REAL-TIME THERMAL CUEING SIMULATION: A PHYSICAL-BASED APPROACH

May 17th, 2018 - Stuttgart (Germany)

Romolo Gordini

LEONARDO
AIRBORNE & SPACE SYSTEMS

ITEC
15-17 May 2018
Stuttgart, Germany

Agenda

- General description of the real equipment
- Targeting/laser pod real-time simulation, status
- Thermal cueing real-time simulation
- Video tracker algorithm
  - overview
  - pre-processing
  - customized klt usage
  - targets detection
  - targets tracking
  - post-processing

GENERAL DESCRIPTION OF THE REAL EQUIPMENT

Targeting/laser PODs mounted on military interceptors, optimized for the air superiority role in beyond visual range and close air combat, can operate both in Air-to-Surface and in Air-to-Air modes, allowing the acquisition and tracking of air and ground targets.

Pilots can select the best POD imaging mode, depending on the specific nature of the mission, and activate Thermal Cueing. This significantly increases the combat effectiveness of the aircraft in all weather conditions during the attack of ground and air targets with a large variety of standoff weapons.

GENERAL DESCRIPTION OF THE REAL EQUIPMENT

Once a target has been located via Thermal Cueing, the laser designator can be used to guide a laser guided bomb towards the reflected laser energy.

PODs purpose normally is:

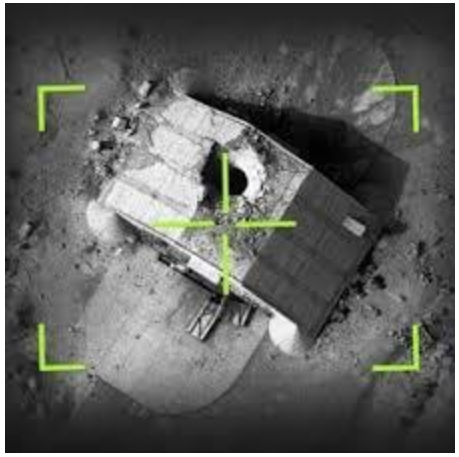
- detection, acquisition, tracking and imaging of possible targets and threats in the air-to-air operation;
- detection of possible ground targets and threats in the flying aid mode in all weather;
- provision of flying and landing aids for the pilot at night, and in adverse weather conditions.

GOAL

The current presentation is focused on Targeting/laser PODs Imaging Modes in IR and its implementation into Full Mission flight simulators for training military pilots.

TARGETING/LASER POD REAL-TIME SIMULATION, STATUS

Within a training system, the Targeting/laser POD real-time simulation (typically at 50 Hz) has got a dual nature.

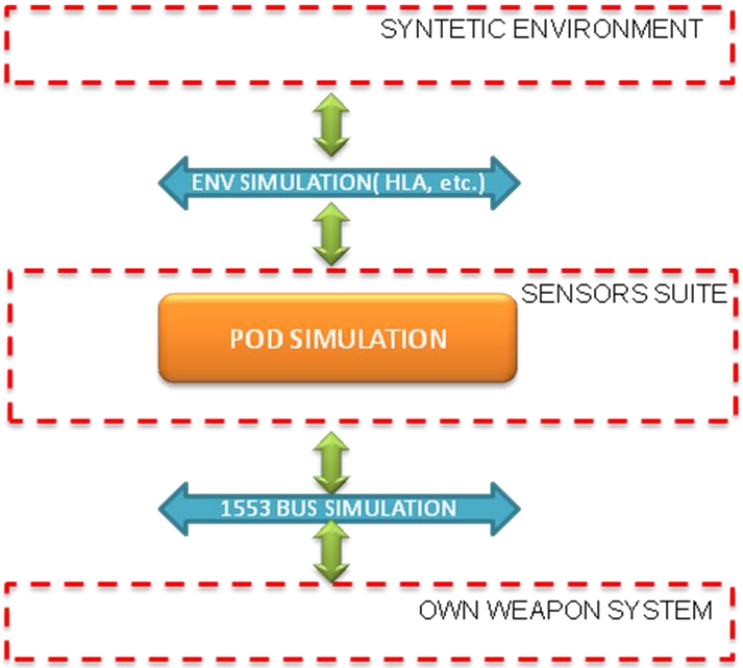


1. It is primarily a sensor;
2. it is also part of the Ownship Weapon System Simulation.

Two architectural solutions are therefore common.

The former (more consolidated and transitional) hosts the POD LRI within the Weapon System Simulation Host, the latter (more innovative and flexible) hosts the POD LRI within the Sensors Host.
In both cases, PODs interface with the simulation and can be summarized as follows.

TARGETING/LASER POD REAL-TIME SIMULATION, STATUS



There are two major logical interfaces to the POD:

- one via the 1553 Bus, either simulated or real
- the other via commercial protocols, typically DIS, HLA, TCP, RMB or other ones with the Synthetic Environment.

The POD System's dual nature dictates for it the following simulation requirements:

1. to be highly correlated with other mission data sets from other weapons sensors, such as radar, IFF, MIDS, etc. and synthesized into a single target information set by the Attack Computer;
2. to be highly correlated with other flight data sets from other mission sensors, such as Visual System, Navigation System, etc. and perceivable by the Pilot through his eyes;
3. to faithfully represent the equipment it is simulating, with all modes, data and performances as design by the equipment OEM.

TARGETING/LASER POD REAL-TIME SIMULATION, STATUS

These needs are frequently conflicting with one another, especially for IR Thermal Cueing.

