ESA Workshop on Avionics Data, Control and Software Systems

Avionics/GNC Architecture and Sensors Suite for Exploration RT

Experience and views of CGS on optimization of Avionics/GNC/Sensors for Exploration (Planetary Landing)

3 Nov 2010

Carlo Gavazzi Space SpA

Headquarters: Via Gallarate 150 - 20151 Milano (Italy)
Tel: +39.02.380481 - Fax: +39.02.3086458
Website: www.cgspace.it - E-mail: cgs@cgspace.it

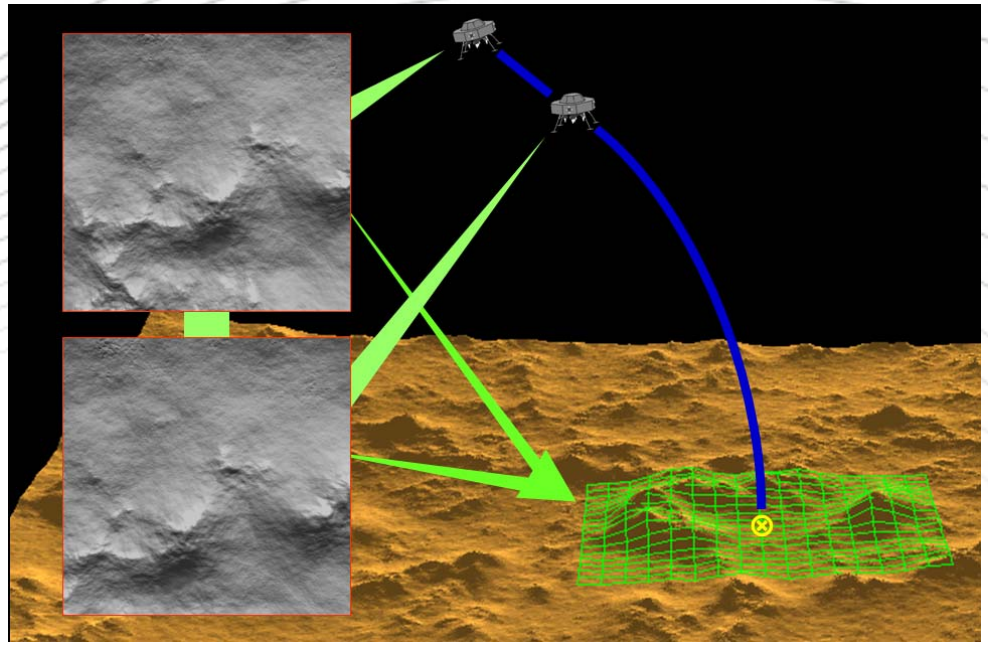
Contents

- Overview of Developments for Exploration GNC at CGS
- Innovative Lander Navigation Concepts
  - Relative Navigation
  - Absolute Navigation
- Innovative Lander Guidance Concepts

Carlo Gavazzi Space SpA

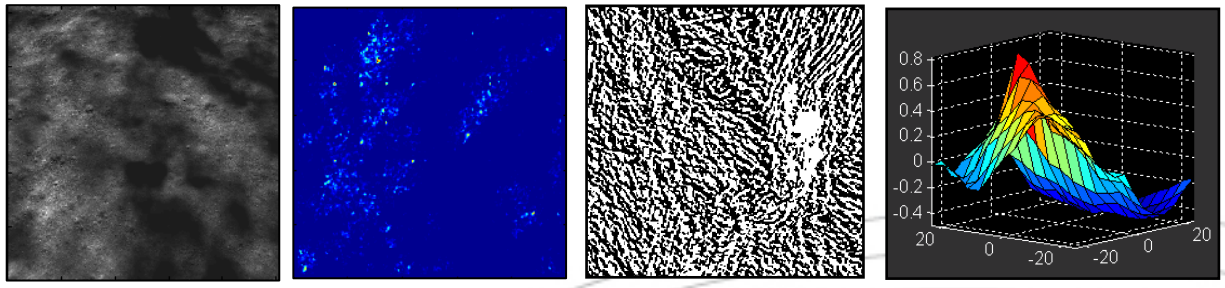
Overview of Developments for Exploration GNC at CGS

- Since 2001 CGS is pursuing research activities in the field of vision-based GNC algorithms for Space Exploration, aimed at identifying, developing and testing innovative and high-performance concepts
- The main considered application is on future planetary landers (“NAVIGATOR” project)
- The developed algorithms were tested using an in-house software simulation system and test bench
- Mars and Moon mission scenarios considered
- Main sensor combination: Camera + IMU
- All key GNC functions developed and tested:
  - Image Processing
  - Navigation
  - Terrain DEM Generation
  - Hazard Map Generation
  - Landing Site Selection
  - Trajectory Generation and Control

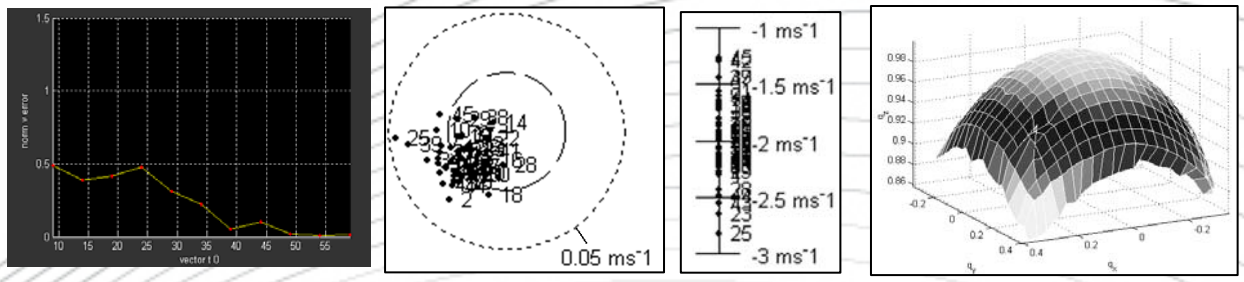


Overview of Developments for Exploration GNC at CGS

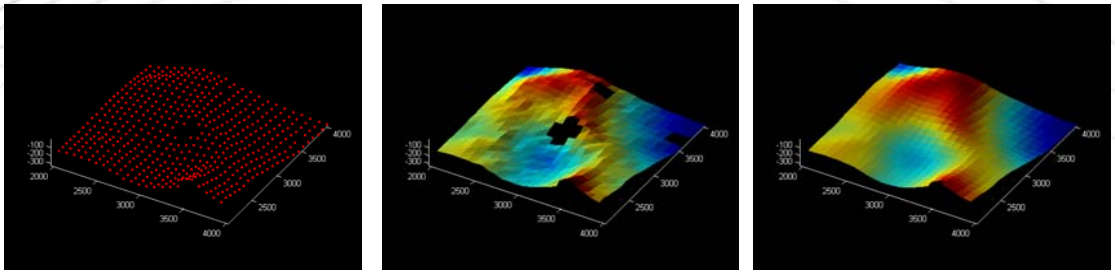
- Image Processing
  - Various techniques for extracting and tracking “feature points” or “landmarks” across a sequence of images



- Navigation
  - Utilizing a Navigation Filter for obtaining real-time vehicle state estimates
  - Both “Relative” and “Absolute” Navigation modes considered

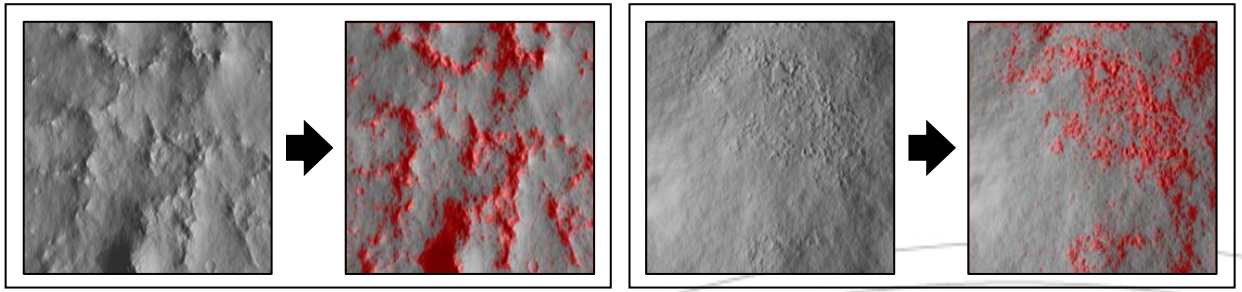


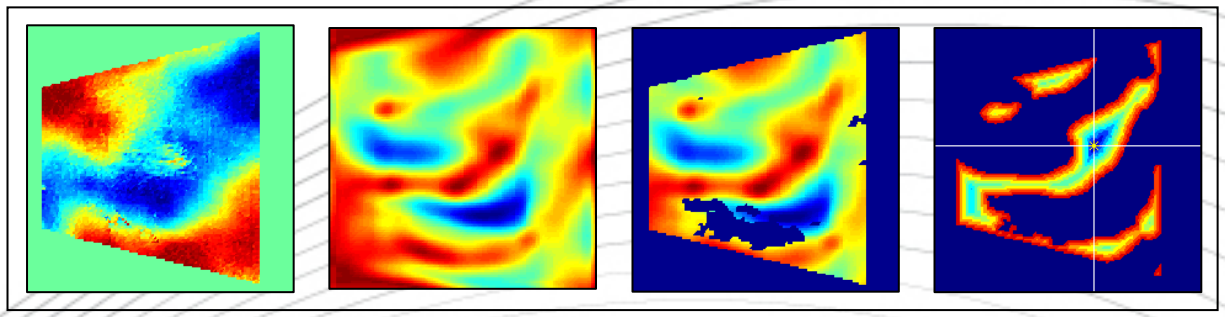
- Terrain DEM Generation
  - Producing and updating an elevation model of the observed terrain during the descent

