Towards Resilient Autonomous Navigation of Drones

Angel Santamaria-Navarro*, Rohan Thakker*, David D. Fan, Benjamin Morrell, and Ali-akbar Agha-mohammadi

angel.santamaria.navarro@jpl.nasa.gov
home page: http://angelsantamaria.eu
NASA-Jet Propulsion Laboratory, California Institute of Technology
Oak Groove Dr. 4800 Oak Grove Dr, Pasadena, CA 91109, USA,

Abstract. Robots and particularly drones are especially useful in exploring extreme environments that pose hazards to humans. To ensure safe operations in these situations, usually perceptually degraded and without good GNSS, it is critical to have a reliable and robust state estimation solution. The main body of literature in robot state estimation focuses on developing complex algorithms favoring accuracy. Typically, these approaches rely on a strong underlying assumption: the main estimation engine will not fail during operation. In contrast, we propose an architecture that pursues robustness in state estimation by considering redundancy and heterogeneity in both sensing and estimation algorithms. The architecture is designed to expect and detect failures and adapt the behavior of the system to ensure safety. To this end, we present HeRO (Heterogeneous Redundant Odometry): a stack of estimation algorithms running in parallel supervised by a resiliency logic. This logic carries out three main functions: a) perform confidence tests both in data quality and algorithm health; b) re-initialize those algorithms that might be malfunctioning; c) generate a smooth state estimate by multiplexing the inputs based on their quality. The state and quality estimates are used by the guidance and control modules to adapt the mobility behaviors of the system. The validation and utility of the approach are shown with real experiments on a flying robot for the use case of autonomous exploration of subterranean environments, with particular results from the STIX event of the DARPA Subterranean Challenge.

1 Introduction

Robots are especially useful for accomplishing tasks that would otherwise be impossible or highly dangerous for humans to perform; for instance, search

*Both authors contributed equally to this manuscript.

The paper has supplementary material including a video showing experimental results of the methods presented in the paper.

Copyright 2019 California Institute of Technology. U.S. Government sponsorship acknowledged.

and rescue missions in collapsed zones, package delivery for disaster response or even exploration of other planetary bodies. As opposed to controlled laboratory environments, deploying autonomous robots in real-world scenarios requires *robustness* and *resiliency* to different types of failures. Such failures can include loss of communications, physical damage to sensors or loss of perceptual data in degraded environments (*e.g.*, dust, smoke or fog); and can result in damage to the robot, injuries to humans or failure of mission-critical tasks. In this work, we focus on a particular class of failures which are related to state estimation. These failures are often catastrophic since they cascade down through the whole autonomy stack and are of particular importance on aerial robots, where autonomous capabilities are most often limited by the state estimation quality. Hence, we define a set of desirable "resiliency principles" to reliably operate a system in real-world, challenging environments:

- $RP1$**: Redundancy** in hardware and algorithms to eliminate single points of failure.
- $RP2$**: Modularity** to easily integrate different components.
- $RP3$**: Self-recovery** from failures in a distributed component-level fashion.
- $RP4$**: Adaptability** of mobility behaviors based on estimation health status to ensure safety.
- $RP5$**: Real-Time performance** under size, weight and power constraints.

These principles are not tackled in most state estimation approaches, which combine data from several sensors into one tightly coupled estimation engine (*e.g.*, [11–14]). These methods are designed and evaluated for accuracy rather than robustness, possess a single point of failure (no $RP1$) and are not able to recover upon breakdown (no $RP3$), leading to catastrophic failures in the navigation stack. For example, KITTI dataset [4] benchmarks accuracy of more than 100 approaches that fuse visual information (monocular, stereo or RGB-D cameras), range measurements (SoNAR or LiDAR) and inertial measurements (IMU). Methods like [1, 5, 9, 10] partially consider $RP1$ by exploiting heterogeneity and redundancy of sensors to achieve resiliency for legged, snake-like and race car robots, respectively, but still possess a single point of failure in the software since they use a single estimation engine. Whereas, [6] partially achieves $RP1$ by detecting only measurement failures and searching for a combination of un-compromised sensors. Furthermore, most state-of-art guidance and control modules [7] either assume that the state estimation module won't fail; or they try to prevent failures through perception-aware planning [2] but in their appearance they don't recover from it (no $RP3$) or adapt (no $RP4$).