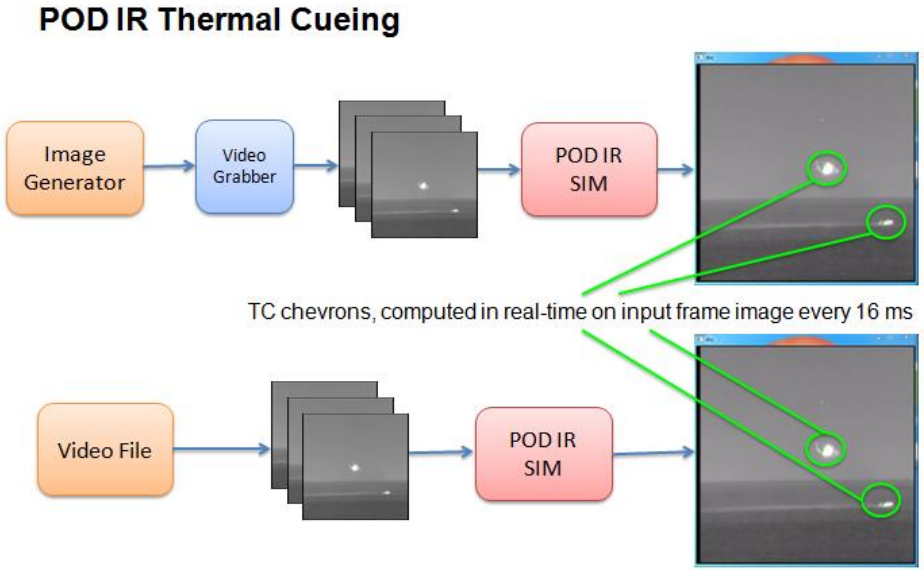
In fact, Thermal Cueing training solutions deployed in simulation – in order to locate hot/white spots on POD images – either use synthetic information coming from the Synthetic Environment (SE), or delegate the Thermal Cueing chevron designations to the Visual System.

1. SE uses synthetic information which is *a-priori* known by the simulation and does not take into account in any respect either the actual POD image nor any physics.
2. On the contrary, the Visual System considers the POD images purely, with a poor, if any, simulation of physical properties of materials in different frequency bands, nor consider in any respect the detection capability of the device itself nor environmental conditions.

These different existing approaches are the only ones capable to guarantee Thermal Cueing basic simulation with the time-critical data I/O operations with the rest of the simulator but they dramatically fail in assuring sensor realism.

THERMAL CUEING REAL-TIME SIMULATION

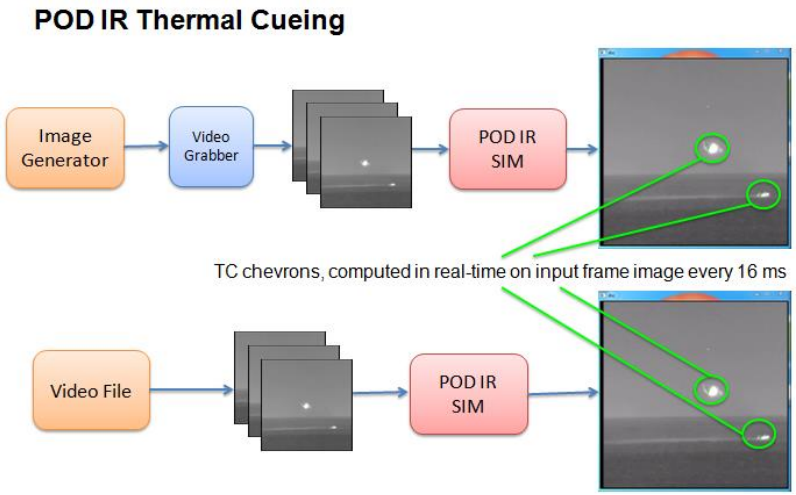
This presentation describes the Leonardo *in-house* TC research ("***MARS FLIR***") using a real-time Thermal Cueing simulation capable of analysing in real-time images coming from sensor cameras (one or more Image Generator channel in any frequency band), which perform the extractions of the most significant targets according to physical rules (hot/white spots), collapse the points passing the filtering phase into a smaller number of significant clusters and track them among consecutive frames of the video flow.



The input image is acquired in real-time via an acquisition board and it is decomposed into a sequence of images which are in turn stored into a queue ready to be processed by the SW.

