About putting NPFG on Multicopter

Dec 15, 2022

PX4 Dev Summit Video

https://youtu.be/LY6hYBCdy-0?t=1472

Jerk-limited Trajectory

`VelocitySmoothing` library has a function to check what max velocity vehicle can have at a certain distance from the target. We can even use this for track-error boundary calculation (instead of depending on time-constant & ground speed based), to decide 'when' to start blending into moving in the direction of the path.

That function seems to be here:

```
/* Compute the braking distance given a maximum acceleration, maximum jerk and a maximum delay acceleration.
 * We assume a constant acceleration profile with a delay of accel_delay_max/jerk
 * (time to reach the desired acceleration from opposite max acceleration)
 * Equation to solve: vel_final^2 = vel_initial^2 - 2*accel*(x - vel_initial*2*accel/jerk)
 *
 * @param velocity initial velocity
 * @param jerk maximum jerk
 * @param accel maximum target acceleration during the braking maneuver
 * @param accel_delay_max the acceleration defining the delay described above
 *
 * @return braking distance
 */
inline float computeBrakingDistanceFromVelocity(const float velocity, const float jerk, const float accel,
                const float accel_delay_max)
{
        return velocity * (velocity / (2.0f * accel) + accel_delay_max / jerk);
}
```

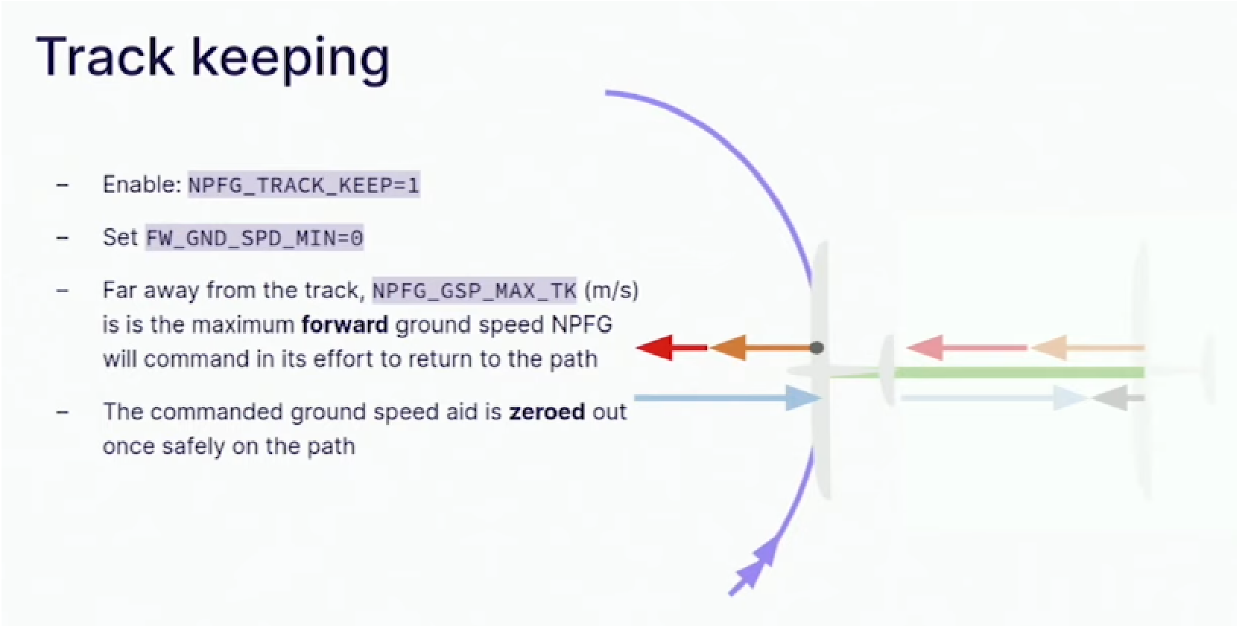
This relates to what david said about computing the ramp-in of path-parallel velocity component & ramp-out of path-orthogonal velocity component, as we approach the path, since we can dynamically calculate depending on the distance to path, a maximum velocity we can have towards the path, or vice versa.

