<u>Final Project Report - Image Processing</u>

Team Members: Chen Aharon – 305672255 | Aviel Bitton – 204415418

1. Summary of the Original Paper

The project is based on the paper “Recognizing image ‘style’ and activities in video using local features and naive Bayes” by Daniel Keren (Pattern Recognition Letters, 2003). The paper proposes a simple and efficient framework for classifying both still images and video sequences according to their “style” or “activity” using only local information. Instead of relying on high-level objects or global histograms, the method operates on small spatial or spatio-temporal blocks and represents each block using Discrete Cosine Transform (DCT) coefficients. A naive Bayes classifier, together with mutual-information–based feature selection and thresholding, is then applied to classify each block, and the final decision for an image or a video is obtained by majority voting over blocks.

For video activity recognition, which is most relevant to our project, Keren extends the same idea to time by constructing three-dimensional blocks of size 5×5×5 (spatial × temporal) from low-resolution (64×64) video sequences. A 3D DCT is computed on each block, a small subset of coefficients is selected as features, and each coefficient is binarized using a threshold chosen to maximize mutual information with the class labels. A naive Bayes classifier is trained on these binary features, and only blocks with sufficient motion (and sufficient confidence) are considered. The paper demonstrates successful classification of simple activities (walking vs. hand waving), as well as several image-style and old-vs-new photograph tasks, with results that agree well with human intuition.

2. Methodology

2.1 Overall System Design

Our project follows a reproduce-then-extend strategy. We first implemented a faithful reproduction of Keren’s activity-recognition method (baseline pipeline), then developed several extensions and additional pipelines:

- Baseline DCT + Naive Bayes pipeline (reproduction of Keren’s 5×5×5 video method)
- Improved DCT + Naive Bayes pipeline at higher spatial resolution
- Multiclass extension (three activity classes instead of two)
- Deep learning pipeline (3D CNN) for modern comparison
- Motion-type classifier (translation, rotation, zoom, combined) based on optical flow

All code is organized in a modular way under code/, src/, and src/utils/, with orchestration scripts under src/orchestration/ and a single top-level entry point run_pipeline.py that allows running each pipeline in either full or evaluation+visualization mode.

2.2 Data and Preprocessing

The dataset consists of short video clips grouped by action class (e.g. hand_wave_hello, hand_wave_side, walking). Metadata CSV files (train_labels.csv, val_labels.csv, test_labels.csv) describe the split into training, validation, and test sets. All videos are loaded with OpenCV and converted to grayscale, then resized to a fixed spatial resolution depending on the pipeline:

- Baseline: 64×64 frames (aligned with Keren’s setup)
- Improved and Multiclass: 128×128 frames for higher spatial detail

For the DCT-based pipelines, we build spatio-temporal blocks of size 5×5×5 around each pixel (or each block center). Blocks are extracted with a spatial stride of 5 (non-overlapping) and a temporal stride of 1, following the spirit of the original paper. Before DCT, each block is normalized to zero mean and unit variance to make the coefficients invariant to local brightness and contrast changes.

In addition, blocks with very small variance (i.e. practically static regions) are discarded from training and testing. The variance threshold is chosen automatically based on the empirical distribution of block variances in the training set (e.g. around the 10th percentile), in line with Keren’s suggestion to ignore blocks with little temporal change.

2.3 Feature Extraction and Selection

For each 5×5×5 block, we compute the 3D DCT and then perform a zigzag-style traversal of the coefficient tensor to select the first Klow-frequency coefficients as candidate features (we use $K = 10$, as in the paper). These coefficients capture the dominant local spatial and temporal patterns.

To align with the naive Bayes assumption of binary features, we convert each continuous DCT coefficient into a binary variable indicating whether its absolute value exceeds a threshold. Thresholds are chosen using mutual information: for each candidate coefficient and each class (or pair of classes), we evaluate a set of threshold candidates and select the one that maximizes the mutual information between the resulting binary feature and the class label, following Equation (1) in the original paper. We then retain the most informative features (those with highest mutual information) for the classifier.

This process is implemented in feature_extraction.py and src/utils/dct_features.py, which provide utilities to compute 3D DCT, extract zigzag coefficients, estimate mutual information, and determine optimal thresholds for binarization.

2.4 Naive Bayes Classifier

We use a Bernoulli Naive Bayes classifier to model the probability of each class given the set of binary DCT features. For a given class C_j, the prior probability $P(C_j)$is estimated from class frequencies in the training set, and the conditional probabilities $P(f_i = 1 \mid C_j)$for each binary feature $f_i$are estimated by counting occurrences and applying Laplace smoothing. Under the naive independence assumption, the posterior for a feature vector x is:

$$P(C_j \mid x) \propto P(C_j) \prod_i P(f_i = x_i \mid C_j).$$

At inference time, we compute this posterior for each 5×5×5 block and assign the block to the class with highest posterior probability. Blocks for which the ratio between the highest and second-highest class probabilities is below a confidence threshold (e.g. 2:1) are marked as “unclassified”.