THERMAL CUEING REAL-TIME SIMULATION

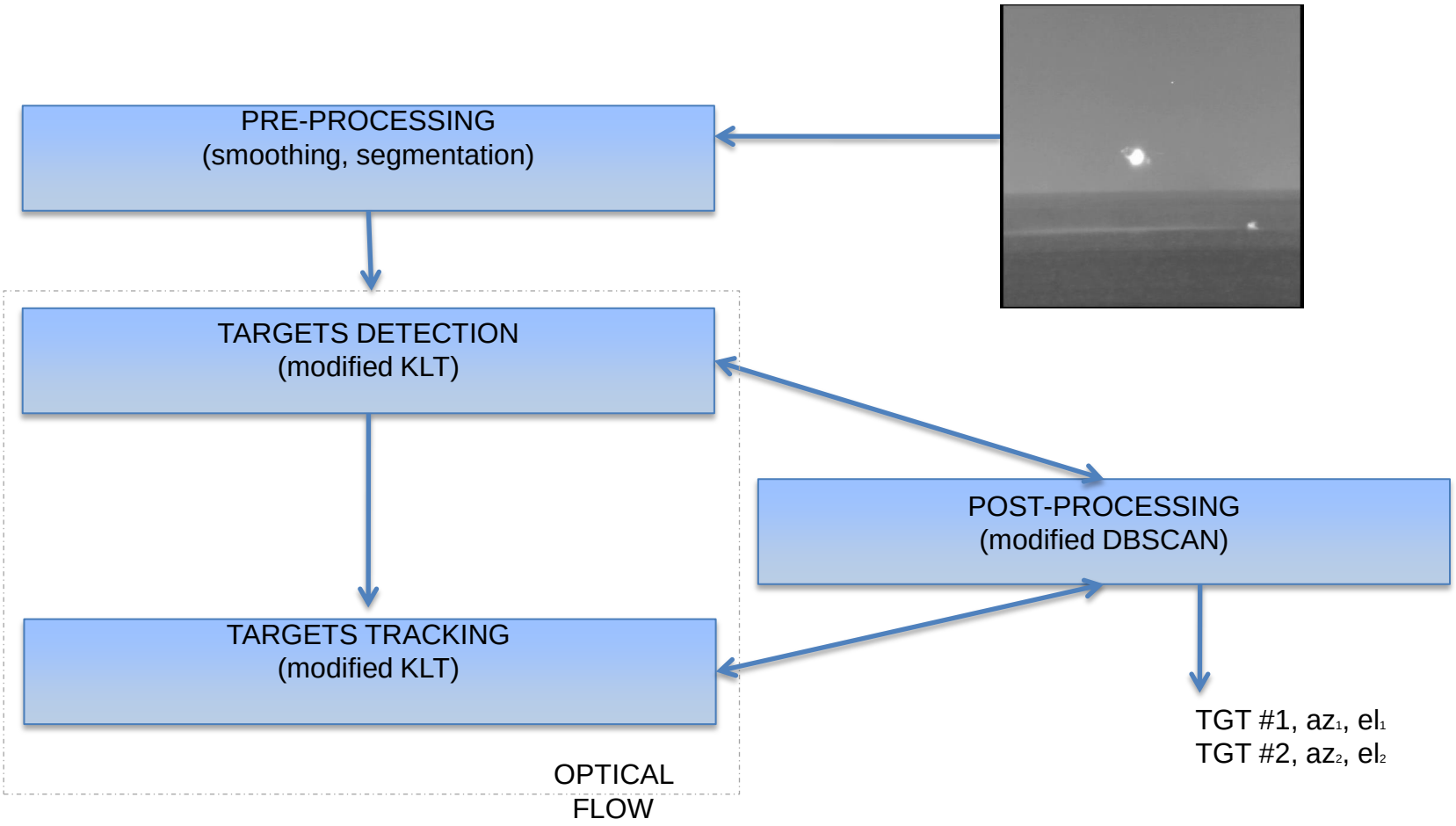
Since the beginning the architectural schema used to implement the solution is original. The simulated sensor, in fact, does not have a proprietary IG resident on the Sensor Suite Server, as for instance in the EF2000 ASTA solution, but it starts from the concept that the unique IG is the OTW one and that the correlation with it is a must have bonus.



Up to now this approach has been generally discarded because it implies acquiring in real-time, via frame grabbers the stream of images and processing them on the fly, with algorithms of image processing. This method is very accurate and precise, but it requires a significant amount of time, and up to now, it was not considered possible to tailor it in order to be used with a processing rate compatible with the one adopted by the simulation.

VIDEO TRACKER ALGORITHM: OVERVIEW

Workflow for targets detection and tracking on an images video stream is the following:



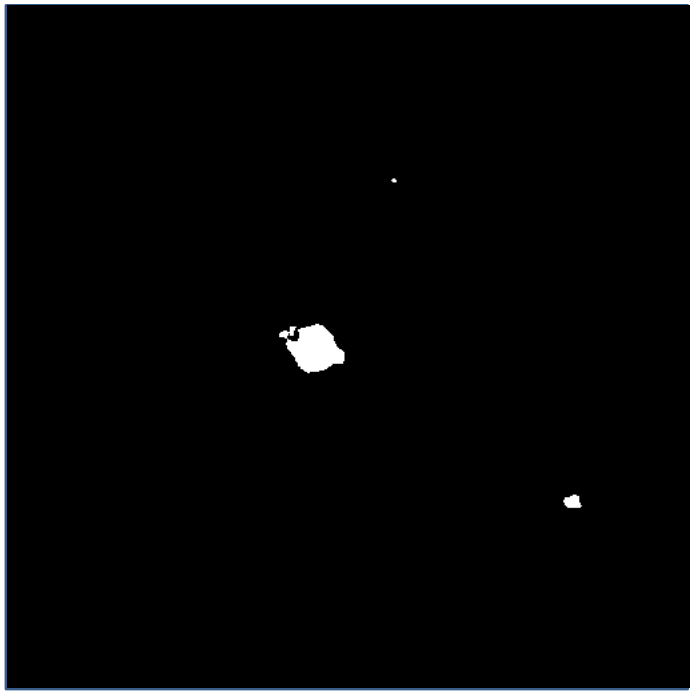
VIDEO TRACKER ALGORITHM: PRE-PROCESSING

The input image is acquired in real-time via an acquisition board, decomposed into a sequence of images, stored into a queue and processed by the SW.

a. Gaussian smoothing (reduce noise)

b. Threshold-based segmentation (speed-up execution)

a)



b)

VIDEO TRACKER ALGORITHM: CUSTOMIZED KLT USAGE

A worldwide known algorithm (KLT) has been used to extract targets from an image. The KLT algorithm has been modified in order to be adapted to strict real-time processing.