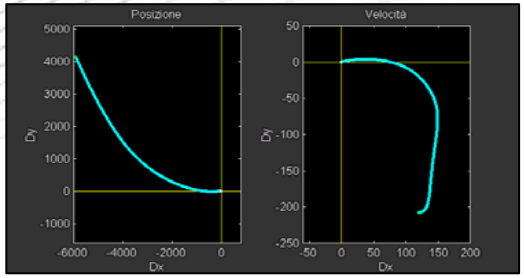


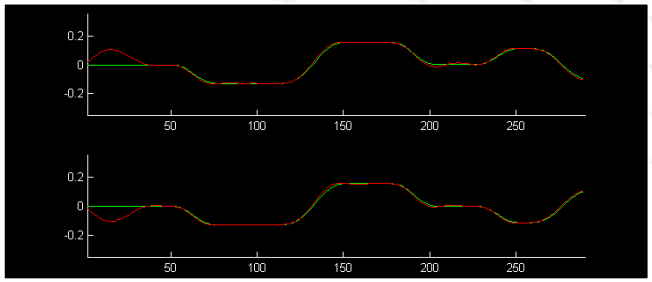
Overview of Developments for Exploration GNC at CGS

- Hazard Map Generation
  - Identifying the position of “hazards” (e.g. boulders) on which the vehicle must not land
- Landing Site Selection
  - Autonomous determination of the “optimal” landing site
  - Taking into account DEM and Hazard Map data, together with fuel consumption
- Trajectory Generation and Control
  - Determination of the trajectory to reach the chosen site
  - Thruster command generation for realizing the trajectory



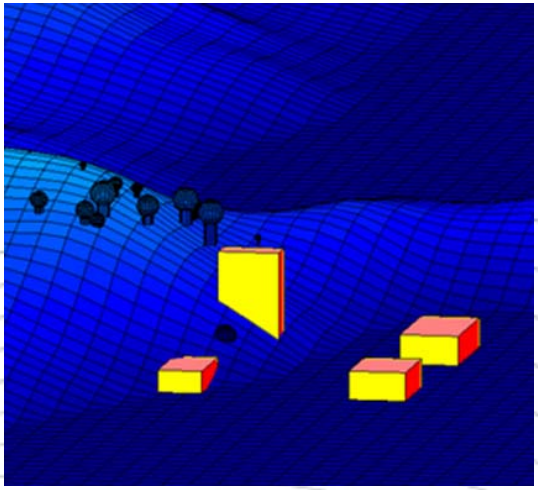


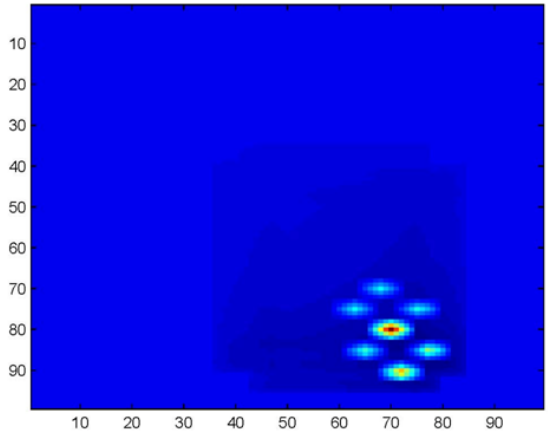


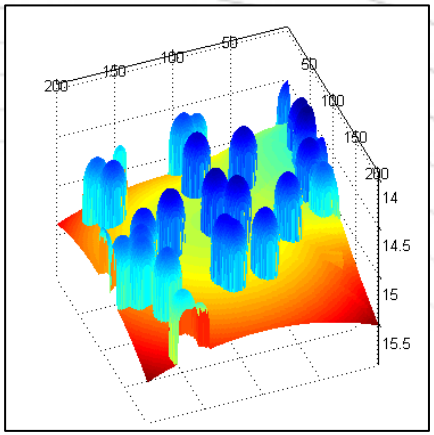


Overview of Developments for Exploration GNC at CGS

- In general, there are many commonalities between lander GNC algorithms, and the ones for “terrestrial” vehicles (e.g. UAVs).
- In the frame of the NEMO project, a multi-purpose GNC SW framework has been established, for both space and terrestrial vehicles, and the key “common” building blocks were developed.
- Main “terrestrial” mission scenarios considered:
  - Forest Fire Detection and Monitoring: here an UAV is used for autonomously detecting and investigating fire zones
  - Manned Landing Support: a LIDAR-based system for aiding the pilot in difficult conditions (high obstacles and low visibility)



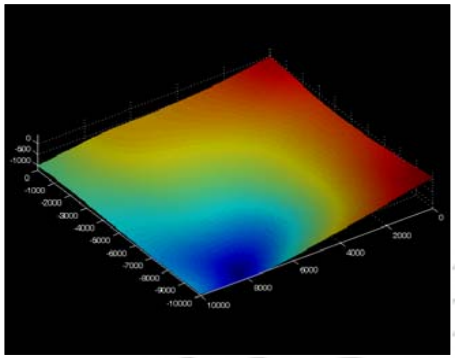


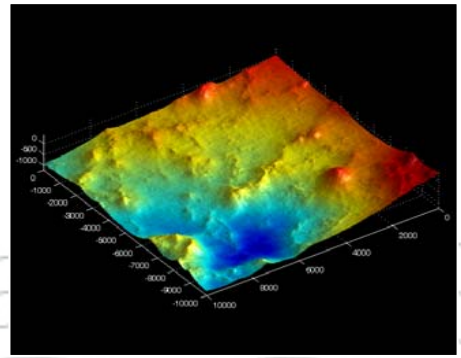


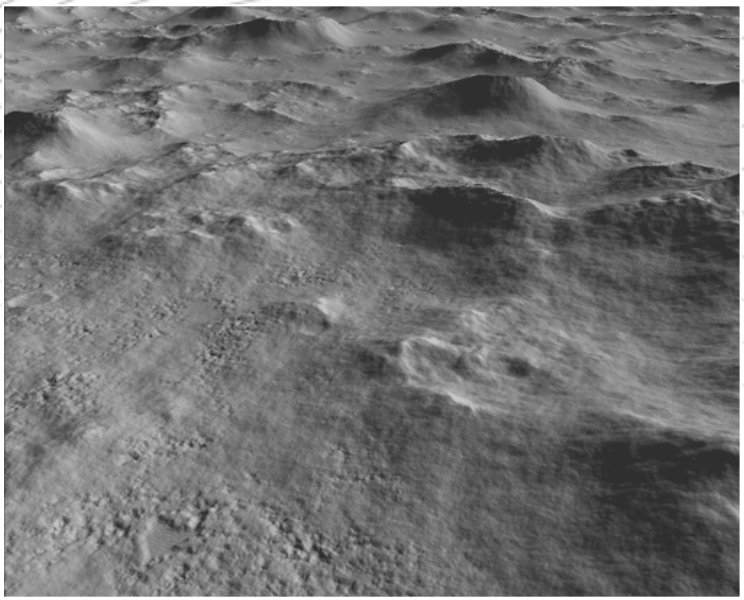
Carlo Gavazzi Space SpA

Overview of Developments for Exploration GNC at CGS

- An utility for generating simulated images of planetary landscapes (Mars and Moon) has been developed for testing GNC Algorithms for Landers and Rovers.
- "Offline" step: Terrain Generation and Preparation
  - Processing pre-existing terrain data (e.g. Mars Global Surveyor MOLA data, or SRTM data)
  - Fractal detail, boulders, dunes and craters
  - Simulating thermal erosion (if necessary) and shadows
- "Online" step: Image generation
  - Dynamic Level of Detail (LOD) control, to ensure image crispness from ~10 km to ~1 m, in-house renderer utilized







Planetary Scene Generator: Sample Images

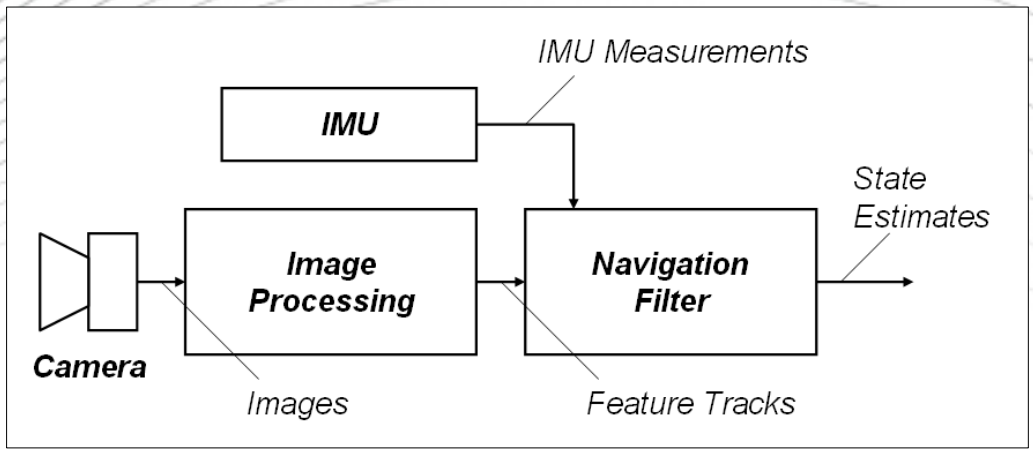
Lander Navigation Development: Objectives

