Circular Object Detection

with the

Vector Gradient Intersection Transform

Henning Lorch, Pascal Ballester

European Southern Observatory, Karl-Schwarzschild-Str. 2, D – 85748 Garching bei München

Introduction

Goal and History

A new transform for the detection of circular objects, the Vector Gradient Intersection Transform (VGIT), was presented in Sep. 2007 at the ADASS conference [1]. It is dedicated to finding circular objects in images, similarly to the well-known multi-stage Hough Transform (HT) [2,3,6]. This poster presents a whole detection process, and provides basic comparisons of both transforms.

Application in Astronomy

In astronomical data processing, circular object characterisation is required on the instrument and control system level to locate and describe for instance projections of pinhole masks, or endings of optical fibres. An automatic detection procedure must cope with major disturbances, like multiple objects in the image, noise and detector defects. Celestial object detection is also possible, although mostly a more advanced task.

Overview

We present below an end-to-end detection process for circular objects in images, using the Vector Gradient Intersection Transform (VGIT). The VGIT combines the locations of intersections of gradient vectors into peak distributions in the centres of circles, which is shown to the right.

Details for an implementation example are provided for each detection step. These steps are namely input preparation, transformation, object / peak detection, second-stage parameter determination, and parameter fitting / optimisation.

References and Download

Please find references and a download link below.

Hough Transform

In terms of computational complexity, the most efficient method previously known for finding circles is the dedicated multi-stage Hough Transform. It takes a gradient field of the original image frame as input, and draws a line for every gradient pixel into an accumulator frame. These intersect at the centre of a circle and sum up to a peak. The radius is found in a second stage. The drawback is that other image content will introduce disturbances, which might obscure the peak completely due to the addition of all lines.

Multi-Stage VS. Single-Stage Transforms

An object-detection image transform normally incorporates a basic model of the shape of the searched objects. The following 2 approaches exist::

- Single-Stage Transform: such a transform contains a model featuring the whole parameter set of a wanted object. One example is the Radon transform [4]. It creates an output hypercube with as many dimensions as object parameters, in which a hyper-peak indicates a fully described object. This approach is straightforward, and thus has a high computational complexity.
- Multi-Stage Transform: only a subset of the model parameters is searched in the first stage, for instance to determine an object's existence and location. From the identified parameters as a starting point, the remaining parameters are now searched in a second stage.

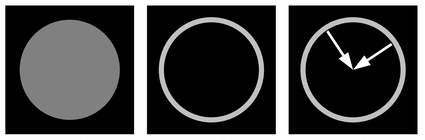
For the example of circular object detection with the basic parameters (x,y,r) (neglecting others like edge width etc.), the single-stage approach using the Radon transform loops over N pixels and $\sqrt{N}$ possible radius values while integrating along the perimeter of length $\simeq\sqrt{N}$. This results in a complexity of $O(N^{\frac{3}{2}})$.

Vector Gradient Intersection Transform

Principle

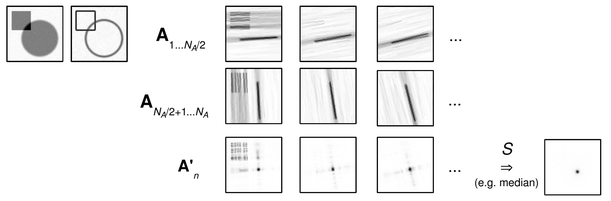
The VGIT is based on the idea that all gradient vectors from a circular object intersect in its centre (like the HT). The prepared input is a gradient field containing information about edges in the input and their orientations.

The locations of the intersections of pairs of gradient vectors are accumulated. This differs from the HT, which accumulates the gradient vector lines, but not the intersection locations themselves.



