UNIVERSITÀ DEGLI STUDI DI PADOVA

Laboratory 5 – Keypoints, Descriptors and Matching

Alberto Pretto

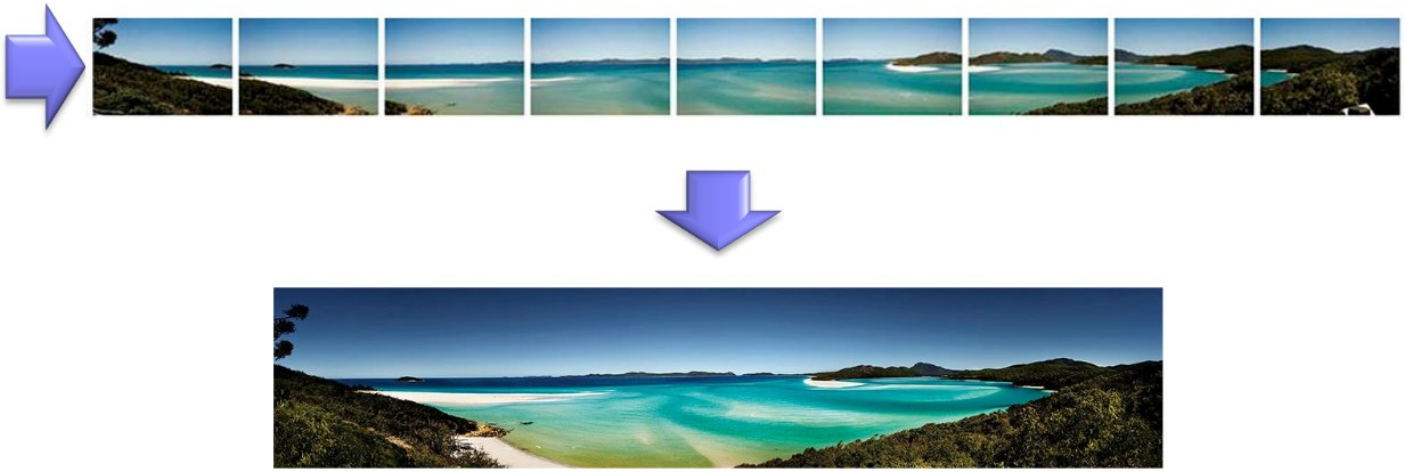
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

INTELLIGENT AUTONOMOUS SYSTEMS LAB

Lab Recap

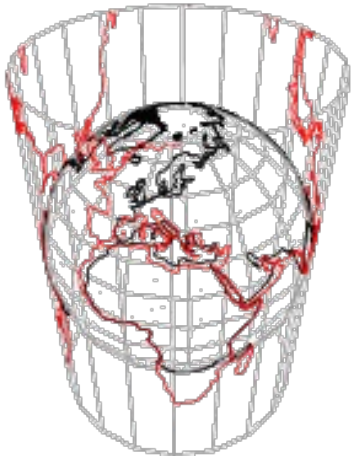
- Panoramic images and cylindrical projection
- Learn how to extract and describe features
- Match features in a robust way
- Stitch images together