There's another similar function (doing inverse), `computeMaxSpeedFromDistance` here:

```
/* Compute the maximum possible speed on the track given the desired speed,
 * remaining distance, the maximum acceleration and the maximum jerk.
 * We assume a constant acceleration profile with a delay of 2*accel/jerk
 * (time to reach the desired acceleration from opposite max acceleration)
 * Equation to solve: vel_final^2 = vel_initial^2 - 2*accel*(x - vel_initial*2*accel/jerk)
 *
 * @param jerk maximum jerk
 * @param accel maximum acceleration
 * @param braking_distance distance to the desired point
 * @param final_speed the still-remaining speed of the vehicle when it reaches the braking_distance
 *
 * @return maximum speed
 */
inline float computeMaxSpeedFromDistance(const float jerk, const float accel, const float braking_distance,
                const float final_speed)
{
        auto sqr = [](float f) {return f * f;};
        float b =  4.0f * sqr(accel) / jerk;
        float c = - 2.0f * accel * braking_distance - sqr(final_speed);
        float max_speed = 0.5f * (-b + sqrtf(sqr(b) - 4.0f * c));

        // don't slow down more than the end speed, even if the conservative accel ramp time requests it
        return max(max_speed, final_speed);
}
```

Track-keeping feature



It is noted that to enable track keeping, the ‘minimum ground speed’ needs to be set to 0 (thus, removing user-defined constraint), and the ‘maximum track keeping minimum ground speed’ needs to be set.

However, as shown in the `minGroundSpeed` function [here](#):

```
float NPFG::minGroundSpeed(const float normalized_track_error, const float feas)
{
	// minimum ground speed demand from track keeping logic
	min_gsp_track_keeping_ = 0.0f;

	if (en_track_keeping_ && en_wind_excess_regulation_) {
		// zero out track keeping speed increment when bearing is feasible
		// maximum track keeping speed increment is applied until we are within
		// a user defined fraction of the normalized track error
		min_gsp_track_keeping_ = (1.0f - feas) * min_gsp_track_keeping_max_ * math::constrain(
						 normalized_track_error / NTE_FRACTION, 0.0f,
						 1.0f);
	}

	// minimum ground speed demand from minimum forward ground speed user setting
	float min_gsp_desired = 0.0f;

	if (en_min_ground_speed_ && en_wind_excess_regulation_) {
		min_gsp_desired = min_gsp_desired_;
	}

	return math::max(min_gsp_track_keeping_, min_gsp_desired);
} // minGroundSpeed
```

Because of the `(1.0 - feas)` term, the track-keeping will never come into effect if the bearing is fully feasible (which is always the case in wind-less environment). This is also how it’s documented in the paper ‘On Flying Backwards’ as well:

B. Track Keeping

To further stay on track in excess wind speeds, an additional speed increment Δv_A^e corresponding to the normalized track-error $\bar{e}$ may be defined:

$$\Delta v_A^e = \Delta v_{A,\max}^e k_{\bar{e}} k_w \left(1 - \mathrm{feas}\left(\lambda, \beta\right)\right) \tag{34}$$

where $\Delta v_{A,\max}^e$ is the maximum allowed speed increment generated from track-error. The gains $k_{\bar{e}}$ and k_w are used to tune track-error and wind speed excess derived saturation ramps:

$$k_{\bar{e}} = \mathrm{sat}\left(\frac{\bar{e}}{\bar{e}_{\mathrm{buf}}}, 0, 1\right) \tag{35}$$

$$k_w = \mathrm{sat}\left(\frac{\Delta w}{\Delta w_{\mathrm{buf}}}, 0, 1\right) \tag{36}$$

$k_{\bar{e}}$ is scaled by a chosen fraction of the normalized track-error $\bar{e}_{\mathrm{buf}}$, setting the proximity at which $\Delta v_{A,\max}^e$ is applied in full, while k_w is scaled by Δw_{buf} to ensure no airspeed increment is applied in the condition that the feasibility function lies within the extended buffer zone below $\beta = 1$. The track offset -based speed increment Δv_A^e assists Δv_A^w by increasing the airspeed enough to overpower the current wind speed, returning the aircraft to the path, at which point the term again zeros out. With both increments in play, the augmented airspeed reference combines them as follows:

$$v_{A,\mathrm{ref}} = v_{A,\mathrm{nom}} + \min\left(\Delta v_A^w + \Delta v_A^e, \Delta v_{A,\max}\right) \tag{37}$$

Therefore, this would result in multicopter (with nominal airspeed set to 0), not coming close to the track, unless there is a wind, and bearing is not fully feasible.

To solve this, I think we can remove the feasibility consideration, as I believe even when the bearing is fully feasible, the ‘ideal’ minimum ground speed should still be commanded based on the track error.