2 phases:

TARGET DETECTIONS: on the first frame of a sequence of images it extracts "outstanding points" (corners);

TARGET TRACKING: it estimates on the next frame of the sequence of images locations of targets detected into the previous phase (sequential mode). A Kalman filter has been added to the logic, in order to cope with situations of tracks fading or cycles of lack of detections.

VIDEO TRACKER ALGORITHM: TARGETS DETECTION

The basic concept of this phase is to analyse the entire starting image, looking for a corner, by using a sliding squared window of known dimensions and centring it on each pixel at each iteration.

$$C_{str} = \begin{bmatrix} c_{xx} & c_{xy} \\ c_{xy} & c_{yy} \end{bmatrix}$$

on (x,y) pixels marked as white in mask

VIDEO TRACKER ALGORITHM: TARGETS DETECTION

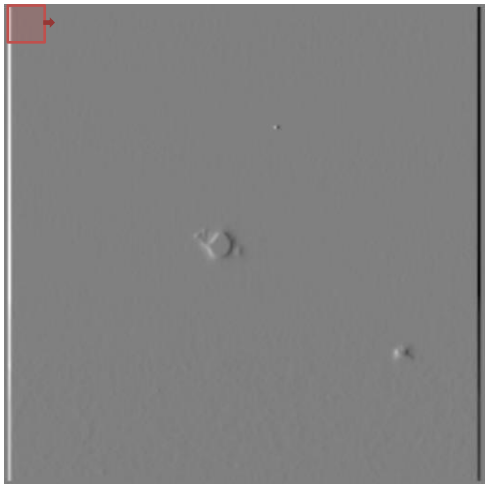
The window identifies the portion of the image to analyse to determine if the pixel on which it is centred is a corner and then a target.

The following step is to analyse the gradient.
Prior of the target selection process, two matrices of gradients, namely "$grad_x$" and "$grad_y$", have to be calculated. The dimensions of these latter matrices are equal to the dimensions of the initial image.

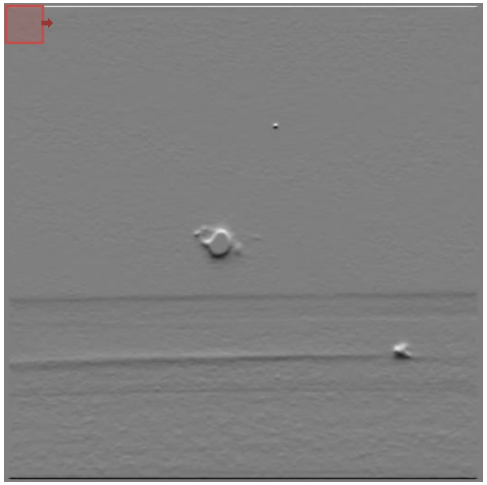
They respectively contain the gradient along the x axis and the gradient along the y axis of the light intensity of the initial image (previously smoothed).

VIDEO TRACKER ALGORITHM: TARGETS DETECTION

x gradient



y gradient



The same "*selection window*" has to be picked up from gradx and grady matrices, centred on the pixel that the algorithm wants to determine whether it is a target or not, and consider the values of these two extracted windows which have to be combined in order to obtain the 2x2 G array, whose elements are: g_{xx}, g_{xy}, $g_{yx} = g_{xy}$, g_{yy}.

VIDEO TRACKER ALGORITHM: TARGETS DETECTION

The matrix G is called "***Structure Matrix***", since its eigenvalues define the presence or absence of edge or corner in the pixel under evaluation. In particular:

- $\mathbf{g_{xx}}$ is the sum of all the squared values of the pixels contained in the window extracted by gradx;
- $\mathbf{g_{yy}}$ is the sum of all the squared values of the pixels contained in the window extracted by grady;
- $\mathbf{g_{xy}}$ is the sum of all the values of the pixels contained in the window extracted by $grad_x$ multiplied by the corresponding values of the pixel matrix extracted from $grad_y$.

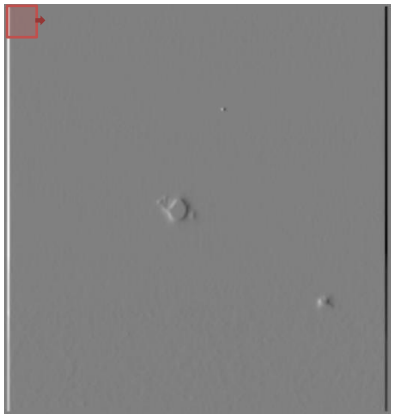
The pixels for which the G matrix has got a non-zero determinant are "***target candidates***".

VIDEO TRACKER ALGORITHM: TARGETS DETECTION

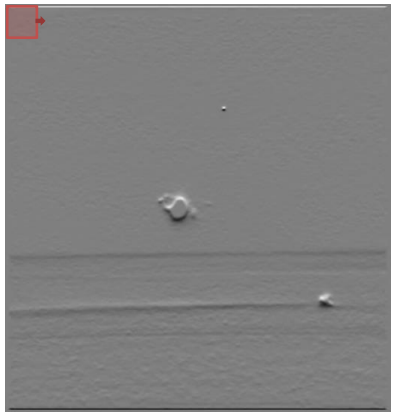
The more the values of elements of the matrix G are smaller, the more probable is the fact that the window under evaluation does not contain variations in light intensity and therefore does not contain corners.

A candidate target becomes effectively a corner point if the smaller eigenvalue of the G matrix exceeds a certain lambda threshold.

x gradient



y gradient



$$C_{str} = \begin{bmatrix} c_{xx} & c_{xy} \\ c_{xy} & c_{yy} \end{bmatrix}$$

$$\lambda_1, \lambda_2 > \lambda_{thr}$$

$$(x, y)$$

VIDEO TRACKER ALGORITHM: TARGETS DETECTION

OUTPUT: **List of candidate corner points**

ordered with eigenvalue descending values.