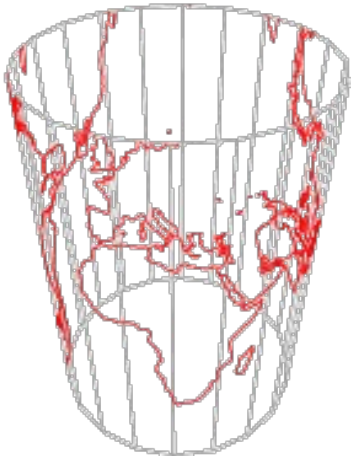
Panoramic Images

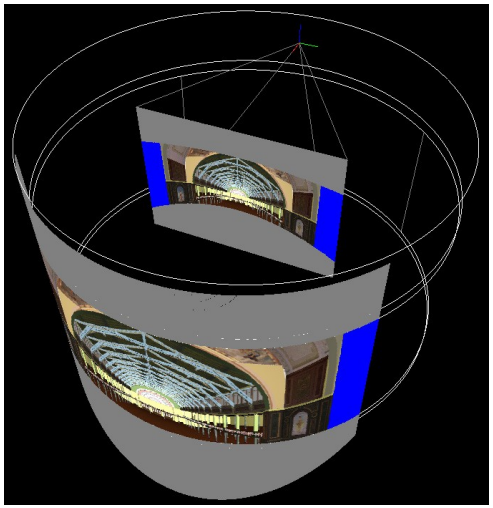


- Pictures covering a (up to) 360° field of view in the horizontal direction
- Panoramic images can be built from a set of pictures taken with a rotating camera from a single viewpoint

Cylindrical Projection





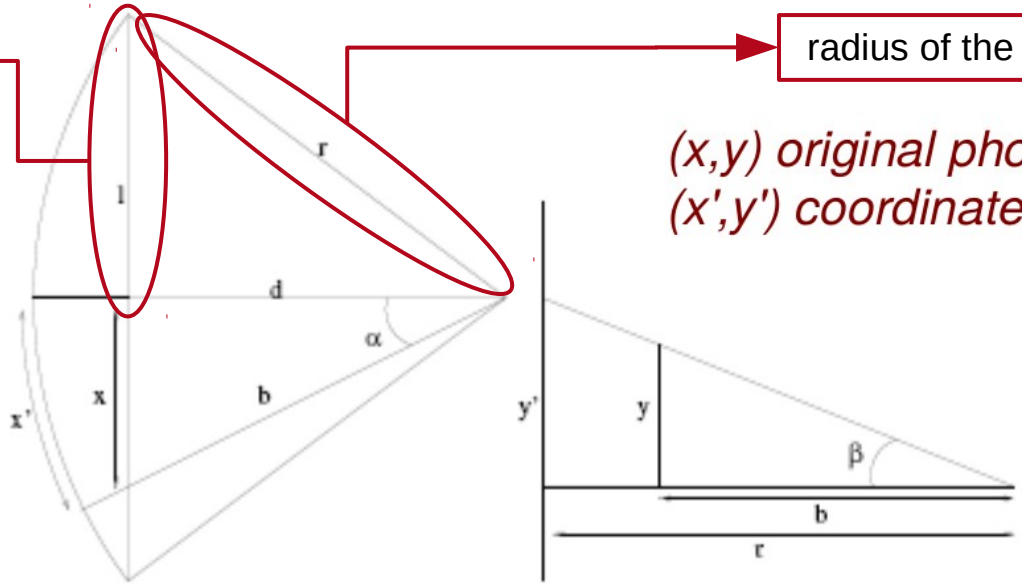


- Images are projected on a cylinder
- After the cylindrical mapping the transformation between the various pictures becomes a simple translation
- See the file ‘cylindrical_projection.pdf’ for the theory and equations of the projection

Cylindrical Projection

image_width / 2

radius of the cylinder



(x,y) original photo coordinates
(x',y') coordinates on the cylinder

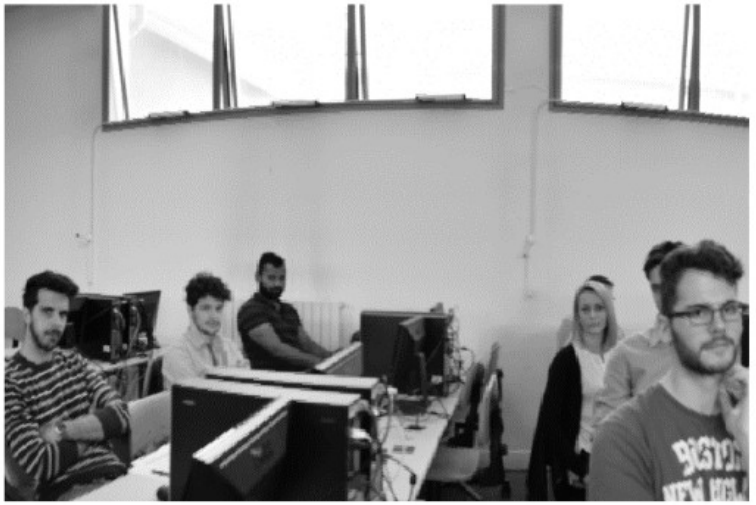
$$x = d\ tan(\alpha) = d\ tan(\frac{x'}{r})$$