Circular Object | Gradient Map | Gradient Vectors

To avoid the time-consuming pairing of gradients (complexity N^2), lines are drawn for all gradient pixels into different accumulator frames,

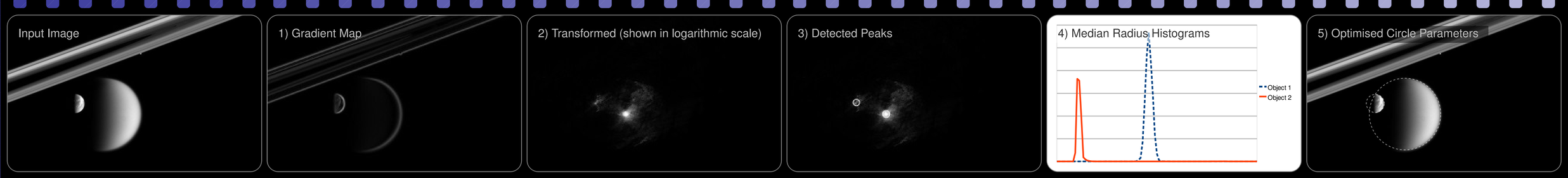


while similar gradient orientations are grouped. Conjunctions of perpendicular groups can then be computed. The algorithm includes the following steps:

- The arc range between 0 and π is divided into N_A sub-ranges,
- N_A accumulator frames $\mathbf{A}_n$ are created,
- for every gradient $\mathbf{G}$, the corresponding accumulator frame is selected, and a line is drawn into this frame, starting at the pixel position of $\mathbf{G}$, in the orientation $\angle\mathbf{G}$, and weighted by the magnitude $w(|\mathbf{G}|)$,
- the conjunctions of perpendicular accumulator frames are performed pixel-wise (here multiplied), this way intersections remain:
$$\mathbf{A'}_n(x,y) \Leftarrow \mathbf{A}_n(x,y) \cdot \mathbf{A}_{n+N_A/2}(x,y),$$
- The resulting accumulator frames (which now contain intersections) are combined using a function S, which could be a sum or median operator:
$$\mathbf{R}(x,y) \Leftarrow S\{\mathbf{A'}_1(x,y),...,\mathbf{A'}_{N_A/2}(x,y)\}.$$

Oppositely directed lines will be conjuncted with the same perpendicular line, therefore these oppositely directed lines can be drawn into the same accumulator frame, and only the angle range $0...\pi$ is divided into N_A sub-ranges.

The average length of a line drawn into a quadratic frame is proportional to one edge length and thus to $\sqrt{N}$. Since for every gradient pixel one line is drawn, the computational complexity converges to $O(N\cdot\sqrt{N})$ for large N. This equals the one of the multi-stage HT.



1. Input Preparation

The available real-world data have very often a different format than the required input for the actual transformation, for instance both the VGIT and the HT require gradient maps. Here, gradients can easily be determined by convolution of the input image with Sobel operators. Other input preparation steps might involve rescaling, noise removal and other means. The complexity of the input preparation can therefore range from triviality to systems more complex than the top detection process by involving deeper layered other detection processes.

Implementation:

- Downscale the input by a factor 2 to reduce pixel noise and decrease computation time.
- Compute the gradient map by respectively convolving with horizontal and vertical Sobel operators, and retrieving both the absolute gradients and the angle field using the arc tangent.

2. Transformation

The prepared input space is transformed to generate easily detectable features, for instance peaks. This is the core, where the selection of the transformation algorithm must be dedicated to the expected or wanted object features (here circular shapes).

Implementation:

- Create the N_A accumulator frames,
- for each pixel entry in the gradient map do:
  - Select the accumulator frame corresponding to the gradient's angle,
  - draw an additive cone into the accumulator frame (drawing a cone takes care of angular noise rather than drawing a line),
- multiply perpendicular accumulator frames pixel-wise to gain distribution maps of intersections,
- combine the distribution maps by summing or taking the median pixel values.

Input Parameters:

- Boolean flag specifying whether to draw the cones in both directions from a gradient's location, or only in positive (negative) gradient orientation. Drawing only in one direction enhances the robustness against other objects in the image, but requires knowledge about whether the searched circular object is brighter or darker than the background.
- The width of the cones, which should be adjusted to really occuring angular deviations of the gradients.
- Choice of sum or median operator for the combination of the distribution maps.

3. Object Detection

The objects are found by searching features in the transformed space, here for instance by peak detection. This step must incorporate knowledge 1st about the wanted output features from the transform (2), and 2nd about possible artefacts created by the transform (2) which should not be detected.

