The Influence of HDD Reference Frame and Path Prediction on Aircraft Taxi Performance

E.M. Reinhoudt
Delft University of Technology
Faculty of Electrical Engineering
Telecommunications and Traffic Control Systems Group

Abstract - In order to increase the passenger handling rate at airports, landing and departure separations of aircraft are being minimised. During low visibility conditions, a bottleneck emerges in handling aircraft on the ground because of highly decreased taxi speeds. This causes increasing delays, aircraft having to fly extra holding patterns and eventually forces arriving aircraft to divert to other airports. This is becoming a great economical and environmental burden.

Eagerness to solve this problem has resulted in world wide research on A-SMGCS systems. Such a system will enable pilots to safely navigate their aircraft at economical taxi speeds from strip to gate and back, under all visibility conditions.

In an effort to develop a cost effective solution for such a system, taxi performance tests have been carried out using a fixed base simulator with the Primary Flight Display as only guiding instrument.

Preliminary results show that the projection of either a 3D exo-centric or 3D ego-centric frame of reference combined with a path predictor, allows a significant increase in average taxi speed while maintaining an acceptable level of path following accuracy.

Professor: prof.dr.ir. D. van Willigen
Mentor: dr.ir. E. Theunissen
Period: September 1996 - June 1997
Date: August 10, 1997

Keywords - Advanced Surface Movement Guidance and Control System (A-SMGCS), Taxi Guidance, Path Prediction, Reference Frame, Taxi Performance

1. Introduction

The last few decades the need for air traffic capacity has increased rapidly. Airports all over the world are expanding to allow the ever increasing amount of passengers to travel comfortably, quickly and most of all safely to their designated destinations.
Minimising aircraft separations during landing and take-off procedures helps increasing an airports passenger handling capacity. Modern airports are equipped with high-tech navigation and ATC systems allowing aircraft to safely land and depart at a reasonable rate, even in low visibility conditions.

However, during low visibility conditions pilots have to reduce their taxi speeds in order to maintain an acceptable level of safety. This results in delayed runway clearances which causes major departure and arrival delays. If the reduced visibility conditions remain for a substantial period of time it can even cause arriving aircraft having to divert to other airports. The financial consequences of this bottleneck are obvious.

With the IMproved Airport GuidancE (IMAGE) project research is carried out to

develop an on board guidance and navigation system which is an essential element in any Advanced Surface Movement Guidance and Control System (A-SMGCS). The goal of the on board system is to provide pilots with the information needed to safely navigate their aircraft at economical taxi speeds from strip to gate and back.

NASA's Low-Visibility Taxi Operations Group has targeted 20 knots (≈ 10 m/s) as the ground speed necessary to achieve optimum traffic flow [6].

The Total System Error (TSE) on accuracy is defined as the difference between actual aircraft position and desired path [1]. The TSE is build up of a Path Definition Error, a Position Estimation Error and the Path Steering Error. The latter is the error dealt with in this research. The 95% TSE value (normal performance) for track widths of 23 meters is 2.2 meters. Half of that value goes to the account of the Position Estimation Error in conditions of Runway Visual Range (RVR) < 250ft. (75 m). This obviously puts a big constraint on the guidance accuracy of an A-SMGCS.

Within the IMAGE project a suitable display configuration has to be found to provide the pilots with all the necessary guidance information.

2. Aspects on A-SMGCS Display Design

To find an answer to the question: "What would be a suitable display configuration for an A-SMGCS?", this question first has to be placed into its context. What is considered to be a 'suitable' solution? Is this 'suitable' configuration feasible in terms of current technology, economical resources, ergonomical aspects and safety standards? In the sections below, some of these aspects will be discussed.

2.1 Integration

The A-SMGCS that is being developed will be put to use in aircraft approximately 5 to 15 years from now (1997). The air fleet at that time will mostly consist of types that are currently in service. This means that the on board system has to be integrated with the avionics existing in current aircraft. This aspect of retrofitting is a major design constraint.

The usage of, for instance, a Head Up Display (HUD) in an aircraft guidance system is not yet very common in civil aviation. One of the few carriers known to date to fly with this display is Alaska Airlines.

The Head Down Display (HDD), on the other hand, is already widely used in the Electronic Flight Instrument System (EFIS) of many commercial airliners. If the A-SMGCS would make use of a HDD solution, this obviously would make integration of the on board system more efficient. The taxi guidance system would then just add an extra piece of functionality to the EFIS.

With a HDD solution, the main question will be whether the display size is adequate for navigation and guidance purposes.

2.2 User Opinions

Interviews with several airline pilots revealed the preference of a conservative display arrangement for a taxi guidance system. That is, preserving conformity with both the Out-The-Window (OTW) scene and the (projected) instruments in the cockpit.

This would imply a combination of two displays. The first one consisting of a 2D (plan view) HDD which shows the airport map and route directions. The

second one would be a 3D, fully conform, perspective view HUD with projections of at least the track centerline, track edges, stop bars, ground speed and all potentially hazardous objects that have to be avoided. The 2D HDD would then replace the well known map-on-the-lap and covers the global awareness aspect. The 3D HUD would ideally project a perfect representation of the OTW-scene delivering the pilot situation awareness.

2.3 Design Aspects

Under the conditions in which the system will be used, typical RVR < 600ft. [9], it is first of all of vital importance that the pilot is able to react in time to, and is at all times aware of, the presence of ground traffic, stop bars and track signs. Secondly, there has to be a constant awareness of the plane's position, the direction in which the craft has to be directed and ground speed. If the pilot constantly has to switch between viewing points during the control task, scanning from head-up to head-down and back, precious time is lost that could and should be used for noticing possible hazards. Taking this into account, it is preferred to implement a 'single scan' (head up or head down) display configuration.

Since the reaction time of the pilot is of prime importance, the A-SMGCS display has to deliver its most relevant information in the shortest period of time possible. Therefore the display should not contain too much (complex) information. It must enhance a pilot's situation awareness, while minimally interfering with the taxiing task.

2.3.1 Aspects on HUDs

In their research, Lasswell & Wickens [6] state that if a HUD is used, it should be configured in fully conform view, projecting the taxi way and its edges at the exact location the pilot would see them in visual meteorological conditions (VMC).

A HUD obviously has the advantage of retaining OTW-view, allowing the pilot to use all the visual cues still available to safely navigate the aeroplane across the airfield.

The disadvantage is that the scanning problem remains as the global awareness still has to be obtained from the map-on-the-lap and the instruments have to be read head-down. This problem can be eliminated by also projecting route directions and crucial instruments, but this introduces the danger of clutter.

A general disadvantage of the current HUDs is their small viewing angle, which makes it difficult to obtain and follow flow-cues during low visibility conditions. The pilot using this HUD only obtains motion cues within the foveal Field Of View (FOV) and lacks peripheral sight.

2.3.2 Aspects on HDDs

A HDD configuration has the advantage of more easily being implemented in the EFIS. This also makes sense from an economical point of view presuming it would reduce the costs of the system.

With a HDD it is also allowable to present a non conformal frame of reference as there is no projection on and therefore no masking of the OTW-scene.

The major disadvantage of a HDD such as the PFD, is the small effective visual area. Within the next 15 years these displays are expected to increase in size, but most of them will not significantly exceed 8 inch squared. In modern aircraft, the size is only 6.5 x 6.5 inch. At a typical viewing distance of 80cm, this leaves a viewing angle of only 11.8 degrees, which is a major limitation to its capabilities of clearly presenting guidance information.

2.4 Results of previous research

A German research team successfully tested a system that uses two head-down displays of the Electronic Flight Instrument System (EFIS). One display projected the airport map while the other displayed a drive-director that indicated the advised ground speed and turning angle.

High speeds were combined with high levels of accuracy. However, collision avoidance was not taken into account.

Lasswell & Wickens [6] compared the effect on taxi performance of a 3D head up display (ego-centred frame of reference, cockpit view) and a 2D head-down display (exo-centric frame of reference, map view). They found that the highest taxi speeds were measured using the head-down 2D plan view display. If ground collision avoidance was also taken into account, the ego-centric HUD was slightly in favour with the researchers because of the increased situation awareness. The HUD also resulted in a slightly better performance on lateral errors.

A remark has to be made concerning the equipment that was used during these tests. High-performance graphics workstations (Silicon Graphics, Evans & Sutherland) were used to project both the head-up and the head-down images. Also, the screen sizes were relatively large. The 6m x 3m HUD, viewed from 3 meter distance, subtended an area of 24 degrees vertically and 40 degrees horizontally. Combined with the quality of the projected images, this setup compared to synthetic vision. The 16 inch HDD, viewed from 1 meter distance, covered viewing angles of 16.5 degrees vertically and 25.5 degrees horizontally. This is also quite large for normal cockpit operations. The symbology and images could therefore be displayed very clearly, which obviously improved the test results.

2.5 Summary & Hypotheses

Summarising all previous aspects, the most 'suitable' A-SMGCS display configuration consists of a head-down solution that delivers adequate situation awareness. From a financial as well as an ergonomical point of view, a single display configuration is preferred. It would be interesting to see whether the objectives, high taxi speeds, low error levels and obstacle detection could be met using only the PFD.

Grunwald [4] has shown that the use of a path predictor in perspective flight path displays reduces navigational errors significantly. In theory, this positive effect on accuracy in airborne applications should apply to the taxiing task as well. At the same time, this could partially compensate for the effects of the small spatial resolution of the PFD.

Further experiments concerning the frame of reference are a logical step in succession of the research of Lasswell & Wickens [6].

The advantages of a 3D ego-referenced viewpoint are its forward sight (preview) to infinity and the conformity with which the data is presented. Velocity cues are obtained through optic flow, just as the pilot would expect them. At a constant ground speed, objects accelerate

and increase in size as they approach the aircraft. Also, the spatial resolution close to the aircraft is large, which eases accurate local navigation.

The limited FOV is the major disadvantage of this reference frame. Threat detection is restricted to this area only, which makes collision avoidance with objects outside the FOV a harsh effort.

The disadvantage of the 3D ego-centred viewpoint, is the biggest pro of the 2D map view. The 360 degrees view of the ground plane delivers excellent situation awareness and threat detection capabilities (e.g. tail view). Furthermore, cross track errors are easy to detect as they occur in this plane of view.

The disadvantage is the non conformity of this reference frame. Velocity cues result from texture flow, which differs distinctively from the ego-referenced viewpoint. The dimensions and speed of the objects approaching the aircraft remain constant at constant ground speeds. Therefore, pilots will constantly have to make a mental transition in estimating the consequences of their speed. The second disadvantage is the trade-off between display range (adequate preview and tail view) and display resolution (adequate guidance accuracy).

A new reference frame combining the two previous viewpoints could well provide a suitable solution. This 3D view compromises between situation and global awareness and should cover practically all the advantages mentioned above.

The viewpoint is placed several meters behind and above the aircraft 'looking' diagonally down (Figure 1)

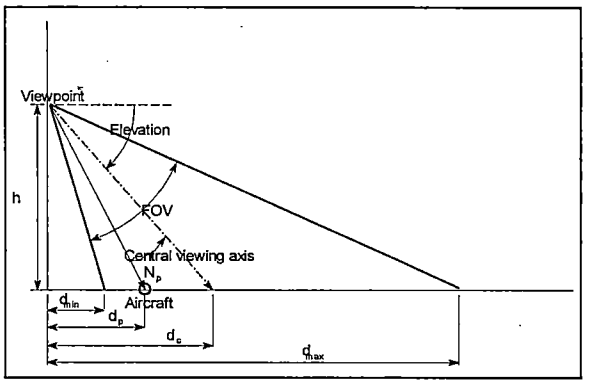

Figure 1. Spatial placement of viewpoint and viewing angles in 3D exo-centric view

This way an adequate preview, 360 degrees view around the aircraft and almost complete conformity are all combined in one frame.

As the effect on taxi performance of both the new reference frame and the taxi-way path prediction tool are practically unknown, in this research their influence will be measured. During these measurements, only the PFD is used as navigational and guidance instrument.

3. Experimental Design

To be able to compare the three different display configurations discussed in the previous chapter, they were tested to their effectiveness in a simulation procedure.

3.1 Measurement Techniques

A 3x2 ANalysis Of VAriance (ANOVA) Repeated Measures Within Subjects test [3] was set up. The two factors in these tests being the reference frame (3, 3D ego/3D exo/2D map) and the predictor (2, enabled/disabled).

Six students of the faculty of Electrical Engineering volunteered as subjects. The reason that no actual pilots were tested lies

in the fact that we just wanted to measure the influence on steering performance using the different display configurations, separate from any taxiing skills.

The main objective of the subjects was to follow a desired trajectory, which was indicated by means of a green centerline, as accurate as possible while maximising the ground speed of the aircraft.

A map, which contained part of Amsterdam's Schiphol Airport (Figure 2), was used to choose six equally 'difficult' routes, with a length varying between 4200 and 5900 meters. All tracks were made 23 meters wide.

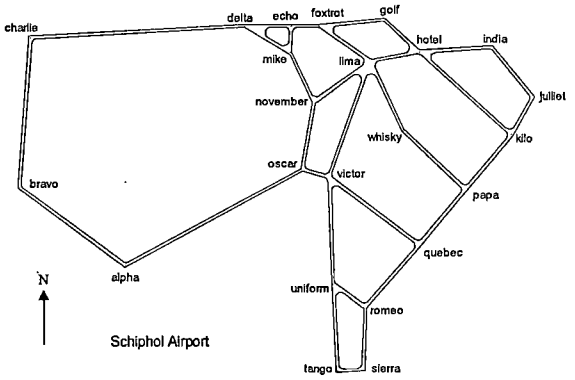

Figure 2. Map of Schiphol Airport used during tests

Subjects first completed a series of training sessions in which they drove with every display condition. This way the learning effect was mostly eliminated from the actual the experiment and subjects were able to familiarise themselves with each configuration.

During the actual experiment, the routes were driven in mixed order, each under every view and predictor condition. This way, each subject completed each track in every possible condition, each in a different order. Using this extensive method of 36 test runs per subject, all possible remaining learning effects between the subjects should average out in the results.

The data needed for the performance analysis was collected on each run at a rate of 37.5 Hz. It contained information on aircraft parameters and control inputs.

During these runs, the maximum occurring cross track error was continuously monitored. If the error exceeded 10 meters, the run was repeated. Subjects could indicate if they were disturbed in any way during a run. In that case too, the run was repeated.

The timetable of one complete test session is given in Table 1.

Table 1.
Typical Test Session Timetable

Action	Duration
Briefing & Training	60 minutes
Break	15 minutes
1st Test session, 12 runs	60-75 minutes
Break	15 minutes
2nd Test session, 12 runs	60-75 minutes
Break	15 minutes
3rd Test session, 12 runs	60-75 minutes

3.2 Display Configurations

To simulate the PFD, a monitor was covered to a size of 6.5 x 6.5 inch. The simulation program displayed anti-aliased images with a resolution of 480 x 480 pixels at an update rate of 37.5 Hz. Subjects sat at a distance of 0.75m - 1m from the display, looking approximately 10 degrees down (Figure 3).

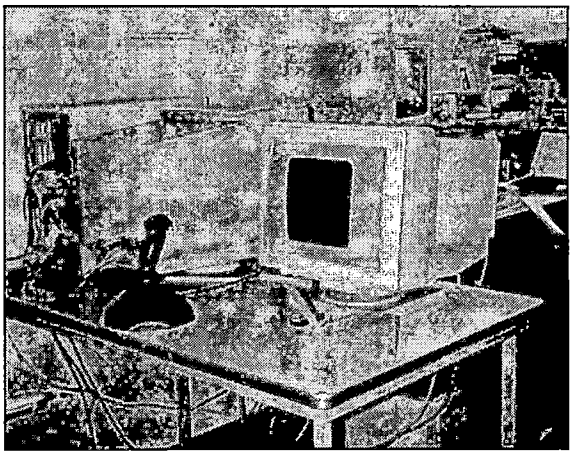

Figure 3. Experimental setup

To be able to compare the flow and position cues within each frame of reference, the gains for these cues will be calculated for each viewpoint.

3.2.1 Ego-referenced Perspective 3D View

The ego-centred frame of reference shows a simplified projection of the OTW-scene as seen from the cockpit (Figure 4). To preserve conformity with the OTW-scene, the projected FOV is 40 degrees horizontally and vertically. Additional symbology show a side stick position indicator (1), a horizontal heading tape with triangular heading indicator (2) and a vertical speed tape (3). Extra flow cues are given via the projection of taxi light poles (4). In the picture below, another aircraft crosses the subject's path (5).

