**2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan**

Vision Aided Automatic Landing System for Fixed Wing UAV

Maximilian Laiacker, Konstantin Kondak, Marc Schwarzbach and Tin Muskardin

Abstract— **In this paper, we present a multi-sensor system for automatic landing of fixed wing UAVs. The system is composed of a high precision aircraft controller and a vision module which is currently used for detection and tracking of runways. Designing the system we paid special attention to its robustness. The runway detection algorithm uses a maximum amount of information in images and works with high level geometrical models. It allows detecting a runway under different weather conditions even if only a small part is visible in the image. In order to increase landing reliability under sub-optimal wind conditions, an additional loop was introduced into the altitude controller. All control and image processing is performed onboard. The system has been successfully tested in flight experiments with two different fixed wing platforms at various weather conditions, in summer, fall and winter.**

I. Introduction

The integration of advanced robotic technologies into UAVs opens new fields of applications for these kinds of robots. Especially the automation of landing for fixed wing UAVs is important because the landing is one of the most critical phases in many mission profiles. First, even small errors in guidance and control could yield system loss or damages. Second, the possibility to operate a system under sub-optimal weather conditions often depends on ability to land safely.

In this paper we present the concept and experimental results for the automatic landing system based on a combination of GPS and vision data processing. This system is currently used for automatic landing of UAVs on runways. Later it should be integrated into a system for automatic landing of our solar powered high altitude platform ELHASPA, s. Fig. 1, on mobile ground vehicles. The wing span of this platform is $23m$ and the mass is about $100kg$. The platform should be operated at the altitude of $18-25km$. The concept of this type of landing is shown in Fig. 2. In the touch-down phase of the landing the relative velocity of a flight platform to a ground vehicle can be controlled close to zero. This allows to minimize the mechanical stress of the plane structure and therefore to minimize its weight. Also the landing gear is not needed anymore.

Considering a landing of UAVs on a runway as well as a landing on mobile platforms two main issues should be taken into account: high precision motion control and the precision and reliability of the sensor system providing the position information. These two issues are discussed in the paper in more detail. The presented system was evaluated in flight experiments. The adaptation of the system to different flight platforms and to different environmental conditions was evaluated in experiments as well.

Institute of Robotics and Mechatronics, German Aerospace Center (DLR), 82230 Wessling, Germany
maximilian.laiacker@dlr.de

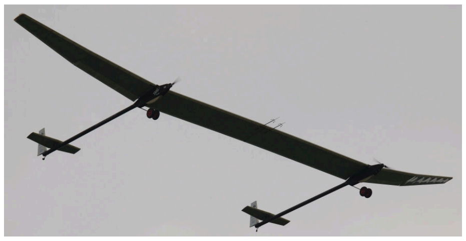

Fig. 1. Solar HALE UAV ELHASPA in flight

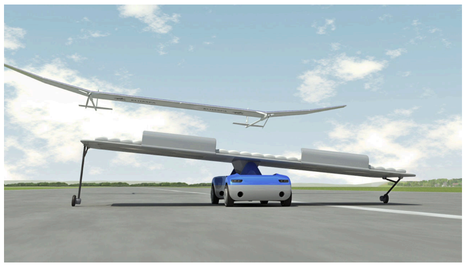

Fig. 2. ELHASPA lands on a mobile ground vehicle (concept)

Automatic landing for military grade fixed wing UAVs on runways is available like in the HERON 1 [1] or the MQ-1B Predator [2] but no detailed information about these systems is available. Despite the fact that landing is often the most critical phase in the whole mission, only very few publications about flight tested automatic landing systems are available. A lot of work has been done in automatic landing of rotary wing UAVs like [3] or [4]. Most recently perching of a quadrotor [5] and perching of a small plane [6] have been shown in controlled indoor environments. Some approaches for vision guided automatic landings of fixed wing UAVs have been published including very early work done by E.D. Dickmanns et al. [7] where a hardware in the loop simulation for automatic landing was presented. The vision system was tested as stand alone in manual flight experiments. In [8] a forward looking camera is used to calculate the position of the UAV relative to a known runway by comparing the camera image with a stack of recorded images. Also this approach was not tested in real flight experiments. In [9] a visual servoing method for landing a fixed wing plane is presented but also no flight experiments were conducted for verification. The experimental UAV developed by BAE SYSTEMS presented in [10], have performed automatic take-off and landing during flight experiments. The landing was done using GPS and IMU. A laser altimeter was used to control the final flare maneuver. Our approach for the automation in the last phase is similar to the one presented in [10] but we control the airspeed until touch down which in our opinion is safer than reducing the throttle to idle.