PARAMETERS: window size, eigenvalue threshold, *minimum distance between two targets.*

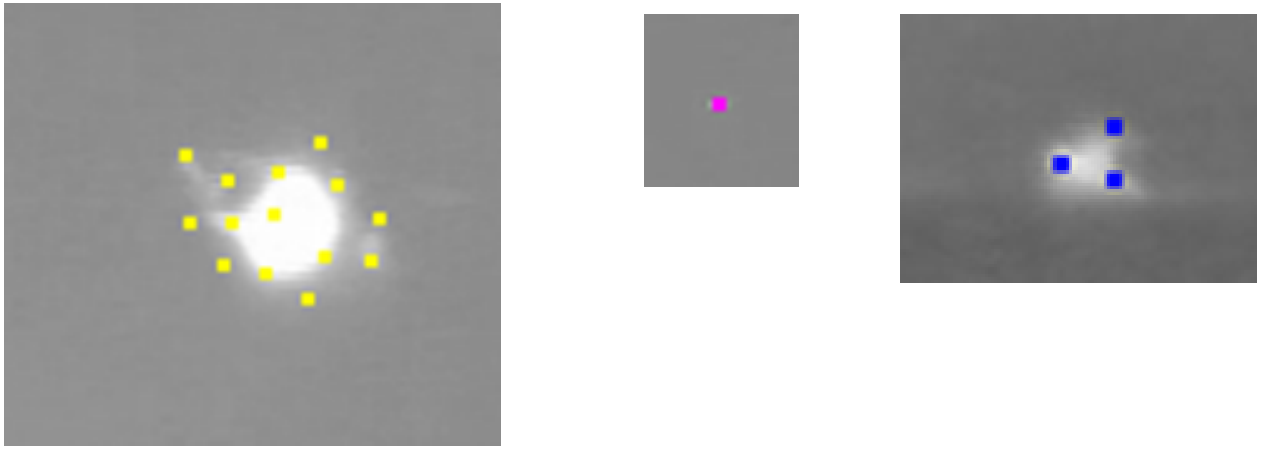
feature	(x,y)=eigenvalue
0	(287,461)= 45
1	(77,461)= 40
2	(463,464)= 37
3	(406,464)= 30
4	(299,459)= 30
5	(446,466)= 29
6	(338,476)= 29
7	(351,423)= 28
8	(316,461)= 28
9	(232,451)= 27
10	(476,431)= 26
11	(268,425)= 26
12	(165,416)= 25
13	(132,457)= 25
14	(309,431)= 25
15	(475,467)= 24
16	(270,477)= 23
17	(189,456)= 23
18	(215,474)= 23
19	(412,410)= 23
20	(46,431)= 22
21	(146,432)= 21
22	(95,457)= 21
23	(452,439)= 20
24	(96,432)= 20
25	(250,418)= 20
26	(436,459)= 19
27	(426,422)= 19
28	(242,464)= 18
29	(169,452)= 17
30	(421,460)= 17
31	(290,476)= 17
32	(82,471)= 17
33	(180,487)= 16
34	(326,468)= 16
35	(379,483)= 16
36	(246,453)= 16
37	(125,485)= 15
38	(276,443)= 15
39	(386,449)= 15
40	(71,420)= 15

points

decimation

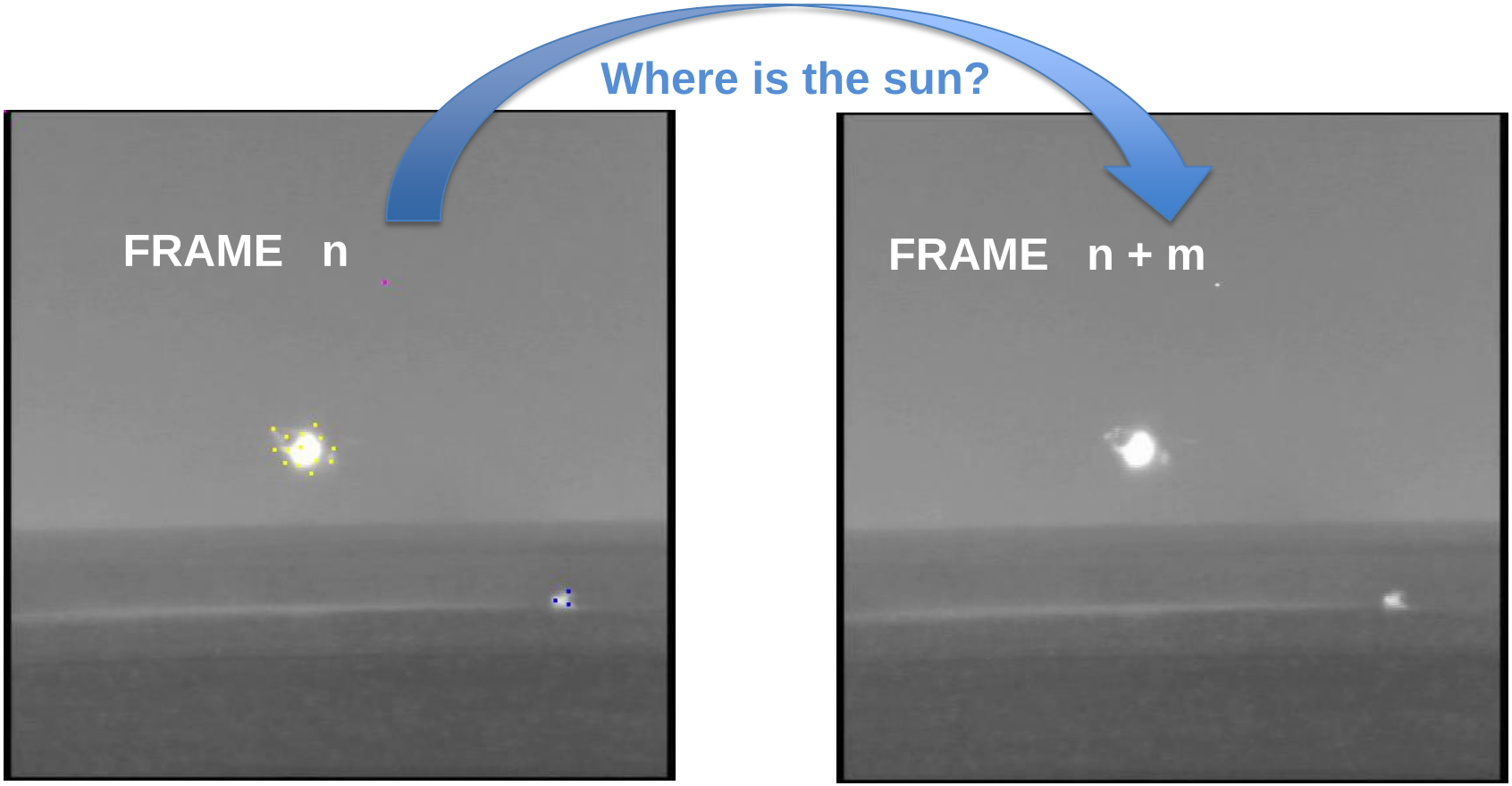
VIDEO TRACKER ALGORITHM: TARGETS DETECTION