In this work, we seek to build towards a "resilient architecture" that follows the resiliency principles stated above, considering the following modules: a) Hardware b) State estimation and c) Guidance and control. First, the hardware resiliency can be achieved using mechanical protections, *e.g.*, propeller guards for drones, and using redundant/heterogeneous sensors and actuators ($RP1$). Secondly, we can prevent single points of failure ($RP1$) in state estimation by combining heterogeneous redundant odometry algorithms. We propose the use

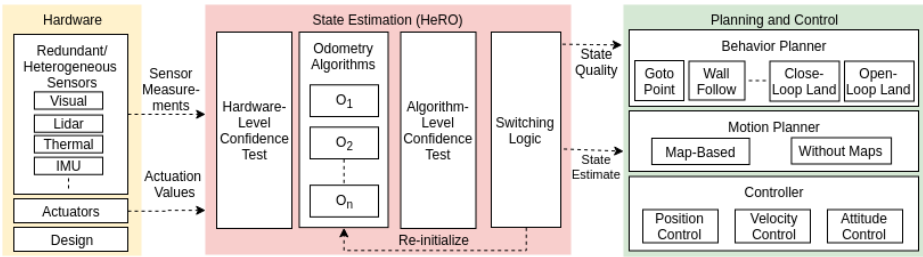


Fig. 1: General resiliency architecture.

of a resiliency logic for supervising the overall estimation pipeline; running confidence checks to verify both sensor data integrity and "health" of the algorithms; re-initializing the failed sensors or methods ($RP3$); and switching (guaranteeing smooth estimates) among all possible estimation streams to provide the best output. The resiliency logic also produces a *state quality* measure which is used by the guidance and control module to adapt the mobility behavior of the system ($RP4$), to ensure safety and perform recovery behaviors ($RP3$); for example, switching to velocity control when position estimates are bad or triggering a safe landing behavior using attitude control when velocity estimates are not reliable. The modular design of HeRO ($RP2$) makes it very easy to leverage COTS (commercial-off-the-shelf) odometry solutions. For example, one can easily combine any number of COTS LiDAR, thermal, and visual odometry algorithms to increase the resiliency of the system. This modular design also allows for a selection of sensors and algorithms to accommodate for particular computational resources, power or payload, allowing to easily run minimal hardware and software architectures in real-time ($RP5$).

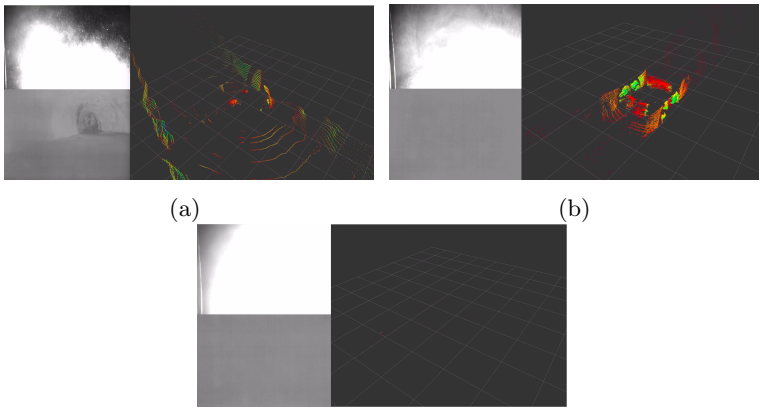
The remainder of this article is structured as follows. In the following section, we describe the main "resiliency architecture", with corresponding concepts and solutions. Validation and experimental results are presented in Section 3, showing the feasibility of the proposed approach through real robot experiments performed live at the STIX event in the DARPA subterranean challenge. Finally, conclusions are given in Section 4.

2 Resiliency Architecture

We propose a framework designed to autonomously detect and adapt to failures and the key concept behind it is to select the best available state estimation stream, based on confidence checks that assess its quality, and adapt the behavior of the robot by selecting accordingly the appropriate planner and controller. This general architecture is depicted in Figure 1 and is described hereafter.

2.1 Hardware

The robot hardware consists of a set of heterogeneous and redundant sensors that are used by the state estimation sub-system (HeRO). This can include,



(a) (b)

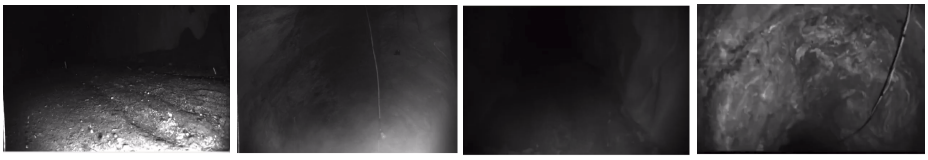
(c)

Fig. 2: Example of motivation for heterogeneity and redundancy of methods. For (a)-(c), top left and bottom left are images from visual and thermal cameras, respectively. Right is a point cloud from a 360 degree LiDAR. (a) Visual methods fail with modest dust. (b) Thermal and visual approaches fail with dust and low thermal gradients. (c) LiDAR, thermal and visual fail with intense dust.

but is not limited to, visible/infrared/thermal cameras, LiDAR (scanners or height sensors), IMU, RaDAR or SoNAR. The sensor set is recommended to be chosen with a diversity of physical phenomena they are based on, to diversify the possible failure scenarios. Examples of such failure scenarios are shown in Figure 2. Here, dust causes issues with visual sensors (Figure 2a), as does low thermal gradient for thermal cameras (Figure 2b). Intense dust can even cause issues with LiDARs (Figure 2c), requiring reliance on proprioceptive sensors such as an IMU.