- In the frame of our activities, a “clean sheet” approach has been used for the development of the lander navigation function, called “QuickNav”, with the key targets of reaching:
  - a simple overall architecture, using a limited number of sensors with high TRL
    - → baseline: a single camera and an Inertial Measurement Unit
  - high performance and robustness
    - → high accuracy of the obtained state estimates
    - → high robustness w.r.t. input errors
  - low computing requirements
    - → straightforward implementation on existing space-qualified computing hardware
- Furthermore, high-flexibility solutions were sought, for example in terms of:
  - ability of interfacing with various categories of sensors (e.g. altimeter, scanning LIDAR)
  - minimization of system-level impacts (e.g. no need for “initialization” state estimates)

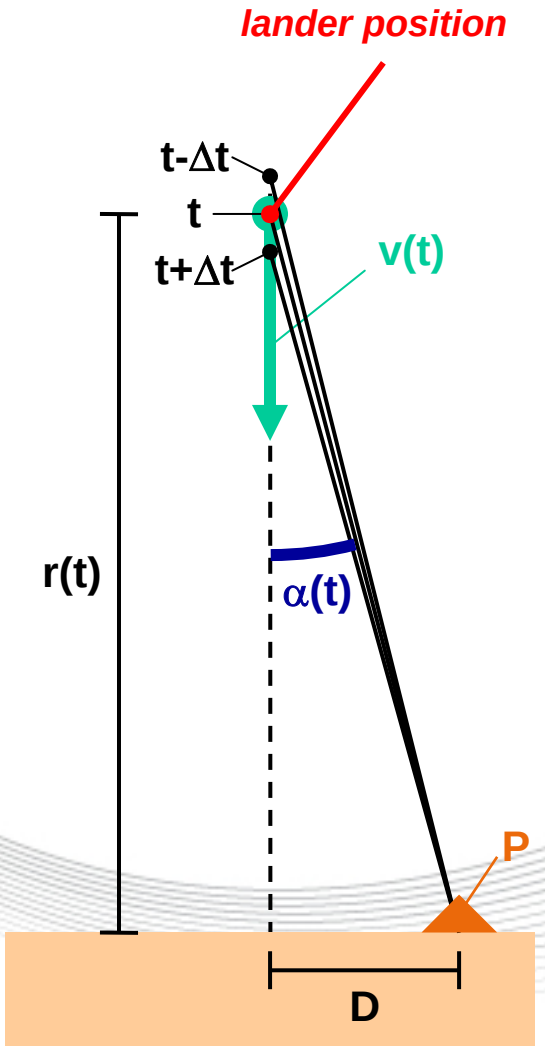
Relative Navigation

- Here the Navigation function has to produce the state estimates without the utilization of onboard maps / known landmarks; the state has to be expressed in a “Relative” terrain-fixed reference frame; also the gravity vector direction (utilized for trajectory generation and control) has to be computed
- It is very important to have a robust and accurate method for obtaining these state estimates, since the estimation errors adversely impact all the “downstream” functions of the GNC processing chain
- Simple “baseline” computing architecture (but also easily expandable) considered here:
  - Image Processing: identifies and tracks “unknown” features / landmarks on images, producing a series of “Feature Tracks” (FTs);

    → reference: FEIC (Univ. of Dundee)
  - Navigation Filter: produces real-time state estimates by merging FTs and IMU data



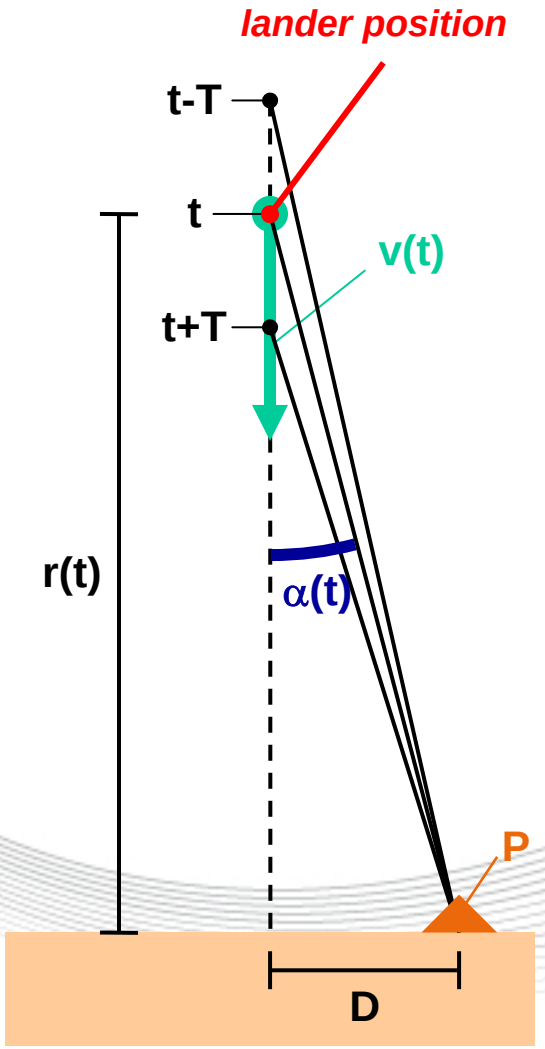
Relative Navigation: “Classical” approach



- Simple 1D case considered for comparison:
  - Vertical descending motion, single tracked point (P)
  - Measured: α = arctg(*D* / *r*) (from FTs), *a* = *r*’’ (from the IMU)
  - Key objective: estimate *D*
- D can be obtained with: $q = \frac{r}{D} = \frac{1}{\tan(\alpha)} \Rightarrow q'' = \frac{r''}{D} \Rightarrow D = \frac{r''}{q''}$
- q’’ can be estimated from q, so: $D \cong \frac{r''(t)}{\frac{1}{(\Delta t)^2}(q(t-\Delta t) - 2q(t) + q(t+\Delta t))}$
- Δt must be low (e.g. ~0.1 sec) for a good approximation of q’’.
- However, since very small angle differences have to be evaluated, the signal-to-noise ratio of q(t) can become quite low (even <<1!)
- Clearly, by filtering and merging many single estimates of *D* the SNR will improve, but: is there a different way of computing *a single estimate* of *D*?

Carlo Gavazzi Space SpA

Relative Navigation: "QuickNav" Approach



- The "classical" approach is based on *differential expressions* → is it possible to increase the accuracy by using *integral expressions* instead?
- 3 instants considered: $t_1 = t_2 - T$, t_2, $t_3 = t_2 + T$ (i.e. T instead of Δt)
- By considering some integrals of the acceleration,

$$A_1 = \int_{t_1}^{t_2}\int_{t_1}^{t} a(\tau)d\tau dt, A_2 = \int_{t_2}^{t_3}\int_{t_2}^{t} a(\tau)d\tau dt, A_3 = \int_{t_1}^{t_2} a(\tau)d\tau$$

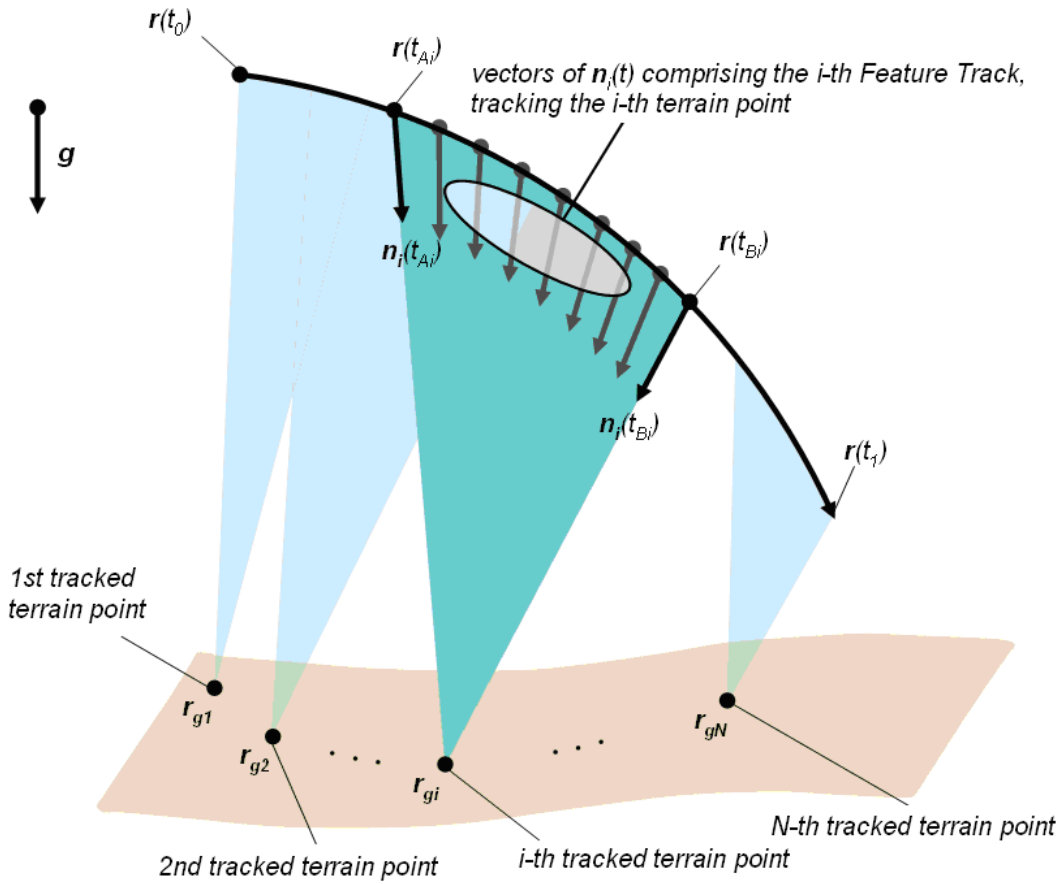
the following expression for D can be obtained:

$$D = \frac{A_1 - A_2 + TA_3}{q(t_2+T) - 2q(t_2) + q(t_2-T)}$$

- Since no approximations have been used in the above expression, it is possible to increase the "temporal baseline" (i.e. the T value) from ~0.1 to ~1 sec → that allows a reduction of the SNR (and of the overall estimation error) of ~10x