In order to be able to extend the presented system for automatic landing on a mobile platform as shown in Fig. 2, its control and navigation algorithms should be able to not only land a UAV safely but also to lead it to a desired touch down point. In Sec. II we give a general description of the system. We describe the navigation algorithm together with the high precision motion control used for automatic landing in Sec. III. The runway detection algorithm is described in Sec. IV. In Sec. V we present results of flight experiments where we validated the system components and automatic landing as integrated functionality. Finally, conclusions are made and future work is outlined.

II. System overview

The functional scheme of the system with main components, their connections and data flows are shown in Fig. 3.

The system components can be divided in two parts. The hard real-time part controls the UAV flight and the vision system that acts as an additional sensor. The real-time system or flight control system (FCS) is a standard setup with one central flight control computer (FCC) running the autopilot software. The FCC is connected to the main sensors needed to control the UAV. The sensors are a MEMS inertial measurement unit, a Novatel differential GPS receiver and an air-data probe which measures airspeed and pressure altitude. The actuator interface that controls the servo motors is also connected to the FCC. A radio modem provides a datalink to the ground station where the UAV operation can be controlled and monitored. The vision system performs the landing target recognition by using state information provided by the FCS and the image captured by the camera.

The FCS software is a self developed modular autopilot system [11] running on top of the QNX operating system. The vision computer is running Linux on a Core 2 Quad running at $2GHz$. The camera we used is a RGB 1620x1220 Pixel 25Hz industrial model with a 70° wide angle lens. The camera is mounted on the UAV looking forward with a 30° downwards angle.

III. Landing trajectory and control

To perform an automatic landing the UAV needs a controller that can follow a trajectory with a high accuracy at different airspeeds and aerodynamic configurations. The UAV is controlled by three separate control loops. The altitude controller uses the elevator to control the altitude H. The course controller uses the ailerons to control the UAV course over ground χ. The speed controller uses the engine throttle to control the air speed V_A.

The control structure of the altitude controller is shown in Fig 4. To adapt the controller for operation at different airspeeds we extended a cascaded P and PI controller with nonlinear terms which are denoted in Fig 4 by means of three nonlinear blocks. Block A calculates the desired path angle

$$\gamma_{cmd} = \arcsin\left(\frac{\frac{\delta H}{\delta t}}{V_k}\right)$$

which represents the trigonometric relation between the absolute ground speed V_k and the desired altitude change $\frac{\delta H}{\delta t}$. The saturation value applied to γ_{cmd} is based on the maximum climb and decent capabilities of the airframe. The nonlinear block B compensates for the rotation velocity q measured by the IMU around the aircraft Y-axis, which points to the right, when the aircraft is not in horizontal flight. During a coordinated turn (no altitude change) with a bank angle ϕ the aircraft must rotate around the Y-axis with

$$q_{turn} = \frac{g}{V_k} \cdot \sin(\phi)\tan(\phi)$$

where g is the gravity. The third block is the elevator gain scheduling needed to compensate for the reduction of elevator effect at lower airspeed. The same applies to the course controller where the aileron gain is scaled according to the current airspeed.