In addition to hardware considerations, an important factor to reduce the chances of failure is with correct sensor placements. For example, a VIO running on a drone has higher chances of failure if using a forward facing camera compared to a downward facing camera while flying at low altitudes. However, in the latter case, the VIO is susceptible to failure if the drone is tilting or moving very close to the ground at high speeds; or if there is a lack of visual features on the ground. Additionally, there may be different amounts of visual texture or lighting in the different directions. This is the case in dark underground environments, where illumination of the scene comes from light sources on the robot (see Figure 3). Hence, using multiple camera orientations reduces the chances of overall system failure at the cost of more mass, power and computational resources.

Finally, a robust mechanical design of the robot hardware can allow autonomy to have significantly higher tolerances to avoid failure. Notice how incorporating redundancy, heterogeneity and mechanical robustness on robots such as drones, can result in significantly lower flight time, due to increased weights of sensors, however it can highly increase the probability of mission success.



(a) (b) (c) (d)

Fig. 3: Contrasts in visual data from cameras pointing to different directions for different robot positions. Here, to overcome the dark environment, we use on-board illumination. (a) Forward facing near the ground: good lighting and texture. (b) Upward facing near the ground: poor lighting and limited detail. (c) Forward facing when flying in a large tunnel: poor illumination and no texture. (d) Upward facing in the same scenario as (d): good illumination and ceiling details.

2.2 Heterogeneous-redundant odometry (HeRO)

Our objective is to estimate the following robot states with respect to a "world" frame along with their quality:

$\mathbf{p} \in \mathbb{R}^3$	Position represented in world frame
$\mathbf{R} \in SO(3)$	Orientation represented in world frame
$\mathbf{v} \in \mathbb{R}^3$	Linear velocity
$\boldsymbol{\omega} \in \mathbb{R}^3$	Angular velocity
$\mathbf{a} \in \mathbb{R}^3$	Linear acceleration
$\boldsymbol{\alpha} \in \mathbb{R}^3$	Angular acceleration
$Q_i \in \{Good, Bad\}$	Quality of i, with $i \in [\mathbf{p}, \mathbf{R}, \mathbf{v}, \boldsymbol{\omega}, \mathbf{a}, \boldsymbol{\alpha}]$

These states are used (when available) for motion planning and control, whereas the quality metrics are used by the behavior planner to select the appropriate mobility service for the current mission task. Note that all attributes are represented in robot body frame unless stated otherwise. Moreover, we restricted the quality of the state to binary values although it can be easily generalized to higher resolutions and even continuous representations.

To obtain the best available estimation, we use HeRO, consisting of three main components: a) A stack of odometry algorithms running in parallel, b) A set of confidence checks on the quality of the estimation and c) A resiliency logic. All these modules are detailed in the following.

A. Odometry Algorithms

Any single estimation algorithm will have circumstances in which may fail; circumstances that are particular to the sensor/algorithm combination. Hence, we run in parallel a stack of heterogeneous odometry algorithms (*e.g.*, visual, LiDAR, thermal, RaDAR or SoNAR), to increase the probability of overall success by having non-overlapping failure scenarios. There is no special requirement on the type of algorithm, with the ability to incorporate either tightly or loosely coupled approaches. However, to take advantage of all possible mobility services, there is a need for estimating position, orientation, velocity and, ideally, acceleration. If the method is only estimating positions, it can be fused with IMU

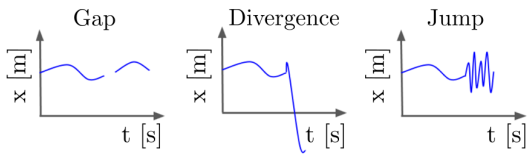


Fig. 4: Different failure modes for odometry algorithms.

measurements (in a loosely coupled fashion) to obtain the rest of the state. This might be the case for pure visual or LiDAR odometry methods (VO and LO, respectively).

B. Confidence Checks and Health Monitoring

The detection of malfunctioning odometry streams is a fundamental feature to select the best state estimation. Hence, it is of interest to define the possible failures in this context. At a high level, a failure is any estimation that jeopardizes robot controls or planning. Early identification of failures can allow for system recovery or, if recovery is not possible, performing safety fall-backs to keep the robot integrity. HeRO detects the failures by running confidence checks at different levels of implementation. First, HeRO performs confidence checks at hardware-level by checking the data from sensors. As an example, this type of checks in case of using cameras or LiDARs might include (but are not limited to):

- Rate of sensor output.
- Overall image intensity or its variation within the image.
- Distance between first and last return of a LiDAR beam.
- Number of invalid scan points.

Next, HeRO performs confidence checks using data from the odometry algorithms, where the goal is to catch the failures depicted in Figure 4, ideally before they occur or at an early stage. These failures can come in a variety of forms, being a gap in the state updates, a divergence of the estimate, or rapid jumps. The failure could be caused by limitations of the sensor or by the overall odometry algorithm. The checks to detect these failures include:

- Rate of algorithm output (to catch gaps).
- Rate of change of the position/velocity estimate (to catch jumps).
- Trace of the estimation covariance matrix (to catch divergence).

Notice how we can also run checks dependant on particularities of the methods; for instance number or quality of features tracked; or number or quality of inlier features (*e.g.*, from matching/RANSAC).

