Project PIV 2025

Page • Tag • 1 backlink

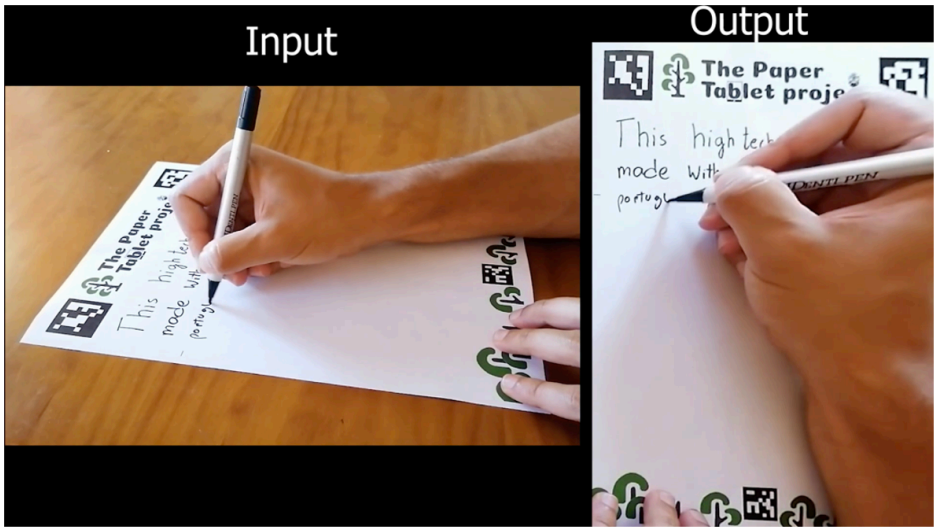
The Paper Tablet Project

Motivation

A moving camera observes a person filling out or manipulating a paper form placed on a table. As the camera or the paper moves, the apparent geometry of the document changes due to perspective distortions.

To enable automatic analysis, digitization, or interaction (e.g., handwriting capture), the system must continuously track the document and produce a **perspective-corrected (frontal) view** of the form in each frame.

In the figure below we illustrate part of the intended output of the project.



However we will go one step further and use 3D information to better render images of the document.

A project in 3 steps

The project develops mainly in two parts plus one optional

Part I - The input is a sequence of images and one template (one image). The input sequence may have images showing the template or not. The camera may drift away from the template but your code must provide a transformation from every image to the template.

Part II - We add depth to the RGB image. The sequence will include both the RGB image and a depth image as captured by a depth camera. This way we will be able

to detect and register objects of any shape and scenes not necessarily planar. In short, this part is intended to use 3D data to its largest scope.

As the figure shows, with depth images a 3D point cloud can be computed which is a completely different representation with different properties that you will learn during the course.

Part III - Time allowing, the use the 2D/3D data and any extra information you may propose to use (using ready-made systems) you may be able to build an interesting/useful/sophisticated application. Examples could be a hand tracker to remove it from the output or use hand position and action to create an interface system using a paper form as "buttons".

This description is an early definition of the main tasks, in the coming days further specifications will be given.

Part I – Homography Estimation from RGB Images

Objective: Develop a robust homography estimation pipeline using only RGB data.

Typical Implementation Steps

1. **Feature Detection and Matching**
    - Detect local features using a scale- and rotation-invariant detector such as **SIFT**, **SURF**, or **ORB**.
    - Compute descriptors and establish correspondences between the current frame and a **reference image** of the form.
2. **Outlier Removal with RANSAC**
    - Many feature matches may be incorrect due to occlusions, texture ambiguities, or lighting changes.
    - Use **Random Sample Consensus (RANSAC)** to robustly estimate a homography by repeatedly sampling minimal sets of four

correspondences and selecting the model with the largest inlier consensus.

3. **Homography Estimation via Least Squares**
    - Once inliers are identified, refine the homography parameters using **linear least squares (LSE)** or the **Direct Linear Transform (DLT)** algorithm, followed by normalization.
    - Apply the homography to warp the current frame, generating a **rectified, perspective-corrected image** of the document.
    - Using SGD (or Newton-like) compute transformation as a nonlinear-optimization problem. (very optional)

Part II – Incorporating Depth Information (RGB-D Extension)

Objective

Enhance the system with an **RGB-D camera** that provides a registered RGB image and depth map at each time instant, allowing 3D geometry-based tracking and object detection.

Implementation Steps

1. **3D Point Extraction and Registration**
    - Using camera intrinsics, convert 2D feature correspondences into **3D point clouds**.
    - Estimate the rigid 3D transformation (rotation + translation) between frames using the **Procrustes method** (orthogonal Procrustes analysis) to minimize alignment error between corresponding 3D points.
2. **Robust Estimation with RANSAC**
    - Integrate **RANSAC** to reject outlier 3D correspondences that violate the planar or rigid transformation model.
    - Improves robustness against mismatched features or moving elements (e.g., hands, pen).
3. **Plane and Object Segmentation**
    - Fit a dominant plane to the 3D points (e.g., via RANSAC plane fitting) to identify the document surface.
    - Points that lie significantly above the plane can be labeled as **out-of-plane objects**, allowing **hand or pen detection** and masking.

Expected Outcome

A two-stage system capable of:

- Tracking and rectifying a planar document in real time under camera motion (**Part I**).
- Leveraging depth information to improve registration accuracy and detect non-planar elements (**Part II**).

The final pipeline demonstrates how **robust 2D and 3D estimation techniques**—including **RANSAC**, **LSE**, and **Procrustes analysis**—can be combined for reliable geometric vision tasks.

Deliverables (preliminary)

- **Code implementation** of both RGB-only and RGB-D pipelines.
- **Short report** (4pages max)describing:
    - Algorithmic details (feature choice, RANSAC parameters, numerical methods).
    - Experimental setup and evaluation.
    - Example results showing rectified views and depth-based segmentation.

Suggested Tools

- Python with **OpenCV**, **NumPy**, and **Open3D or Jupyter K3D if you use Jupyter notebooks**
- Optional visualization using **Matplotlib** or **PCL** tools.

Constraints

- The main modules must be implemented "from scratch" that is, you may use only scipy and numpy. Regressions and ransac must be implemented from the ground up. Details to be defined later

DATASETS

you are encouraged to create your own datasets but we will provide some at this address:

https://drive.google.com/drive/folders/1t-0vKIpNe_9e5Z-gyiMhTLwA3QxTz8-L?usp=drive_link