Implementation:

- Background subtraction from the transformation output. The background can for instance be created using a lowpass filter.
- Determine the standard deviation of the brightness. Exclude zero-value pixels since there can be uncovered areas.
- Slightly lowpass the frame to decrease artefacts (real peaks remain due to their weight).
- Threshold the frame e.g. to 6-sigma.
- Determine the thresholded regions to find the peaks.
- Search the peaks' maximum locations, estimate the full-width-half-maximum (FWHM) values.
- Roughly refine the peaks' positions by for instance fitting a gaussian in the +/- FWHM region around the peaks. If unsuccessful (e.g. uncertainty too high), use a barycentre computation respectively.

4. Further Parameter Detection

Further object parameters are determined in the input or prepared input space, like the radius of a circle whose centre was found. This is the 2nd stage in the "multi-stage Hough Transform", which is also the principle of the VGIT. Single-stage transforms do not require this step.

Implementation:

- For each found peak (i.e. for each found object) determine the radius:
  - Create a set of histograms, each one corresponding to an orientation range:
    - From the relative angle between the gradient pixel and the peak select the corresponding histogram.
    - For each gradient pixel in the gradient map compute a weighting indicating whether it points towards the peak (using a tolerance angle value similar to the width of the cones in step 2). Allow missing the peak by 1 pixel for quantisation reasons.
    - Further weight by the inverse of the distance (to compensate for different perimeters).
    - And weight by the gradient intensity.
    - Add the weighting to the right histogram.
  - Compute the median histogram of all the angular histograms.
  - Search the peak(s) in the median histogram and fit a gaussian. Iterate by selecting the +/- 3 sigma region respectively and refitting.

This is valid for circular objects featuring a clear edge. Otherwise select other means than a gaussian to characterise the histogram peaks.

5. Object Modelling and Optimisation

The parameters usually need to be optimised by all-parameter modelling of the detected object in the input or in the prepared input space, since the detection steps are normally dedicated to the detection itself (what is a non-linear step) and cannot guarantee accurate numerical results.

Implementation:

- Define a function to return a matching value between circular parameters (x,y,r) and the gradient map. For clear-edge objects this can be the mean along the perimeter of the weightings which indicate whether a gradient pixel points towards the centre (see (4)) multiplied with the respective gradient intensities. For objects with ramp edges this function should compute the mean also over the estimated edge width (gained from the histogram peak evaluation in (4)).
- Maximise the matching value by refining the circular parameters using an appropriate optimisation procedure, for instance the downhill-simplex method [5].
- Scale back the resulting circular parameters according to rescaling that took place in (1).
- For very small radiuses (i.e. peak-like circular objects in the input frame) correct for the broadening of the objects by the rescaling and the slightly blurring Sobel operators in (1), or reject these objects and remodel them by e.g. appropriate fitting in the input image.
- Return the results.

References

1. Lorch H., 2007, *Vector Gradient Intersection Transformation*, ADASS XVII, ASP Conference Series Vol. 394
2. Ballester P., 1994, *Hough transform for robust Regression and automated Detection*, A&A Vol 286, pp. 1011-1018
3. Illingworth J. and Kittler J., 1988, *A survey of the Hough transform*, CVGIP Vol. 44, pp. 87 – 116
4. Helgason S., 1980, *The Radon Transform*, Boston
5. Nelder J.A. and Mead R.A., *A Simplex Method for Function Minimization*, Computer Journal, 1965, 7
6. Hough, 1962

Download

An example implementation (running on Unix, Linux, Mac) is provided for download at:

- ftp.eso.org/pub/general/hlorch/vgit/

Requirements:

- CPL (www.eso.org/cpl)
- cfitsio (www.eso.org/cpl) (Version 2.5.1 for CPL 4.x)

Comparison

VGIT

- ✓ Tolerant against input disturbances by using a median operator
- ✓ Only intersections remain
- ✗ But sensitive to bad input SNR because creating intersections is non-linear and high compression

HT (multi-stage)

- ✓ Tolerant against bad input SNR since the output is just an accumulation
- ✗ But sensitive to input disturbances (e.g. other objects in the image creating artefacts in the output)

Which to Choose

		# Disturbances / Objects	
		low	high
SNR	good	both are fine	VGIT
	bad	HT	try