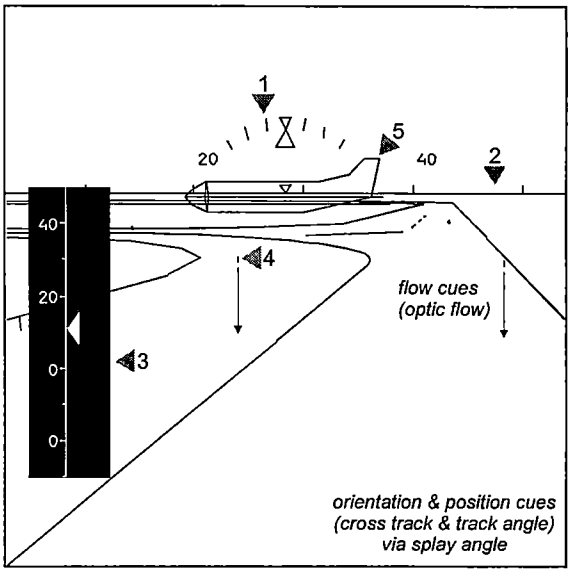

Figure 4. Ego-referenced perspective 3D view with cue indications

Wolpert et al. (1983) refer to the angle between the track edge lines and the line perpendicular to the horizon as the *splay angle* (Figure 5).

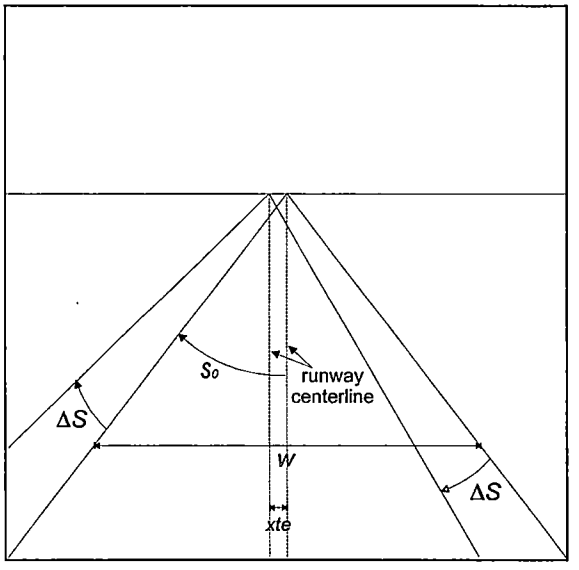

Figure 5. Influence of cross-track error (xte) on splay angle S_0

In the absence of errors the splay angle is equal to:

$$S_0 = \arctan\left(\frac{w}{2 \cdot h}\right) \quad (1)$$

w being the track width and h the (cockpit) viewing height.

Horizontal translations of the viewpoint result in changes in the splay angle. Theunissen [8], states that if the lateral position errors are small compared to the track width w, ΔS can be approximated by

$$\Delta S = xte \cdot \frac{K_{wh}}{w} \quad (2)$$

(clockwise rotation is positive). In this formula K_{wh} is the $w/2.h$ ratio. Further, the rate of change of the splay angle for lateral orientation errors is given by

$$\dot{S} = tae \cdot \frac{K_{wh} \cdot v}{w} \quad (3)$$

in which v represents the ground speed of the aircraft and *tae* equals the track angle error.

A new parameter is introduced that represents the level of conformity. It defined through the viewing or elevation angle θ (Figure 1) by

$$K_c = \cos(\theta) \quad (4)$$

Now all typical characteristics of the different viewpoints can be given. For the ego-centred viewpoint, they are listed in Table 2.

Table 2.
Overview of parameter gains and values in the ego-centred reference frame

Parameter	Gain / Value
conformity	G: $K_c = 1$
xte cue	G: $\frac{K_{wh}}{w}$
tae cue	G: $\frac{K_{wh} \cdot v}{w}$
speed cue	G: 1 (=optical flow cues)
preview	V: ∞ m
tail view	V: 0 m

3.2.2 2D Plan View

As a result of maintaining a FOV of 40 degrees, a chosen preview of 150 meters and a tail view of 50 meters the exo-centric 2D viewpoint (Figure 6) 'hovers' 284 meters directly above the aircraft. The projection shows a plane icon taxiing on a virtual map of the airfield. The speed tape is shown as the only additional instrument.

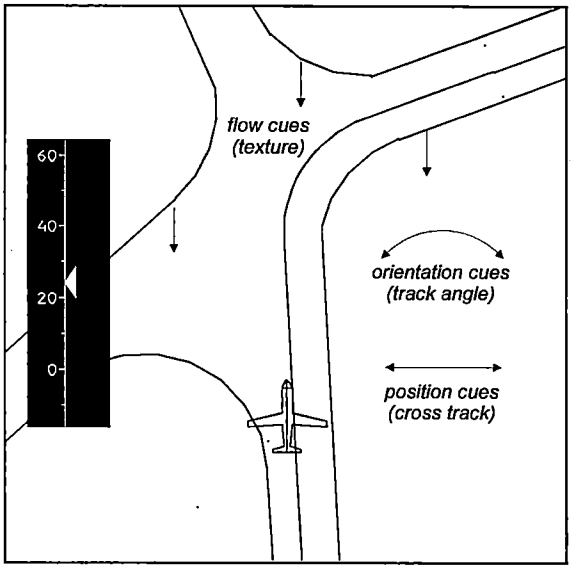

Figure 6. 2D plan view with cue indications

Table 3.
Overview of parameter gains and values in 2D plan view during tests

Parameter	Gain / Value
conformity	G: $K_c = 0$
xte cue	G: $\frac{1}{h}$
tae cue	G: 1
speed cue	G: 0 (= texture flow cues)
preview	V: 150 m
tail view	V: 50 m

Table 4.
Overview of parameter gains and values in 3D exo-centric reference frame

Parameter	Gain / Value
conformity	G: $K_c = 0.87$
xte cue	G: $\frac{K_c \cdot K_{wh}}{w} + \frac{1-K_c}{h}$
tae cue	G: $\frac{K_c \cdot K_{wh} \cdot v}{w} + (1-K_c)$
speed cue	G: K_c
preview	V: $d_{\max} - d_p = 375$ m
tail view	V: $d_p - d_{\min} = 30$ m

3.2.3 Exo-centric Perspective 3D View

The third view is the combination of the first two configurations. The FOV is maintained at 40 degrees which in combination with the chosen distance d_p and elevation angle of 30 degrees results in a viewpoint 'hovering' h=84 meters above and 100m behind the airplane (Figure 1&7). In this view, the speed tape is also the only additional instrument shown.

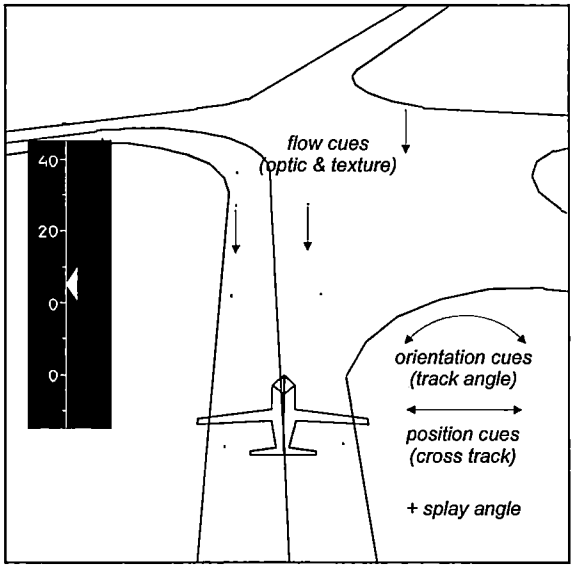

Figure 7. Exo-centric perspective 3D view with cue indications

3.3 Simulation aspects

During the display performance tests, the most common ground operations situations have to be simulated as real as possible. This means that ground operations procedures, dynamic scenery, aircraft controls and the dynamics of the aircraft need to be embedded in these tests.

3.3.1 Ground Operations Procedures

During low visibility conditions, navigation on the ground relies on carefully following 'on track' guiding signs, and obeying ATC-calls and light gun signalling [9,10]. Table 5 shows how these signals were implemented in the simulation program.

Table 5.
Ground Operations Signals

Signals at Airfields	Signals shown during Test Procedure
stop bars	stop bar projection
'on track' guiding signs and lights	taxi route centerline & stop bar projection
light gun signalling	ATC vocal cues and taxi route centerline & stop bar projection
ATC vocal guidance cues	taxi route centerline projection
ATC vocal 'HALT' cues	stop bar projection

3.3.2 Dynamic Scenery

During each run, 2 other aircraft also drove around the airfield (Figure 4), having intersecting paths with the subject's route. It depended on the subject's speed whether, when and where the aircraft met each other. Obviously, subjects were instructed to keep a safe distance and avoid colliding with these aircraft at all costs.

If a subject came within 200 meters of other aircraft, an event file was recorded. This file contained speed, position and control inputs of all aircraft involved in the 'event', which facilitated post experimental evaluation.

3.3.3 Aircraft Controls

In modern day aircraft, steering the nose wheel is either done by stick, by pedals or by wheel. Thrust level is always adjusted by levers and deceleration is obtained via thrust reversal and braking with the pedals. Nose wheel steering was implemented using the movement in the x-axis of an ordinary joystick. The y-axis was not used, dismissing any coupling between the two degrees of freedom of the joystick. The stick was placed front-left of the subjects, just like it is in a B737 next to the captain's seat.

A software based nose wheel controller (Appendix Z) was developed. This controller simulates the latency effects caused by the side-stick to nose wheel servo transition. The angular velocity of the nose wheel varied with time according to an S-curved profile (Figure 8).

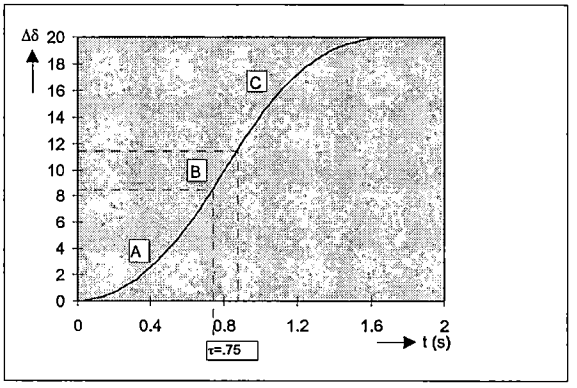

Figure 8. Change of nose wheel angle, $\Delta\delta$, for +20 degrees change of controller input.

The figure shows areas A, B and C. During the 'acceleration area' A, the angular velocity increases until it reaches its maximum value after τ seconds. In area B, δ changes at this constant, maximum rate. During the 'deceleration area' C, the angular velocity decreases to zero, leaving δ at the required angular change.

A separate throttle box was used to adjust the amount of thrust. The lever-box was placed front-right of the subject. During normal taxi operations, thrust reversal is seldom used and was therefore disabled in the simulation.

A pedal set accounted for the brakes. Asymmetric pushing of the two pedals during taxiing resulted in a slight rotational movement of the aeroplane. To prevent the subjects from using this feature for making small course corrections, a dead- zone &

step function was applied to the output (Figure 9).

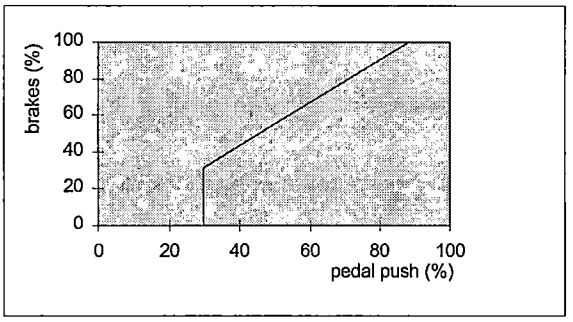

Figure 9. Brake level output

3.3.4 Aircraft Dynamics Model

To simulate the aircraft's dynamics on the ground a basic vehicle model (Appendix A) was adjusted to the weight and size of an average, mid-sized aeroplane (B737-400) to cover the inertia and handling aspects (Table 6).

Table 6.
Parameters in Aircraft Model [2]

mass	50000 kg
wheel track	12.45 m
wheel span	5.23 m
cockpit height	4 m
idle thrust	10000 N
max. thrust	195800 N
max. brake [5]	0.6 G m/s^2

3.4 Path Predictors

To allow pilots using the A-SMGCS to see the effect of their navigation actions in advance, a path predictor was implemented (Appendix Z).

This feature projects an arbitrary shape (e.g. arrow) in front of the aircraft describing the path the aircraft would follow if the control inputs were held fixed in their current positions. (Figure 10a+b,11,12).

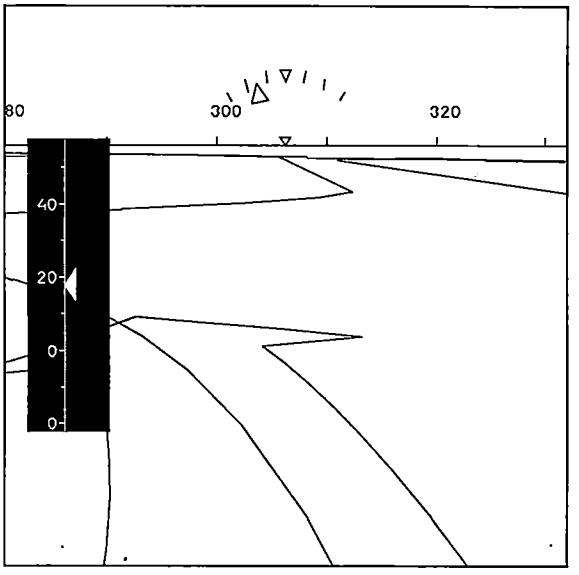

Figure 10a Predictor in Ego-centred 3D view

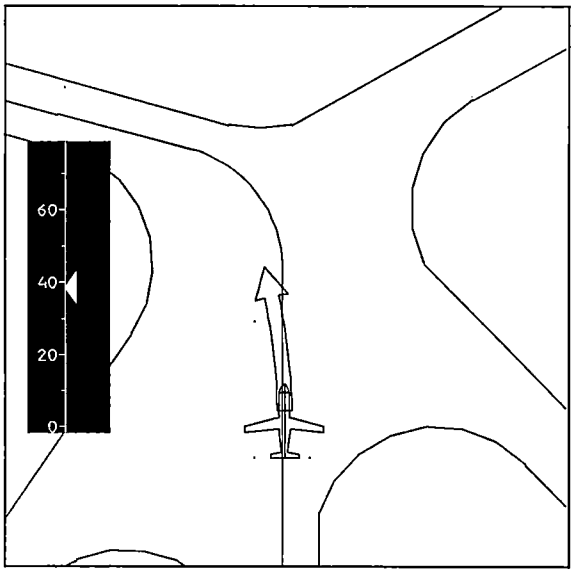

Figure 11. Predictor in 2D Map view

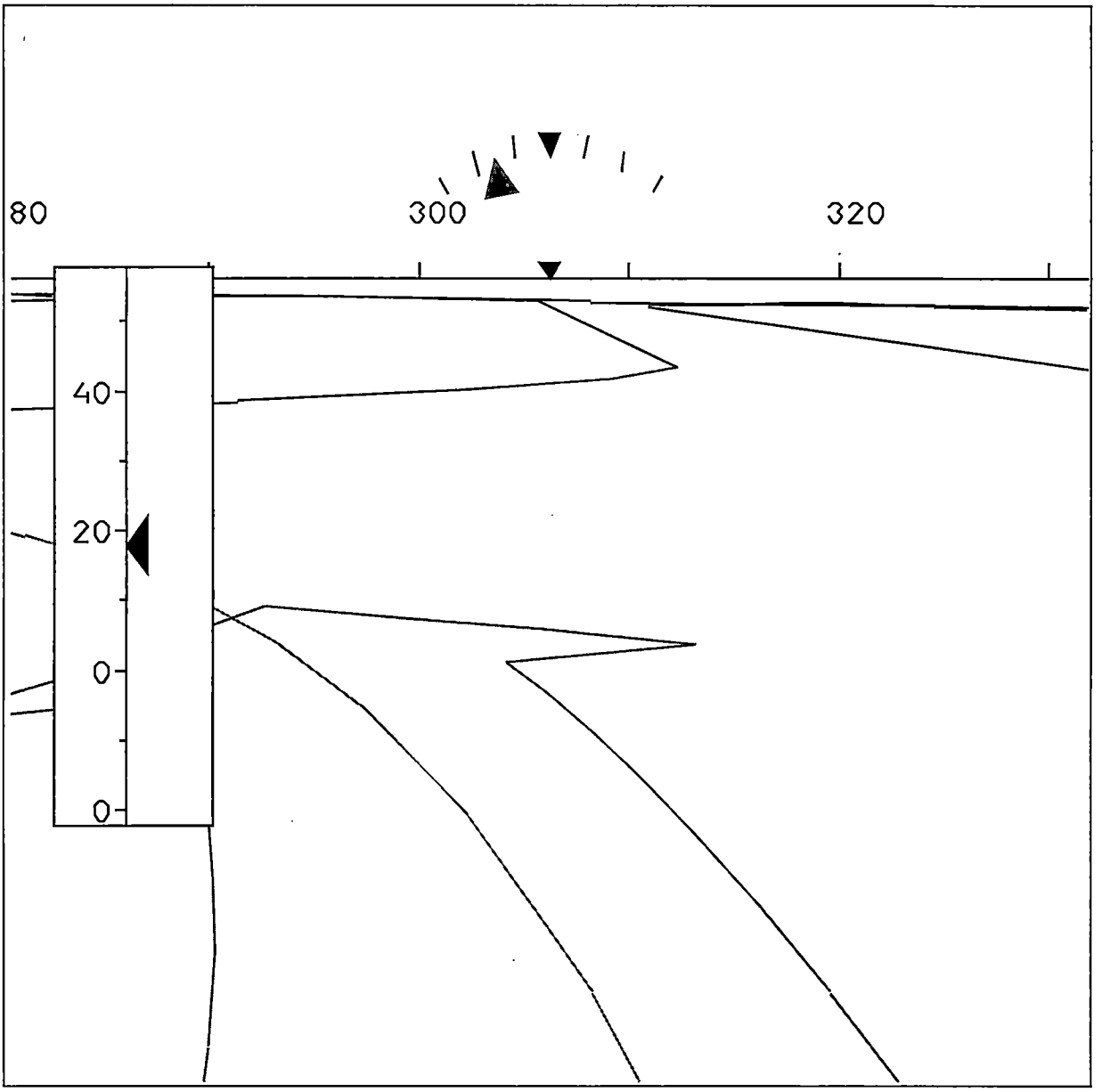

Figure 10b. Color coded, life size screen dump. (black=white, white=black, red=yellow)

Figure 10b shows a life size, colour coded screen dump of Figure 10. Some colours in this picture differ from the simulation display. In the picture black and white are inverted and red equals yellow.