Final result of the detection phase is a “cloud” of one or more targets associated to each real-world target.



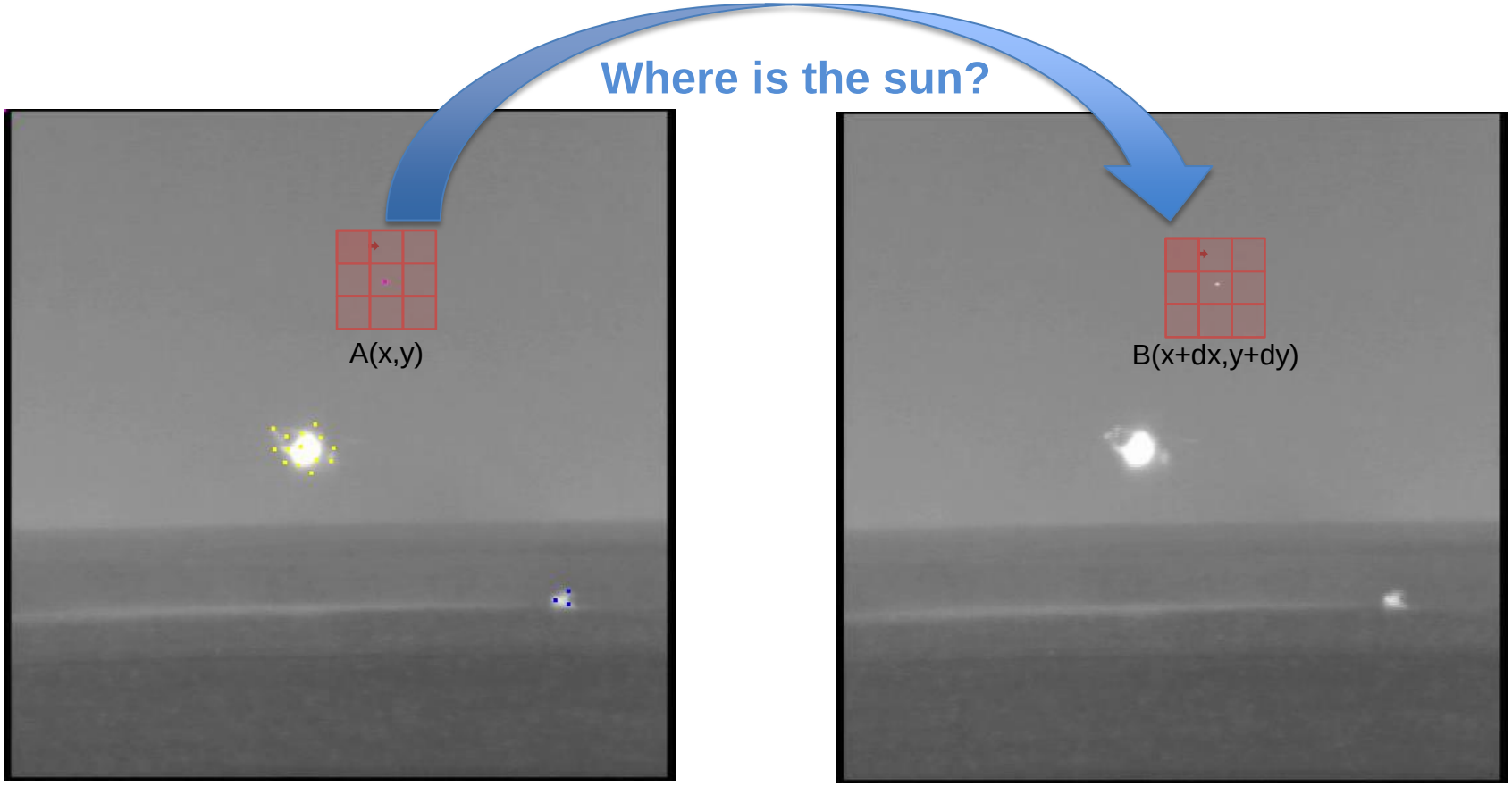
VIDEO TRACKER ALGORITHM: TARGETS TRACKING

The tracking algorithm takes the input list of targets obtained from the previous step and derives the estimated position (x + dx, y + dy) for each target in the next image of the sequence (less time-consuming).



