Autonomous Systems

- Definitions
- Applications
- Technologies
  - Mission Planning
  - Mission Control Perception
  - Intelligence
  - Embedded computing
  - Autonomous systems swarms
  - Communications
Autonomous System definitions

A fully autonomous system can:
• Gain information about the environment.

• Work for an extended period without human intervention.

• Move either all or part of itself throughout its operating environment without human assistance.

• Avoid situations that are harmful to people, property, or itself unless those are part of its design specifications.
Autonomous System definitions

Sensorial signals (video, acoustic, tactile, radio signals) should be processed by an AS in real time to:

• interpret the external situation in which it operates;
• relate such a situation to its internal state, by observing it with other proprioceptive sensors, so that it becomes self-aware;
• to use representations to help its own control blocks to drive its actuators;
• to be able to explain at sub-symbolic and symbolic level the reasons of its own choices.
Autonomous System definitions

Autonomous car structure

Courtesy L. Marcenaro, C. Regazzoni
Autonomous Systems

• Definitions
• Applications
  • Cars
  • Drones
  • Marine systems
  • Robots
• Technologies
Autonomous system applications

- Autonomous cars
Autonomous system applications

- Autonomous car sensors and perception
Autonomous system applications

Drones

- Flight control unit + On-board computers
- GPS
- Possible Parachute
- LTE
- FPV camera
- Battery
- LIDAR
- Gimbal
- AV Camera
- Altimeter
- GPS
Autonomous system applications
Drone swarms
Autonomous system applications
Undewater vehicles
Autonomous system applications
Merchant ships
Autonomous system applications
Robots
Autonomous System technologies

- Autonomous car structure
Autonomous System technologies

• Mission Planning and Control
• Perception and Intelligence
• Embedded computing
• Swarm systems
• Communications
• Societal technologies
Autonomous system mission

• Autonomous car mission
  • List of navigation actions
  • Motion along a 2D trajectory (path)
• Autonomous drone AV Shooting Mission: list of actions
  • Shooting Actions: drone + camera
    e.g., Lateral Tracking, Fly-Over, Orbit, ...
  • Navigation Actions: drone action only, does not involve shooting
    e.g., Take-off, Land, Go-to-waypoint, ...
Autonomous system mission planning

• Autonomous car mission planning
  • Find the best (2D) trajectory from start to destination
  • Planning constraints:
    • Road map (e.g., Google maps)
    • Regulatory restrictions (one way streets)
    • Traffic load
  • Use of semantic (2D) maps
Autonomous system mission planning

• Google maps path planning.
Autonomous system mission planning

• Drone mission planning.
• Planning of:
  • Drone flight
  • Payload (e.g., camera) actions
• Use of semantic 3D maps
Mission example: Giro d’Italia

Finish line
Expected at 04:00:00

Closeup on
Actual at 02:05:10

Actual at 00:11:00
Long shot on Cuneo bridge

<<Accident Detected>>
Path Planner

- This submodule is used by:
  - High-level Planner to estimate drone paths and flying times.
  - Onboard Scheduler to compute a path to a landing position in case of emergency.

- Navigation map implemented as a grid. Obtained from Semantic Map.
  - Semantic annotations are indicated as KML features.
  - Geodesic coordinates translated into Cartesian.
  - No-fly polygons become occupied cells in grid.

Path Planner Example

- Path from one corner to the other. Buildings labeled as no-fly zones (obstacles represented as red crosses in the grid).
- Solved in 66 ms.
Autonomous car control

- Car dynamic modelling
- Interfacing car perception to car control
- Levels of car control automation