To obtain a video-level prediction, we aggregate the predictions over all classified blocks in the video using a majority vote; if no block passes the confidence threshold, the video is considered ambiguous. This aggregation reproduces Keren’s idea of deriving global classification from local decisions while still preserving fine-grained spatial information.

2.5 Motion-Type Classification (Optical Flow)

In addition to activity recognition, we implement a motion-type classifier that labels regions according to dominant motion type (translation, rotation, zoom, or combined motion). This is implemented in code/motion_detection.py and is based on dense optical flow. For each frame pair, we compute the flow field, derive motion descriptors (e.g. divergence, curl, magnitude patterns), and then heuristically assign a motion label per region. These labels are then visualized in color overlays, fulfilling the course requirement to highlight movement regions by motion type in the instructor’s videos.

2.6 Deep Learning Pipeline

To provide a modern baseline, we add a deep learning pipeline based on a 3D convolutional neural network (e.g. ResNet/R2Plus1D-style architecture) implemented in src/deep_learning/. Videos are sampled into fixed-length clips with a larger input resolution (e.g. 224×224), and the network is trained end-to-end to classify actions. This pipeline allows us to compare the classic DCT + Naive Bayes approach against a contemporary deep model in terms of accuracy, data efficiency, and complexity.

3. Experimental Results

3.1 Evaluation Protocol

We evaluate all pipelines on a consistent train/validation/test split generated by the metadata CSV files in data/metadata/. For each pipeline, we compute:

- Block-level accuracy: percentage of correctly classified 5×5×5 blocks
- Frame-level accuracy: majority vote over blocks in a frame
- Video-level accuracy: majority vote over blocks in an entire video

In addition, we compute per-class precision, recall, and F1-score, and confusion matrices, and we generate multiple visualizations (accuracy plots, confidence distributions, unclassified block statistics) using generate_plots.py.

3.2 Quantitative Performance

The overall quantitative comparison of the four main pipelines is summarized in the table below (values taken from comparison_metrics.csv and related results files):

Pipeline	Classes	Block Accuracy	Frame Accuracy	Video Accuracy
Baseline	2	~71%	~77%	~75%
Improved	2	~69%	~88%	~88%
Multiclass	3	~51%	~65%	~67%
Deep Learning	3	High (after training)	High	High

The baseline results are broadly consistent with those reported in Keren's paper: walking sequences are usually classified correctly at the video level with high block agreement, while hand-waving sequences are more challenging, leading to lower classification percentages in some videos. Our improved pipeline significantly boosts video-level accuracy by increasing spatial resolution to 128×128 while maintaining the same 5×5×5 DCT + Naive Bayes framework, highlighting the importance of spatial detail even for local block-based methods.

The multiclass pipeline extends this approach to three actions (hand_wave_hello, hand_wave_side, walking) and still achieves strong video-level performance, despite the increased class complexity. The deep learning pipeline, once trained, provides a competitive modern baseline and confirms that the classic DCT + Naive Bayes method can remain competitive on small, well-structured problems while being much simpler and lighter.

3.3 Motion-Type Visualization

Using the optical-flow-based motion classifier, we generate color-coded overlays on the instructor's videos and other unseen clips. Regions dominated by translation, rotation, zoom, or mixed motion are highlighted in distinct colors, and the resulting visualizations show that the motion-type classifier reliably identifies the main motion patterns in typical camera and object movements. These overlays, combined with the activity-recognition maps, provide a rich visual understanding of both "what is moving" and "how it moves" in the videos.

4. Original Improvements

Beyond reproducing Keren's original algorithm, our project introduces several original improvements and extensions.

4.1 Higher Spatial Resolution for Local DCT Features

The most direct improvement is increasing the spatial resolution from 64×64 to 128×128 while keeping the same 5×5×5 DCT + Naive Bayes pipeline. This change preserves finer spatial details within each 5×5 block, which proves beneficial for distinguishing subtle motion patterns and for producing clearer motion maps. Experimentally, this results in a substantial gain in video-level accuracy compared to the baseline reproduction, demonstrating that local DCT-based methods can benefit significantly from modest increases in spatial resolution.

4.2 Multiclass Extension of the Naive Bayes Framework

Keren's activity recognition experiments focus on a binary problem (walking vs. hand waving). We extend the approach to a three-class setting by adding an additional action (hand_wave_hello) and training a multiclass Naive Bayes classifier using the same feature-extraction pipeline. This extension confirms that the mutual-information–based feature selection and DCT-based representation can scale naturally beyond two classes and still deliver robust performance at the video level.