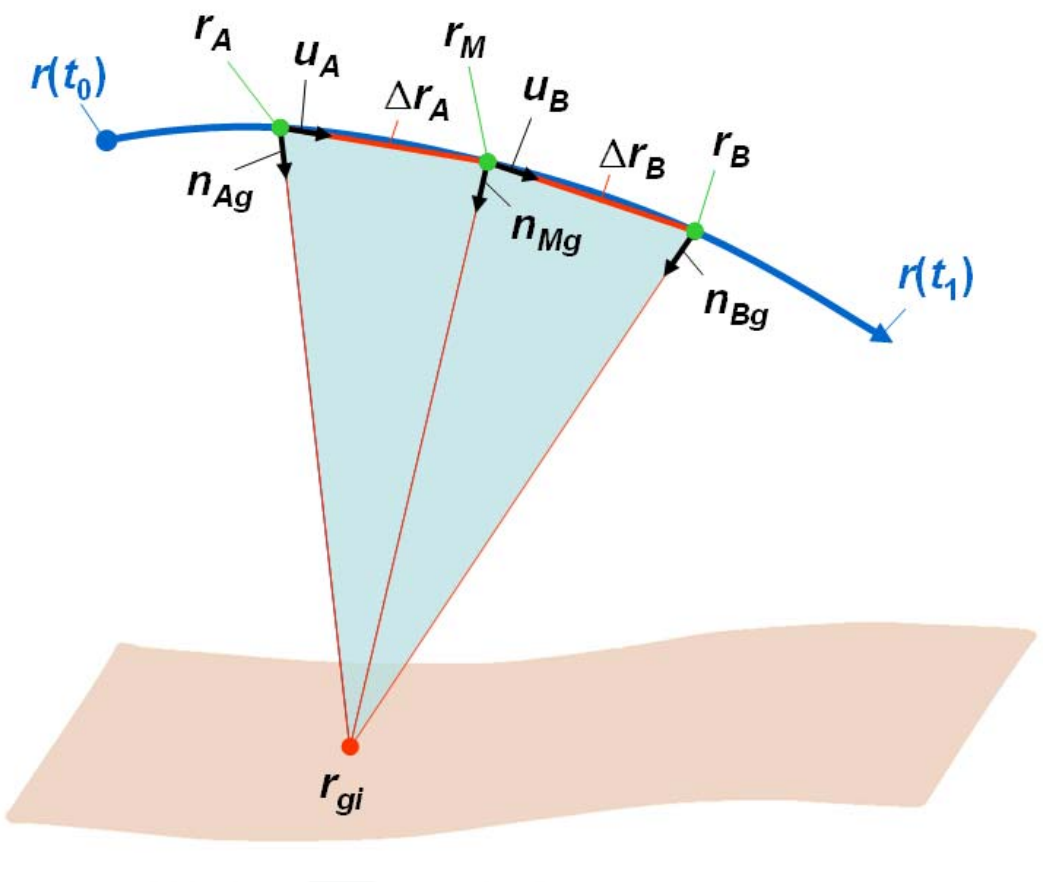
Non-simplified "3-D Vision-Based Navigation" algorithm overview: General characteristics

- **Descent time interval: $[t_0, t_1]$, initial reference frame origin chosen in order to have $\mathbf{r}(t_0) = 0$**



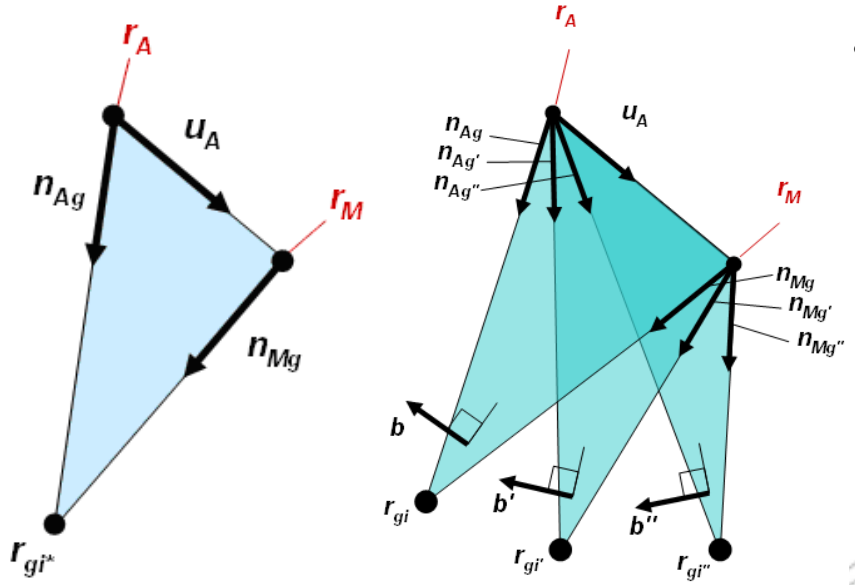
- **Three-step operation: Pre-processing step, Core step, Post-processing step**
- **Inputs to the Core step:**
  - **Profile of $\mathbf{a_m}(t) = \mathbf{a}(t) - \mathbf{g}$ (i.e. the "measurable" part of the acceleration) for $t \in [t_0, t_1]$**
  - **A series of "feature tracks", expressed using the direction vectors $\mathbf{n_i}(t)$**
- **Outputs of the Core step:**
  - **An estimate of the velocity vector at the beginning of each "feature track", $\mathbf{v}(t_A)$**
  - **An estimate of the gravity acceleration vector, g**
- **Using $\mathbf{v}(t_A)$, g and the $\mathbf{a_m}(t)$ profile, it is possible to reconstruct the entire position and velocity profile**

Non-simplified “3-D Vision-Based Navigation” algorithm overview: Core step



- **For each feature track, starting at $t = t_A$ and ending at $t = t_B$, the following points are considered:**
  - **r_A, r_B and r_M: vehicle positions respectively at $t = t_A$, $t = t_B$ and $t = t_M = (t_A + t_B) / 2$**
  - **r_{gi}: tracked point on the ground**
- **The following position differences are defined then:**
  - **$\Delta r_A = r_M - r_A$**
  - **$\Delta r_B = r_B - r_M$**
- **The first operation involves the computation of the directions of these two vectors, i.e.**
  - **$u_A = \Delta r_A / l_A$, where $l_A = | \Delta r_A |$**
  - **$u_B = \Delta r_B / l_B$, where $l_B = | \Delta r_B |$**

Non-simplified "3-D Vision-Based Navigation" algorithm overview: Core step (continued)

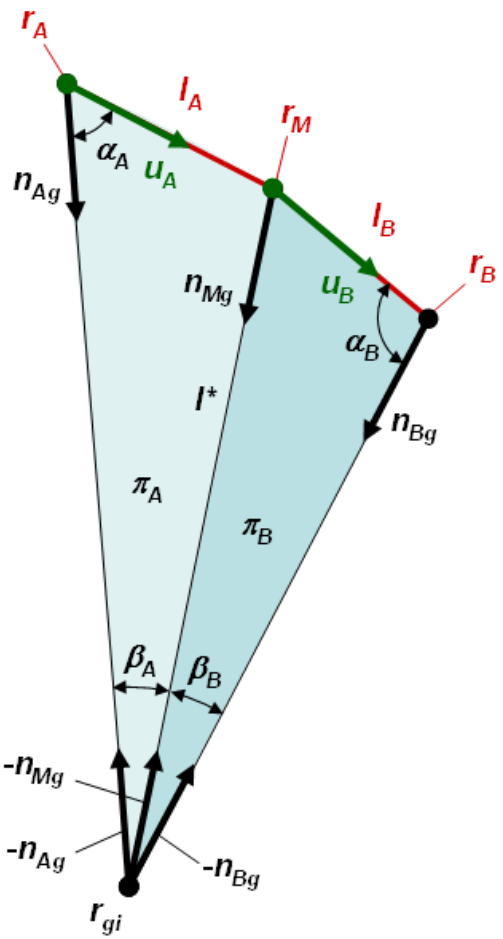


- In order to compute u_A, the following is done:
  - For each feature track starting before $t=t_A$, and ending after $t=t_M$, the plane containing the following points is defined: r_A, r_M and r_{gi^*} (the point tracked by that track)
  - Since u_A is contained in all the planes of the resulting set, it can be obtained by computing the vector that is "most orthogonal" (in the least-squares sense) to the set of vectors $\{b_j\}$ that are normal to these planes

$$J(\mathbf{u}) = \sum_{j=1}^{M} (\hat{\mathbf{b}}_j \cdot \mathbf{u})^2$$

- In order to minimize that function, subject to the constraint |u| = 1, it necessary to find the points, on the unit sphere, for which the gradient of *J*(u), $\nabla J(\mathbf{u})$, is perpendicular to the sphere's tangent plane. That means that $\nabla J(\mathbf{u})$ will have to be parallel to u, i.e. $\nabla J(\mathbf{u}) = k\,\mathbf{u}$ for some scalar *k*. The expression for the gradient of J is $\nabla J(\mathbf{u}) = W\,\mathbf{u}$, where W is a matrix obtained from the components of the vectors b_j. Consequently, u_A will have to satisfy the condition $W\,u_A = k\,u_A$ for some *k*. In other words, u_A is found among the eigenvectors of W.