C. Resiliency Logic

The resiliency logic in HeRO is based on the various odometry estimates and respective confidence checks, to output a resulting state with its quality measure, which will be used by the planner and controller. Moreover, it supervises the individual odometry estimates and requires them to re-initialize in case of detecting failures.

An initial version of the logic can use a predefined ranking of odometry algorithms to select the best estimation method, thus selecting the approach with highest priority that is not failing. When that algorithm fails a confidence check, it is required to re-initialize while the next algorithm in the ranking is selected. These switches are provoked only in case of failure of the current source in order to minimizes the number of switches. A more advanced logic can consider a selection criteria weighted by the quality of the algorithm. This comparison between streams also provides a third avenue for confidence checks; for instance, with three or more streams a voting scheme can be used to identify outliers.

The resiliency logic has to guarantee continuity in the resulting estimate while switching between streams. This consistency in the output can be achieved by iteratively composing an incremental state from the selected odometry source to the state from the previous estimate (*e.g.*, a method estimating pose will re-start with its origin at $\mathbf{0}$, hence if we switch to it after its re-initialization, we have to concatenate its estimation with the previous resulting state of HeRO).

The resulting quality of the estimation is generated depending on the confidence checks in the odometry source being used. Depending on this quality we can enable different mobility services, shown in Table 1 and detailed in the following section. Note that in Table 1 we incorporated the platform height estimation (${}^{g}p_z$) as a separate column because it can be directly used for the controller (altitude hold) and is an intermediate step between having all position estimates ($\mathbf{p}$) or just velocities ($\mathbf{v}$). Similarly, this sensor can be used to estimate vertical velocities (v_z). These velocity-only states can appear either because they are directly provided by a method (*e.g.*, optical flow) or because the logic decided to keep using an approach which is partially failing; for instance, using the output of a method that runs on IMU only while we reset the other update sources. In the latter, velocities can be trusted for short periods. In Table 1, we assume the IMU is not failing, providing good attitude ($\mathbf{R}$), angular velocities ($\boldsymbol{\omega}$) and accelerations ($\mathbf{a}$,$\boldsymbol{\alpha}$) with biases estimated by those methods previously working. If that is not the case we consider an immediate landing in "open loop" control mode.

2.3 Planning and Control

The estimated state will be used for planning and control (see Figure 1). In this work we consider three main layers for these modules:

- **Behavior planning:** A state machine which chooses the type of behaviors the robot should execute depending on the available quality of the estimated state. The possible decisions are depicted in the right-most column of Table 1.

Case	State Quality					Mobility
No.	$\mathbf{p}$	${}^{g}p_z$	v_x, v_y	v_z	$\mathbf{R}$, $\boldsymbol{\omega}$, $\mathbf{a}$, $\boldsymbol{\alpha}$	
1						Global
2						Local
3						Local
4						Closed Loop on z
5						Attitude

Table 1: Summary of available mobility services given different estimation qualities resulting from HeRO. The colour code is Good Bad.

- **Motion planning:** Generates desired trajectories for the robot according to the decisions of the behavior planner.
- **Control:** Tracks desired trajectories and closes the loop directly on the provided state estimates from HeRO.

With regards to the state estimation quality, we distinguish the following mobility services, which include the above mentioned layers:

- **Global:** This group of services require good robot position estimates and may include (but are not limited to) building occupancy-grid maps, running path planning algorithms for reaching a desired location in the map, or reasoning and executing high-level *global* goals. Notice how many of these planning and control algorithms rely on continuous estimates of position, which HeRO provides.
- **Local:** Without reliable position estimates, the robot is restricted to plan and execute *local* actions depending on reliable attitude, velocity and acceleration estimates. Examples of such behaviors can include wall-following, obstacle avoidance or hover-in-place. These behaviors are often robust and locally optimal [3]; however, they often require some tuning and assumptions on the topology of the environment (*e.g.*, flying through a long tunnel can be accomplished with wall-following behavior).
- **Attitude:** If it is not possible to maintain high-quality estimation of either positions or velocities, then specific *attitude* behaviors can be performed to keep the integrity of the robot or even to recover. For example, a drone can land using attitude control and slowly reduce its thrust. Once landed, the chances of being able to re-initialize odometry algorithms dramatically increase (*e.g.*, camera artifacts like motion blur are reduced by not moving or the effect of dust is mitigated by not spinning the propellers).

Notice how some odometry algorithms can give mixed results on their state estimation quality; for instance, estimating reliably velocities but intermittently the positions. In this work, we take advantage of as much reliable state information as possible. An example is obtaining height estimations from a single ranger, helping to close the loop only on the vertical axis (see case 4 in Table 1).