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Driver</th>
<th>DEM2</th>
<th>DDTF3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No automation</td>
<td>HD4</td>
<td>HD</td>
<td>HD</td>
</tr>
<tr>
<td>1</td>
<td>Driver assistance</td>
<td>HD &amp; system</td>
<td>HD</td>
<td>HD</td>
</tr>
<tr>
<td>2</td>
<td>Partial automation</td>
<td>System</td>
<td>HD</td>
<td>HD</td>
</tr>
<tr>
<td>3</td>
<td>Conditional automation</td>
<td>System</td>
<td>System</td>
<td>HD</td>
</tr>
<tr>
<td>4</td>
<td>High automation</td>
<td>System</td>
<td>System</td>
<td>System</td>
</tr>
<tr>
<td>5</td>
<td>Full automation</td>
<td>System</td>
<td>System</td>
<td>System</td>
</tr>
</tbody>
</table>
Autonomous car control
Autonomous car control

• Steering control
• Braking control
• Power control
Drone Mission Planning and Control Architecture

MULTIDRONE Planning

- Dashboard
- Event manager
- Mission Controller
- High-level Planner
- Path Planner
- Onboard Scheduler
- Action Executor
- Controllers

Dashboards
Director events
Events
Mission actions
Plan
Drone actions
Drone actions
Commands
Drone Control Objectives – Trajectory Tracking

Track a trajectory.
Realistic model.
Robustness to disturbances.
Bounded actuation.
Large basin of attraction.
Drone Controller

1. Drone Status
2. Target Status
3. Shooting Action parameters
4. Reference
5. Drone Velocity Command
Onboard Drone control Architecture

- **Perception**
- **Scheduling**
- **Execution**

**Onboard CPUs/GPUs**

**Autopilot**
- Sensors (IMU, GPS)

**Controller**

**Output**

**Additional Sensors**

**Onboard CPUs/GPUs**
- Perception
- Scheduling
- Execution

**Comms**

**Gimbal**

**LiDAR**

**Navigation Camera**

**RTK GPS**

**On-board CPUs/GPUs**
- Perception
- Scheduling
- Execution

**Comms**

**LTE / WiFi / RC**

**Shooting Camera**
Car collision avoidance

• Sensors for:
  • Vehicle detection/localization
  • Pedestrian detection
  • Real-time car trajectory replanning for collision avoidance.
Drone collision avoidance

- Collision hull defined as a cylinder (yellow).
- Horizontal conflict when reserved cylinder (green) overlaps with others.
- Vertical conflict when blocking cylinder overlaps with others.
- Cylinders allow drones to brake on time and maneuver to avoid collision.
Autonomous System technologies

• Mission Planning and Control
• Perception and Intelligence
• Embedded computing
• Swarm systems
• Communications
• Societal technologies
Autonomous car sensors

• Front/roof cameras
Autonomous car sensors

• Front/roof Lidars
3D localization sensors: GPS

- Other Satellite systems: GLONASS (Russia), BeiDou (China), Galileo (EU).
- RTK-GPS uses measurements of the phase of the signal carrier wave, in addition to the information content of the signal and relies on a single reference ground station (or interpolated virtual station) to provide real-time corrections, providing up to cm-level accuracy.
**Drone Sensors: IMU**

- **Inertial Measurement Unit (IMU):**
  - It measures and reports a body's specific force, angular motion rate and, sometimes, the magnetic field surrounding the body.
  - It uses a combination of accelerometers, gyroscopes and, sometimes, also magnetometers.
Drone Sensors: Laser altimeter

• It measures the altitude (height) above a fixed ground level.
• It emits laser pulses which travel to the ground surface, where they are reflected.
• Part of the reflected radiation returns to the laser altimeter, is detected, and stops a time counter started when the pulse was sent out.
• The distance is then easily calculated by taking the speed of light into consideration.
2D maps

• Google maps.
• OpenStreetMaps.

• Semantic annotated information:
  • (roads, POIs, landing sites) in KML format in Google Maps.
  • roads in OSM (XML) in case of OpenStreetMaps.

• Google Maps JavaScript API.
• OpenStreetMaps API.
3D maps

• Formats:
  • 3D triangle mesh.
  • 3D Octomap.