Non-simplified "3-D Vision-Based Navigation" algorithm overview: Core step (continued)



- At this point, assuming that also u_B has been estimated (using the approach used for u_A), it is possible to consider two triangles π_A and π_B (in general not coplanar) that have, respectively, the following vertices:
  - r_G, r_M and r_A
  - r_G, r_B and r_M
- The two triangles share a side of length /* (the value of /* is not known at this stage by the algorithm)
- The values of α_A, α_B, β_A, and β_B can be obtained by computing the angles between some of the unit vectors (whose values have been previously computed)

$$\alpha_A = \arccos(\mathbf{u_A} \cdot \mathbf{n_{Ag}}),\ \alpha_B = \arccos(-\mathbf{u_B} \cdot \mathbf{n_{Bg}})$$

$$\beta_A = \arccos(\mathbf{n_{Ag}} \cdot \mathbf{n_{Mg}}),\ \beta_B = \arccos(\mathbf{n_{Bg}} \cdot \mathbf{n_{Mg}})$$

- Using these values, it is possible to compute the value of ρ, which is the ratio between l_B and l_A:

$$\rho = \frac{l_B}{l_A} = \frac{\sin(\alpha_A)\sin(\beta_B)}{\sin(\alpha_B)\sin(\beta_A)}$$

Non-simplified "3-D Vision-Based Navigation" algorithm overview: Core step (continued)

- **Since**

$$\Delta\mathbf{r}_A = \mathbf{v}_A T + \int_{t_A}^{t_M}\int_{t_A}^{t} \mathbf{a}_m(\tau)\,d\tau dt + \tfrac{1}{2}\mathbf{g}T^2 \qquad \Delta\mathbf{r}_B = \mathbf{v}_M T + \int_{t_M}^{t_B}\int_{t_M}^{t} \mathbf{a}_m(\tau)\,d\tau dt + \tfrac{1}{2}\mathbf{g}T^2 \qquad \mathbf{v}_M = \mathbf{v}_A + \int_{t_A}^{t_M} \mathbf{a}_m(t)dt + \mathbf{g}T$$

it is possible to introduce

$$\mathbf{h}'_{AM} = \int_{t_A}^{t_M} \mathbf{a}_m(t)dt \qquad \mathbf{h}''_{AM} = \int_{t_A}^{t_M}\int_{t_A}^{t} \mathbf{a}_m(\tau)\,d\tau dt \qquad \mathbf{h}''_{MB} = \int_{t_M}^{t_B}\int_{t_M}^{t} \mathbf{a}_m(\tau)\,d\tau dt$$

and write

$$\Delta\mathbf{r}_B = \mathbf{v}_A T_{i*} + \mathbf{h}'_{AM} T_{i*} + \mathbf{h}''_{MB} + \tfrac{3}{2}\mathbf{g}T^2 \qquad \Delta\mathbf{r}_A = \mathbf{v}_A T + \mathbf{h}''_{AM} + \tfrac{1}{2}\mathbf{g}T^2$$

- **By combining these with the equation $\rho = l_B / l_A$, the following equation is obtained**

$$\hat{\mathbf{u}}_B l_A \rho - \hat{\mathbf{u}}_A l_A = \mathbf{h}'_{AM} T_{i*} + \mathbf{h}''_{MB} - \mathbf{h}''_{AM} + \mathbf{g}T_{i*}^2$$

- **This expression can be simplified by using the following values (all computable by the algorithm):**

$$\mathbf{s}_0 = \hat{\mathbf{u}}_B \rho - \hat{\mathbf{u}}_A \qquad \mathbf{s}_1 = \mathbf{h}'_{AM} T_{i*} + \mathbf{h}''_{MB} - \mathbf{h}''_{AM} \qquad s_2 = T_{i*}^2$$

- **The resulting equation is**

$$\mathbf{s}_0 l_A = \mathbf{s}_1 + \mathbf{s}_2 \mathbf{g}$$

Carlo Gavazzi Space SpA

Non-simplified "3-D Vision-Based Navigation" algorithm overview: Core step (continued)

- **By using also the condition $|\mathbf{g}| = g_S$, it is possible to obtain a single quadratic equation with l_A as the unknown:**

$$|\mathbf{s}_0|^2 l_A^2 - 2(\mathbf{s}_0 \cdot \mathbf{s}_1) l_A + |\mathbf{s}_1|^2 - g_S s_2^2 = 0$$

- **This equation has two solutions for l_A; the correct one can be identified via simple checks of the resulting estimates (e.g. typically the "wrong" root will lead to gravity pointing upwards).**
- **Once l_A is obtained, then the following values can computed in sequence: (by using the previously shown expressions): g, l_B, $\Delta\mathbf{r}_A$, $\Delta\mathbf{r}_B$ and $\mathbf{v}_A$. Finally, $\mathbf{v}(t_1)$ can be obtained simply by integration:**

$$\mathbf{v}(t_1) = \mathbf{v}_A + \int_{t_A}^{t_1} \mathbf{a}_m(t)dt + (t_1 - t_A)\mathbf{g}$$

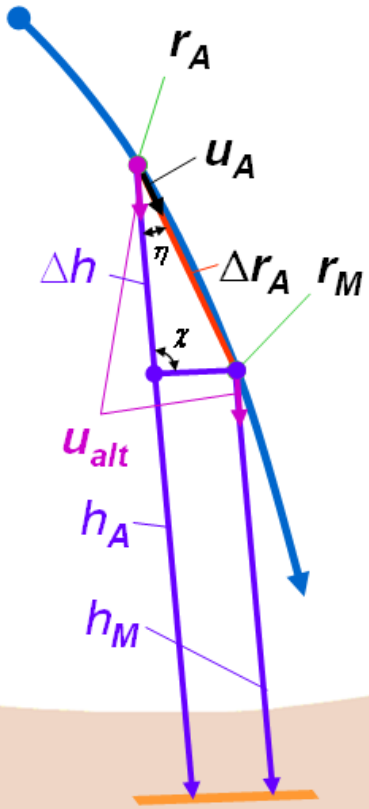
- **The velocity profile is simply computed by back-integrating the estimate $\mathbf{v}(t_1)$ using $\mathbf{a}_m(\cdot)$ and g:**

$$\mathbf{v}^A(t) = \mathbf{v}^A(t_1) + \int_{t_1}^{t} \mathbf{a}_m^A(\tau)d\tau + (t - t_1)\mathbf{g}^A$$

- **Since $\mathbf{r}_A(t_0) = 0$ by definition, the position profile is then simply computed by integrating velocity:**

$$\mathbf{r}^A(t) = \int_{t_0}^{t} \mathbf{v}^A(\tau)d\tau$$

Non-simplified "3-D Vision-Based Navigation" algorithm: Addition of an altimeter



- **By adding an altimeter, it is possible to increase the state estimation accuracy, by providing an alternative method for estimating one of the key involved values, l_A (the norm of Δr_A):**

$$l_A = | \Delta r_A | = | r_M - r_A | = | r(t_M) - r(t_A) |$$

- **In the figure, the orientation of the altimeter is indicated with the vector u_{alt}**
- **The angle between u_{alt} and u_a is indicated with η; it can be obtained using only the feature tracks and gyro measurements**
- **Since $\chi \approx 90°$, $l_A = \Delta h / \cos \eta$**
- **In turn, Δh is obtained with $\Delta h = h_A - h_M$, where h_A and h_M indicate the altimeter measurements at t_A and t_M respectively**

Additional information about the algorithm can be found in the paper "NEMO: an Advanced Cross-Application Vision-Based GNC SW Platform and Simulator" (Vukman et al.), presented at IAC 2010, Prague

Relative Navigation: Features and Performance

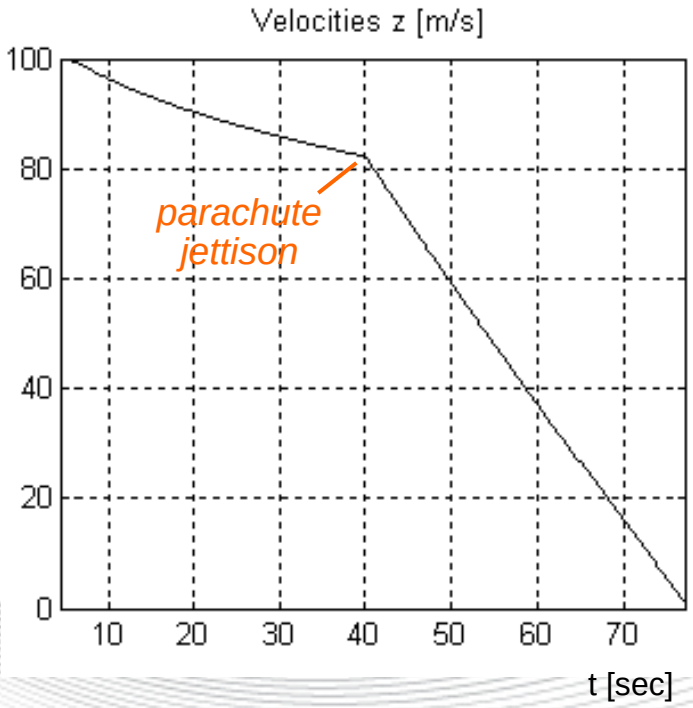
- "Qualitative" advantages of the QuickNav algorithm, w.r.t. "Classic" (based on the Kalman Filter and its variants) solutions:
  - No need for first guess/initialization values (like the initial vehicle velocity or height) at the beginning of the filter operation;
  - No possibility of filter divergence/instability
  - No approximations that could reduce performance (e.g. linearizations) were utilized
  - It is simple to evaluate the effect of input data errors on the overall filter accuracy - no long testing campaign required for that
  - High robustness to IMU noise, and to "outliers" in the feature tracks
  - All the operations that are performed by the algorithm are relatively simple (e.g. no operations on large matrices required) and therefore easy to execute in real-time without requiring excessive computing power
  - The estimation of the direction of the local vertical - which is essential for Guidance and Control - is done accurately, also when no altimeter is present