VIDEO TRACKER ALGORITHM: TARGETS TRACKING

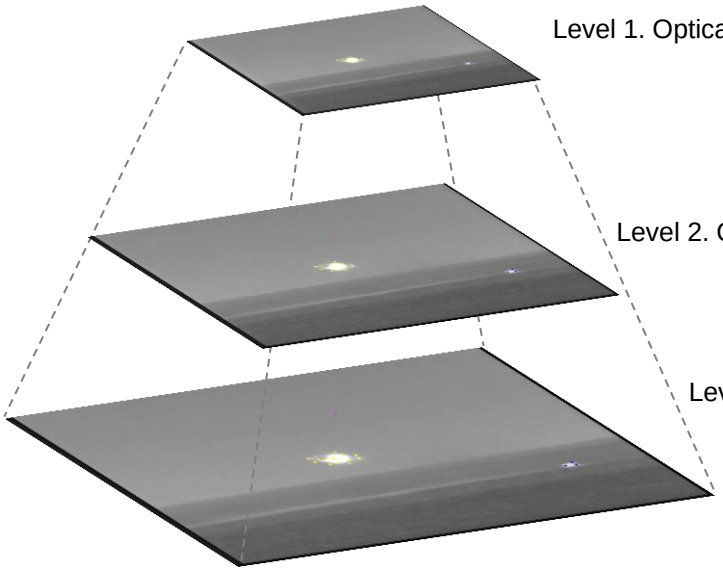
In a subsequent frame, find location (x+dx, y+dy) of target (x,y) selected in the previous frame (less time-consuming).



dx,dy = min(diff(A,B))

VIDEO TRACKER ALGORITHM: TARGETS TRACKING

The computation generates a pyramid of images for each frame of the image stream, in which each layer represents the original image, with a resolution reduced by a factor of 2 compared to the image of the previous level.

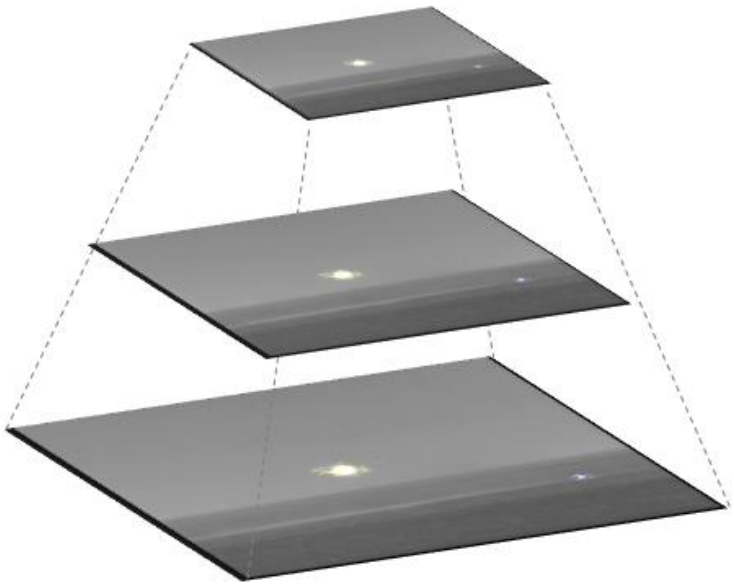


Level 1. Optical flow computation at lowest resolution (subsampling with factor 8, 64x64 pel)

Level 2. Optical flow computation at next higher resolution (subsampling with factor 4, 128x128 pel)

Level 3. Final optical flow computation at original image (512x512 pel)

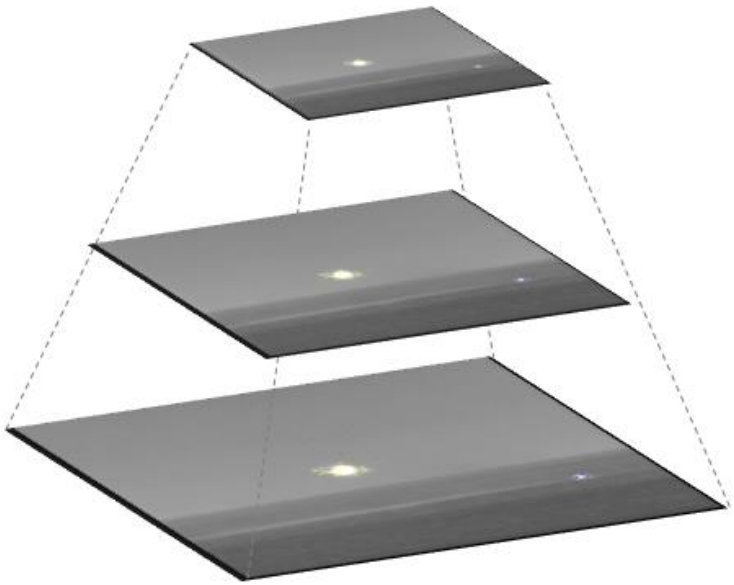
VIDEO TRACKER ALGORITHM: TARGETS TRACKING



In this way the algorithm can follow targets with more than one level of results, starting from the lowest resolution layer of the image.