• Octomap:
  • The octomap is a fully 3D model representing the 3D environment, where the UAV navigates.
  • It provides a volumetric representation of space, namely of the occupied, free and unknown areas.
  • It is based on octrees and using probabilistic occupancy estimation.
Geometrical mapping

Lidar mapping

Repeatability

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mean Error (m)</th>
<th>Median Error (m)</th>
<th>Min Error (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1377</td>
<td>0.1073</td>
<td>0.00098</td>
</tr>
<tr>
<td>2</td>
<td>0.1053</td>
<td>0.0769</td>
<td>0.00045</td>
</tr>
<tr>
<td>3</td>
<td>0.0847</td>
<td>0.0578</td>
<td>0.00083</td>
</tr>
<tr>
<td>4</td>
<td>0.1074</td>
<td>0.0792</td>
<td>0.00078</td>
</tr>
<tr>
<td>5</td>
<td>0.1722</td>
<td>0.1560</td>
<td>0.00130</td>
</tr>
</tbody>
</table>
Geometrical mapping

Validation with a TOTAL STATION
Visual odometry
3D Scene Reconstruction from Uncalibrated Multiple Cameras

Images obtained from Google Earth

3D models reconstructed in 3DF Zephyr Free using 50 images from Google Earth
Visual SLAM

https://youtu.be/sr9H3ZsZCzc
Why is place recognition difficult

Likely algorithm answer:

- NO  NO  TRUE NEGATIVE
- NO  YES  FALSE POSITIVE
Semantic 3D mapping
Crowd detection
Semantic 3D Map annotation

- 2D Crowd region analysis and mapping
Semantic information projection on 3D maps
3D world modeling

• 3D road modeling
• Lane detection
Object detection

- Pedestrian, cars/vans/cyclist, road sign detection
- Current neural detectors are very capable of accurately detecting objects
- SSD, YOLO
Object detection

- But require domain-specific training or fine-tuning
Object detection

- Both can be trained when suitable annotations are available,
  - e.g., YOLO for face and human detection, trained on WIDER dataset
Object detection acceleration

- Examples of acceleration techniques:
  - Input size reduction.
  - Specific object detection instead of multi-object detection.
  - Parameter reduction.
  - Post-training optimizations with TensorRT, including FP16 computations.
Object detection acceleration

- YOLO: good precision in general, but too heavyweight
  - small objects are more challenging to detect.
- Evaluation on VOC (Mean average precision, time):

<table>
<thead>
<tr>
<th>Input Size</th>
<th>FPS</th>
<th>mAP</th>
<th>Forward time (ms) No TensorRT</th>
<th>Forward time (ms) TensorRT</th>
<th>Forward time (ms) FP16</th>
</tr>
</thead>
<tbody>
<tr>
<td>608</td>
<td>2.9</td>
<td>71.26</td>
<td>241.5</td>
<td>128.8</td>
<td>69.3</td>
</tr>
<tr>
<td>544</td>
<td>3.2</td>
<td>73.64</td>
<td>214.4</td>
<td>121.2</td>
<td>64.3</td>
</tr>
<tr>
<td>480</td>
<td>5.4</td>
<td>74.50</td>
<td>155.4</td>
<td>62.3</td>
<td>35.7</td>
</tr>
<tr>
<td>416</td>
<td>6.4</td>
<td>73.38</td>
<td>155.3</td>
<td>56.5</td>
<td>32.5</td>
</tr>
<tr>
<td>352</td>
<td>7.8</td>
<td>71.33</td>
<td>111.0</td>
<td>45.0</td>
<td>24.3</td>
</tr>
<tr>
<td>320</td>
<td>8.5</td>
<td>70.02</td>
<td>103.0</td>
<td>40.4</td>
<td>22.8</td>
</tr>
</tbody>
</table>
UAV Object detection & tracking
Object Tracking specs for car vision

• 2D visual tracking will be employed for target following.
• Satisfactory performance in road footage is required.
• Target tracking should be performed in real-time, i.e., > 25 fps.
• Embedded implementation is required and low computational complexity is preferred.
• Parallel or parallelizable methods (e.g., with CUDA implementations) should be preferred as well.
• Assuming 2D target tracking methods operate faster than combining target detection and recognition methods, long-term object tracking is also preferred.
Joint Detection & Tracking

- **Tracker**: Given the initialized position of a target, the tracker $T$ is responsible for estimating the bounding box of the target in the subsequent frames.