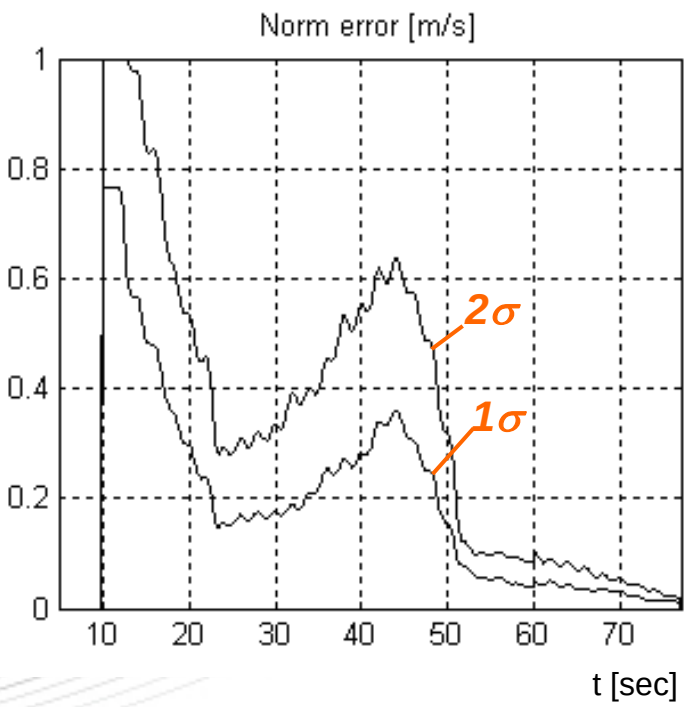
Relative Navigation: Features and Performance

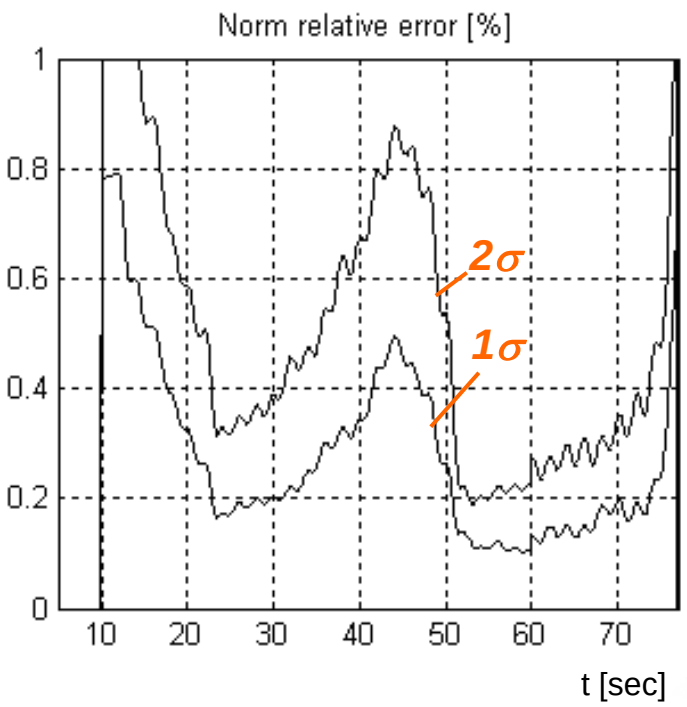
- The validation of the algorithm has been performed at Proof-of-Concept level, using an initial MATLAB implementation of the algorithm, and a software test bench for generating simulated filter inputs in open loop; the current TRL reached is 3
- The outcome of the validation activities, involving a series of Monte-Carlo tests, has shown that the “QuickNav” Navigation Filter also goes well beyond the (E)KF-based filters for what concerns key “quantitative” metrics, i.e:
  - State estimation accuracy (velocity/position error): >10x error reduction achieved
  - Required computing power: >10-50x reduction achieved
- Main testing scenario: Mars Landing, key features:
  - Initial altitude: ~5 km, initial velocity ~100 m/s (~20°w.r.t. the vertical),
  - Powered descent begins after parachute jettison, after ~40 sec
  - For the Image Processing block, FEIC (Feature Extraction and Image Correlation)-like performance has been assumed
  - The trajectory is contained in the x-z plane (z is vertical)

Relative Navigation: Features and Performance

- Key value: velocity estimation error (the position error is proportional to it)
- 1σ and 2σ values plotted, for both absolute (norm of the error vector) and relative (ratio between the norm of the error vector, and the true velocity norm) errors





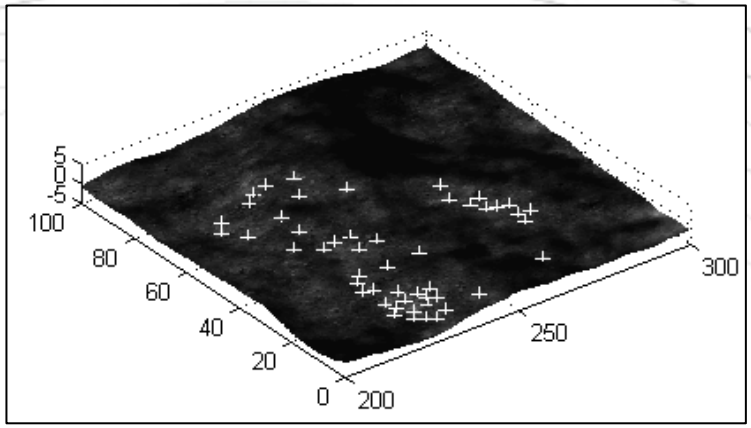


- It can be seen that after 15 seconds of filter operation, the error stays always below 1% (2σ)
- Camera+IMU combination used for these tests; an altimeter would further reduce the errors

Carlo Gavazzi Space SpA

Absolute Navigation

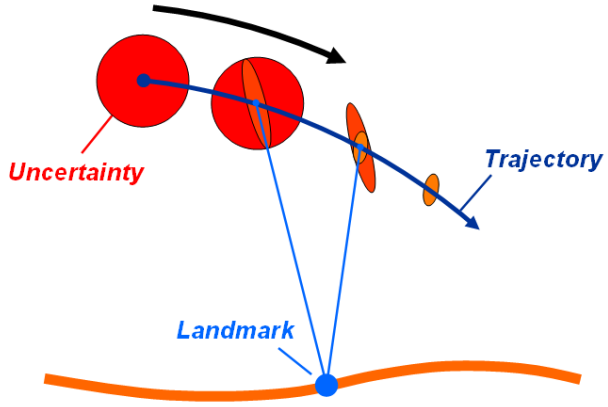
- Here the Navigation function has to produce the state estimates expressed in an “Absolute” terrain-fixed reference frame (ARF), that has been identified before the mission; it is possible to realize the “pinpoint” precision landing functionality (i.e. landing at a pre-specified landing zone)
- Here a set of pre-stored maps (DEMs + orthoimages) / known landmark positions is used
- Two-step approach (for a Moon landing mission):
  - Coarse localization: here the onboard-obtained orthoimage is compared with pre-stored one, to obtain the x,y position components; then, also the z coordinate is computed
  - Fine localization: here, a set of “known landmarks” is identified on the images, and using the knowledge of their position in the ARF, the lander position is obtained; the input data involved is the following:
    - r_{g1}, r_{g2}, ... r_{gN}: positions of the “known landmarks”
    - n_1, n_2, ... n_N: apparent directions in which these landmarks are seen; these define N semi-rays

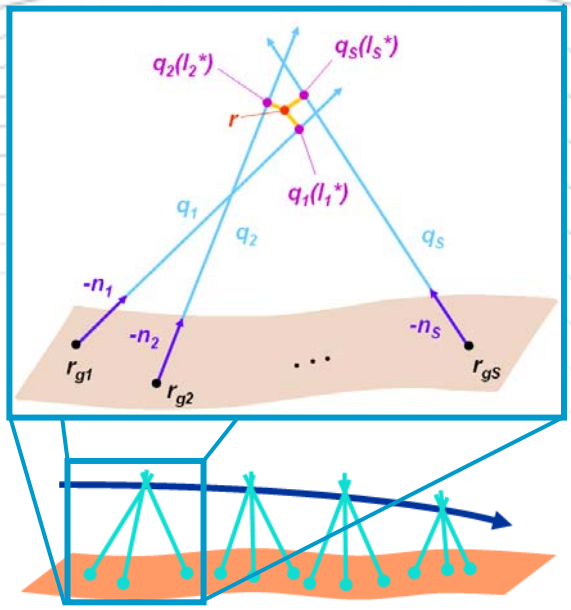


Absolute Navigation

- Classical approach:
  - The (Extended) Kalman Filter (or one of its variants) is applied; $n_i = h_i(r, r_i)$ is the key measurement equation
  - The initial vehicle state uncertainty, expressed via a covariance matrix, gets continuously reduced throughout the descent by using the measurement data
- Approach considered here:
  - A maximum-likelihood estimator is built for obtaining single estimates of r given a set of $\{r_{gi}\}$ and $\{n_i\}$
  - Result: r is obtained by minimizing $f(\mathbf{r}) = \sum_{i=1}^{S} d_i^2(\mathbf{r})$

    where $d_i(r)$ is the distance between the i-th semi-ray and r;
  - That is easy, since $\nabla f(\mathbf{r}) = \mathbf{A}\mathbf{r} + \mathbf{b}$, where $\mathbf{A}$ is a 3x3 matrix
  - Then, using the estimates of r in time, the velocities are estimated with a “wide baseline” approach

