Advanced Guidance
(Guided Mode)

Leonard Hall
First a quick update

• Attitude control
  • thrust vector and heading control seems to be working well

• Separation of ACRO and PILOT parameters
  • Expo, Rate TC, Roll / Pitch / Yaw rates (no more Rate_P)

• S-Curves are in and operating well.
  • Fixed short waypoint problem in 4.2
  • Pause mission with calculation of tangential acceleration
  • Separate Corner acceleration – Should we have a parameter?

• Real time S-Curves
  • Reduce reaction time to small changes
  • Apply Real time S-Curves to Guided mode

• Position Control
  • Prioritize cross track error over tangential error
  • Removed override in precision landing (uses Real Time S-Curves)

• Remove Loiter from Takeoff and Landing
60° Turn

Snap = 5 m/s^4
Jerk = 1 m/s^3
Accel = 2 m/s^2
Vel = 10 m/s
Corner Rad = 5 m
Corner Accel = 2 m/s^2
$65^\circ$ Turn

Snap = 5 m/s$^4$
Jerk = 1 m/s$^3$
Accel = 2 m/s$^2$
Vel = 10 m/s
Corner Rad = 5 m
Corner Accel = 2 m/s$^2$
190° Turn

Snap = 5 m/s^4
Jerk = 1 m/s^3
Accel = 2 m/s^2
Vel = 10 m/s
Corner Rad = 5 m
Corner Accel = 2 m/s^2
190° Turn

Snap = 5 m/s⁴
Jerk = 1 m/s³
Accel = 2 m/s²
Vel = 10 m/s
Corner Rad = 5 m
Corner Accel = 4 m/s²
190° Turn

Snap = 5 m/s^4  
Jerk = 1 m/s^3  
Accel = 2 m/s^2  
Vel = 10 m/s  
**Corner Rad = 50m**  
Corner Accel = 4 m/s^2
**90° Turn**

- **Snap** = 5 m/s$^4$
- **Jerk** = 1 m/s$^3$
- **Accel** = 2 m/s$^2$
- **Vel** = 10 m/s
- **Corner Rad** = 20m
- **Corner Accel** = 2 m/s$^2$
90° Turn

Snap = 5 m/s^4
Jerk = 1 m/s^3
Accel = 2 m/s^2
Vel = 10 m/s
Corner Rad = 20m
Corner Accel = 4 m/s^2
Max Corner Acceleration

Is it worth an extra parameter?
How do we control an aircraft

• We provide directions......
  • What directions?

• Are the directions complete?
  • How do we handle the ambiguities?

• Do we provide instructions on HOW to get there?

• Highly dependent on aircraft dynamics!

• Users tend to be Narcissistic
  • They want a command to be only just complicated enough to completely handle their current problem.
  • All unspecified behaviour should support their current problem.
How do we control an aircraft

- Attitude
  - Roll, Pitch, Yaw
  - Roll, Pitch, Yaw Rate
  - Roll Rate, Pitch Rate, Yaw Rate

- Altitude
  - Throttle
  - Vertical Rate
  - Vertical Position

- What do we provide the User?
  - Users Answer: What I think I need before I work out that I need something else.
How do we control an aircraft

How do people think about providing directions?

• Destination and limits
  • Go to this location with a maximum speed of X
  • Planning is done in the AutoPilot
  • Simple for User but inflexible
  • What limits should be defined in the message?

• Position, Velocity, Acceleration
  • This is exactly what I want you to be doing right now
  • Planning is done by the User
  • Complex for User but flexible
The Micro Air Vehicle Link is a communication protocol for unmanned systems.