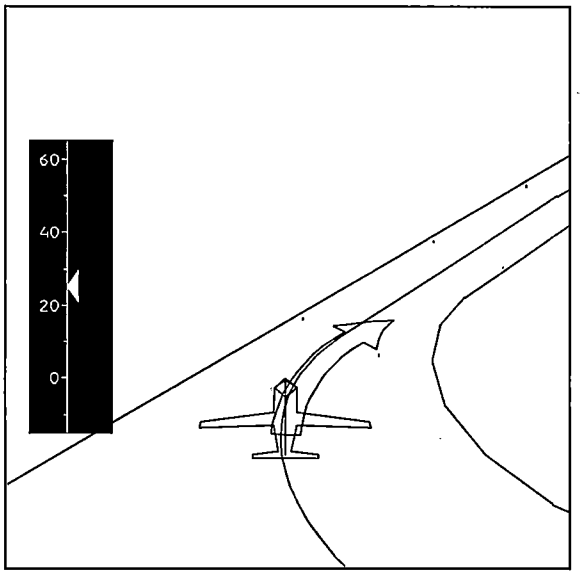

Figure 12. Predictor in Exo-centric 3D view

Two approaches towards the implementation of such a tool can be used. The first one consists of a 'fixed distance' predictor which calculates the path a fixed amount of metres in advance. The second one is a so called 'fixed-time' predictor which calculates the path a fixed amount of time in advance.

The implemented prediction tool is of the fixed time type. It was set to a 3 seconds prediction during simulation.

This type has been chosen because the length of the shape varies with the speed and the acceleration of the aircraft, delivering the pilot extra control cues. The disadvantage of this approach, the fact that the predictor will not be visible at speeds close to zero, only applies to the ego-centred reference frame. In that case, the predictor disappears underneath the aircraft. However, this problem is almost completely eliminated by implementing the side-stick position indicator in this viewpoint.

3.5 Computer

The simulation program was run on a 120 MHz Pentium with 32 MB of RAM. The computer was equipped with an ordinary PCI S3 1MB video card assisted by a Monster 3Dfx graphics accelerator board. This combination produced smooth graphics at a rate of 37.5 Hz.

4. Evaluation of Results

4.1 Performance Measures

The parameters listed in Table 7 were used as a basis of comparison between the performance of the different display configurations. As unit for the parameter control, 'step' has been chosen. This parameter represents the deflection of the joystick. The joystick had a resolution of 125 steps over the interval of [-62, +62].

The parameters xte and tae both represent the accuracy of taxiing, while control gives an indication of the control activity.

Table 7.
Overview of Parameters in Performance Analysis

Parameter	Unit	Explanation
xte_S	m	average absolute cross track error on straight segments
tae_S	deg	average absolute track angle error on straight segments
$speed_S$	m/s	average speed on straight segments
$control_S$	step	average absolute control input on straight segments
xte_C	m	average absolute cross track error on curved segments
tae_C	deg	average absolute track angle error on curved segments
$speed_c$	m/s	average speed on curved segments
$control_c$	step	average absolute control input on curved segments
std xte -	m	standard deviation of cross track error on complete run without prediction tool
std tae -	deg	standard deviation of track angle error on complete run without prediction tool
avg speed -	m/s	average speed on complete run without prediction tool
avg control -	step	average absolute control input on complete run without prediction tool
std xte +	m	standard deviation of cross track error on complete run with prediction tool
std tae +	deg	standard deviation of track angle error on complete run with prediction tool
avg speed +	m/s	average speed on complete run with prediction tool
avg control +	step	average absolute control input on complete run with prediction tool

4.2 Training Sessions

The training session covered between 6 and 8 runs. Each subject drove the even numbered runs without and the odd numbered runs with the use of the predictor. The reference frame and track number were randomised during training.

Figures 17-26 in Appendix B show the performance of the different subjects during these sessions. The training data of subject 3 was corrupted, so only 5 pair of graphs are shown.

The learning effect from run 1 to 4 is clearly visible for all subjects, especially in the non-predictor configurations.

It is also noticeable that subjects performed much better right from the start of the training if they were using the prediction tool.

4.3 Experiment Sessions

The files recorded during the test sessions were split into two separate files containing the data on straight segments and data on curved segments. First, a per-subject analysis was performed. Secondly, all data together was put through the ANOVA repeated measurements test.

4.3.1 Per Subject Performance Analysis

First the results of the six subjects will be compared separately. Figures 27-50 in Appendices C-F show the results of mean performance on the two crucial parameters xte and speed. Table 8 summarises those results.

Table 8.
Summary of overall performances per subject

Subject	Most Accurate	Highest Speed
1	ego + pred	map + pred
2	ego + pred	exo + pred
3	ego + pred	map + pred
4	ego	exo + pred
5	ego + pred	map + pred
6	ego + pred	map + pred

Without exception, the most accurate navigation is performed by all subjects using the ego-centred frame of reference.

The prediction tool proved to add extra accuracy to the performance of all subjects within this ego-centred frame of reference, with the exception of subject 4. Strangely, he stated that he found the predictor *confusing in corners but helpful on straights*, whereas the results show that especially on straight segments he performs worse using the predictor.

Within both exo-centric frames of reference, the positive or negative influence of the predictor is not directly evident from the graphs.

The influence of the prediction tool on the steering behaviour of subjects in turns is very much evident from Figures 13&14.

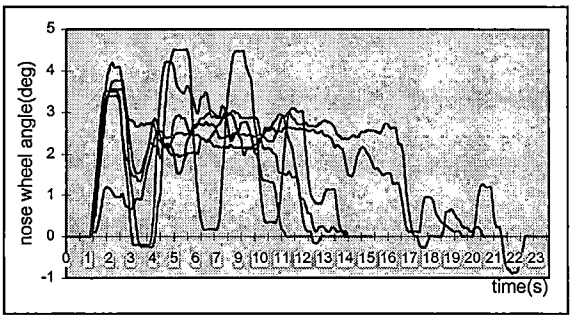

Figure 13. Time history of nose wheel deflection in one turn without predictor

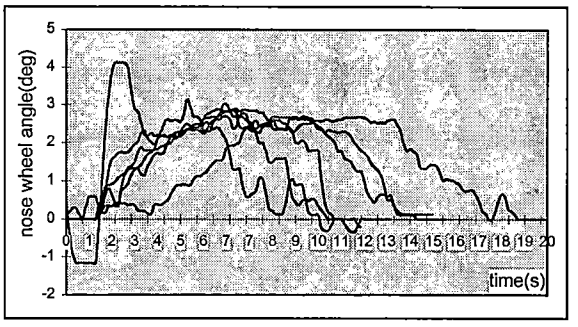

Figure 14. Time history of nose wheel deflection in one turn with predictor

The time history of the nose wheel deflection shows more fluent steering actions when the predictor is enabled. Both figures represent the same turn in an ego-centred viewpoint.

The highest speeds were measured in either one of the exo-centric views, all with the help of the prediction tool. Here too, a more distinctive conclusion cannot be made.

It is likely that the more realistic velocity cues in the ego-centric view caused the pilots to drive at a lower average speed, allowing them to increase tracking performance.

4.3.2 The Post Experimental Questionnaire

After their test runs, subjects were asked to fill out a questionnaire. They were asked for their preferred combination of predictor and frame of reference. Further they could comment on the simulation procedure as a whole.

Three subjects found the map view with predictor the most comfortable to drive. They all stated that the decreased tendency to correct small errors, due to the small spatial resolution, led to less steering actions. This made taxiing a *less stressful* task.

Two subjects preferred the ego-centred view because of the increased *realism factor*. One of these subjects preferred not to drive with the predictor. He found it to be *confusing in corners*. *"I tried to keep the arrow's point exactly on the centerline at all times and always came out wide"*.

Subject 3 preferred the exo-centric view with predictor. He liked the combination of *adequate preview and all-around-the-aircraft-sight* and stated that this view point was *ideal for avoiding other aircraft.*

The main comment all subjects gave, was the inability to avoid aircraft coming from either side of the aircraft in the ego-centred view.

4.3.3 Full Data Performance Analysis - The ANOVA Test -

In order to test both the prediction tool and the different reference frames on having significant effects on the parameters xte, tae, speed and control, the ANOVA test was deployed on all the data. After this test it should be possible to draw a well considered conclusion as to which reference frame / path predictor combination is preferred. Table 9 shows the outcome of the test.

Table 9.
Summary of ANOVA repeated measurements test.
Results that are significant (α=0.05) are accentuated in grey.

Parameter	Reference Frame	Predictor	Interaction
xte_S	F=39.36, p<0.0005	F=2.07, p=0.152	F=0.017, p=0.984
tae_S	F=20.39, p<0.0005	F=0.019, p=0.892	F=1.46, p=0.234
$speed_S$	F=26.48, p<0.0005	F=30.22, p<0.0005	F=1.70, p=0.185
$control_S$	F=3.06, p=0.048	F=3.813, p=0.052	F=1.48, p=0.229
xte_C	F=3.36, p=0.036	F=21.76, p<0.0005	F=33.34, p<0.0005
tae_C	F=13.02, p<0.0005	F=12.95, p<0.0005	F=8.71, p<0.0005
$speed_C$	F=23.96, p<0.0005	F=120.46, p<0.0005	F=7.83, p<0.0005
$control_C$	F=19.05, p<0.0005	F=38.25, p<0.0005	F=14.39, p<0.0005

- Straight Segments

The figures 51-54 in Appendix G show the results on the first 4 parameters of Table 9.

There is a significant effect of the reference frame on all 4 parameters. The predictor only affects $speed_S$. In all cases, there is no interaction between the two factors.

As can be seen in Figures 51&52 the xte_S and tae_S increase significantly as the level of exo-centrism increases. This effect is most probably due to the decrease in the xte and tae cue- gain.

Figure 53 shows that the average speed is much higher in both exo-centric views than in cockpit view. This can be explained by the non-conformity of the speed-cues within these reference frames. Almost all subjects commented that their experience of speed was much less in these views. This led to a tendency to drive faster. The fact that the peak does not lie in the 2D reference frame, can be led back to the small preview.

The predictor was obviously also used as a strong cue for $speed_S$ (and braking distance!), explaining the overall significantly higher average speeds.

Figure 54 shows that the control activity of the subjects depended very much on the reference frame, especially when the predictor was used (shows in Table 9 as almost being significant). The control activity in the exo-centric viewpoint was highest, while its value in

cockpit and map view was very much lower. There are several explanations to the fact that control activity in cockpit view is lower. One of them could be that the smaller position and orientation errors causes less steering actions. Another one is that due to the large spatial resolution close to the aircraft, steering actions are performed more fluent which leads to a smaller average control activity. The lower control activity in map view can be led back to the small spatial resolution (xte and tae cue gains), which decreases the tendency to correct small tracking errors.

- Curved Segments

The Figures 55-58 in Appendix H show the results on the last 4 parameters of Table 9.

There is a significant effect of both reference frame and predictor on all 4 parameters. Furthermore, there is an interaction between the two factors on all 4 parameters.

In the case of the xte_C, Figure 55 shows that the influence of the predictor seems to be caused only by the condition of the ego-centred viewpoint. This explains their interaction. Clearly, the smallest xte_C is achieved in the ego-centred viewpoint with the help of the predictor.

It seems that the same effect appears with the tae_C, only here the performance without the predictor looks to be 'inverted' to that of the xte_C. Also, the effect of the predictor is now caused by two reference frames, namely the ego-centric and the map view. Nevertheless, the outcome is the same as with the xte_C: ego-centric view plus predictor performs best.

The interacting effect between reference frame and predictor on $speed_C$ is caused by the exo-centric 3D viewpoint (Figure 57), where the highest speeds are measured with prediction enabled. This was also the case on straight segments and is therefore not surprising. The overall positive effect of the prediction tool is again very clear.

Figure 58, which shows the parameter $control_C$, is an interesting one. The opposite of the effect on straight segments occurs: In the second frame of reference and with the prediction tool, the least control activity is shown. This is also the main cause of the interaction between the two factors.

CONCLUSION: If a high level of accuracy has to be combined with acceptable speeds, the choice will be the ego-centred reference frame. If speed is more important, the exo-centric 3D viewpoint is the best alternative frame of reference. In either case the path prediction tool increases performance.

The PFD proved to be an adequate taxi guidance tool, combining both speed and accuracy aspects.

4.3.3 Obstacle Avoidance Performance

- Stop Bars

The stop bar projection delivered an adequate cue to all subjects. Every time it was projected, a safe distance to the bar was attended and the aircraft was successfully brought to a safe stop.

- Collision Avoidance

Within every reference frame it seemed very difficult at some points to avoid other aircraft crossing the subject's path. Especially vehicles approaching from either left or right of the aircraft often caused major problems. The ego-referenced view performed worst in this case, which could be expected from its limited FOV. The best collision avoidance performance was achieved in the 3D exo-centric view.

CONCLUSION: Collision avoidance is not adequately supported by an on board taxi guidance system that uses only the PFD. Other solutions have to be sought for to keep aircraft at a safe distance from each other and other ground obstacles.

5. Conclusions and Recommendations

5.1 Conclusions

- The results of the experiment indicate that the main goals of the on board taxi guidance system, economical speeds and high levels of accuracy, can be achieved by only using the PFD.
- Collision avoidance needs further investigation, as the PFD proved to perform inadequate on that subject.
- The ego-centred frame of reference proved to be the most accurate head-down single-display solution. This is also the frame of reference pilots wish to see in a future A-SMGCS.
- The exo-centric 3D view delivered the highest values of taxi speeds and presented a fruitful combination of situation and global awareness to the subjects.
- The path prediction tool has definitely proven its worth in its non-airborne form and will be implemented in the on board system.
- The projection of stop bars and holdings has proven to be a perfect cue.

5.2 Recommendations

- Actual testing in either a flight simulator or test aircraft will have to reveal the real potentials of the one-display configuration. During these tests the performance in both the 3D views (ego-centric and exo-centric) will have to be measured to be able to draw a well considered conclusion in the trade-off between economical speeds and necessary levels of accuracy.
- A solution to the collision avoidance problem could be the usage of a second display. This display could present a (zoomable) map of the airfield showing all (moving) obtacles. This information will have to be provided via datalink. This second display could either be interpreted by the captain or the co-pilot.
- Another collision avoidance solution could be a non-airborne version of TCAS.