Increasing the number of levels of the pyramid allows the algorithm to handle cases in which the offset between corresponding targets in successive frames is large, at the cost of an increase in the computational load. The number of levels of a pyramid is calculated when creating the tracking context during the pre-processing phase.

VIDEO TRACKER ALGORITHM: TARGETS TRACKING

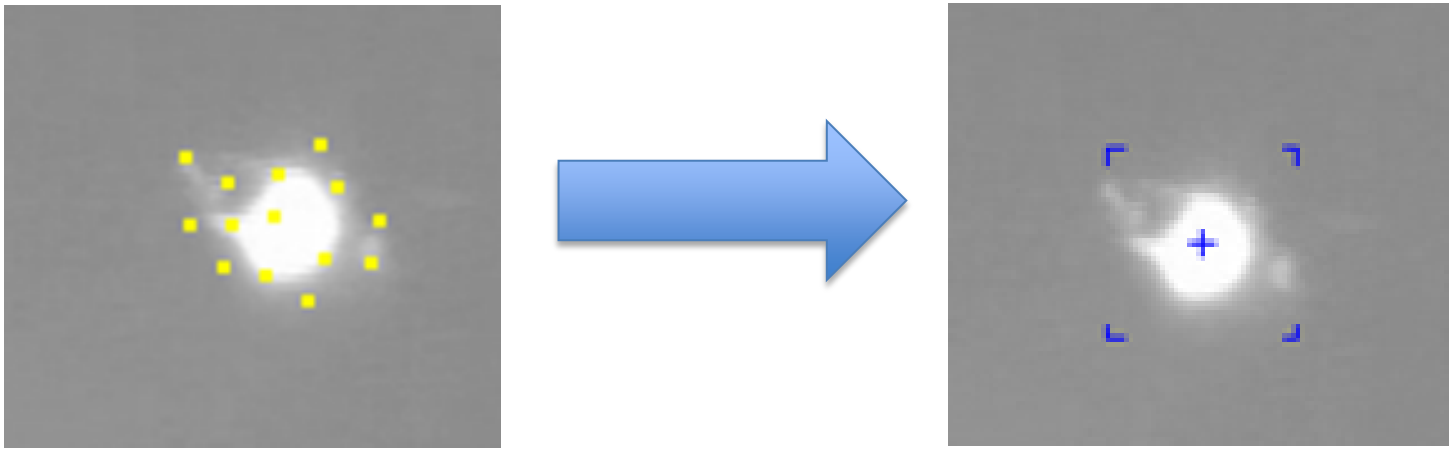


Each level of the pyramid is obtained through sampling the previous level and applying a factor of 2 in width and height. The tracking algorithm starts from tracking targets in the lower resolution level and continues up to convergence. The result obtained on one level is propagated to the next level as an initial guess for the position of the target in the frame. In this way, the tracking is refined in each level more and more, up to the original image. The use of pyramid levels allows the tracker to handle larger movements of the neighborhood size.

VIDEO TRACKER ALGORITHM: POST-PROCESSING

After the detection and tracking steps of the algorithm, for each object with a “thermal outstanding signature” into the image frames targets are extracted and estimated.

The final goal is to have only one target per object for such cases. A post-processing phase on the image is therefore needed.
For this step the **DBSCAN** algorithm has been selected, again, suitably modified in order to sustain hard real-time processing.

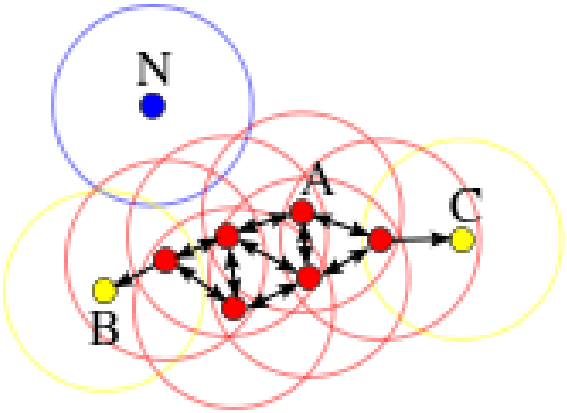


VIDEO TRACKER ALGORITHM: POST-PROCESSING

DBSCAN: *Density-Based Spatial Clustering of Applications with Noise*

\+ *Arbitrary* number of clusters with *arbitrary* shape

\+ Recognize ***noise***



Cluster: set of **mutually** *density-connected* points (**border points** are only *density-connected*).

Points ***p*** and ***q*** are ***density-connected*** if it exists point ***o*** that is *density-reachable* from ***p*** and ***q***.

Two points ***p*** and ***q*** are ***density-reachable*** if it exists a sequence of points $\boldsymbol{p_1, \ldots, p_n}$, with $\boldsymbol{p_1 = p}$ and $\boldsymbol{p_n = q}$, *s.t.* $\boldsymbol{p_{i+1}}$ is *directly reachable* from $\boldsymbol{p_i}$.

Point ***q*** is **directly reachable** from point **p** if:
- *dist*(***p***,***q***) < **ε**, and
- ***p*** has at least **minPts** points in its neighborhood.

VIDEO TRACKER ALGORITHM: POST-PROCESSING

DBSCAN: *Density-Based Spatial Clustering of Applications with Noise*

\+ *Arbitrary* number of clusters with *arbitrary* shape

\+ Recognize ***noise***

THANK **YOU** FOR YOUR ATTENTION

LEONARDO
AIRBORNE & SPACE SYSTEMS