image-stitching

The code in this repository demonstrates stitching of images captured by cameras observing a **planar target**. We analytically derive homographies by assuming the camera pose wrt. to the target are known.

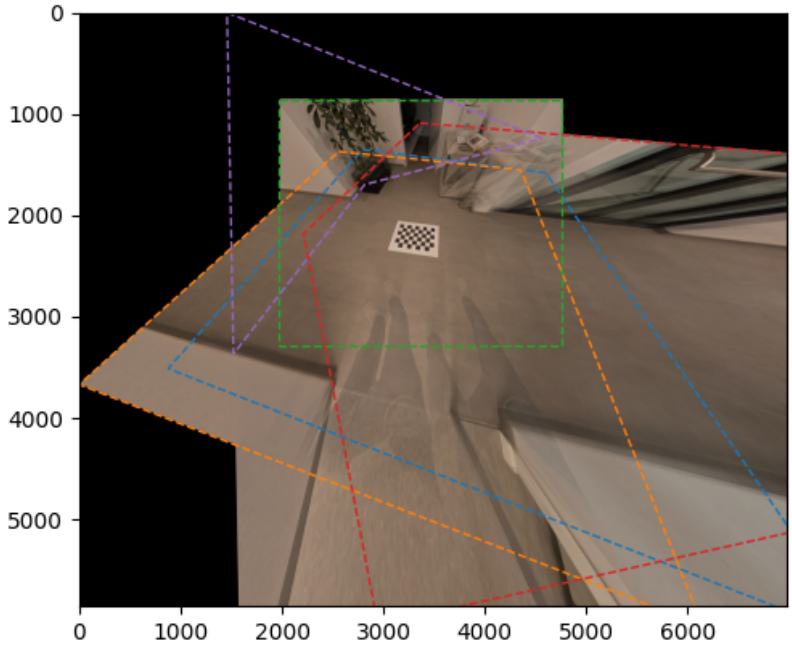


Image shows 4 views stitched in the green reference camera frame. Ghosting artefacts caused by blending moving objects and objects that violate the in-plane assumption.

Introduction

Image stitching is the process of fusioning multiple overlapping images to produce a composite image. Depending on use case, the resulting image is formed on a planar, cylindrical or spherical canvas. The focus of this project is the planar mode, also called scan mode, which requires the target to be sufficiently flat (i.e images captured by a sattelite). In this scenario the cameras can be translated and rotated arbitrarily. The latter two are constrained to cameras sharing a common origin but rotated arbitrarily. The resulting image is usually viewed interactively to avoid severe distortion.

Assumptions

We assume to be given

1. N images of a planar target residing in plane π located T_π^w wrt. to a world frame w,
2. non-linearities are removed from each image I_n,
3. intrinsic camera matrices K_n and view matrices T_w^n

Outputs

The output of our algorithm is a composite image I_c and associated intrinsic matrix K_c. Since all objects, cameras and the target plane, are related through homographies, the image I_c can be formed wrt. to any of the object coordinate systems. Here we look at two special cases

1. Stitching is performed wrt. a chosen reference camera r
2. Stitching is performed wrt. to virtual camera whose image plane aligns with π.

The first option stiches the images alongside of I_r in the viewport of camera r. The second option fusions the images in a virtual camera whose image plane corresponds to π. This results in an image having pixel resolution of $R = \frac{px}{m}$ and allows for image based measurements.

Background

Consider a point

$$p_\pi = [u \quad v \quad 1]^T$$

in the plane π. Perspectively we can express p_π in terms of camera i as

$$p_i = K_i D T_w^i T_\pi^w U p_\pi,$$

where equality is defined up to scale. Here

$$U = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

lifts p_π from UV plane coordinates to homogeneous XYZ world space, $T_w^i \in \mathcal{R}^{4x4}$ is the view transform of camera i wrt. world, likewise $T_\pi^w \in \mathcal{R}^{4x4}$ is the location of the plane in world, $K_i \in \mathcal{R}^{3x3}$ is the intrinsic camera matrix and

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

drops the homogeneous fourth coordinate.

When we map the same point p_π to camera i and j we get two perspective relations

$$p_i = K_i D T_w^i T_\pi^w U p_\pi$$
$$p_j = K_j D T_w^j T_\pi^w U p_\pi.$$

The term $(K_i D T_w^i T_\pi^w U) = L_\pi^i$ represents an invertible 3x3 matrix. This follows from the fact that a) each individual matrix has rank 3 and b) the rank of a matrix product is related to the rank of the individual matrices and c) that a square matrix is invertible if and only if it has full rank. See rank properties for details.