References

[1] Cassel, R. et al., *Airport Surface RNP (Required Navigation Performance) - Implications for GNSS.* Rannoch Corporation, Alexandria, VA.

[2] Gillespie, T.D., *Fundamentals of Vehicle Dynamics.* Society of Automotive Engineers, Warrendale, PA, 1992.

[3] Girden, Eilen R., *Anova; repeated measures.* Sage University Papers, Sage, Newbury Park, 1992.

[4] Grunwald, A.J., *Tunnel Display for Four-Dimensional Fixed-Wing Aircraft Approaches.* Journal of Guidance, Vol 7., No 3., pp. 369-377, 1984.

[5] Jane's Information Group, *International ABC aerospace directory.* Interavia, Geneva, 1936.

[6] Lasswell, J.W. and C.D. Wickens, *The Effects of Display Location and Dimensionality on Taxi-Way Navigation.* Technical Report prepared for NASA Ames Research Center, Moffet Field, CA, May 1995.

[7] Levine, O.H., *Effect of Visual Display Parameters on Driving Performance in a Virtual Environments Driving.* M.Sc. Thesis presentation, North-eastern University, Boston, Massachusetts, August 31, 1995.

[8] Theunissen, E., *Integrated Design of a Man-Machine Interface for 4-D Navigation.* PhD Thesis, Delft University Press, 1997.

[9] U.S. Department of Transportation & Federal Aviation Administration, *Advisory Circular: Surface Movement Guidance and Control Systems.* December 19, 1996.

[10]U.S. Department of Transportation & Federal Aviation Administration, *Airport Markings, Signs and Selected Lighting - introducing SMGCS.* May 1995.

[11]Wolpert, L.,et al., *The isolation of optical information and its metrics for the detection of descent.* Optical flow and texture variables useful in simulating self motion (II), Columbus, OH, 1983.

Appendix A. Aircraft Dynamics Model

The first step in setting up the model is to look at all forces effecting the aircraft's motion [7]. Summing up all forces gives the total effective force on the craft (1).

$$\mathbf{F}_{\mathbf{eff}} = \mathbf{F}_{\mathbf{thrust}} + \mathbf{F}_{\mathbf{brake}} + \mathbf{F}_{\mathbf{drag}} + \mathbf{F}_{\mathbf{roll}} \qquad (5)$$

As those forces only apply to one dimension (direction of drive), the simplified scalar form will be used.

The drag force and rolling resistance force on a vehicle are defined by:

$$F_{drag} = -\text{sign}(v) \cdot c_{drag} . v^2 \qquad (6)$$

$$F_{roll} = -\text{sign}(v) \cdot c_{road} \cdot m \cdot g \qquad (7)$$

The value of c_{road} was set to a value of 0.02, which is an empirical factor for dry paved roads. Because only low speed navigation was involved the parameter c_{drag} was set to 0 during tests.

The next step describes the rotational motion of the craft. This is implemented via the change in effective nose wheel angle δ_{eff} (4).

$$\delta_{eff} = \delta_{servo} + \delta_{Fangle} \qquad (8)$$

In this addition δ_{servo} represents the value that comes from the nose wheel controller algorithm. The other, δ_{Fangle}, is a result of applying different forces to the two brake pedals (Figure 15).

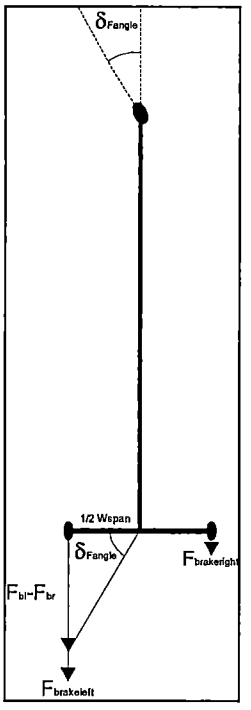

Figure 15. Asymmetric braking with c_{asym}=1

This asymmetric braking affects the rotational motion of the plane. The angle is calculated via

$$\delta_{Fangle} = \arctan\left(\frac{c_{asym} \cdot \left(F_{brakeleft} - F_{brakeright}\right)}{\frac{1}{2} \cdot wheelspan}\right) \qquad (9)$$

in which c_{asym} represents the sensitivity of asymmetric braking. It was set at an empirical value of 0.03125 during the experiments.

To be able to calculate the change in direction, the radius of turn must be calculated (Figure 16).

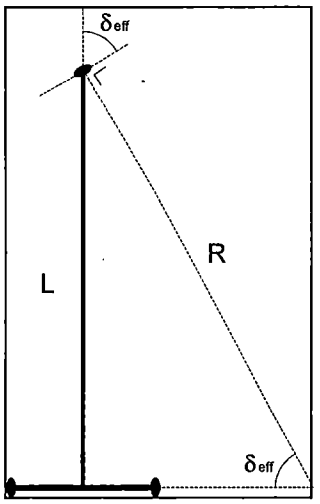

Figure 16. Calculation of radius R

$$R = \frac{L}{\sin(\delta_{eff})} \tag{10}$$

The length L equals the wheel track of the aircraft.

To calculate the new position of the aircraft, the time axis has been discretised. The new compass heading follows from the previous heading angle by (11).

$$\gamma_{heading} = \gamma_{old} + \frac{360 \cdot \sqrt{(x - x_{old})^2 + (y - y_{old})^2}}{R} \tag{11}$$

In this formula the approximation has been made that for small position changes, the arc connecting the points ($x_{old}y_{old}$) and (x,y) has the same length as the straight line between these points.

The last step involves the calculation of the new (x,y) position. Basic physics involving acceleration a and speed v are used (12-15).

$$a = \frac{F_{eff}}{m} \tag{12}$$

$$v = v_{old} + a \cdot T \tag{13}$$

$$x = x_{old} + \left(v_{old} \cdot T + \tfrac{1}{2} \cdot a \cdot T^2\right) \cdot \cos(\delta_{eff}) \tag{14}$$

$$y = y_{old} + \left(v_{old} \cdot T + \tfrac{1}{2} \cdot a \cdot T^2\right) \cdot \sin(\delta_{eff}) \tag{15}$$

The position and prediction calculations in the program were synchronised with every two refreshes of the simulator screen. This way, a constant sample time T was obtained:

$$T = \tfrac{2}{75Hz} \approx 27ms \tag{16}$$

The software implementation of this model is described in Appendix Z.

Appendix B. Results of Training Sessions

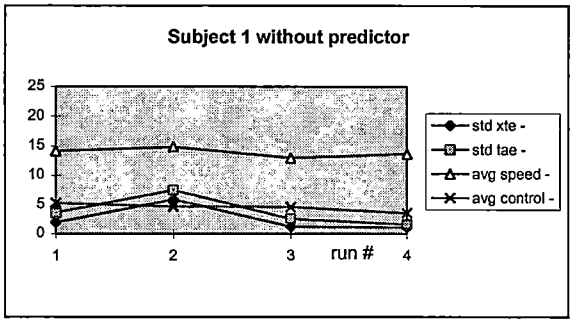

Figure 17. Performance of subject 1 without predictor

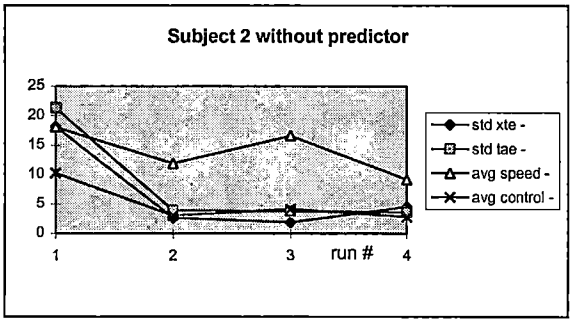

Figure 18. Performance of subject 2 without predictor

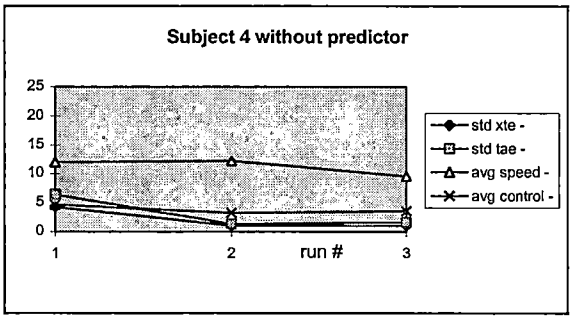

Figure 19. Performance of subject 4 without predictor

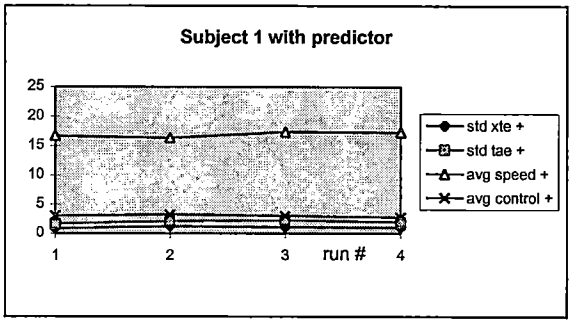

Figure 20. Performance of subject 1 with predictor

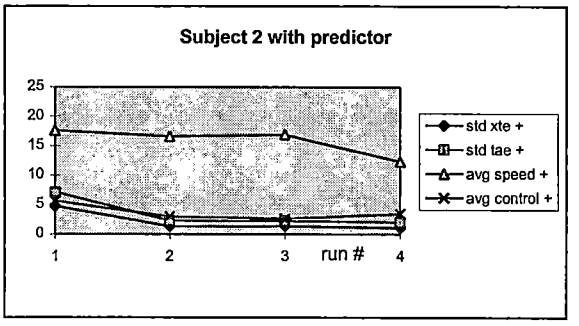

Figure 21. Performance of subject 2 with predictor

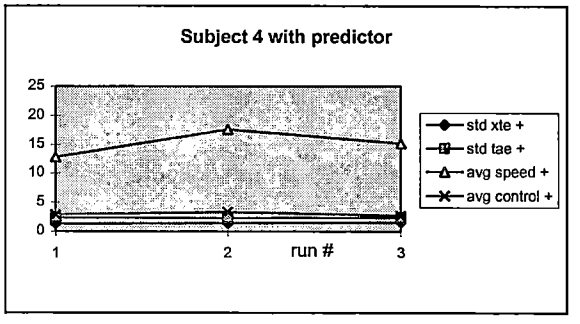

Figure 22. Performance of subject 4 with predictor

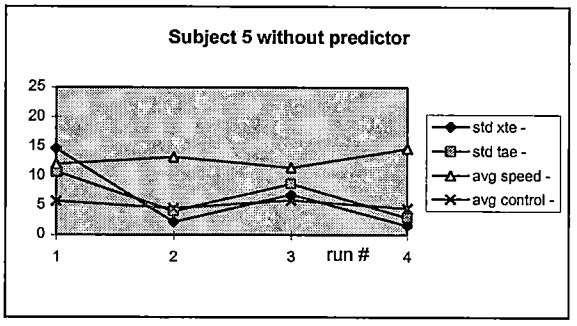

Figure 23. Performance of subject 5 without predictor

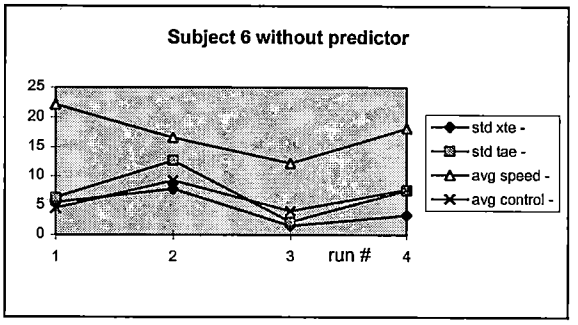

Figure 24. Performance of subject 6 without predictor

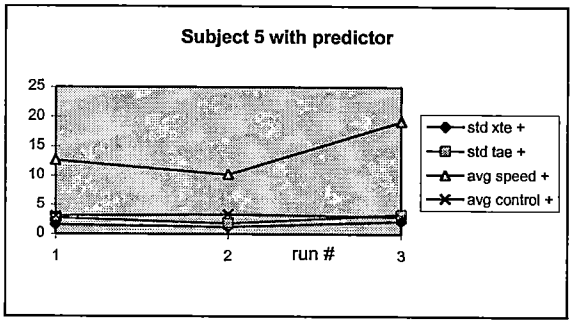

Figure 25. Performance of subject 5 with predictor

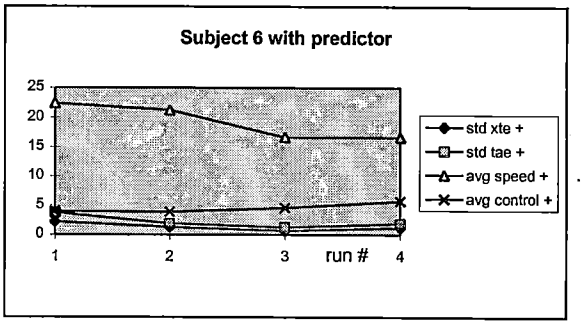

Figure 26. Performance of subject 6 with predictor

Appendix C. Results on XTE_s for separate subjects

REFFRAME 1=ego-centred
2=3D exo-centric
3=2D exo-centric (map)
PRED 1/2 = predictor disabled/enabled