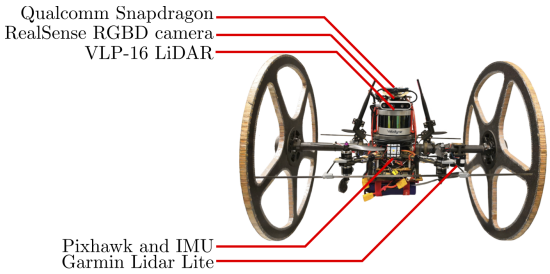


Fig. 5: The *Roll-o-copter*: a hybrid terrestrial and aerial multi-rotor robot

3 Validation and experiments

In this section, we show the validity of our state estimation framework with a specific implementation. We set up the resiliency logic supervising three odometry algorithms using vision, a LiDAR and an IMU as sensing modalities. The first algorithm is a loosely coupled VIO with stereo cameras facing forward, a tightly coupled VIO with a monocular camera facing up, and a LiDAR-inertial odometry (LIO) algorithm with a 360 degree, 16 channel LiDAR. This setup provides redundancy of sensors and heterogeneity in the algorithmic solution. In the following we describe in detail the hardware, odometry algorithms, confidence checks and multiplexing approaches used. Moreover we include experimental results obtained from the STIX event of the DARPA subterranean challenge[1].

3.1 Hardware

We make use of the *Roll-o-copter*: a hybrid terrestrial and aerial multi-rotor equipped with two passive wheels (see Figure 5). Although this robot is designed to take advantage of both aerial and ground terrains, in this work we focus on the aerial mobility of the vehicle and use it as flying-only platform. In flight, *Roll-o-copter* behaves like a normal multi-rotor, with a standard set of electronic speed controllers (ESCs), motors and propellers. It possesses a Pixhawk[2] v2.1 as flight controller, as well as an on-board Intel NUC i7 Core computer. In addition to the standard multi-rotor hardware, we equipped the robot with the following sensors:

- **RealSense[3] RGBD camera.** Composed by a stereo pair of infrared (IR) cameras, an RGB camera and a structured light IR projector.
- **Velodyne** 360° **VLP-16[4] LiDAR.** Rotary head with 16 LiDAR rangers, providing a point cloud with a 360° azimuth angle and ±15° elevation angle field of view.

[1] https://www.subtchallenge.com
[2] http://www.pixhawk.org
[3] https://software.intel.com/en-us/realsense/d400
[4] https://velodyneLiDAR.com/vlp-16.html

- **Garmin LiDAR Lite v3.** We use a 1D LiDAR pointing downwards.
- **Pixhawk v2.1 flight controller.** Composed by an on-board IMU (3-axis gyroscopes and accelerometers) and processing, including a Kalman filter for its own state estimation.
- **Qualcomm Snapdragon[5].** A self-contained flight controller accompanied by an IMU (3-axis gyroscopes and accelerometers), a high resolution forward facing camera and a low resolution upward facing camera for tightly coupled VIO.

3.2 Algorithm Stack

We validate the HeRO approach with two VIOs and one LIO in the stack of methods, which provide the required capabilities to fly in complex and perception-challenging environments. These methods are running in parallel at different frequencies and using different sensor sources ($RP1$), as detailed in the following.

- **Infrared Stereo Visual Inertial Odometry.** We take advantage of ORB-SLAM2 (OS2) [8] running using images from the IR stereo camera (RealSense RGBD). This approach produces 6D pose estimates (3D translations and 3D rotations) and can run up to 60Hz. These estimations are fused with IMU data (running at a frequency of 1kHz) using an EKF (running in the Pixhawk flight controller). Using a stereo odometry algorithm enables us to re-initialize it in flight without the dependence of parallax movements such as in monocular VIO methods. To have continuous odometry estimates when available we disable the loop closure modules of this approach.
- **Monocular Visual Inertial Odometry.** We use the Snapdragon Flight platform from Qualcomm (QSF) running their COTS VIO from mvSDK as the second source of odometry. As in the case of OS2 odometry, here we also incorporated some modifications to allow re-initialization during flight ($RP3$). The incorporation of this approach also demonstrates the capability of our framework to easily integrate closed-source commercial VIO solutions ($RP2$). The state estimation runs at 25Hz.
- **LiDAR Inertial Odometry.** The third source of odometry consists on fusing 6D pose estimates from a LiDAR odometry (LO) approach with IMU data within a regular EKF scheme. The pose estimates from the LO are produced at 20Hz and we use the same IMU as for the IR Stereo VIO, but this time with the data externalized from the Pixhawk flight controller running at 200Hz.

Notice how the above choices allow us to use three different sensing modalities (IR stereo forward-facing, monochrome monocular upward-facing and LiDAR) with three different estimation algorithms ($RP1$). All sensors and approaches are prone to different possible failures, minimizing the occasions where they all fail simultaneously, thus achieving the required level of robustness, as show in the following sections and the accompanying video.

[5] https://developer.qualcomm.com/hardware/qualcomm-flight-pro

(a) (b) (c)

Fig. 6: Real experiment. (a) Frontal camera with the robot idle on the ground. (b) Frontal camera during flight. (c) External view during flight.

3.3 Resiliency logic

In these experiments we use the following confidence tests, set by observing the types of failures the methods are prone to.