- **Detector/Verifier**: Given a bounding box defining the target in a specific frame produced by the tracker, the detector $D$ is responsible for verifying this result, and then provide the appropriate feedback to the system. If the verification fails this module is responsible for detecting the target in a local search area and provide the correct bounding box to the master node $M$.

- **Master**: $M$ is responsible for the coordination of the two aforementioned modules. The node provides the necessary services to control the verification, the detection and the tracking tasks and controls the communication between the different parts of the system.
Joint Detection & Tracking

- Target re-initialization by the detector in hard tracking cases when tracking algorithms fail
Joint Detection & Tracking

- Target re-initialization by the detector in hard tracking cases when tracking algorithms fail
Multi-Target Tracking

- The implementation is extended to support the tracking of multiple targets while maintaining real-time performance.
3D/6D target localization

- 3D target localization using 3D maps
- Lidar localization
- GPS target localization
- Target location and pose are desired
Target Pose Estimation

• **Computer Vision Approach**
  • Relies on detecting a set of *predefined points* (e.g., facial landmarks) and then using a method for solving the respective *Perspective-n-Point (PnP) problem*, i.e., estimation of the camera position with respect to the object.

• **Limitations:**
  • The 3-D coordinates for the landmark points must be known, i.e., a 3-D model of the object is needed
  • The landmarks points must be precisely tracked, i.e., the texture of the object must allow for setting enough discriminative landmarks
Target Pose Estimation

- **Machine Learning Approach**
  - A neural network receives the object and directly *regresses* its pose
  - Only a set of pose-annotated object pictures are needed
    - There is no need to manually develop 3-D models
    - The models are more robust to variations of the object for which we want to estimate its pose
    - The pose estimation can run entirely on GPU and (possibly) incorporated into a unified detection+pose estimation neural network
  - Very few pre-trained models are available
    - Models must be trained for the objects of interest (faces, bicycles, boats, etc.)
Target Pose Estimation

• **Machine Learning Approach**
  • We integrated a pre-trained yaw estimation model of facial pose (DeepGaze library) into the SSD-300 object detector (trained to detect human faces)
  • Varying illumination conditions seem to affect the estimation.
Pedestrian pose estimation (Openpose)
Pedestrian pose estimation
Advanced autonomous car Intelligence

- Self-awareness
- Driver status modelling/ recognition
- Affective computing
- Attention
- Human (e.g., pedestrian) intention prediction
Autonomous System technologies

• Mission Planning and Control
• Perception and Intelligence
• **Embedded computing**
• Swarm systems
• Communications
• Societal technologies
GPU and multicore CPU architectures. Algorithm mapping

• NVIDIA embedded processing boards
  • NVIDIA Jetson TX2
  • NVIDIA Jetson Xavier
• GPU and multicore CPU architectures
  » Multicore CPUs
  – GPUs
• Algorithm mapping:
  Convolutions
GPU and multicore CPU architectures. Algorithm mapping

- NVIDIA embedded processing boards
  - NVIDIA Jetson TX2
  - NVIDIA Jetson Xavier
- GPU and multicore CPU architectures
  - Multicore CPUs
  - GPUs
- Algorithm mapping:
  Convolutions
Processing Units

- **Multicore (CPU):**
  MIMD.
  Focused on latency.
  Best single thread performance.

- **Manycore (GPU):**
  SIMD.
  Focused on throughput.
  Best for embarrassingly parallel tasks.
ARM Cortex-A57: High-End ARMv8 CPU

• ARMv8 architecture
  • Architecture evolution that extends ARM’s applicability to all markets.
    – Full ARM 32-bit compatibility, streamlined 64-bit capability.