$$y = y' \frac{d}{r} \frac{1}{cos(\frac{x'}{r})}$$

see the PDF on elearning for details

Just use the provided `PanoramicUtils::cylindricalProj()` function!

Example (Projection)



Project overview

- Project the images on the cylinder
- Extract and describe SIFT (or ORB) features
- Match features
- Estimate the translation between couples of adjacent images starting from the SIFT matches using a robust estimator (RANSAC)
- Build the panoramic image
- Visualize it

Load and project images

- Load into a `std::vector< cv::string >` the filenames of the input images. You may use the OpenCV function `cv::utils::fs::glob()` with pattern a filter pattern based on '*'/'?' symbols (e.g., img*.bmp).
- Load and project all images by using the provided `PanoramicUtils::cylindricalProj()` function.

  → It just require an angle, use for instance ~30 degrees
- Store projected images into a `std::vector< cv::Mat >`

Feateure detection and matching

- Compute keypoints and descriptors of each projected image (you can use either SIFT or ORB features, by creating the corresponding object with `SIFT::create()` or `ORB::create()`, respectively).
- Compute the matches between consecutive projected images as described in the lab5.pdf document.

Feateure detection and matching

- ```
  static cv::Ptr<cv::SIFT> cv::SIFT::create(
      int    nfeatures = 0,
      int    nOctaveLayers = 3,
      double    contrastThreshold = 0.04,
      double    edgeThreshold = 10,
      double    sigma = 1.6 )
  ```

- ```
  virtual void cv::Feature2D::detectAndCompute(
      InputArray    image,
      InputArray    mask,
      std::vector< KeyPoint > &  keypoints,
      OutputArray   descriptors,
      bool   useProvidedKeypoints = false )
  ```

Feateure detection and matching


- ```
  static Ptr<BFMatcher>cv::BFMatcher::create(
    int normType = NORM_L2,
    bool crossCheck = false)
  ```

- ```
  void   match(
    InputArray queryDescriptors,
    InputArray trainDescriptors,
    std::vector< DMatch > &matches,
    InputArray mask=noArray()) const
  ```

Example (Matching)

Compute the image displacements

- For each pair of consecutive images, estimate the x, y translation in pixels. This can be done by calculating the average translation dx, dy between the matched keypoints.
- To be robust against outliers, don't use all the matches: call the `findHomography()` function, with `cv::RANSAC` as third parameter.
- `findHomography()` will provide a rigid body transformation between the two images but also a mask that highlights the inlier points actually used to estimate the transformation. **To compute the average translation, just use the points marked as inlier**.

Compose the panoramic image

- Prepare a large output image in which to draw the global landscape. For example, to compute the width of such image, consider the projected images widths, and the translations along x.
- Draw each projected image into a submat of the output image, considering the computed translations. To select a submat of an image, you may use the operator `cv::Mat operator() (cv::Range rowRange, cv::Range colRange)`.
- To (try to) improve the final result, you could equalize the projected images with the function `cv::equalizeHist()` before copying them to the output image.

Examples (Panoramic Image)

c++ - OpenCV - SIFT

c++ - OpenCV - ORB

UNIVERSITÀ DEGLI STUDI DI PADOVA

Laboratory 5 – Keypoints, Descriptors and Matching

Alberto Pretto

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

INTELLIGENT AUTONOMOUS SYSTEMS LAB