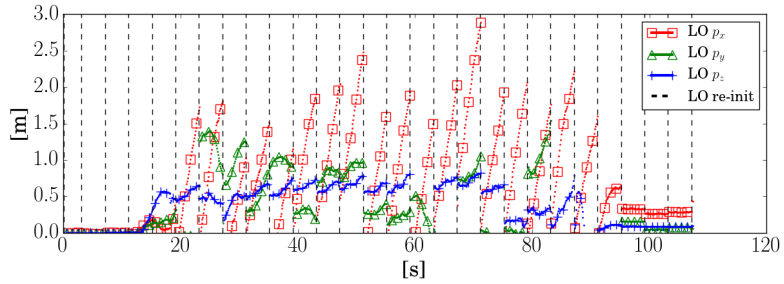
- **Frequency:** The most common mode of failure of OS2 occurs due to feature tracking failures when the drone is executing a fast motion or due to presence of featureless environments. In this case, the frequency of the measurement updates goes down while failing. This policy helps catching "data gaps".
- **Estimation covariance:** The uncertainty of the estimated pose from QSF significantly increases during failure, the state estimate starts to diverge. Hence, we detect these failures by setting a threshold on the trace of the estimation covariance matrix (experimentally determined). This policy helps catching "data divergence".
- **Sudden position changes:** If the estimation method results are inconsistent, it might still produce an output although the covariance of the estimations might not reflect it. To detect these failures we set a confidence check on sudden position changes to catch "data jumps".

3.4 Navigation in a mine, example of usage in the STIX event of DARPA subterranean Challenge

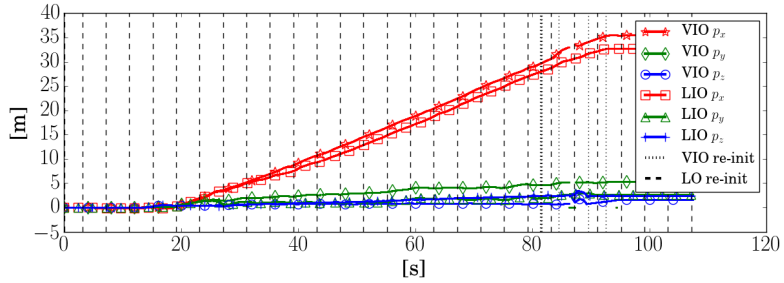
The experimental scenario is part of the DARPA subterranean challenge[6] and, specifically, we show results from the official integration event (STIX). We provide an accompanying video with some of the flights. In these experiments we deploy the *Roll-o-copter* in the entrance of an iron ore mine and set an autonomous exploration mission. Figure 6 shows an example of one of the experimental runs, demonstrating how challenging the environment is in terms of perception and justifies the use of a resilient state estimation approach (*e.g.*, notice the dusty environment comparing Figure 6a where the robot did not start with 6b and 6c while flying). In these experimental runs of the main STIX event we were able to accomplish our missions with *Roll-o-copter*, exploring two different entrances of the mine and validate our resilient state estimation strategy.

An example of these runs (mine entrance 1) is shown in Figure 7 with regular re-initialization of LIO. Here we show the different types of re-initialization,

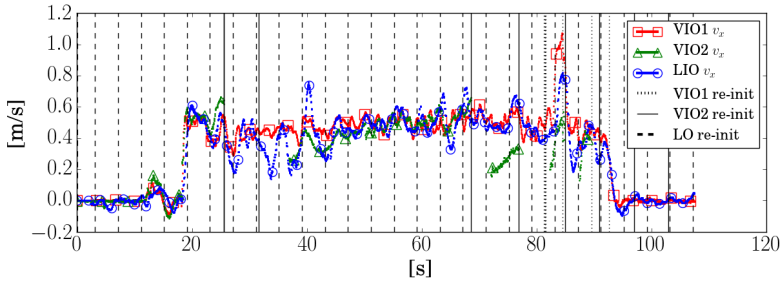
[6] https://www.subtchallenge.com/



(a) LiDAR odometry (LO) X-Y-Z positions.



(b) VIO (OS2) and LIO X-Y-Z position estimates.



(c) VIO (OS2) and LIO X velocity estimates.

Fig. 7: Results from a real experiment with the *Roll-o-copter* flying autonomously in a tunnel (see Figure 6).

depending on the effected odometry stream. A first case consists of partly re-initializing a loosely-coupled odometry approach (LO, in this case), for example resetting one of the inputs of the IMU fusion algorithm. In these cases, we can detect discontinuities by checking the individual estimations (*e.g.*, LO). To avoid inconsistencies, whenever we re-initialize a particular singular source (*e.g.*, VO or LO) that is subsequently fused with an IMU, we keep track of pose discontinuities with a transform manager, similarly as done by the resiliency logic for its estimation output while switching between odometry streams. Figure 7a shows this behavior, and how re-initializing manually every 5s a LiDAR odometry (LO) provokes LO position estimates to jump. Thanks to the transform manager, the

fusion of this LO with an IMU (LIO) is kept consistent and continuous (see Figure 7b where we show the LIO output and an OS2 VIO for comparison purposes). Figure 7c shows an example of resulting x-body axis velocities of all methods in the HeRO stack, running two VIOs (VIO1 is OS2 and VIO2 is QSF) and the LIO test for the same experiment. In this run the resiliency logic was directly selecting VIO1 (used for autonomous navigation) and it requested several times a re-initialization of the VIO2 due to failures.