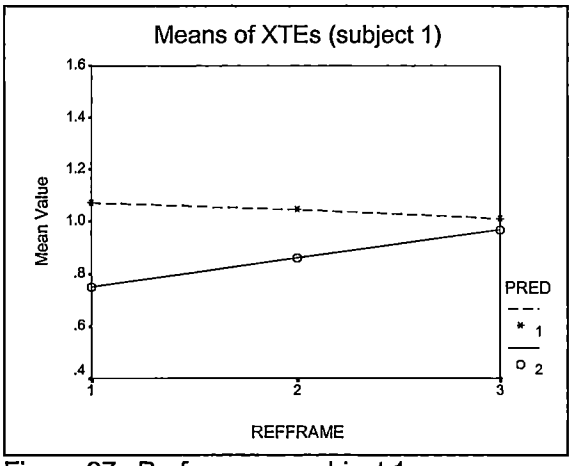

Figure 27. Performance subject 1

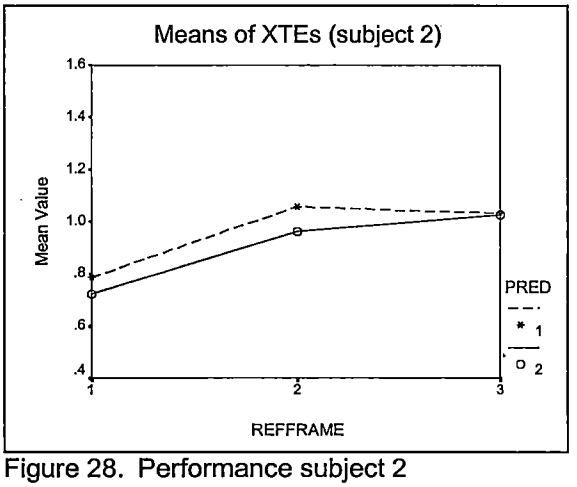

Figure 28. Performance subject 2

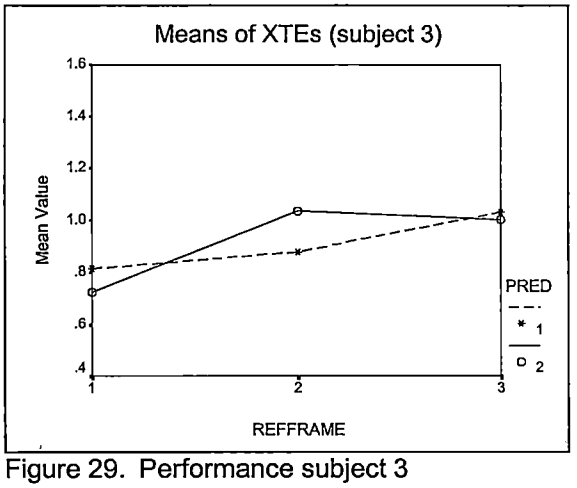

Figure 29. Performance subject 3

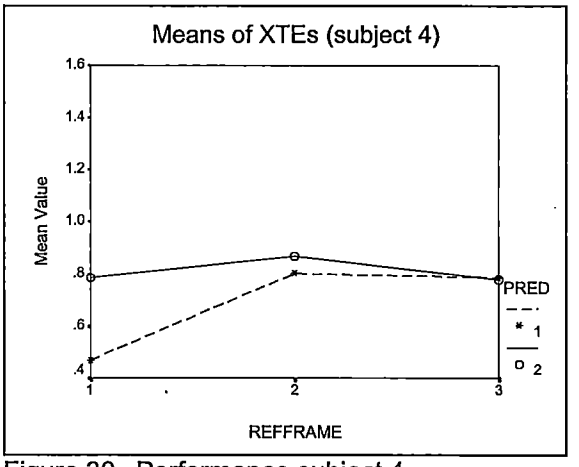

Figure 30. Performance subject 4

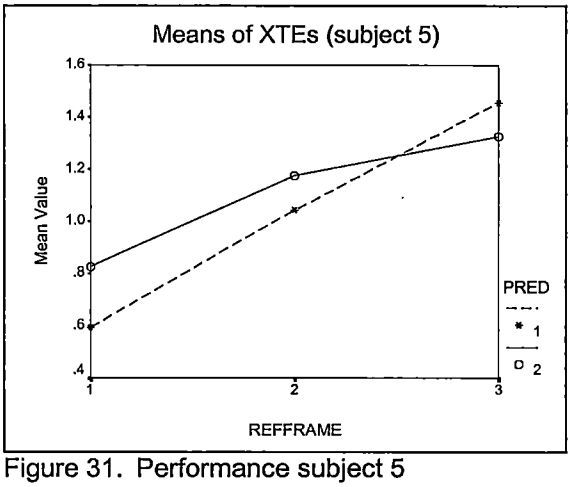

Figure 31. Performance subject 5

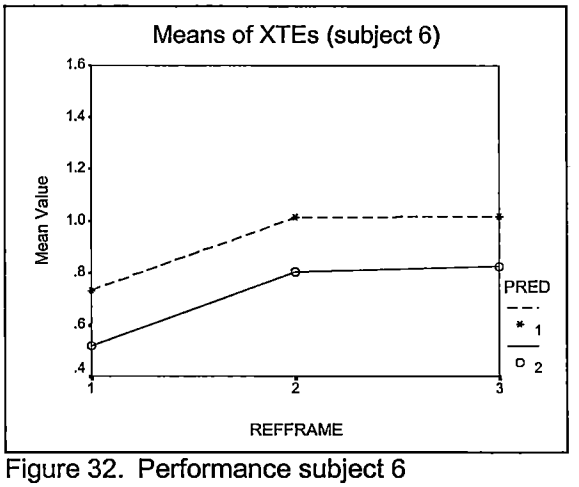

Figure 32. Performance subject 6

Appendix D. Results on SPEED$_s$ for separate subjects

REFFRAME 1=ego-centred
2=3D exo-centric
3=2D exo-centric (map)
PRED 1/2 = predictor disabled/enabled

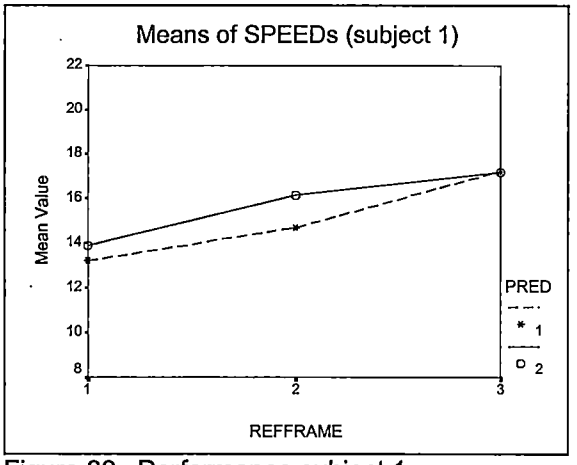

Figure 33. Performance subject 1

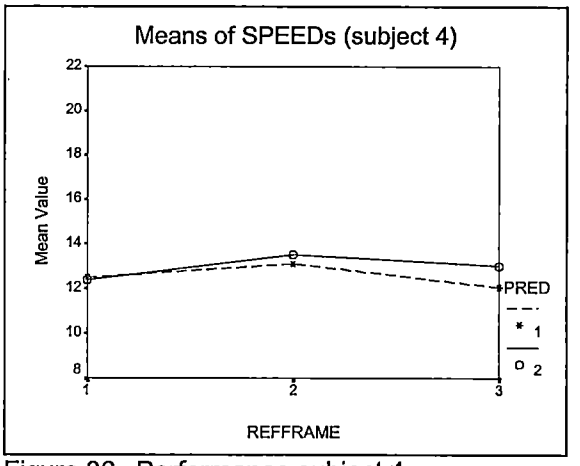

Figure 36. Performance subject 4

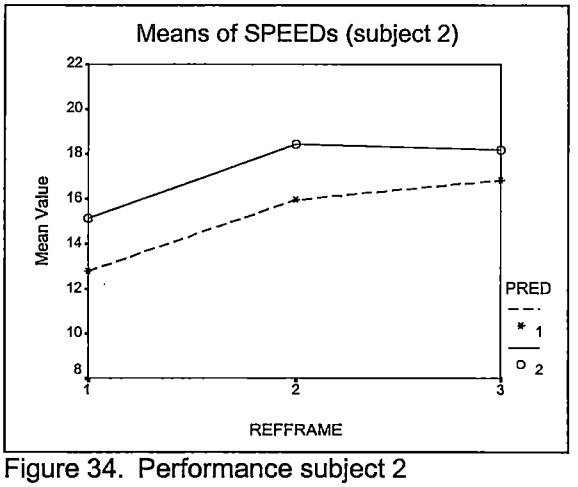

Figure 34. Performance subject 2

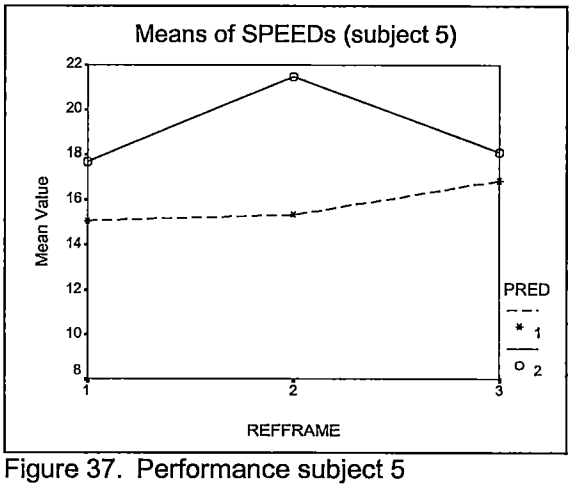

Figure 37. Performance subject 5

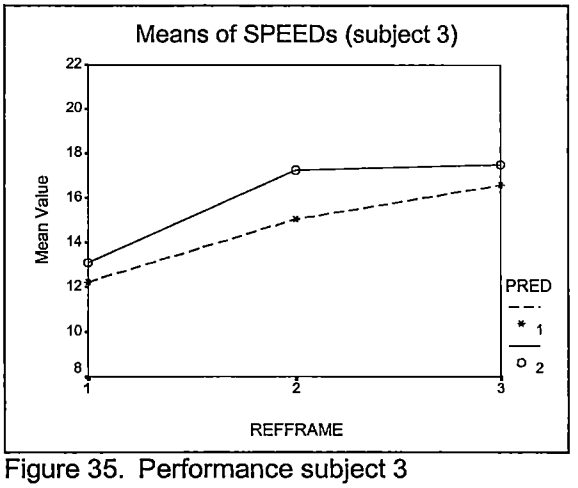

Figure 35. Performance subject 3

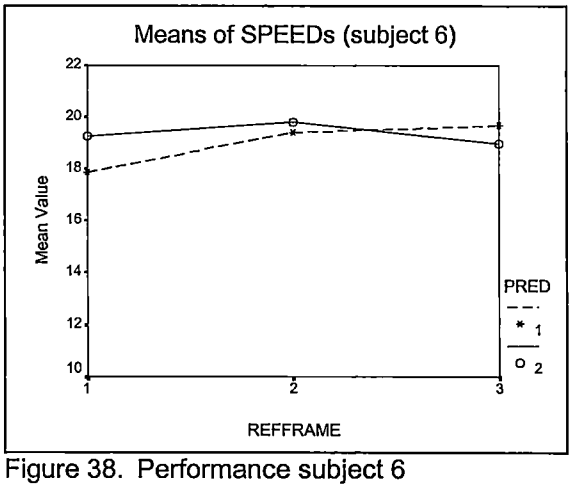

Figure 38. Performance subject 6

Appendix E. Results on XTE_c for separate subjects

REFFRAME 1=ego-centred
2=3D exo-centric
3=2D exo-centric (map)
PRED 1/2 = predictor disabled/enabled

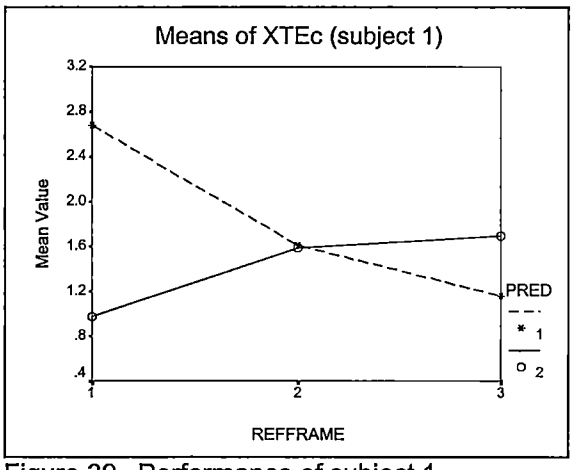

Figure 39. Performance of subject 1

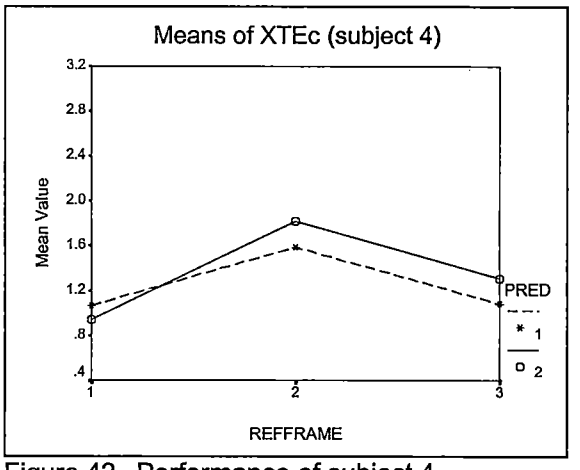

Figure 42. Performance of subject 4

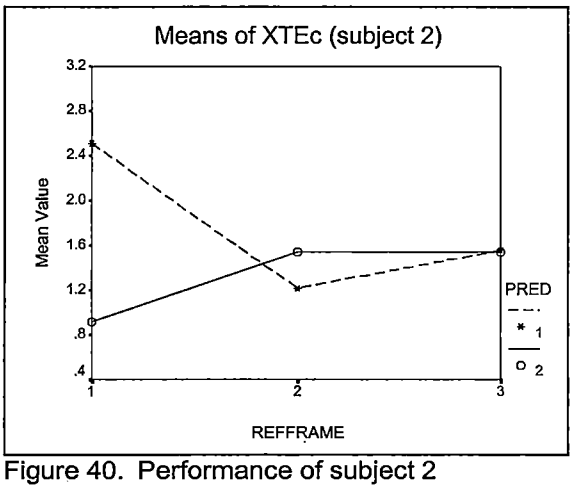

Figure 40. Performance of subject 2

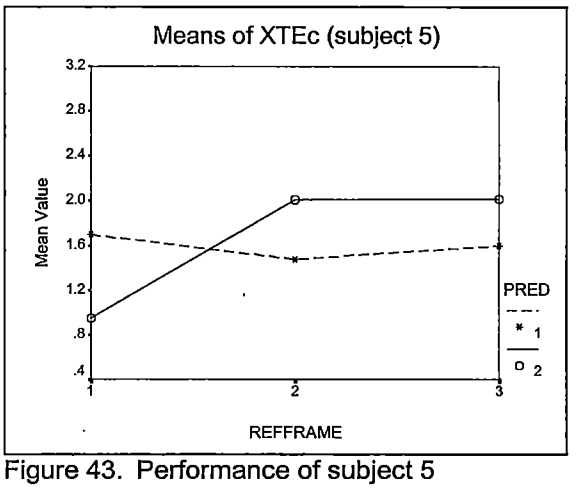

Figure 43. Performance of subject 5

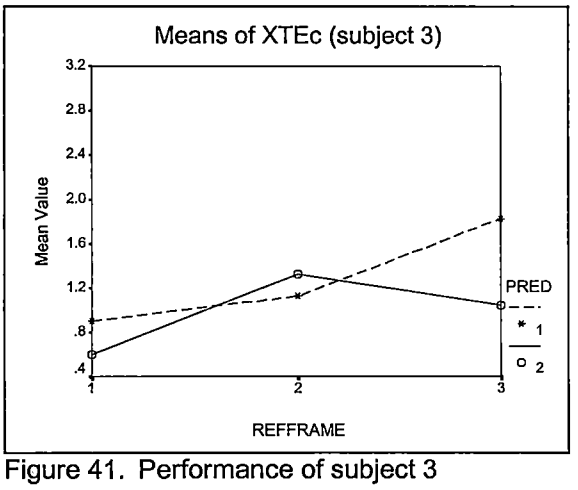

Figure 41. Performance of subject 3

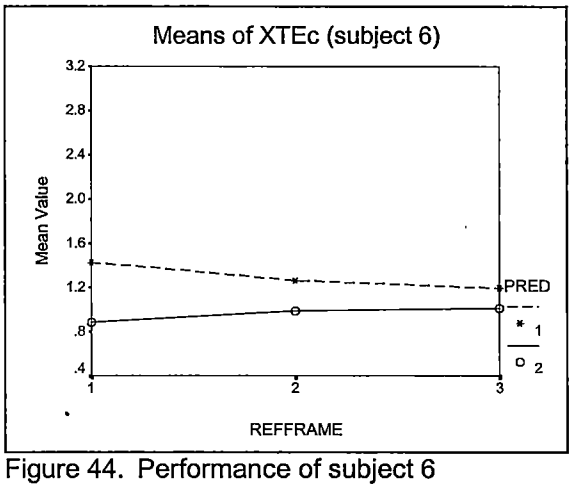

Figure 44. Performance of subject 6

Appendix F. Results on SPEED$_c$ for separate subjects

REFFRAME 1=ego-centred
2=3D exo-centric
3=2D exo-centric (map)
PRED 1/2 = predictor disabled/enabled