- LGPL license
- Vehicle agnostic

- Three general control commands:
  - SET_POSITION_TARGET_LOCAL_NED
  - SET_POSITION_TARGET_GLOBAL_INT
  - SET_ATTITUDE_TARGET

- Matching reply or status commands
- Potential to add additional commands
• Attitude quaternion (w, x, y, z order, zero-rotation is 1, 0, 0, 0)
• Body roll rate \( \text{rad/s} \)
• Body pitch rate \( \text{rad/s} \)
• Body yaw rate \( \text{rad/s} \)
• Collective thrust ArduCopter also supports vertical velocity
• 3D thrust setpoint in the body NED frame, normalized to -1 .. 1

• ATTITUDE_TARGET_TYPEMASK
  • Ignore body roll rate
  • Ignore body pitch rate
  • Ignore body yaw rate
  • Use 3D body thrust setpoint instead of throttle
  • Ignore throttle (could be used to select climb rate)
  • Ignore attitude
**SET_ATTITUDE_TARGET** ( #82 )

- Attitude + Throttle
- Attitude + Angular Velocity + Throttle
- Angular Velocity + Throttle
- Attitude + Vertical Velocity
- Attitude + Angular Velocity + Vertical Velocity
- Angular Velocity + Vertical Velocity

- What limits should the autopilot apply?
  - Vertical Velocity requires an angle limit
  - Throttle does not require an angle limit

- 6 Control Combinations
Attitude + Angular Velocity

• How can we specify both an attitude and an angular velocity?
  • The command specifies what the current state should be.
  • The command does not specify HOW to achieve the commanded state.

• Example: A person is facing north
  • Attitude only:
    • Face South
  • Attitude + Rate:
    • Face North turning CW at 10 deg/s
    • Wait 18 seconds
    • Face South turning 0 deg/s
SET_ATTITUDE_TARGET ( #82 )

Control input NOT supported by this Message:

• Partial attitude specifications:
  • Roll, Pitch + Rate Yaw

• Relative attitude changes:
  • Yaw by X degrees

• Altitude (Vertical control is not formally part of the message)

• Single axis rate commands (others must assume zero)

• Stabilize and Alt_Hold control is not possible with this message alone
• Provides a complete attitude control solution
• Provides a FAST response when angular velocity is provided
• Any attitude interface can be replicated with the right transformations.

• People keep discussing some sort of Stabilize or Alt_Hold instruction.
  • Is this simply the first thing people think of because it is how they control the aircraft?
  • Is there some real advantage for a companion computer, something I am missing?
### SET_POSITION_TARGET

**SET_POSITION_TARGET_LOCAL_NED ( #84 )**
- **X** m  X Position in NED frame
- **Y** m  Y Position in NED frame
- **Z** m  Z Position in NED frame

**SET_POSITION_TARGET_GLOBAL_INT ( #86 )**
- **lat_int**  X Position in WGS84 frame
- **lon_int**  Y Position in WGS84 frame

**Common**
- **vx** m/s  X velocity in NED frame
- **vy** m/s  Y velocity in NED frame
- **vz** m/s  Z velocity in NED frame
- **afx** m/s/s  X acceleration or force
- **afy** m/s/s  Y acceleration or force
- **afz** m/s/s  Z acceleration or force
- **yaw** rad  yaw setpoint
- **yaw_rate** rad/s  yaw rate setpoint

**POSITION_TARGET_TYPEMASK**
- Ignore position x
- Ignore position y
- Ignore position z
- Ignore velocity x
- Ignore velocity y
- Ignore velocity z
- Ignore acceleration x
- Ignore acceleration y
- Ignore acceleration z
- Use force instead of acceleration
- Ignore yaw
- Ignore yaw rate

**MAV_FRAME - LOCAL_NED**
- MAV_FRAME_LOCAL_NED = 1,
- MAV_FRAME_LOCAL_OFFSET_NED = 7,
- MAV_FRAME_BODY_NED = 8,
- MAV_FRAME_BODY_OFFSET_NED = 9

**MAV_FRAME - GLOBAL_INT**
- MAV_FRAME_GLOBAL = 0 or 5,
- MAV_FRAME_GLOBAL_RELATIVE_ALT = 3 or 6,
- MAV_FRAME_GLOBAL_TERRAIN_ALT = 10 or 11
SET_POSITION_TARGET

Navigation Control:
- Position, Velocity, Acceleration
- Position, Velocity
- Position
- Velocity, Acceleration
- Velocity
- Acceleration

Heading Control:
- Heading, Heading Rate
- Heading
- Heading Rate

- Separation of X, Y and Z axis may be achieved using the ignore flags
  - 108 combinations – 756 with frames

- No way to enable or disable stabilization of ignored axis
  - Stop Position Stabilization
  - Stop Position and Velocity Stabilization
  - 300 control combinations – 2100 with frames

- Does not specify limits or path when velocity and acceleration are ignored
  - Fastest?
  - Straight line?