Using the invertability property we rearrange the second equation to

$$(L_\pi^j)^{-1} p_j = L_j^\pi p_j = p_\pi,$$

and substitute for p_π in first equation to get

$$p_i = L_\pi^i L_j^\pi p_j,$$

which relates a pixel in camera j to camera i. We define

$$H_j^i = L_\pi^i L_j^\pi \in \mathcal{R}^{3x3},$$

to be the homography between camera j and i.

Algorithm

The (simplified) stitching algorithm in this repository works as follows.

1. Choose reference camera intrinsics K_r and view matrix T_w^r.
2. Compute the homography H_i^r for each camera i.
3. Find the extent of the output image I_c by computing the bounding box of image corners transformed by H_i^r.
4. Compute the translation matrix S necessary to avoid negative pixel coordinates and let $\hat{H}_i^r = SH_i^r$ be the final homographies.
5. Warp image I_i using $\hat{H}_i^r$ giving $\hat{I}_i$.
6. Form the output image I_c from all $\hat{I}_i$ via blending of overlapping pixels.
7. Return I_c and $K_c = SK_r$.

Stitching in physical camera viewpoint

In order to stitch in camera k let $T_w^r := T_w^k$ and $K_r := K_k$ and run the above algorithm.

Stitching in target plane π

When stitching in the target plane, we construct a virtual camera whose image plane aligns with plane π. In particular we need to construct a suitable view matrix T_w^r and intrinsic matrix K_r.

Let $T_w^r = (T_\pi^w T_r^\pi)^{-1}$ with

$$T_r^\pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

being the virtual camera frame with respect to plane π. Here we have chosen the image axes x, y aligned to u, v. The camera is offsetted 1 units in negative z, so that the cameras image plane aligns with π. Note, $z = -1$ and $z = 1$ are both possible, but one will give a horizontally flipped image. Which one to choose depends on how the coordinate frame π is defined. The flipped image occurs because of a 'see-through' effect.

K_r becomes a pure scaling matrix to account or metric to pixel conversion

$$K_r = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where $R = \frac{px}{m}$ is a user defined pixel per meter ratio. Note, here we use meters as metric units, but in fact it should be measured in the same units as your extrinsic matrices.

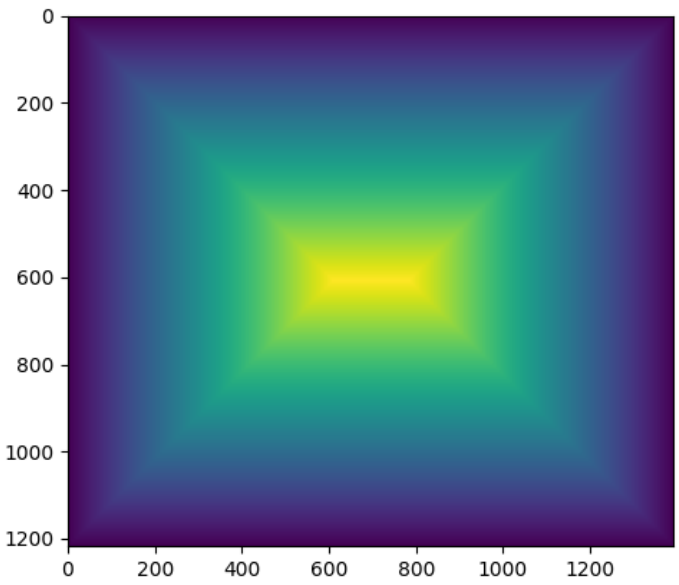
Blending

This repository implements a simple blending strategy that gives central pixels more weight.

This strategy is justified for two reasons:

1. When T_π^i is computed from calibration patterns the camera position is typically chosen such that the calibration pattern appears central in the camera image. The pattern is often of finite (small) size, leading to registration errors that amplify towards the image borders.
2. The observed brightness of real lenses decreases towards the edges of an image decreases due to an effect called lens vignetting.