• High-performance next-generation microarchitecture
  – Improved performance on all workloads – 32b/64b integer, FP / SIMD.
  – Optimized for modern high-end workloads.
  – Significant improvements in power efficiency.
GPU and multicore CPU architectures. Algorithm mapping

- NVIDIA embedded processing boards
  - NVIDIA Jetson TX2
  - NVIDIA Jetson Xavier

- GPU and multicore CPU architectures
  - Multicore CPUs
  - GPUs

- Algorithm mapping:
  Convolutions
GPU Optimization

- Spawn threads.
- Use registers.
- Loop unrolling.
- Use SIMD capabilities.
- Take data locality into consideration.
- Trust the compiler.
Pascal microarchitecture

Pascal microarchitecture

https://devblogs.nvidia.com/inside-pascal/gp100_sm_diagram/
GeForce GTX 1080

- Microarchitecture: Pascal.
- SMs: 20.
- CUDA cores: 2560.
- Clock (base/boost): 1607/1733 MHz.
- GFLOPs: 8873.

- DRAM: 8 GB GDDR5X at 10000 MHz.
- Memory bandwidth: 320 GB/s.
- L2 Cache: 2048 KB.
- L1 Cache: 48 KB per SM.
- Shared memory: 96 KB per SM.
NVIDIA Jetson Xavier

• AI Computer for autonomous machines
• Designed for robots, drones and other
• Multiple operating modes (10/15/30 W)
• Comparison to TX2:
  Greater than 10x the energy efficiency.
  More than 20x the performance
CUDA

- Compute Unified Device Architecture (CUDA) is a parallel programming framework.
- Developed by Nvidia.
- Started as an attempt to give C/C++ programs access to GPU resources.
- Microarchitectures are name after famous physicists (Kepler, Maxwell, Pascal, Turing, Volta).
CUDA

- Data in CPU RAM are moved to device RAM, then device L2 cache then SM L1 cache.
- The CUDA kernel is the function that will run in parallel.
- When a kernel is launched, threads are grouped into blocks and all blocks form the CUDA grid for the kernel.
- Blocks are assigned to SMs in thread warps.
- Each CUDA kernel can handle 4 threads.
- GPU usage can be monitored through command line tools (nvidia-smi) or NVIDIA’s API (NVAPI).
GPU and multicore CPU architectures. Algorithm mapping

- NVIDIA embedded processing boards
  - NVIDIA Jetson TX2
  - NVIDIA Jetson Xavier
- GPU and multicore CPU architectures
  - Multicore CPUs
  - GPUs
- Algorithm mapping:
  Convolutions
Introduction to fast CNN convolution algorithms

- Typical 2D convolutional layer \( l \) of a CNN:

\[
x(i, j, c_{l+1}, l+1, k) = f(b(l, k) + \sum_{c=1}^{C_l} \sum_{i'=0}^{H_{l,k}} \sum_{j'=0}^{W_{l,k}} h(i', j', l, k) x(i-i', j-j', c, l, k))
\]

input feature map \( x_l : N_l \times M_l \times C_l \)-dimensional 3D tensor

- \( w_{l,k} : N_{l,k} \times W_{l,k} \times C_l \)-dimensional 3D tensor
- \( b(l, k) \): bias term
- \( f \): nonlinear activation function
Fast 1D convolution algorithms with minimal computational complexity

- Winograd convolution algorithms
  
  \[ Y = C(Ax \otimes Bh) \]

- Require only \(2N - \nu\) multiplications in their middle vector product, thus having minimal multiplicative complexity
Autonomous System technologies

• Mission Planning and Control
• Perception and Intelligence
• Embedded computing
• Swarm systems
• Communications
• Societal technologies
Autonomous Systems swarms

• Car platoon control
Autonomous Systems swarms

- Lane-less highways
- Collision avoidance
- Fluid dynamics principles
Drone swarms
Leader-following for drone formation control

• Main idea:
  Trailer-like behavior for the followers.