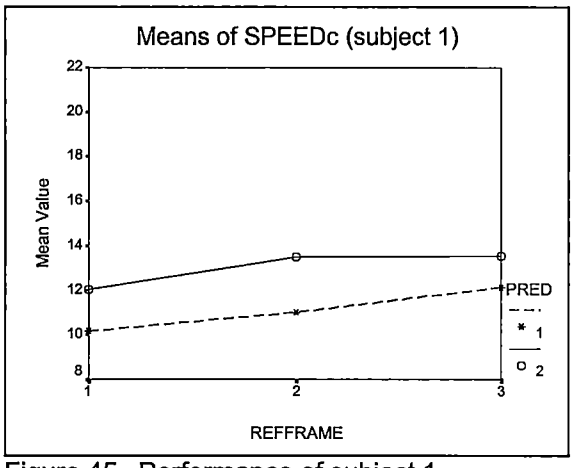

Figure 45. Performance of subject 1

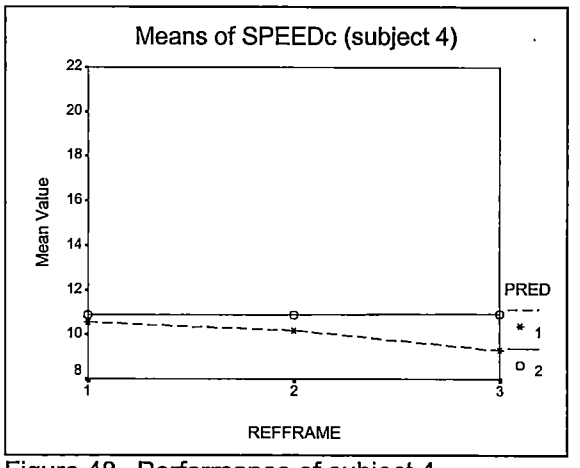

Figure 48. Performance of subject 4

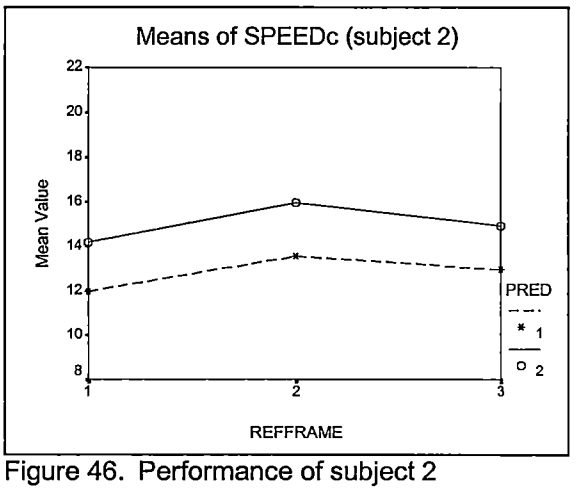

Figure 46. Performance of subject 2

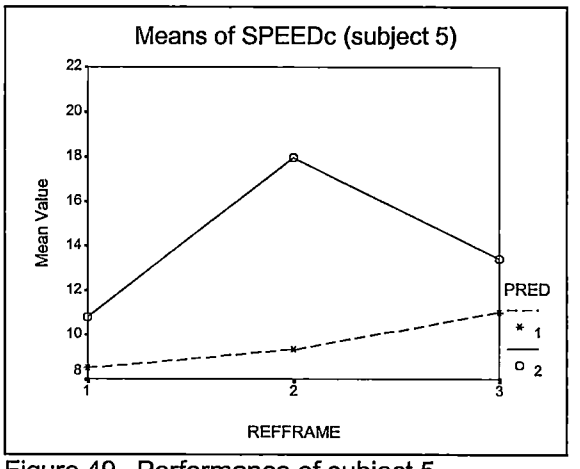

Figure 49. Performance of subject 5

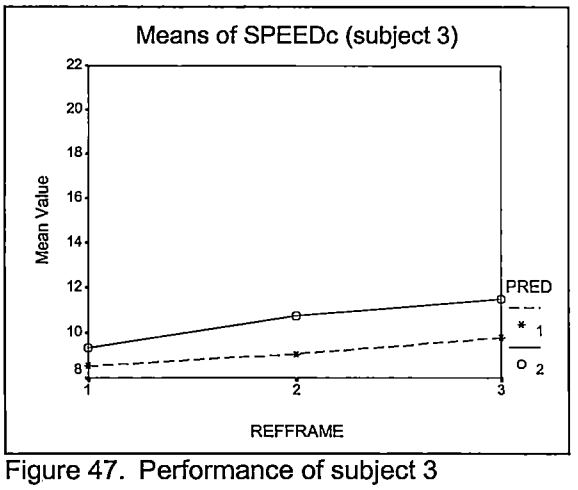

Figure 47. Performance of subject 3

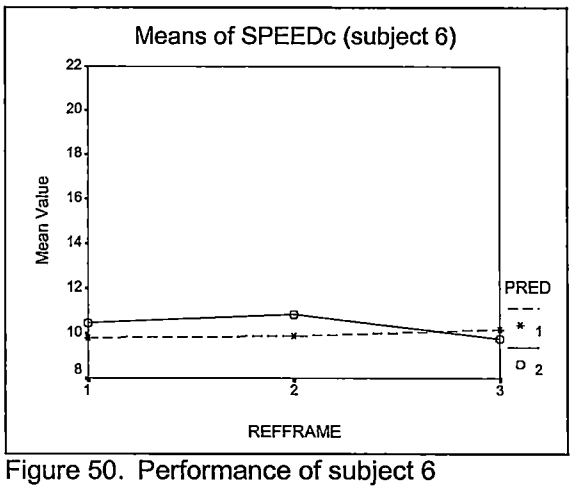

Figure 50. Performance of subject 6

Appendix G. Final results on straight segments

REFFRAME 1=ego-centred
2=3D exo-centric
3=2D exo-centric (map)
PRED 1/2 = predictor disabled/enabled

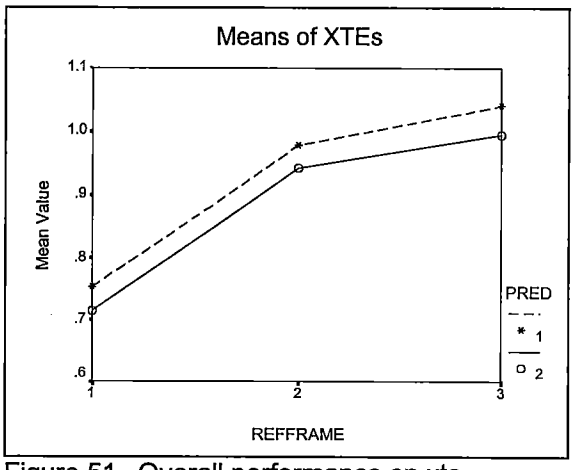

Figure 51. Overall performance on xte_S

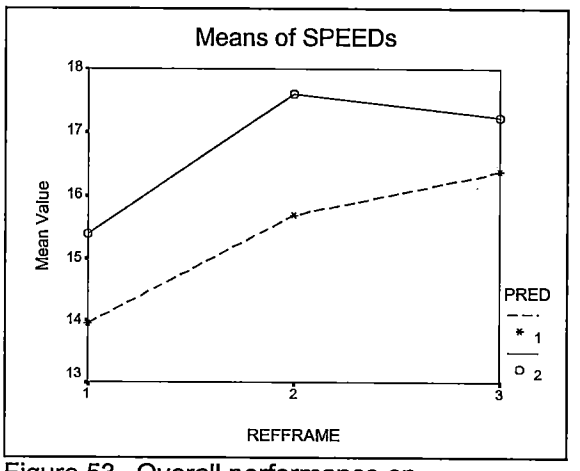

Figure 53. Overall performance on $speed_S$

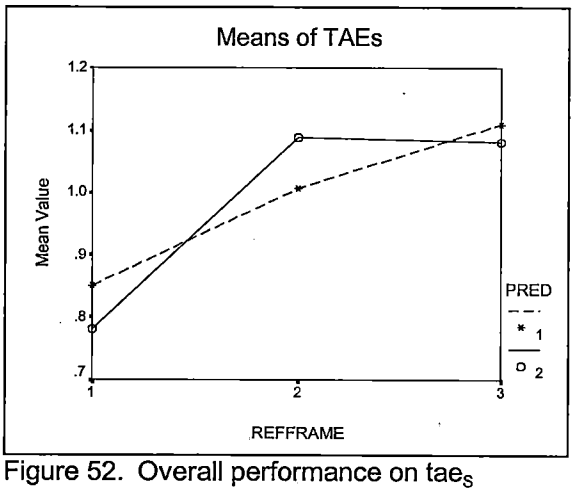

Figure 52. Overall performance on tae_S

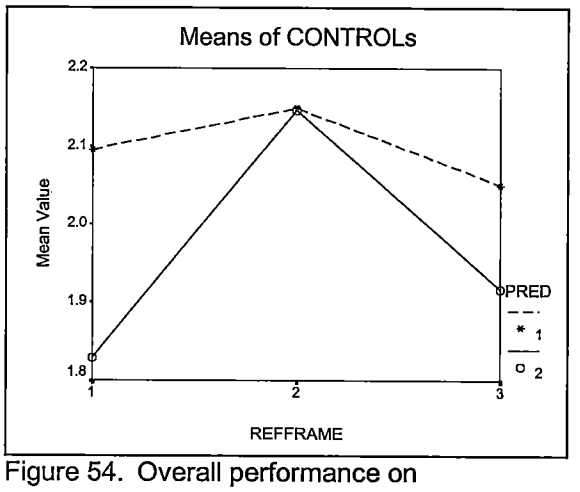

Figure 54. Overall performance on $control_S$

Appendix H. Final results on curved segments

REFFRAME 1=ego-centred
2=3D exo-centric
3=2D exo-centric (map)
PRED 1/2 = predictor disabled/enabled

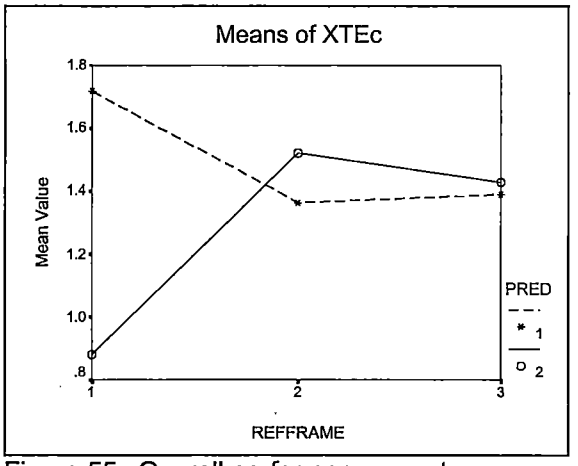

Figure 55. Overall performance on xte_C

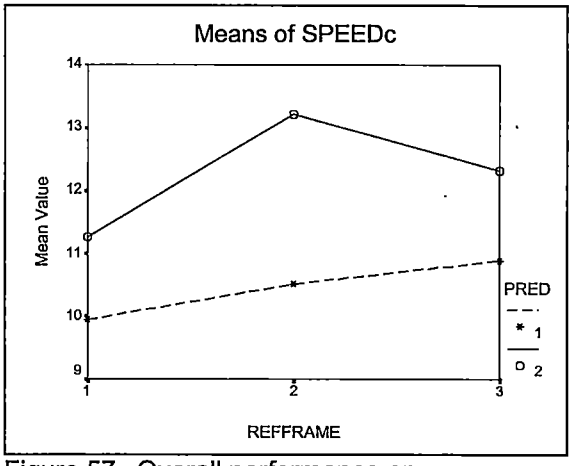

Figure 57. Overall performance on $speed_C$

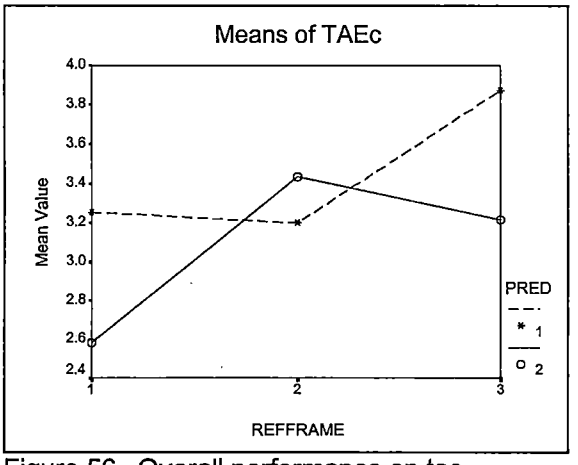

Figure 56. Overall performance on tae_C

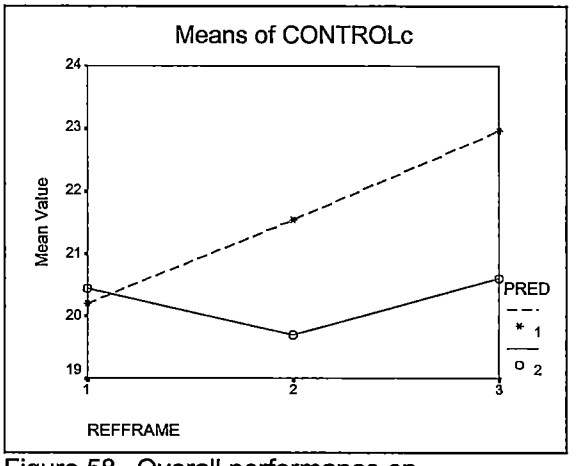

Figure 58. Overall performance on $control_C$

Appendix Z The IMAGE Taxi Software

The software described here is embedded in the ***DELPHINS II*** software of the Telecommunications and Traffic Control Systems Group.

Z.1 Decleration file `IM_DECL.H`

In the file `IM_DECL.H` (Appendix Z.5), all variables and functions that are used in this part of the software are declared.

Z.1.1 Description of Variables

- `struct PATH_PRED`

`xpos[i]`	: x-position of prediction point `i` in the predictor shape
`ypos[i]`	: y-position of prediction point `i` in the predictor shape
`angle[i]`	: heading angle at prediction point `i` in the predictor shape

- `struct SHAPE`

`dist[i]`	: distance perpendicular to reference point `i` the prediction path
`refpoint[i]`	: reference point `i` in the prediction path

- `struct JOYSTICK`

`jxm`	: joystick's x center position
`jym`	: joystick's y center position
`blm`	: left brake pedal initial position
`brm`	: right brake pedal initial position
`pitchrate`	: y-axis joystick output after corrections
`rollrate`	: x-axis joystick output after corrections
`jx1`	: x-axis joystick readout
`jy1`	: y-axis joystick readout
`jx2`	: left brake pedal readout
`jy2`	: right brake pedal readout
`butter_x1[i]`	: i^{th} stored value of `jx1-jxm` for filter correction
`butter_y1[i]`	: i^{th} stored value of `jy1-jym` for filter correction
`butter_x2[i]`	: i^{th} stored value of `jx2-blm` for filter correction
`butter_y2[i]`	: i^{th} stored value of `jy2-brm` for filter correction
`brakeleft`	: left brake pedal readout
`brakeright`	: right brake pedal readout
`max_brake`	: maximum brake level
`max_thrust`	: maximum thrust level
`idle_thrust`	: thrust level at idle
`rollgain`	: joystick gain
`brakegain`	: brake pedal gain

- struct SERVO

delta	: nose wheel deflection
max_delta	: maximum nose wheel deflection
rate	: angular velocity of nose wheel
max_rate	: maximum angular velocity of nose wheel
change	: difference between nose wheel deflection and joystick input
T	: sample time
tau	: acceleration time for rate from 0 to max_rate
step	: rate increase/decrease step size
rico	: direction (-1,0,+1) of change

- struct POSITION

pi	: pi
x	: x position of aircraft
x_old	: previous x position
y	: y position of aircraft
y_old	: previous y position
v	: aircraft ground speed
a	: aircraft acceleration
inf	: value for ∞
radius	: radius of turn
angle	: aircraft heading
steerangle	: effective nose wheel angle
F	: total effective force on aircraft
Fthrust	: thrust force
Fdrag	: drag force
Froll	: rolling resistance force
Fbrake	: brake force
Fangle	: nose wheel angle due to asymmetric braking
Fangle_sens	: sensitivity of asymmetric braking
G	: gravitational acceleration
drag	: drag resistance constant
road	: road resistance constant
m	: aircraft mass
wheelbase	: aircraft wheel base
wheelspan	: aircraft wheel span
direction	: direction (+1,0,-1) of v relative to aircraft heading
looptime	: runtime of 1 program loop (Zen-timer)

- struct PREDICTION

pred	: structure with parameters of all points in prediction path
shapedata	: structure containing the form of the shape
resolution	: number of points in prediction path
timefix	: number of seconds that path prediction covers
delta_store	: stored value of delta
rate_store	: stored value of rate
v_store	: stored value of v
x_store	: stored value of x

`y_store`	: stored value of `y`
`x_old_store`	: stored value of `x_old`
`y_old_store`	: stored value of `y_old`
`angle_store`	: stored value of `angle`
`shapepoints`	: number of points in shape form

- `struct ASMGCS`

`*joystick`	: pointer to the `JOYSTICK` structure
`*servo`	: pointer to the `SERVO` structure
`*position`	: pointer to the `POSITION` structure
`*prediction`	: pointer to the `PREDICTION` structure

Z.2 Module file `IM_PROC.C`

In the file `IM_PROC.C` (Appendix Z.6), all procedures are defined that influence the aircraft dynamics on the ground, the 2D position calculation and the prediction tool. A flow chart of the complete ASMGCS simulation program is given in Figure Z.1 after the procedure descriptions.