The course controller is shown in Fig. 5. The nonlinear

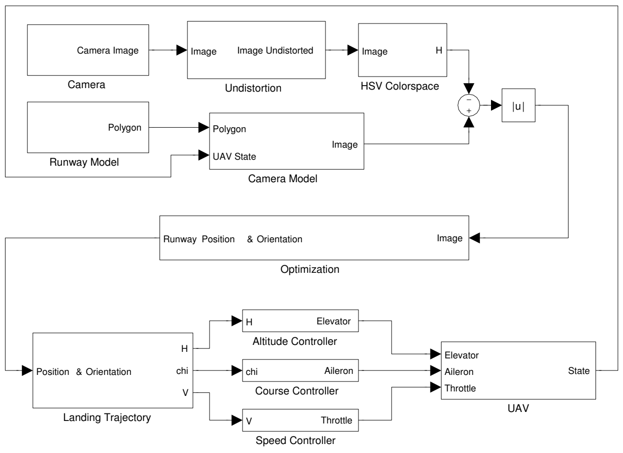

Fig. 3. System structure used for the automatic landing system

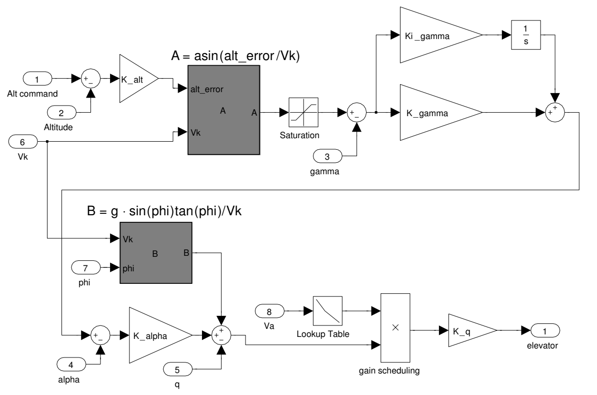

Fig. 4. Structure of the altitude controller

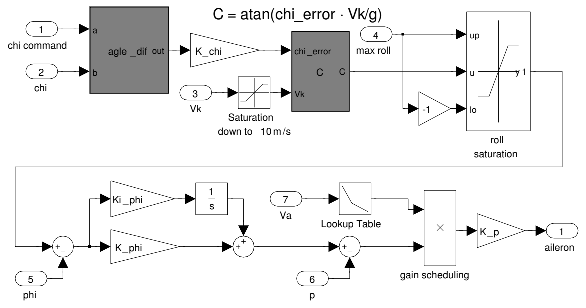

Fig. 5. Structure of the course controller

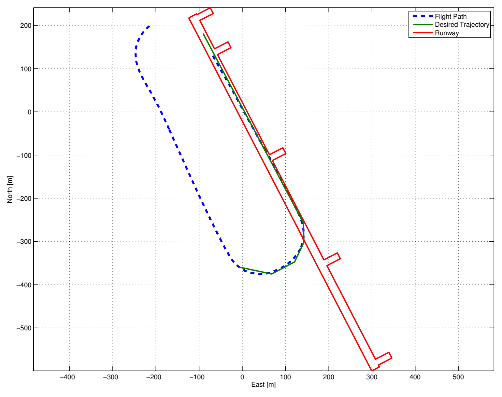

Fig. 6. A typical landing trajectory and the runway outline with taxiway exits

block C calculates the desired roll angle

$$\phi_{cmd} = \arctan\left(\frac{\delta\chi}{\delta t}\frac{V_k}{g}\right)$$

based on the desired course change rate $\frac{\delta\chi}{\delta t}$. The saturation of the desired roll angle ϕ_{cmd} is set according to the current flight phase. Different to the widely used altitude controllers for small size UAVs, in the proposed altitude controller we use one additional nested loop in order to increase the performance required for a precision landing.

The airspeed V_A is controlled with the engine throttle by a PI controller with a feed forward look-up-table derived from the desired airspeed V_{cmd}. The feed forward look-up-table is also adapted to the current flap setting. Feed forward gains for different airspeed and flap settings have been calculated from mean throttle values recorded during flight experiments at different airspeeds.

To prepare our controller for an automatic landing, different flight tests with different speeds and flaps settings were conducted in order to determine all required parameters. The initial values of parameters were calculated using the linearized system model.

This control architecture doesn't reflect all physical couplings between the control loops. E.g. the coupling between course and altitude controller when the aircraft is banked as well as the coupling between changes in altitude and airspeed. The presented architecture is a trade off between complexity and performance. It allows an easy set up and tuning because every loop can be disabled/enabled and tested separately. The disabled loops are controlled by a safety pilot. In addition, the roll angle during landing should be small so the coupling of altitude and course controller is minimal. With the altitude changes being small and constant during the final landing phase the speed and altitude control can work independently.

Commands for the controller are generated by a navigation module. The module has different modes which can be activated remotely. In flight mode it is able to guide the UAV by a list of way-points commanding the course to the next way-point and its altitude.

The landing mode is implemented in the block *Landing Trajectory* as a state machine, s. Fig. 3. The control loops for normal way-point flight and for landing are the same. As seen in Fig. 6 and 12 the landing has four phases. The landing trajectory can be modified by the following parameters:

- Approach turn direction (left turn, right turn)
- Glide path start altitude H_{glide}
- Glide path angle γ_{glide}
- Glide path airspeed V_{glide}
- Flare start altitude above the runway H_{flare}
- Flare flight path angle γ_{flare}
- Flare airspeed V_{flare}
- Position of the desired touch down point $\vec{X}_{land} = [x_{land}, y_{land}, H_{land}]^T$
- Heading of the runway χ_{runway}

The landing phases work as follows:

1) The - turn - phase is a 180° turn approximated by a few way-points that will lead the UAV to the beginning of the glide-slope. The turn way-points are modified based on the landing parameter. If the end of the turn is reached and the altitude is close to the beginning of the glide slope the next phase is activated. If not, the turn is repeated until the desired altitude is reached. This turn close to the start of the glide slope is needed to lead the UAV to the glide path from any approach direction where the landing was activated.
2) The – glide – phase will lead the UAV on a straight line along the runway center line and from H_{glide} down to H_{flare} with a constant flight path angle of γ_{glide}. The commanded altitude H_{cmd} is calculated based on the current distance to the desired touch down point r_{land}.

$$r_{land} = \left\| \begin{bmatrix} x_{land} \\ y_{land} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right\|$$

$$H_{cmd} = \tan(\gamma_{glide})\left(r_{land} - \frac{H_{flare}}{\tan(\gamma_{flare})}\right) + H_{flare} + H_{land}$$

This input to the controller results in a controlled decent on the glide path. The glide path speed V_{glide} is commanded and the flaps are extended to the landing position. If $r_{land} < \frac{H_{flare}}{\tan(\gamma_{flare})}$ the next phase is activated.

3) In this phase - flare - the desired path angle is reduced to the flare path angle γ_{flare} by commanding

$$H_{cmd} = \tan\left(\gamma_{flare}\right) r_{land} + H_{land}$$

and the commanded speed is reduced to the flare speed V_{flare}. The maximum allowed commanded roll angle is reduced to $10°$. This prevents touch down with a high roll angle. This phase continues until the UAV touch down on the runway which activates the last phase.

4) During the - roll - phase the desired speed is reduced to zero and the altitude and course control loops are disabled and the flaps are retracted. The UAV is now controlled with the rudder to keep it aligned with the runway during roll out.

The parameters of the landing can be changed in real time. This simplifies the tuning of the landing algorithm during flight experiments and is required for landing on a moving target where the desired touch down point $\vec{X}_{land}$ needs to be updated continuously.

IV. Runway detection

The main idea behind the presented runway detection algorithm is to use as much information of the image as possible to make the detection more reliable. By using a threshold to detect a line as done in [12] a lot of information of the image is lost. The usage of point features like presented in [8] is only possible with small changes of the view point compared to the reference image. This method does not work when we want to approach the runway from different angles as described in Sec. III. Our goal was to detect the runway from all viewing angles and over a large scale.

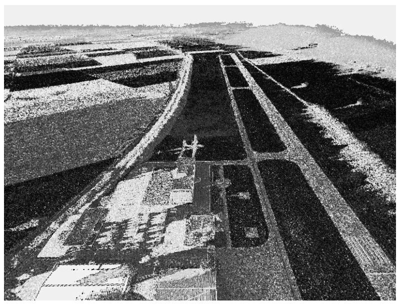

Fig. 8. Example of a shifted hue channel image of the runway

The runway detection is done using the image taken by a forward looking RGB camera mounted on the UAV. The runway detection algorithm uses the color difference between the runway surface and the surface surrounding it. The algorithm compares the real image with a computer generated image of the runway model. The algorithm is described in detail below.

The first step in the image processing is the undistortion of the camera image. The image needs to be undistorted so that we can apply the pinhole camera model in the following steps. The undistorted RGB image I is transformed to the HSV color space image $\hat{I} = [H, S, V]$. The H channel of $\hat{I}$ which represents the color of a pixel is subtracted by the color h of the surface surrounding the runway.

$$H' = |H - h|$$

An example of H' is shown in Fig. 8. We use an optimization to determine the exact position and orientation of the known runway by comparing a rendered image of the runway model with the camera image. The runway model we used is shown in Fig.6 It is a polygon of the outline of the runway including the taxiway exits. We use a bounding-box around the runway model to limit the image area that needs to be processed. The used bounding-box is shown in Fig. 7 as magenta colored lines around the detected runway model which is marked with green lines.

To generate an image of the runway we need to know the location of the camera in the inertial north-east-down frame:

$$C_C^I = C_F^I C_C^F$$

Here C_F^I is the homogeneous transformation of a point defined in the inertial frame I to the UAV fuselage frame F. The position and orientation of the UAV relative to the inertial frame can be measured by IMU and GPS. The camera is rigidly mounted to the fuselage so the location of the camera with respect to the fuselage C_C^F is known and fixed. The location of the runway in the inertial frame C_R^I is only known approximately and this is what we want to measure with the runway detection algorithm. To calculate the difference between the runway model and the camera image we transform the runway model points and runway model bounding box points $\vec{r_i}$ to image coordinates $\vec{p_i}$ using the pinhole camera model with the focal length f.