In inertial frame:
Translated identical paths

In trailer frame:
Different paths, no superposition
SA1 - Constant relative positions
Autonomous System technologies

- Mission Planning and Control
- Perception and Intelligence
- Embedded computing
- Swarm systems
- Communications
- Societal technologies
Autonomous System Communications

- Communication infrastructure
- Video streaming
Autonomous System communications

- Communication infrastructure
  - Vehicle2ground
  - Vehicle2vehicle
- 5G/LTE/WiFi communications
- Ground2vehicle command communications
- Sensor data, telemetry communications
- Communication latency
- QoS in communications
- Robustness, security
Drone Swarm Communication infrastructure

- Drone2Drone Communication.
- Drone2Ground communication.
- Live broadcasting.
Drone Communications Infrastructure

Objective: Secured and resilient transparent IP access to drones / ground station (LTE and WiFi).

Subnet 2 (LTE) INTERNAL (10.10.40.0/24)
Subnet 3 (WiFi mesh) INTERNAL (11.11.0.0/24)
Subnet 4.1 (Drone coms board #1) 192.168.1.0/24
Subnet 4.2 (Drone coms board #2) 192.168.2.0/24
Subnet 4.3 (Drone coms board #3) 192.168.3.0/24

Private network (no connection with external wide network)
Drone Communications Infrastructure

• LTE & Wi-Fi communications
• Default IP gateway to the ground and to other drones.
• Route traffic to/from wireless link interfaces (LTE & Wi-Fi)
  • Transparent to the users of the com. module.
• QoS : mark and Schedule IP flows depending on applications
• QoS priority recognized thanks to:
  • IP-mark in DSCP field or
  • Planned IP 5-tuple (@src, @dest, Psrc, Pdest, proto).
Drone Communications Infrastructure

• LTE & Wi-Fi communications
• Admission control.
• Traffic shaping (when congestion occurs).
• Communications authentication, encryption and other security related mechanisms.
• Monitor and report communication link availability.
5G Communications Infrastructure

- Internet of Vehicles
- Massive deployment, throughput
- Ultra low latency networks
- Robustness
- Edge/cloud computing
Autonomous System Communications

- Communication infrastructure
- Video streaming
Drone Digital Video Streaming

Problem 1: Network

Considerations:

**Wireless** communication with receiver - weak & subject to failure (distance, obstacles, other wireless networks etc).

Good quality video is **massive** in terms of **Mbps** required to transfer it

1 second of 720p (1280x720) **8-bit video** requires 65.92MBytes – prohibitive.

Video **compression** must be used prior to streaming:

H264 & H265 coding are great candidates...

... but they inevitably introduce **delays** (compression + decompression)

Lossy: must find **trade-off between latency & quality**.

Which **network protocol** should be used?

Real-time Transport Protocol (**RTP**) with User Datagram Protocol (**UDP**) TCP is also standardized for use with RTP, but favors reliability instead of timeliness.
Drone Digital Video Streaming
Problem 1: Network

Compression takes place on-board the drone NVIDIA’s Jetson TX2 module offers. Hardware accelerated image/video compression.

Also a 256-core Pascal @ 1300MHz GPU with capability comparable to an Intel Xeon E5-2960 v4 CPU in Deep Learning tasks.
Problem 1: Network

RTP Packets

timestamp: 32 bits

The timestamp reflects the sampling instant of the first octet in the RTP data packet.

The initial value of the timestamp is random, as for the sequence number. Several consecutive RTP packets may have equal timestamps if they are (logically) generated at once, e.g., belong to the same video frame.
Drone Digital Video Streaming

Problem 1: Network

How to synchronize all streams?