The dense pixel weights W_i are computed by a simple distance transform that computes the distance for each pixel to the nearest border as shown below (yello is



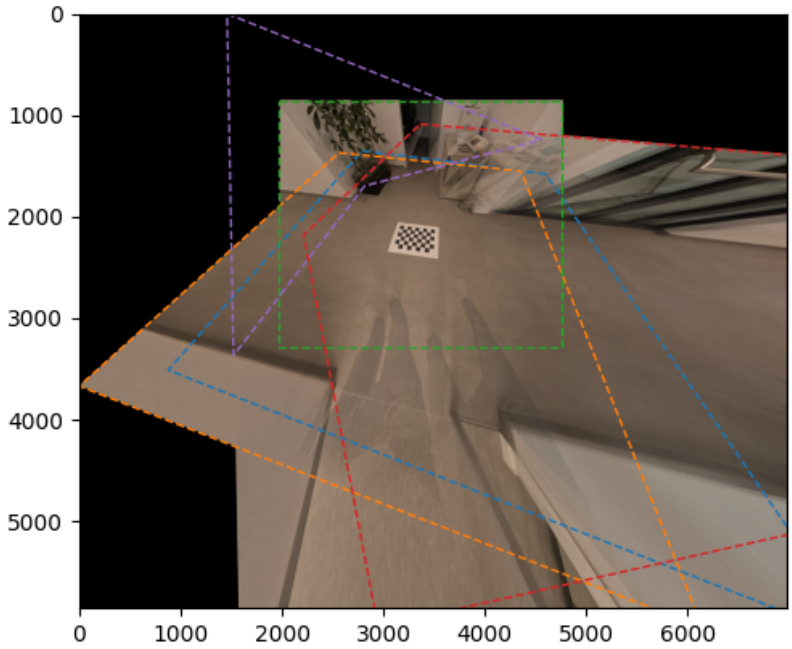
large weight)

We then warp the W_i alongside with I_i and then compute I_c as the weighted average.

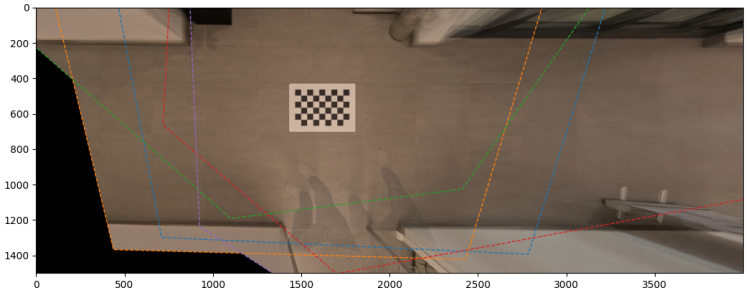
Usage

The code provided is for demonstration purposes only. It is limited to a scenario in which a moving fisheye camera observes a ground floor. The extrinsics are computed from knowing the fisheye intrinsics/distortions and the pattern configuration.

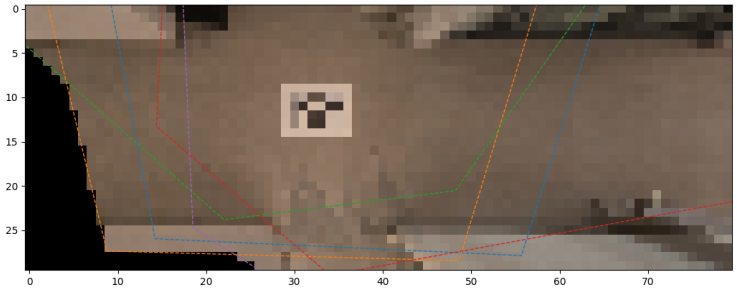
```
# Stitch in camera 3 view (index starting at zero)
python stitch.py -r 2
```




```
# Stitch in plane pi using px/m of 500
python stitch.py -r -1 -px-per-m 500
```




```
# Stitch in plane pi using px/m of 10
python stitch.py -r -1 -px-per-m 10
```



Limitations

Boundary detection

To detect the boundaries of the composite image I_c the bounding box of the transformed image corners of each source view is computed. When source and reference planes associated with the homography H_i^r are angled in a specific ways, corners may be reflected in r. This causes the the transformed view boundaries to become non-convex and bounding box detection might fail.

Shared intrinsics

The current code assumes a single moving camera with shared intrinsics.

Ghosting

Moving and out-of plane objects cause ghosting artefacts due to the blending approach taken.