/hand_landmarker>

[14] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields,” *CVPR*, 2017. doi: 10.1109/CVPR.2017.143