The modularity of the presented state estimation framework allow us to run different methods on the HeRO stack. For example, in the case of a drone where the available payload does not allow carring a 3D LiDAR, we can still use redundant VIOs. An example of this case is shown in Figure 8 corresponding to a different experiment in the entrance 2 of the mine. In these figures we present the following observations that validates the use of the presented approach:

- **VIO estimations:** we show the estimated position (first three plots) and velocities (latter three plots) of OS2 (VIO1 in red with squares) and QSF (VIO2 in green with triangles), together with the respective output of the resiliency logic (blue solid line). Notice how the resilient output does not always overlap with the estimations because it keeps continuity during switches.
- **Channel selection:** On all Figure 8 we overlap the channel selected by the resiliency logic (magenta and specification on the right axis) while switching between methods.
- **Re-initialization triggers:** Vertical lines in all plots.
- **Mobility services:** Available services depending on the quality of the state estimation, shown in all plots with coloured areas. This includes: Take-off, waypoint navigation and landing using positions (global), wall-following strategies (local) or hovering and landing (attitude). In particular for this experiment, for safety reasons we land after 3s of dead-reckoning flight. This strategy helps us to let the dust settle down and thanks to the re-initialization of the methods, we are able to recover the full global mobility services.

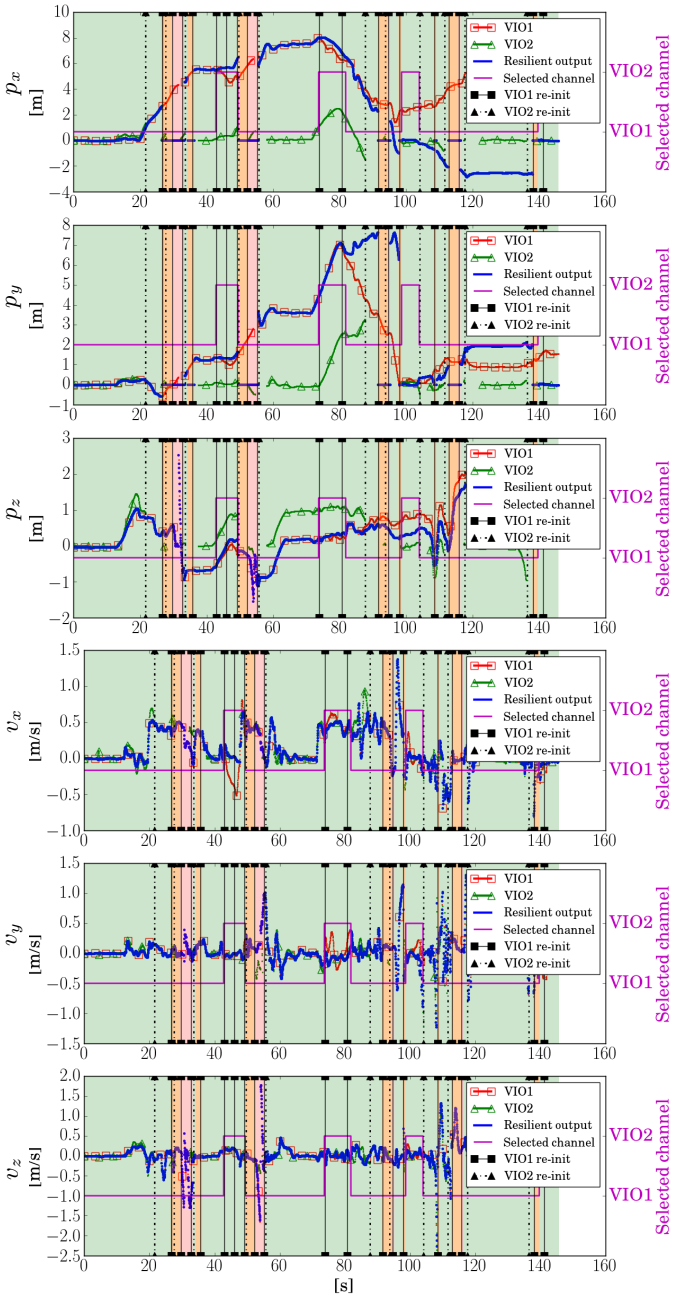


Fig. 8: Results of a real experiment running two VIOs (VIO1 is OS2 and VIO is QSF) together with the resiliency logic. In these plots we include: a) Estimations; b) Resiliency logic output (selection of (a) guaranteeing continuity); c) Channel selection to form (b); d) Triggers of re-initialization of methods (vertical lines); and resulting mobility services available depending on the status (coloured areas and right vertical axis).