Z.2.1 Description of procedures

`struct ASMGCS *init(char *datafilename, char *shapefilename)`

- Description of local variables

`data`	: structure containing array of variables with initial values read from `*datafilename (ASMGCS.INI)`
`datafile`	: file handler for `*datafilename`
`shapefile`	: file handler for `*shapefilename (SHAPE.SHP)`
`i`	: counter
`v`	: variable counter
`readvar`	: set if variable is being read [0,1]
`read_pos`	: set to column number of `*shapefile` [1,2,3]
`ch`	: character read from file
`buffer[i]`	: array of length `i`, containing a combination of characters `ch`

- Procedure calls: none

- Affected variables: all

- Actions:

1. Build memory model, allocating memory for structures ASMGCS, JOYSTICK, SERVO, POSITION, PREDICTION and setting pointers to the different memory blocks
2. Read data file with initial values (ASMGCS.INI)
3. Read predictor shape file (SHAPE.SHP)
4. Initialise loop timer (Zen Timer)
5. Initialise all variables
6. Return pointer to main structure ASMGCS

void reset_asmgcs(struct ASMGCS *asmgcs)

- Description of local variables

i : counter

- Procedure calls: none

- Affected variables:

struct JOYSTICK:	butter_x1
	butter_y1[i]
	butter_x2[i]
	butter_y2[i]
	rollrate
	pitchrate
	brakeleft
	brakeright
struct SERVO:	delta
	rate
	change
struct POSITION:	F
	Fthrust
	Fdrag
	Froll
	Fbrake
	Fangle
	a
	v
	x
	y
	x_old
	y_old
	radius
	steerangle
struct PREDICTION:	delta_store
	rate_store
	v_store
	x_store
	y_store
	x_old_store
	y_old_store

- Actions:

1. Reset joystick filter
2. Reset input variables
3. Reset aircraft parameters

`void init_line_connect(struct ASMGCS *asmgcs, struct threedobject *pred1_tdo)`

- Description of local variables

`i` : counter

- Procedure calls: none

- Affected variables: external variables belonging to structure `pred1_tdo`

- Actions:

1. Appoint points in predictor shape to the appropriate lines in the shape form

`void calc_shape(struct ASMGCS *asmgcs, struct threedobject *pred1_tdo)`

- Description of local variables:

`i` : counter

- Procedure calls: none

- Affected variables: external variables belonging to structure `pred 1_tdo`

- Actions:

1. Calculate all points in the predictor shape

`void readstick(struct ASMGCS *asmgcs)`

- Description of local variables:

`reg` : register pack needed for joystick readout
`i` : counter

- Procedure calls: none

- Affected variables:

```
struct JOYSTICK:  butter_x1[i]
                  butter_y1[i]
                  butter_x2[i]
                  butter_y2[i]
                  pitchrate
                  rollrate
                  brakeleft
                  brakeright
```

- Actions:

1. Shift joystick values `jx1`, `jx2`, `jy1`, `jy2` into sliding window averaging filter
2. Store filtered values in joystick parameters `rollrate`, `pitchrate`, `brakeleft` and `brakeright`
3. Adjust final values to dead-zone and maxima
4. Apply gain

`void nosewheel(struct ASMGCS *asmgcs)`

- Description of local variables: none

- Procedure calls: none

- Affected variables:

```
struct SERVO:     change
                  rico
                  delta
                  rate
```

- Actions:

1. Check for existence static situation (`rate`=0) and very small input change.
2. If so, let nose wheel deflection change directly on input (no S-curve), else
3. let nose wheel deflection follow S-curved profile
4. Adjust final value to dead-zone and maxima

`void force(struct ASMGCS *asmgcs)`

- Description of local variables: none

- Procedure calls: none

- Affected variables:

```
struct POSITION:   direction
                   Froll
                   Fbrake
                   Fdrag
                   Fthrust
                   F
                   Fangle
```

- Actions:

1. Calculate all forces
2. Calculate asymmetric braking angle

`void taxi(struct ASMGCS *asmgcs)`

- Description of local variables:

```
i                  : counter
```

- Procedure calls:

```
external:          LZTimerOff()
                   LZTimerCount()
                   LZTimerOn()
internal:          readstick(asmgcs)
                   nosewheel(asmgcs)
                   force(asmgcs)
```

- Affected variables:

```
struct SERVO:      delta
                   rate
                   T
                   step
struct POSITION:   looptime
                   v
                   x
                   y
                   x_old
                   y_old
                   angle
                   a
                   steerangle
                   radius
```

```
struct PREDICTION:delta_store
                  rate_store
                  v_store
                  x_store
                  y_store
                  x_old_store
                  y_old_store
                  angle_store
                  pred.xpos[i]
                  pred.ypos[i]
                  pred.angle[i]
```

- Actions:

1. Stop loop timer, collect loop time, start loop timer
2. In case i=0, restore aircraft parameters and calculate new position, else
3. In case i=1, store all aircraft parameters and start calculating prediction path using same path calculation routine, else
4. finish calculating prediction path, using same path calculation routine

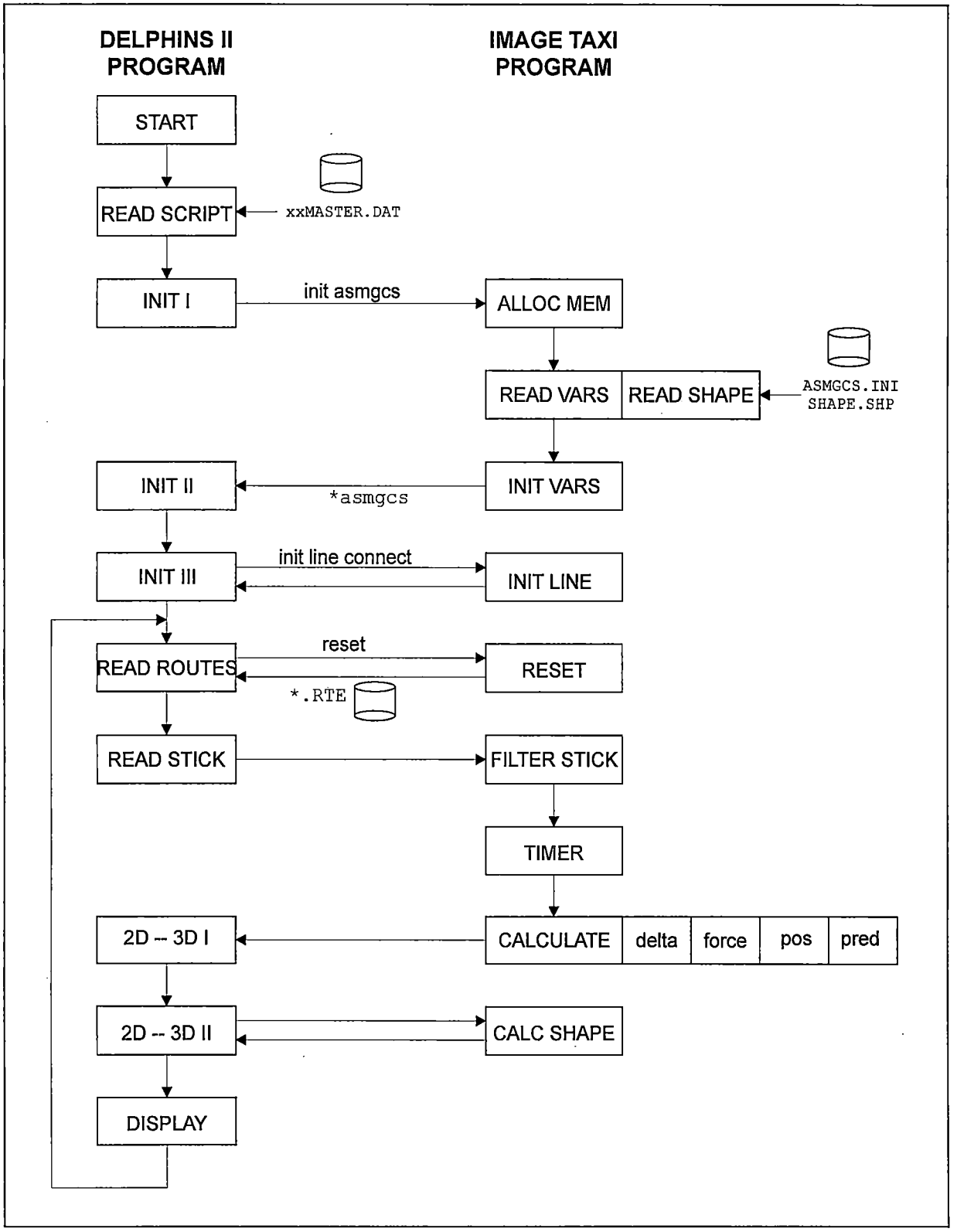

Figure Z.1. Flow chart of ASMGCS simulation program

Z.3 ASMGCS.INI

```
                    ***** ASMGCS parameter file *****

Values between the 'equal to' and the 'dot-comma' are recognized
as parameter values. Use ONLY THESE two characters as starter and
terminator symbols. Spaces are allowed.

!DO NOT CHANGE ORDER OF PARAMETERS AS THEY ARE READ SEQUENTIALLY!

joystick rollrate gain = 10.0;
joystick brakerate gain = 10.0;
idle thrust = 10000;
maximum thrust = 195800;
maximum brake = 332400; 0.6*G
maximum nose wheel angle (degrees) = 45;
maximum angle rate of nose wheel (degrees/sec) = 12.5;
seconds to maximum angle rate : tau =.75;
sensitivity of asymmetric braking =.003125;
gravity acceleration constant (m/s2) = 9.81;
aircraft drag coefficient =0; EMPERICAL VALUE
road constant (0.02 represents dry paved road) =0.02; EMPERICAL VALUE
start heading angle = 76;
infinite = 9E+99;
aircraft mass (kg) = 50000;
wheelbase (nosewheel-maingear) (m) = 12.45;
main gear wheelspan (m) = 5.23;
amount of seconds to extrapolate prediction path = 3;
number of points in prediction path = 15;
```

Z.4 SHAPE.SHP

***** Path Predictor Shape description file *****

Values between the 'equal to' and the 'dot-comma' are recognized as parameter values. Use ONLY THESE two charachters as starter and terminator symbols. Spaces are allowed.

The shape is build up of points left and right perpendicular to reference points. These reference points are the points of the prediction path. The shape is 'curved' around it via a so called forcing function.

Distances are in meters.

INDEX	REF POINT	LEFT DISTANCE	RIGHT DISTANCE
1	=1;	=0;	=0;
2	=1;	=2.615;	=2.615;
3	=2;	=2.615;	=2.615;
4	=3;	=2.615;	=2.615;
5	=4;	=2.615;	=2.615;
6	=5;	=2.615;	=2.615;
7	=6;	=2.615;	=2.615;
8	=7;	=2.615;	=2.615;
9	=8;	=2.615;	=2.615;
10	=9;	=2.615;	=2.615;
11	=10;	=2.615;	=2.615;
12	=11;	=2.615;	=2.615;
13	=12;	=2.615;	=2.615;
14	=12;	=6;	=6;
15	=13;	=4;	=4;
16	=14;	=2;	=2;
17	=15;	=0;	=0;

Z.5 IM DECL.H

```
#define MAX_PRED_POINTS 100
#define MAX_SHAPE_POINTS 100

struct PATH_PRED
{
        double xpos[MAX_PRED_POINTS],ypos[MAX_PRED_POINTS],angle[MAX_PRED_POINTS];
};

struct SHAPE
{
      double dist[MAX_SHAPE_POINTS];
      int refpoint[MAX_SHAPE_POINTS];
};

struct JOYSTICK
{
        /* joystick parameters */
        double jxm,jym,blm,brm,pitchrate,rollrate;
        double jx1,jy1,jx2,jy2;
        double butter_x1[5],butter_y1[5],butter_x2[5],butter_y2[5];
        double brakeleft,brakeright,max_brake,max_thrust,idle_thrust;
        double rollgain,brakegain;
};

struct SERVO
{
        /* nosewheel servo parameters */
        double delta,max_delta,rate,max_rate,change;
        double T,tau,step;
        int rico;
};

struct POSITION
{
        /* position parameters */
        double pi,x,x_old,y,y_old;
        double v,a,inf,radius,angle,steerangle;
        double F,Fthrust,Fdrag,Froll,Fbrake,Fangle;
        double Fangle_sens,G,drag,road,m,wheelbase,wheelspan;
        int direction;
        long looptime;
};

struct PREDICTION
{
        /* path prediction parameters */
        struct PATH_PRED pred;
        struct SHAPE shapedata;
        double resolution,timefix;
        double delta_store,rate_store;
        double v_store,x_store,y_store,x_old_store,y_old_store;
        double angle_store,radius_store;
        int shapepoints;

};

struct ASMGCS
{
        struct JOYSTICK *joystick;
        struct SERVO *servo;
        struct POSITION *position;
        struct PREDICTION *prediction;
};

struct ASMGCS *init(char *datafilename, char *shapefilename);
void reset_asmgcs(struct ASMGCS *asmgcs);
void calc_shape(struct ASMGCS *asmgcs, struct threedobject *pred1_tdo);
void init_line_connect(struct ASMGCS *asmgcs, struct ADDRESS *pred1_obj);
void readstick(struct ASMGCS *asmgcs);
void nosewheel(struct ASMGCS *asmgcs);
void force(struct ASMGCS *asmgcs);
void taxi(struct ASMGCS *asmgcs);
```

Z.6 IM_PROC.H

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdarg.h>
#include <string.h>
#include <time.h>
#include <dos.h>
#include <malloc.h>
#include <math.h>
#include "d3s.h"
#include <conio.h>
#include <io.h>
#include <glide.h>
#include <mgraph.h>

#include <fcntl.h>

#include "pmode.h"
#include "ztimer.h"

#include "im_decl.h"


#define BUFSIZE 100
#define MAXVARS 100

/* INITIALIZING PROCEDURE */
struct ASMGCS *init(char *datafilename, char *shapefilename)
{
        struct VARS
        {
              double var[MAXVARS];
        };

        struct VARS data;
        int datafile,shapefile,i,v,readvar,read_pos;
        char ch,buffer[BUFSIZE],endptr;

        /* set pointers */
        struct ASMGCS *asmgcs;
        struct JOYSTICK *joystick;
        struct SERVO *servo;
        struct POSITION *position;
        struct PREDICTION *prediction;

        union REGPACK reg;

        /* allocate memory for variables */
        asmgcs=(struct ASMGCS *)malloc(sizeof(struct ASMGCS));
        joystick=(struct JOYSTICK *)malloc(sizeof(struct JOYSTICK));
        servo=(struct SERVO *)malloc(sizeof(struct SERVO));
        position=(struct POSITION *)malloc(sizeof(struct POSITION));
        prediction=(struct PREDICTION *)malloc(sizeof(struct PREDICTION));

        /* check for enough memory */
        if ((asmgcs==NULL)||(joystick==NULL)||(servo==NULL)
           ||(position==NULL)||(prediction==NULL))
        {
                printf("Not enough memory store parameters\n");
                exit(1);  /* terminate program if out of memory */
        }

        asmgcs->joystick=joystick;
        asmgcs->servo=servo;
        asmgcs->position=position;
        asmgcs->prediction=prediction;

        v=0;
        i=0;
        readvar=0;

        /* open datafile and check existence */
        if ((datafile = open(datafilename,O_RDONLY | O_TEXT)) == -1)
        {
                perror("Error:");
                exit(1);
```

```
	}

	/* read chars from the datafile until it EOF is reached */
	while (!eof(datafile))
	{
		read(datafile, &ch, 1);
		/* convert to double when terminator ';' is reached */
		if (ch==';')
		{
			buffer[i]='\0'; /* terminate string */
			data.var[v]=strtod(buffer,NULL);
			i=0;
			readvar=0;
			v++;
			if (v==MAXVARS)
			{
				printf("v too large\n");
				exit(-1);
			}
		}
		/* start storing buffer after the '=' char */
		else if (ch=='=')
			readvar=1;
		else if (readvar==1)
		{
			buffer[i]=ch;
			i++;
			if (i==BUFSIZE)
			{
				printf("i too large\n");
				exit(1);
			}

		}
	}
	close(datafile);

	v=0;
	i=0;
	readvar=0;
	read_pos=1;
	prediction->shapepoints=0;

	/* open datafile and check existence */
	if ((shapefile = open(shapefilename,O_RDONLY | O_TEXT)) == -1)
	{
		perror("Error:");
		exit(1);
	}

	/* read chars from the datafile until it EOF is reached */
	while (!eof(shapefile))
	{
		read(shapefile, &ch, 1);
		/* convert to value when terminator ';' is reached */
		if (ch==';')
		{
			buffer[i]='\0'; /* terminate string */
			/* read reference point */
			if (read_pos==1)
			{
				prediction->shapepoints++;
				if (prediction->shapepoints==MAX_SHAPE_POINTS)
				{
					printf("too many shape points\n");
					exit(-1);
				}
				prediction->shapedata.refpoint[prediction-
>shapepoints]=atoi(buffer);
				read_pos=2;
			}
			else
			{
				/* read distance to reference point */
				v++;
				if (v==MAXVARS)
				{
					printf("v too large\n");
					exit(-1);
				}
				prediction->shapedata.dist[v]=strtod(buffer,NULL);
				if (read_pos==2)
```

```
                                read_pos=3;
                        else
                                read_pos=1;
                }
                i=0;
                readvar=0;
        }
        /* start storing buffer after the '=' char */
        else if (ch=='=')
                readvar=1;
        else if (readvar==1)
        {
                buffer[i]=ch;
                i++;
                if (i==BUFSIZE)
                {
                        printf("i too large\n");
                        exit(1);
                }
        }
}

close(shapefile);