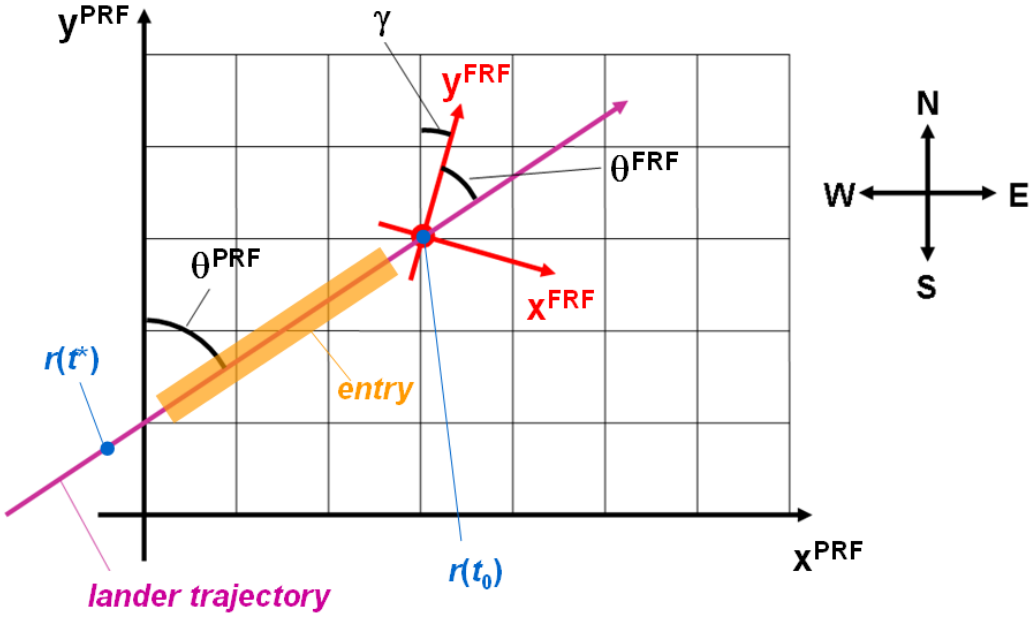




- **Sequence of steps for performing Absolute Navigation:**

1. **Utilization of the "Base" QuickNav algorithm for determining the state profile in I=[t_0, t_1] in FRF (and the gravity vector g) using the camera images and IMU measurements during that period**
2. **Determination of the rotation angle γ between FRF and PRF as the difference between the velocity heading, θ (i.e. the velocity direction in the x-y plane) in FRF and PRF, for a t* before t_0:**

$$\gamma = \theta^{PRF}(t^*) - \theta^{FRF}(t^*)$$

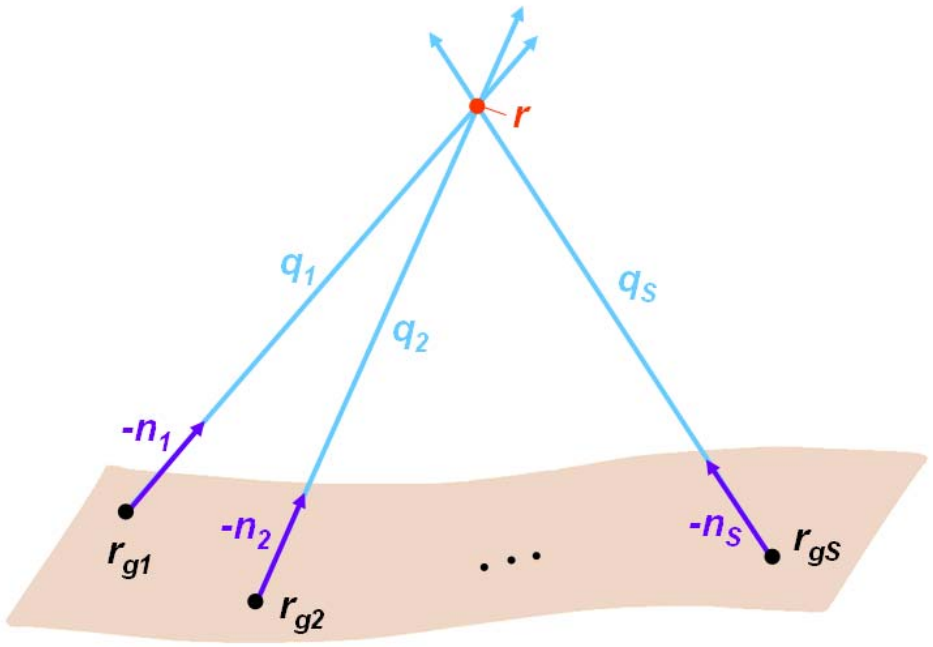


- **where:**
  - **$\theta^{PRF}(t^*)$ can by obtained from pre-entry orbital parameters**
  - **$\theta^{FRF}(t^*)$ can be obtained by back-integrating $v^{FRF}(t_0)$ (calculated by the "Base" QuickNav algorithm) using the acceleration measurements for $t \in [t^*, t_0]$**
- **consequently, it remains only to determine the coordinates, in PRF, of the FRF origin, i.e.**

$$r_O = r^{PRF}(t') - r^{FRF}(t'), \text{ for } t' \in I$$

3. The determination of $r_O = r^{PRF}(t') - r^{FRF}(t')$ is performed in two steps:
   a) Coarse estimation of the x and y components, i.e. r_{0x} and r_{0y}:
      a) An orthoimage (i.e. graylevel map in the xy plane) of the terrain observed by the camera, expressed in FRF, is calculated (by projecting the acquired image(s) on the DEM or terrain plane, computed by the onboard GNC SW)
      b) That orthoimage is matched (using e.g. interest points) against the pre-stored map (expressed in PRF); the search area size depends (also) on the IMU dead-reckoning error
      c) The optimum match corresponds to a (x,y) displacement equal to (r_{0x}, r_{0y})
   b) Coarse estimation of the z component, i.e. r_{0z}, by comparing the z values of the onboard map (in PRF) with the z values of the DEM (or terrain plane) computed by the onboard GNC SW in FRF
   c) Fine estimation of the x, y and z components of r_O
      a) A set of "landmarks" on the observed terrain is chosen, for which the position on PRF is known (since stored on the onboard map); these are indicated with r_{g1}, r_{g2}, ... r_{gS}
      b) The position of these landmarks on the acquired images, $p_i(t')$, is determined (the coarse estimation of r_O, is used for determining the centre of the search space on the images)
      c) Since the attitude of the lander in PRF is available, it is possible to compute from $p_i(t')$ the associated direction unit vector $n_i(t')$ (for each $i \in \{1, ... S\}$) in the PRF

ALGORITHM IN-DEPTH DESCRIPTION (BACKUP SLIDES)



- At this point, the following is available:
  - a set of landmark positions, r_{g1}, r_{g2}, ... r_{gS}, expressed in FRF
  - a set of corresponding direction unit vectors, $n_1(t')$, $n_2(t')$, ... $n_S(t')$ (expressed in FRF), that indicate the direction of each landmark, w.r.t. the position of the lander, r(t'); in other words,

$$\hat{\mathbf{n}}_i(t') = \frac{\mathbf{r}_{gi} - \mathbf{r}(t')}{|\mathbf{r}_{gi} - \mathbf{r}(t')|}$$

(*Note: in order to simplify the notation, the indication of time t' will now be omitted)*

- Consequently, it is possible to define, for each landmark - vector pair, i.e. r_{gi} and n_i, a semi-ray q_i

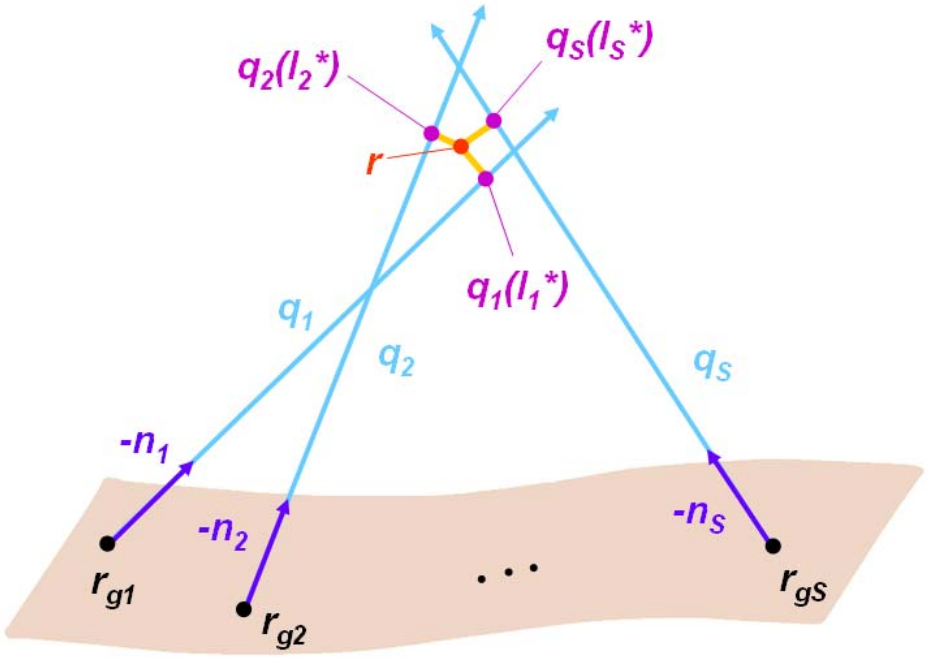
$$q_i(l_i) = r_{gi} + n_i\, l_i$$

such that, in absence of errors, each semi-ray at a certain point intersects the lander position r; i.e.