4 Conclusions

In this work, we present a robust and resilient navigation architecture pursuing redundancy ($RP1$), modularity ($RP2$), self-recovery ($RP3$), adaptability ($RP4$) and real-time performance ($RP5$). This architecture has considerations on the hardware, state estimation and planning and control modules.

In particular, for hardware we propose to use redundant measurement sources ($RP1$), based on different physical phenomenas to minimize the probability of failures, by mounting sensors in different configurations to maximize the information gain. Moreover, we advocate for a robust mechanical design of the robot to have significantly higher tolerances to avoid failure. For state estimation, we present HeRO: An heterogeneous and redundant odometry estimator which enables resilient state estimation in perceptually degraded environments. We run several odometry estimation approaches in parallel($RP1$) that use data from redundant-heterogeneous sensors which are supervised by a resiliency logic. HeRO is agnostic to the type of odometry approach used and also allows an easy incorporation of commercial-of-the-shelf solutions ($RP2$). The resiliency logic checks the estimation integrity by running confidence tests on sensor data and odometry estimations and, in case of failure, triggers re-initialization of malfunctioning elements ($RP3$). Moreover, HeRO provides quality estimates which are used by planning and control modules to adapt the mobility behavior of the robot to ensure safety ($RP4$). In that sense, we use different mobility services depending on the overall state estimation quality, entailing global (*i.e.*, with good global localization), local (*i.e.*, with good velocity estimates) or attitude (*i.e.*, attitude-only estimation) modalities. Having accurate confidence checks is a critical component to the proposed architecture, an area where further research is welcomed. Such research is not only critical to achieving resilient state estimation but can also be in planning behaviors to prevent failures (*e.g.* perception-aware planning).

Our experiments at the DARPA's STIX event of the Subterranean Challenge showed that using the proposed approach we are able to safely navigate in perceptually degraded environments in presence of heavy dust. Redundancy ($RP1$) allows the drone to fly for longer distances without failure since it required only one of several estimation sources to work. The system is capable of running in real-time ($RP5$) using COTS solutions ($RP2$). It is able to adapt the behavior ($RP4$) on estimation failures by triggering safety landing and was able to recover ($RP3$) once the dust settled down, resuming the mission. This architecture, which follows the resiliency principles, enables us to bring the drones in the real-world where failure is not an option.

Acknowledgement

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1. Carnevale, D., Martinelli, F.: State estimation for robots with complementary redundant sensors. International Journal of Advanced Robotic Systems **12**(10) (2015) 138
2. Falanga, D., Foehn, P., Lu, P., Scaramuzza, D.: PAMPC: Perception-aware model predictive control for quadrotors. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2018) 1–8
3. Florence, P., Carter, J., Tedrake, R.: Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In: Workshop on the Algorithmic Foundations of Robotics (WAFR). (2016)
4. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: IEEE Conference on Computer Vision and Pattern Recognition. (2012) 3354–3361
5. Gosala, N., Bühler, A., Prajapat, M., Ehmke, C., Gupta, M., Sivanesan, R., Gawel, A., Pfeiffer, M., Bürki, M., Sa, I., Dubé, R., Siegwart, R.: Redundant perception and state estimation for reliable autonomous racing. In: IEEE International Conference on Robotics and Automation. (2019) 6561–6567
6. Kim, J., Lee, C., Shim, H., Eun, Y., Seo, J.H.: Detection of sensor attack and resilient state estimation for uniformly observable nonlinear systems having redundant sensors. IEEE Transactions on Automatic Control **64**(3) (2019) 1162–1169
7. Morrell, B., Thakker, R., Merewether, G., Reid, R., Rigter, M., Tzanetos, T., Chamitoff, G.: Comparison of trajectory optimization algorithms for high-speed quadrotor flight near obstacles. IEEE Robotics and Automation Letters **3**(4) (2018) 4399–4406
8. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras. IEEE Transactions on Robotics **33**(5) (2017) 1255–1262
9. Nobili, S., Camurri, M., Barasuol, V., Focchi, M., Caldwell, D., Semini, C., Fallon, M.: Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots. In: Robotics: Science and Systems Foundation. (2017)
10. Rollinson, D., Choset, H., Tully, S.: Robust state estimation with redundant proprioceptive sensors. In: ASME 2013 Dynamic Systems and Control Conference. (2013)
11. Santamaria-Navarro, A., Loianno, G., Solà, J., Kumar, V., Andrade-Cetto, J.: Autonomous navigation of micro aerial vehicles using high-rate and low-cost sensors. Autonomous Robots **42**(6) (2018) 1263–1280
12. Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I.L., Ruess, F., Suppa, M., Burschka, D.: Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. IEEE Robotics Automation Magazine **19**(3) (2012) 46–56
13. Weiss, S., Achtelik, M.W., Lynen, S., Achtelik, M.C., Kneip, L., Chli, M., Siegwart, R.: Monocular vision for long-term micro aerial vehicle state estimation: A compendium. Journal of Field Robotics **30**(5) (2013) 803–831
14. Zhang, J., Singh, S.: Visual-lidar odometry and mapping: low-drift, robust, and fast. In: IEEE International Conference on Robotics and Automation. (2015) 2174–2181