RTP Control Protocol (RTCP) may be used in conjunction with RTP


NTP timestamp: 64 bits

Indicates the [wallclock time when this report was sent](https://tools.ietf.org/html/rfc1889) so that it may be used in combination with timestamps returned in reception reports from other receivers to measure round-trip propagation to those receivers.
Drone Digital Video Streaming
Problem 2: Synchronization

- From wikipedia:
  Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks.
  - NTP is used to synchronize the clocks of all servers & clients.
    - This ensures all participating devices use the same clock.
Drone Digital Video Streaming

Problem 2: Synchronization

Scenario 2: multiple drones - one ground station.
Drone Digital Video Streaming
Problem 2: Synchronization

Alongside the stream, visual analysis of the video frames must take place.

More delays -> more synchronization problems.

Metadata needs to accompany each video frame, such as:

- NTP timestamp corresponding to the moment of the frame’s capture
- Drone telemetry status
- Gimbal status
- Camera status.

Metadata can be sent as a separate stream, but synchronization of metadata & video frames must take place at the receiver – problematic. They may be inserted into the stream, but they must survive the compression (no watermarking).

Better yet (probably): insert metadata as RTP header extension.
Gstreamer is written in C, but offers bindings in multiple languages:
https://gstreamer.freedesktop.org/bindings/
Recommended: original C or C++ or Python.
Sample streamer + receiver are provided in Python:
They show how to access pipeline elements & modify them, intercept buffers etc
https://lazka.github.io/pgi-docs/#Gst-1.0 python bindings
Gstreamer official documentation:
https://gstreamer.freedesktop.org/documentation/
Useful elements for custom streams: appsrc and appsink.
GStreamer: open source multimedia framework

“pluggable components [that] can be mixed and matched into arbitrary pipelines”

Example using gst-launch (command line pipeline parser):

```bash
gst-launch-1.0 v4l2src ! autovideosink
```

Opens a usb camera (if one is plugged in) & displays the image in a window

```bash
gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-raw,width=640,height=480 ! autovideosink
```

Caps: set various properties of the stream according to the device's capabilities
Solutions & Tools: Gstreamer examples

H264 compression + RTP streaming:
```
gst-launch-1.0 v4l2src ! x264enc ! video/x-h264, stream-format=byte-stream ! h264parse ! rtph264pay config-interval=1 ! udpsink
```
port=5000

Receive + display:
```
gst-launch-1.0 udpsrc port=5000 caps="application/x-rtp,media=(string)video,clock-rate=(int)90000,encoding-name=(string)H264" ! rtph264depay ! avdec_h264 ! autovideosink
```

gst-launch is great for understanding gstreamer concepts but for more complex matters, code must be written
RptBin element: RTP/RTCP functionality

Sender:
gst-launch-1.0 rtpbin name=rtpbin ntp-time-source=ntp rtcp-sync-send-time=false v4l2src device=/dev/video0 do-timestamp=true ! timeoverlay ! x264enc bitrate=3000000 ! video/x-h264, stream-format=byte-stream ! h264parse ! rtph264pay config-interval=1 ! rtpbin.send_rtp_sink_0 rtpbin.send_rtp_src_0 ! udpsink port=5000 rtpbin.send_rtcp_src_0 ! udpsink port=5001 sync=false async=false udpsrc port=5005 ! rtpbin.recv_rtcp_sink_0

Receiver:
gst-launch-1.0 -v rtpbin name=rtpbin ntp-sync=true ntp-time-source=ntp buffer-mode=synced udpsrc caps="application/x-rtp,media=(string)video,clock-rate=(int)90000,encoding-name=(string)H264" port=5000 ! rtpbin.recv_rtp_sink_0 rtpbin.recv_rtcp_src_0 ! udpsink port=5001 ! rtpbin.recv_rtcp_sink_0 rtpbin.send_rtcp_src_0 ! udpsink port=5005 sync=false async=false

Autonomous System technologies

• Mission Planning and Control
• Perception and Intelligence
• Embedded computing
• Swarm systems
• Communications

• Societal technologies:
  • Security
  • Safety
  • Privacy protection
Safety, Security and ethics

• Misuse avoidance
  no specific legislation prescribes protective measures against misuse and vulnerability exploitation.
  Vehicle hacking, GPS signal jamming, weak security in communications can also allow obtaining the video captured by the drone, or its intended flight path.
  Redundant active perception methods (vehicle localization), secure and signed autopilot firmware updates, as well as autopilot input commands filtering, can be employed to this end.