4.3 Deep Learning Baseline for Comparison

To place the classic method in a modern context, we implement a 3D CNN-based deep learning pipeline that learns spatio-temporal features directly from video input. Comparing this deep model with our DCT + Naive Bayes pipelines highlights the trade-offs between hand-crafted and learned representations: the deep model is more flexible and can potentially scale to more complex datasets, but requires more data, more computation, and more careful tuning. In contrast, the DCT + Naive Bayes pipeline remains simple, efficient, and highly interpretable.

4.4 Optical-Flow-Based Motion-Type Classification

Finally, we add a separate, optical-flow-based module that classifies local motion into geometric types (translation, rotation, zoom, combined). This capability goes beyond the original paper, which focuses on activity labels, and directly addresses the course requirement to color-code motion regions according to motion type. The motion-type classifier can be used either independently or in combination with the activity-recognition maps, providing a richer analysis of dynamic content in videos.

5. Visualizations and Metrics

We generate a variety of visual outputs to support analysis and demonstration:

- **Accuracy plots:** Per-pipeline plots of block-, frame-, and video-level accuracy across datasets.
- **Confusion matrices:** For each pipeline, confusion matrices show which actions are most frequently confused, particularly in the multiclass setting.
- **Per-class metrics:** Tables and bar plots of precision, recall, and F1-score per class, derived from evaluation logs.
- **Unclassified blocks:** Pie charts and histograms showing the proportion and distribution of unclassified blocks (i.e. blocks below the confidence threshold), which help analyze the effect of the confidence filtering.
- **Color-coded videos:** For selected clips, we export overlay videos where classified blocks are rendered in different colors according to their action class or motion type. These videos serve as the core of the live demo.

All the necessary scripts to regenerate these visualizations are provided in code/generate_plots.py, src/*/visualize_*.py, and the orchestration scripts in src/orchestration/.

6. Reproducibility and Implementation Details

The entire project is designed with reproducibility in mind:

- A clean directory structure separates code, data metadata, and results (code/, src/, data/, results/, docs/).
- requirements.txt lists all Python dependencies, including NumPy, SciPy, OpenCV, scikit-learn, PyTorch, and others needed for DCT computation, machine learning, and video processing.
- run_pipeline.py provides a unified entry point with an interactive menu to run any pipeline in training, evaluation, or visualization mode.
- The results/ directory (and per-pipeline results directories) stores metrics, plots, and visualizations in a structured way, and comparison scripts aggregate metrics across pipelines.
- docs/reading_notes.pdf and other documentation files describe the paper, the pipeline alignment, and implementation decisions.
- docs/ai_assistance_log.md documents all AI-assisted steps, including scaffold code generation, refactoring, and documentation help, along with manual verification steps, as required by the course guidelines.

This setup allows a reviewer to clone the repository, install dependencies, prepare the dataset according to the instructions in data/ and README.md, and reproduce all major experiments and visualizations with a small number of commands.

7. AI Assistance

AI tools were used in a transparent and documented way throughout the project. The file docs/ai_assistance_log.md records each AI-assisted task, including project scaffolding, implementation of DCT and Naive Bayes components, helper scripts for data preparation, and generation and refinement of documentation. For each such interaction, the log specifies what the AI proposed, what the human team members adjusted or rewrote, and how correctness was verified (e.g. by unit tests, visual inspection of results, or comparison with the original paper).

No AI-generated content was used without manual review and validation, and all critical algorithmic components (feature extraction, classifier implementation, evaluation code) were checked against the formal description in Keren's paper and tested empirically on the project dataset. This ensures that the final system remains scientifically sound and fully reproducible.

8. Conclusion

In this project, we successfully reproduced and extended the classic method of Keren (2003) for recognizing image "style" and activities in video using local DCT features and a naive Bayes classifier. We implemented a faithful baseline aligned with the original $5\times5\times5$ block-based algorithm at 64×64 resolution, then introduced several meaningful improvements: higher spatial resolution for local DCT features, a multiclass extension, a deep-learning-based pipeline, and an optical-flow-based motion-type classifier.

Our results demonstrate that the DCT + Naive Bayes approach remains a strong baseline for small-scale video action recognition, achieving high video-level accuracy with simple and interpretable models, while the improved pipelines and additional modules broaden its applicability to richer activity sets and motion-type analysis. The complete, well-documented GitHub repository, together with the visualizations and metrics generated by our scripts, fulfills the course requirements for reproducibility, original contribution, and a live demonstrable system.