- ArduCopter supports
  - 18 Basic Combinations
  - 36 including disabling XY Stabilization
  - 2 straight line options
Oh, I forgot to mention:

• We should be able to swap seamlessly between control systems!
• We want to be able to control yaw manually!
• Collision avoidance should work!
• We should not breach the fence!
• Should the reply message be:
  • What we sent,
  • Our current state?
  • Our current target state?
“But I think it should be............”
• We should be able to swap seamlessly between control systems!
• We want to be able to control yaw manually!
• Collision avoidance should work!
• We should not breach the fence!

• Handle collision avoidance at the position controller level
  • Bendy ruler algorithm may work effectively based only on the information in the position controller.
  • Fence limits may also be implemented at the Pos Control level.
  • This may generalise and extend these features to all Position controller-based modes. (Loiter, Follow, Pos Control .....)
Should the reply message be:

- What we sent,

OR

- Our current state?

OR

- Our current target state?

Time stamp may be used to differentiate.

We need some formal decision we can live with.

I would suggest

- what was sent with received time as time stamp
- Full message with current time stamp representing current target (not current state)
GET_POSITION_TARGET
GET_POSITION_TARGET

- Guided command
  - Last instruction from the companion computer

- Real Time S-Curves
  - Project last instruction forward in time
  - Remove integration errors
  - Generate a Jerk limited “Target” trajectory following the command

- Estimation
  - Combine sensor data to generate Pos, Vel, Accel

- Position Controller
  - Make the Estimation as close as possible to the Target
Guided mode

Pos Control Tools:
- input_pos_xyz
- input_pos_vel_accel (xy / z)
- input_vel_accel (xy / z)
- input_accel (xy / z)
- stop_pos_xy_stabilisation
- stop_vel_xy_stabilisation

Waypoint Navigation
- set_wp_destination

Pos Control Tools:
- Real time S-Curve trajectory
- Natural blending between commands
- Mix and match without discontinuity
- Path depends on state and limits

Waypoint Navigation
- Trigonometric S-Curve path
- Strict path following
- Initialised using stopping point
- Discontinuous if called when moving
Real Time S-Curves

• Accepts Commanded
  • Position
  • Velocity
  • Acceleration

• Given Limits
  • Velocity
  • Acceleration
  • Jerk

• Move Pos, Vel, Accel towards commanded Pos, Vel, Accel using a Kinematically consistent, Jerk limited path.

• Output is an Acceleration
Real Time S-Curves
Guided Mode Future Development

• Structure Guided mode handling of SET_POSITION_TARGET into vertical and horizontal functions
• Could we use SET_ATTITUDE_TARGET and SET_POSITION_TARGET together:
  • Attitude for thrust vector control.
  • Pos, Vel, Accel Z to control altitude.
  • Is it worth the complexity?
• Add a new message based on Destination + Limits
  • SET_ATTITUDE_LIMITS – Probably won’t help users
  • SET_POSITION_LIMITS – Potentially useful for users
  • Stabilize / Alt Hold equivalent command – hard to justify when SET_ATTITUDE_TARGET is so complete.
• LUA scripting
  • Great care needs to be used when defining interface functions.
  • Guided mode interface is limited by the Mavlink commands.
  • Direct access to AC_PosControl and AC_AttitudeControl is desirable
  • Custom LUA flight mode may be required for safe operation.
Looking forward to 4.3

• Ground / Air transition handling
  • Throttle time constant
  • Ground separation detection

• Complete Guided Mode implementation
  • Finalize and implement “GET” messages

• S-Curves
  • Add function to implement fast stop to assist WP based guided

• Follow Mode
  • Use Real time S-Curves to generate high-rate trajectory data

• Collision Avoidance Structure development
  • Support natively as some sort of Position Controller integration
Questions