• Data security
  Footage data collected by vehicles raise privacy concerns.
Data security requirements

- The types of data that must be protected are:
  - data stored within vehicles:
    - On-vehicle data encryption, allowing access to authenticated people only.
  - data stored in ground infrastructure.
  - data transmitted over the air:
    - Wifi and radio data transmitted should be encrypted.
    - Data protection can be achieved with ciphering and authentication mechanisms, e.g. IPSec over LTE for transmitted data.
  - data that are to be publicly distributed (e.g., AV datasets)
Privacy and data protection

• Protection of personal data must be ensured in the acquired video and/or images.
• The EU’s General Data Protection Regulation 2016/679), repealing the 1995 Data Protection Directive.
• “Member States shall protect the fundamental rights and freedoms of natural persons and in particular their right to privacy, with respect to the processing and distribution of personal data.”
Data protection issues in cars

- Location/trajectory data
- Car sensor data
- Driver performance data
- Data communication security/privacy issues
  - Vehicle2vehicle communications
  - Vehicle to road/ground infrastructure communications
Data protection issues in drones

• Public perceives drones as machines infringing privacy.
• No flights above private property.
• Distinguish between:
  • actors, spectators, crowd
  • public events, private events.
• Data protection issues for AV shooting:
  • for broadcasting
  • for creating experimental data bases.
• Use of data de-identification algorithms when doing AV shooting.
Privacy Protection

• An issue of ethics and security
• Post-production stage
• Approaches
  • Face de-detection (Face detector obfuscation)
    • Naïve approach
    • SVD-DID
  • Face de-identification (face recognizer obfuscation)
    • Gaussian blur
    • Hypersphere projection
Privacy Protection: acceptable facial image quality?

- Original Image
- Gaussian blur with std. deviation of 5
- Hypersphere projection with radius of 8
Application on drone videos

Video Capture with Drone

New Video

Face Detection

Face Recognition

De-Identification

[Diagram showing the process flow]
Face recognition/de-identification/privacy protection
UAV flight regulations in EU

- UAVs < 2kg are allowed within a 50m flight radius without professional pilot license.
- Pilot license and drone insurance are required for all professional applications.
- UAVs > 2kg of weight may be required to carry emergency parachutes (France).
- UAVs exceeding 15kg of weight might require special license or even be prohibited (Germany).
UAV flight regulations in EU

• Maximum flight altitude is typically restricted to 120m or 150m (400ft or 500ft) within several European countries.

• Line of sight must be maintained by the licensed pilot of the UAV at all times, either physically, or using visual aids (e.g., VR-goggles).

• Horizontal distance between the drone and the pilot is typically limited to specific meters (e.g., 500m).

• Outdoor UAV flight is restricted/prohibited above congested areas, crowds of people and airports, leading to permissible flight zones delineated by law.

• Inherently complying with such a complex and varying web of regulations (geofencing) is a challenge for all autonomous UAV applications (e.g., DJI app automatically downloads and determines permitted flight zones).
Other UAV safety issues

- Potential Landing Site Detection
- Crowd detection and avoidance
Mission simulations

• Simulations in Gazebo
• Simulations in Unreal Engine and AirSim
• Simulations for training data generation
Pilot Study - Test Content

Object Models

O1 Motorcycles

Background Environment

B1 Down Town

O2 Sports Car

B2 Industrial City
Test Sequence Example II: S2

VIDEO: Scenario 2 with drone height of 1, 2, 6, 10 and 14m.
Bibliography


Q & A

Thank you very much for your attention!

Contact: Prof. I. Pitas
pitas@csd.auth.gr