/* initialize loop timer */
ZTimerInit();
LZTimerOn();

for (i=1;i<5;i++)
{
        joystick->butter_x1[i]=0;
        joystick->butter_y1[i]=0;
        joystick->butter_x2[i]=0;
        joystick->butter_y2[i]=0;
}
joystick->jxm=0;
joystick->jym=0;
joystick->blm=0;
joystick->brm=0;
joystick->rollrate=0;
joystick->pitchrate=0;
joystick->brakeleft=0;
joystick->brakeright=0;
joystick->rollgain=data.var[0];
joystick->brakegain=data.var[1];
joystick->idle_thrust=data.var[2];
joystick->max_thrust=data.var[3]-data.var[2];
joystick->max_brake=data.var[4];

servo->delta=0;
servo->max_delta=data.var[5]; /* maximum nose wheel angle */
servo->rate=0;

servo->max_rate=data.var[6]; /* maximum rate of turn */
servo->change=0;
servo->tau=data.var[7]; /* seconds to maximum angle rate */

servo->T=0;
servo->rico=0;

position->pi=M_PI;

position->F=0;
position->Fthrust=0;
position->Fdrag=0;
position->Froll=0;
position->Fbrake=0;
position->Fangle=0;
position->Fangle_sens=data.var[8]; /* asymmetric braking */
position->G=data.var[9]; /* gravity acceleration constant */
position->drag=data.var[10]; /* aircraft drag coefficient */
position->road=data.var[11]; /* road constant */

position->looptime=0;
position->a=0;
position->v=0;
position->x=0;
position->y=0;
position->x_old=0;
position->y_old=0;
position->radius=0;
position->angle=data.var[12]*position->pi/180;
```

```
        position->steerangle=0;
        position->inf=data.var[13];

        position->m=data.var[14]; /* aircraft mass */
        position->wheelbase=data.var[15]; /* wheelbase */
        position->wheelspan=data.var[16]; /* main gear wheelspan */

        position->direction=0;

        prediction->timefix=data.var[17]; /* extrapolating seconds */
        prediction->resolution=data.var[18]; /* pixels in prediction path */

        prediction->delta_store=0;
        prediction->rate_store=0;
        prediction->v_store=0;
        prediction->x_store=0;
        prediction->y_store=0;
        prediction->x_old_store=0;
        prediction->y_old_store=0;
        prediction->angle_store=data.var[12]*position->pi/180;

return(asmgcs);
}

/* RESET PARAMETERS */
void reset_asmgcs(struct ASMGCS *asmgcs)
{
        struct JOYSTICK *joystick;
        struct SERVO *servo;
        struct POSITION *position;
        struct PREDICTION *prediction;

        int i;

        joystick=asmgcs->joystick;
        servo=asmgcs->servo;
        position=asmgcs->position;
        prediction=asmgcs->prediction;

        for (i=1;i<5;i++)
        {
                joystick->butter_x1[i]=0;
                joystick->butter_y1[i]=0;
                joystick->butter_x2[i]=0;
                joystick->butter_y2[i]=0;
        }

        joystick->rollrate=0;
        joystick->pitchrate=0;
        joystick->brakeleft=0;
        joystick->brakeright=0;

        servo->delta=0;
        servo->rate=0;

        servo->change=0;

        position->F=0;
        position->Fthrust=0;
        position->Fdrag=0;
        position->Froll=0;
        position->Fbrake=0;
        position->Fangle=0;

        position->a=0;
        position->v=0;
        position->x=0;
        position->y=0;
        position->x_old=0;
        position->y_old=0;
        position->radius=0;
        position->steerangle=0;

        prediction->delta_store=0;
        prediction->rate_store=0;
        prediction->v_store=0;
        prediction->x_store=0;
        prediction->y_store=0;
        prediction->x_old_store=0;
        prediction->y_old_store=0;
```

```
}

/* SET POINTS IN SHAPE TO ACCORDING LINES */
void init_line_connect(struct ASMGCS *asmgcs, struct ADDRESS *pred1_obj)
{
        struct PREDICTION *prediction;
        struct threedobject *pred1_tdo,*pred1_tdo1;

        int i;

        pred1_tdo=pred1_obj->tdo;
        pred1_tdo1=pred1_obj->tdo1;
        prediction=asmgcs->prediction;

        /* make line connection list */
        for(i=1;i<=prediction->shapepoints-1;i++)
        {
                pred1_tdo->line_a[i-1].begin=2*i-1;
                pred1_tdo->line_a[i-1].end=2*i+1;
                pred1_tdo->line_a[i-1].color=3;
                pred1_tdo->line_a[prediction->shapepoints+i-1].begin=2*i;
                pred1_tdo->line_a[prediction->shapepoints+i-1].end=2*i+2;
                pred1_tdo->line_a[prediction->shapepoints+i-1].color=3;
                pred1_tdo1->line_a[i-1].begin=2*i-1;
                pred1_tdo1->line_a[i-1].end=2*i+1;
                pred1_tdo1->line_a[i-1].color=3;
                pred1_tdo1->line_a[prediction->shapepoints+i-1].begin=2*i;
                pred1_tdo1->line_a[prediction->shapepoints+i-1].end=2*i+2;
                pred1_tdo1->line_a[prediction->shapepoints+i-1].color=3;
        }
        pred1_tdo->line_a[prediction->shapepoints-1].begin=2*prediction->shapepoints-1;
        pred1_tdo->line_a[prediction->shapepoints-1].end=2*prediction->shapepoints;
        pred1_tdo1->line_a[prediction->shapepoints-1].begin=2*prediction->shapepoints-1;
        pred1_tdo1->line_a[prediction->shapepoints-1].end=2*prediction->shapepoints;
        pred1_tdo->line_a[prediction->shapepoints-1].color=3;
        pred1_tdo->numberofpoints=2*prediction->shapepoints+1;
        pred1_tdo->numberoflines=2*prediction->shapepoints-1;
        pred1_tdo1->line_a[prediction->shapepoints-1].color=3;
        pred1_tdo1->numberofpoints=2*prediction->shapepoints+1;
        pred1_tdo1->numberoflines=2*prediction->shapepoints-1;
}

/* CALCULATE ALL POINTS IN SHAPE */
void calc_shape(struct ASMGCS *asmgcs, struct threedobject *pred1_tdo)
{
        struct PREDICTION *prediction;

        int i;

        prediction=asmgcs->prediction;

        /* calculate all coordinates in shape */
        for (i=1;i<=prediction->shapepoints;i++)
        {
                pred1_tdo->tdpoint[2*i-1].x=prediction->pred.xpos[prediction-
>shapedata.refpoint[i]]+prediction->shapedata.dist[2*i-1]*cos(prediction-
>pred.angle[prediction->shapedata.refpoint[i]]-M_PI/2);
                pred1_tdo->tdpoint[2*i-1].y=0;
                pred1_tdo->tdpoint[2*i-1].z=prediction->pred.ypos[prediction-
>shapedata.refpoint[i]]-prediction->shapedata.dist[2*i-1]*sin(prediction-
>pred.angle[prediction->shapedata.refpoint[i]]-M_PI/2);
                pred1_tdo->tdpoint[2*i].x=prediction->pred.xpos[prediction-
>shapedata.refpoint[i]]+prediction->shapedata.dist[2*i]*cos(prediction->pred.angle[prediction-
>shapedata.refpoint[i]]+M_PI/2);
                pred1_tdo->tdpoint[2*i].y=0;
                pred1_tdo->tdpoint[2*i].z=prediction->pred.ypos[prediction-
>shapedata.refpoint[i]]-prediction->shapedata.dist[2*i]*sin(prediction->pred.angle[prediction-
>shapedata.refpoint[i]]+M_PI/2);
        }
}


/* FILTER JOYSTICK & BRAKE PEDAL INPUTS TO CORRECT FOR NOISE */
void readstick(struct ASMGCS *asmgcs)
{
        struct JOYSTICK *joystick;
        union REGPACK reg;
        int i;

        joystick=asmgcs->joystick;

        /* shift down previous joystick values */
```

```
        for (i=0;i<3;i++)
        {
                joystick->butter_x1[4-i]=joystick->butter_x1[3-i];
                joystick->butter_y1[4-i]=joystick->butter_y1[3-i];
                joystick->butter_x2[4-i]=joystick->butter_x2[3-i];
                joystick->butter_y2[4-i]=joystick->butter_y2[3-i];
        }

        /* store joystick value and correct for centre point jitter */
        joystick->butter_x1[1]=joystick->jx1-joystick->jxm;
        joystick->butter_y1[1]=joystick->jy1-joystick->jym;
        joystick->butter_x2[1]=joystick->jx2-joystick->blm;
        joystick->butter_y2[1]=joystick->jy2-joystick->brm;
        /* 4th order sliding window averager to reduce stick noise */
        joystick->rollrate=(.25*joystick->butter_x1[1]+.25*joystick-
>butter_x1[2]+.25*joystick->butter_x1[3]+.25*joystick->butter_x1[4]);
        joystick->pitchrate=(.25*joystick->butter_y1[1]+.25*joystick-
>butter_y1[2]+.25*joystick->butter_y1[3]+.25*joystick->butter_y1[4]);
        joystick->brakeleft=(.25*joystick->butter_x2[1]+.25*joystick-
>butter_x2[2]+.25*joystick->butter_x2[3]+.25*joystick->butter_x2[4]);
        joystick->brakeright=(.25*joystick->butter_y2[1]+.25*joystick-
>butter_y2[2]+.25*joystick->butter_y2[3]+.25*joystick->butter_y2[4]);
        /* dead zone and maximum outputs */
        if (fabs(joystick->pitchrate)<2)
                joystick->pitchrate=0;
        if (fabs(joystick->pitchrate)>30)
                joystick->pitchrate=(joystick->pitchrate/fabs(joystick->pitchrate))*30;
        if (fabs(joystick->rollrate)<2)
                joystick->rollrate=0;
        if (fabs(joystick->brakeleft)<20)
                joystick->brakeleft=0;
        if (fabs(joystick->brakeleft)>50)
                joystick->brakeleft=(joystick->brakeleft/fabs(joystick->brakeleft))*50;
        if (fabs(joystick->brakeright)<20)
                joystick->brakeright=0;
        if (fabs(joystick->brakeright)>50)
                joystick->brakeright=(joystick->brakeright/fabs(joystick->brakeright))*50;
        /* adjust joystick sensitivity */
        joystick->brakeleft/=joystick->brakegain;
        joystick->brakeright/=joystick->brakegain;
        joystick->rollrate/=joystick->rollgain;
}


/* CALCULATE NEW NOSE WHEEL ANGLE VIA S-CURVE PROFILE */
void nosewheel(struct ASMGCS *asmgcs)
{
        struct JOYSTICK *joystick;
        struct SERVO *servo;
        servo=asmgcs->servo;
        joystick=asmgcs->joystick;

        /* calculate difference between joystick and nose wheel angle */
        servo->change=joystick->rollrate-servo->delta;
        if (fabs(servo->change)==0)
                servo->rico=0;
        else
                servo->rico=servo->change/fabs(servo->change);

        if ((servo->rate==0) && (fabs(servo->change)<1.1*servo->step*servo->T))
                servo->delta=joystick->rollrate;
        else
        {
                if ((fabs(servo->change)-.1125)>.5*fabs(servo->rate)*((fabs(servo-
>rate)+servo->step)/servo->step)*servo->T)
                        servo->rate+=(servo->step*servo->rico);
                else
                {
                        if ((fabs(servo->rate)>servo->step)
                        && (servo->rate>0))
                                servo->rate-=servo->step;

                        else if ((fabs(servo->rate)>servo->step)
                        && (servo->rate<0))
                                servo->rate+=servo->step;
                        else
                        {
                                servo->rate=0;
                                servo->delta=joystick->rollrate;
                        }

                        /* limit angle rate */
```

```
                        if (fabs(servo->rate)>servo->max_rate)
                                servo->rate=servo->max_rate*servo->rico;

                        /* new nose wheel angle */
                        servo->delta+=(servo->rate*servo->T);
                }
        /* dead zone */
        }
        if (fabs(servo->delta)<.05)
                servo->delta=0;

        /* limit nose wheel angle */
        if (fabs(servo->delta)>servo->max_delta)
              servo->delta=(servo->delta/fabs(servo->delta))*servo->max_delta;
}

/* CALCULATE FORCES ON AIRCRAFT */
void force(struct ASMGCS *asmgcs)
{
        struct JOYSTICK *joystick;
        struct POSITION *position;
        joystick=asmgcs->joystick;
        position=asmgcs->position;

        /* Froll, Fdrag & Fbrake depend on direction of groundspeed */
        if (position->v==0)
                position->direction=0;
        else
                position->direction=position->v/fabs(position->v);

        position->Froll=position->direction*position->road*position->m*position->G;
        position->Fbrake=position->direction*joystick->max_brake*(joystick-
>brakeright+joystick->brakeleft)/10;
        position->Fdrag=position->direction*position->drag*pow(position->v,2)/25.92;
        position->Fthrust=joystick->idle_thrust-joystick->max_thrust*joystick->pitchrate/30;
        position->F=position->Fthrust+position->Fbrake-position->Fdrag-position->Froll;

        /* set angle when asymmetric braking occurs */
        position->Fangle=atan((position->Fangle_sens*position->direction*(joystick->brakeleft-
joystick->brakeright))/(position->wheelspan/2));
}

/* MAIN POSITION & PATH PREDICTION PROCEDURE */
void taxi(struct ASMGCS *asmgcs)
{
        int i,count;

        struct SERVO *servo;
        struct POSITION *position;
        struct PREDICTION *prediction;
        servo=asmgcs->servo;
        position=asmgcs->position;
        prediction=asmgcs->prediction;

        readstick(asmgcs);

        /* calculate sample time in position calculation */
        servo->T=position->looptime/1E+06;
        servo->step=servo->T*servo->max_rate/servo->tau;

        count=1;

        for (i=0;i<(prediction->timefix/servo->T)+1;i++)
        {
                if (i==0)
                {
                        /* return stored values */
                        servo->delta=prediction->delta_store;
                        servo->rate=prediction->rate_store;
                        position->v=prediction->v_store;
                        position->x=prediction->x_store;
                        position->y=prediction->y_store;
                        position->x_old=prediction->x_old_store;
                        position->y_old=prediction->y_old_store;
                        position->angle=prediction->angle_store;

                        /* erase last aircraft dot */
                }
                else if (i==1)
                {
                        /* temporary storage of all current values */
                        prediction->delta_store=servo->delta;
```

```
                        prediction->rate_store=servo->rate;
                        prediction->v_store=position->v;
                        prediction->x_store=position->x;
                        prediction->y_store=position->y;
                        prediction->x_old_store=position->x_old;
                        prediction->y_old_store=position->y_old;
                        prediction->angle_store=position->angle;
                        if (position->radius==position->inf)
                                prediction->radius_store=-1;
                        else
                                prediction->radius_store=position->radius;

                        /* calculate sampletime predictionloop */
                }

                nosewheel(asmgcs);
                force(asmgcs);

                /* new acceleration */
                position->a=position->F/position->m;

                /* effective steering angle (nosewheel+asymmetrisch braking */
                position->steerangle=servo->delta+180*position->Fangle/position->pi;

                /* set radius of turn; avoid deviding by 0 */
                if (position->steerangle==0)
                        position->radius=position->inf;
                else
                        position->radius=position->wheelbase/sin(position->pi*position-
>steerangle/180.0);

                /* new position */
                position->x+=(position->v*servo->T+.5*position->a*pow(servo-
>T,2))*cos(position->angle);
                position->y-=(position->v*servo->T+.5*position->a*pow(servo-
>T,2))*sin(position->angle);

                /* new groundspeed */
                position->v+=position->a*servo->T;

                /* dead zone */
                if (fabs(position->v)<.001)
                        position->v=0;

                /* new heading */
                position->angle+=2*position->pi*(sqrt(pow(position->x-position-
>x_old,2)+pow(position->y-position->y_old,2)))/position->radius;
                position->x_old=position->x;
                position->y_old=position->y;
                if (!(i==0))
                {
                        if (i>=count*prediction->timefix/(servo->T*prediction->resolution))
                        {
                                /* prediction is relative to plane's position */
                                prediction->pred.xpos[count]=position->x-prediction->x_store;
                                prediction->pred.ypos[count]=position->y-prediction->y_store;
                                prediction->pred.angle[count]=position->angle;
                                count++;
                                if (count==MAX_PRED_POINTS)
                                {
                                    printf("count too large\n");
                                    exit(-1);
                                }
                        }
                }

        }

}
```