$$q_i(l_i^*) = r, \text{ for a certain } l_i^*$$

- Clearly, in practice these semi-rays will have no intersection

ALGORITHM IN-DEPTH DESCRIPTION (BACKUP SLIDES)



- Consequently, in order to find the point r in practice, it is useful to look to the point that is the most probable
- The function for minimization that is obtained is the following:

$$f(\mathbf{r}) = \sum_{i=1}^{S} d_i^2(\mathbf{r})$$

where $d_i(\mathbf{r})$ is the distance between the i-th semi ray and a point r; consequently,

$$d_i(\mathbf{r}) = |\, \mathbf{r} - \mathbf{q}_i(l_i^*) \,|$$

where $\mathbf{q}_i(l_i^*)$ is the point of the semi-ray $\mathbf{q}_i$ that is closest to r; for that point, the value of l_i^* is given by

$$l_i^* = \hat{\mathbf{n}}_i^T (\mathbf{r} - \mathbf{r}_{gi})$$

- Using that expression, it is possible to write $d_i(\mathbf{r})$ as

$$d_i(\mathbf{r}) = |\, \mathbf{r} - \mathbf{q}_i(l_i^*) \,| = |\, \mathbf{r} - \mathbf{r}_{gi} - \hat{\mathbf{n}}_{gi}\hat{\mathbf{n}}_{gi}^T(\mathbf{r} - \mathbf{r}_{gi}) \,| = |\, \mathbf{B}_i(\mathbf{r} - \mathbf{r}_{gi}) \,|$$

where $\mathbf{B}_i$ is a 3x3 matrix:

$$\mathbf{B}_i = \mathbf{I} - \hat{\mathbf{n}}_{gi}\hat{\mathbf{n}}_{gi}^T$$

Carlo Gavazzi Space SpA

- Therefore,

$$d_i^2(\mathbf{r}) = (\mathbf{B}_i(\mathbf{r} - \mathbf{r}_{gi}))^T(\mathbf{B}_i(\mathbf{r} - \mathbf{r}_{gi})) = (\mathbf{r} - \mathbf{r}_{gi})^T \mathbf{C}_i(\mathbf{r} - \mathbf{r}_{gi})$$

  where $\mathbf{C}_i$ is a symmetric matrix:

$$\mathbf{C}_i = \mathbf{B}_i^T \mathbf{B}_i$$

- The gradient of f(r) is therefore given by

$$\nabla f(\mathbf{r}) = \sum_{i=1}^{S} \nabla(d_i^2(\mathbf{r})) = \sum_{i=1}^{S} 2\mathbf{C}_i(\mathbf{r} - \mathbf{r}_{gi}) = \mathbf{D}\mathbf{r} - \mathbf{h}$$

  where

$$\mathbf{D} = 2\sum_{i=1}^{S} \mathbf{C}_i \qquad \mathbf{h} = 2\sum_{i=1}^{S} \mathbf{C}_i \mathbf{r}_{gi}$$

- Therefore, the value of r for which f(r) is minimum can be obtained with the expression

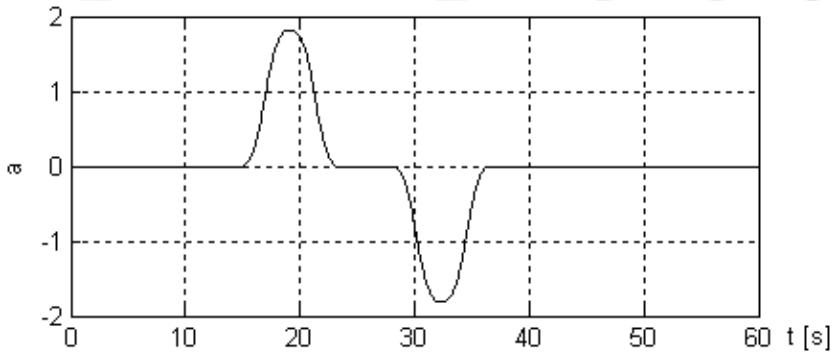
$$\mathbf{r} = \mathbf{D}^{-1}\mathbf{h}$$

- That value is the position of the lander for t=t' in the Planetocentric reference frame, i.e. $r^{PRF}(t')$, and after subtracting $r^{FRF}(r')$ (obtained from the "Base" QuickNav algorithm) from it, the value of r_0 (i.e. the position of the FRF origin, in the PRF) is obtained
- The value of r_0, together with the value of γ, allow the transformation of the FRF state profile in the PRF state profile; that is the final output of the "Absolute Navigation" functionality

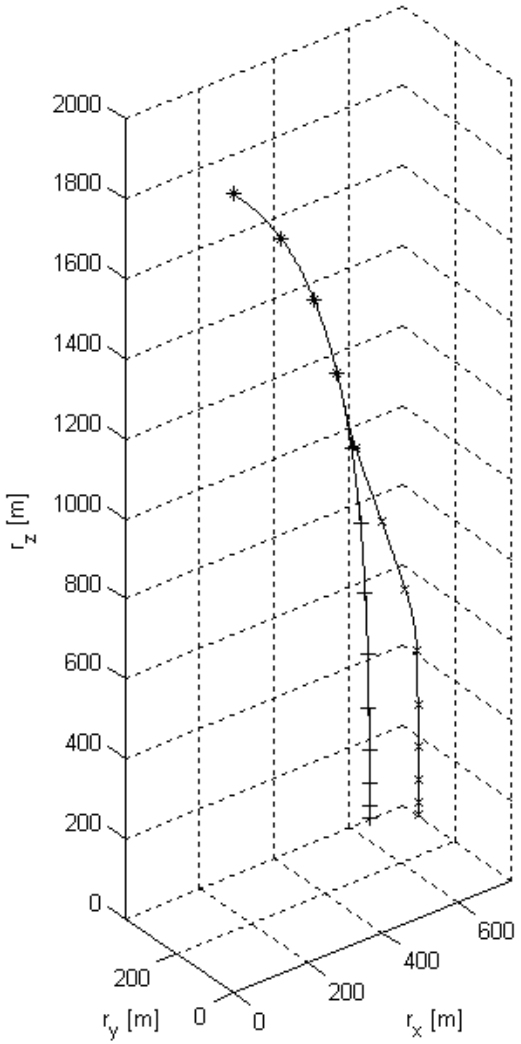
Lander Guidance: Retargeting

General considerations

- Basic descent dynamics: $a(t) = F_T(t) / m(t) + g$, where $F_T(t) = (F_{Tx}(t), F_{Ty}(t), F_{Tz}(t))$
- Main thrust direction is controlled via attitude: $n_x(t) = F_{Tx}(t) / F_{Tz}(t)$ and $n_y(t) = F_{Ty}(t) / F_{Tz}(t)$
- Initial state: $r(t_0)$, $v(t_0)$, finding the control $u(\cdot)$ to meet the target condition $r(t_f) = r_{site}$, $v(t_f) = v_{site}$
- Trajectory parameterization (e.g. from optimal control theory): $u(\cdot) = \varphi(s)$
- The determination of the trajectory can be reduced to non-linear zero-finding, i.e. the computation of s_{land} such that with $u_{land}(\cdot) = \varphi(s_{land})$, the target condition is met
- Typically the chosen parameterization has to take into account various criteria / aspects; in particular it has to allow efficient retargetings (i.e. re-generation of the trajectory after a significant modification of the x,y components of rsite)
- A high retargeting capability can be achieved via a minimum-time/fuel control in the x and y components
  - Result: “bang-bang”-like profiles of n_x and n_y
  - Mars case choice: $u = (n_x, n_y, F_{Tz})$, since $F_{Tz} \approx - |F_T|$



Lander Guidance: Retargeting



- Due to the parametrization, the finding of $u_{land}(\cdot) = \varphi(s_{land})$, often implies the utilization of a “shooting” method (e.g. via Netwon-Raphson), generating a trajectory for each attempted s.
- Problem: the evaluation of $r(t_f)$ and $v(t_f)$ for each s can be time-consuming (due to numerical processing and integration of $u(\cdot)$)
- Potential solution: utilization of polynomial functions: only the coefficients (c_0, c_1, ... c_N) are stored and processed; if p(t) and q(t) are polynomials, then their product and integrals are polynomials
- However: in practice, the required profiles (e.g. the “bang-bang” shapes of n_x and n_y) often require a large number of coefficients
- Solution: instead of simple polynomials, use *piecewise polynomial functions* (the above properties, used for polynomials, that enable “analytical processing”, are also valid for such functions)
- When compared to the numerical approach, a ~100x reduction of the number of operations for each retargeting has been obtained

Conclusions

- One of the targets of our Exploration GNC developments is the identification of algorithms that allow an optimization of the overall GNC subsystem architecture and its efficiency
- In particular, for what concerns the lander navigation function, the “QuickNav” concept has been identified (by using a “clean sheet of paper” approach, and relying on customized solutions) and validated at proof-of-concept level
- Such a concept allows:
  - a simple overall architecture:
    - → baseline sensor set: camera and IMU
    - → processing chain: Image Processing + Navigation Filter
  - high robustness, flexibility and performance; in particular:
    - → efficient hybridization of camera and inertial data (e.g. <1% estimation error at 2σ)
    - → low computational demands (~1-5 MFLOP/sec), no risk of divergence
- Its utilization can have both subsystem-level and system-level benefits

Carlo Gavazzi Space SpA

Headquarters: Via Gallarate 150 - 20151 Milano (Italy)
Tel: +39.02.380481 - Fax: +39.02.3086458
Website: www.cgspace.it - E-mail: cgs@cgspace.it
Other offices in Italy: Benevento - Bologna - Perugia
Rivalta Scrivia (AL) - Roma - S. Giorgio del Sannio (BN)