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \left(C_R^I P_n\right)^T C_C^I \vec{r_i}$$

$$\vec{p_i} = \frac{f}{Z_i} \begin{bmatrix} X_i \\ Y_i \end{bmatrix}$$

The runway model points $\vec{p_i}$ are then inserted as a filled polygon with the value 1 in a virtual image $\tilde{I}$. A score s_n for this parameter P_n is calculated as

$$s_n = 1 - \frac{1}{L}\sum_{l=1}^{L} \mid \tilde{I}(l) - H'(l) \mid$$

overall pixel l that are inside the projected bounding box. The number of pixels inside the bounding box is L.

In the runway detection case the parameters are:

$$P_n = \begin{bmatrix} \cos\left(\alpha_n + \alpha_0\right) & \sin\left(\alpha_n + \alpha_0\right) & 0 & x_n + x_0 \\ -\sin\left(\alpha_n + \alpha_0\right) & \cos\left(\alpha_n + \alpha_0\right) & 0 & y_n + y_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The value $\alpha_n = [-20 \ldots 20°]$ is the runway heading offset. The values $x_n = [-50 \ldots 50]$ and $y_n = [-50 \ldots 50]$ are the runway north/east offset. For each frame we calculate a number of s_n distributed around the initial parameters α_0, x_0, y_0.

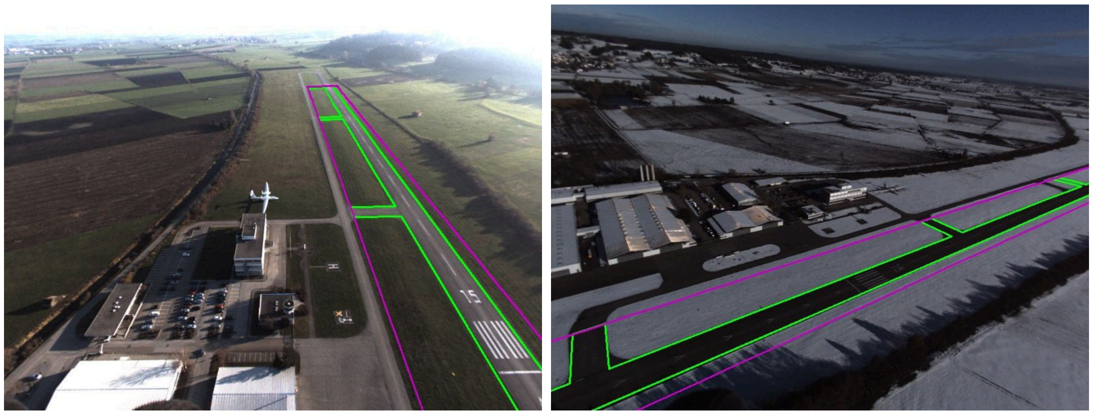

Fig. 7. Example images with detected runway recorded during flight experiments

If the object is detected the algorithm switches into the tracking mode. In the tracking mode a different number and distribution of the parameters is used. During detection we use $\alpha_n = \{-9°, -6°, -3°, 0°, 3°, 6°, 9°\}$ and $x_n = y_n = \{-27m, -18m, -9m, 0m, 9m, 18m, 27m\}$. During tracking $\alpha_n = \{-1.1°, 0°, 1.1°\}$ and $x_n = y_n = \{-3m, 0m, 3m\}$. The score s_n is calculated for every combination of parameters. If the highest s_n is above the threshold τ the model is detected with the parameter P_n.

Using $C_R^I P_n$ and the runway model the desired touch down point can be calculated and transmitted to the *Landing Trajectory* module as shown in Fig. 3. The landing trajectory is updated according to the detected runway parameter as described in Sec. III.

As long as the runway is in the field of view of the camera the position and heading of the runway can be measured. In case of a loss of GPS signal which is most likely to happen near the ground the landing does not have to be aborted because the position of the UAV can be estimated based on IMU measurements. The drift in position caused by integration of IMU measurements can be compensated using the runway position relative to the UAV calculated by the visual system. The altitude can be estimated by a pressure altimeter. Combing the image based runway detection results with a pressure altimeter, it is possible to land the UAV safely even when the GPS is lost in the glide or flare landing phase.

V. Experimental validation

Fig. 9. The UAV ELWIRA in flight

We used two different fixed wing UAVs for our experiments. The system described in Sec. II was integrated in both UAVs. The first UAV, FRAUKE, has a wingspan of $2.8m$ and total takeoff mass of $15kg$ which results in a flare speed of about $14\frac{m}{s}$. The second UAV, ELWIRA, shown in Fig. 9, has a wingspan of $4.5m$ and the total takeoff mass of $36kg$ which results in a flare speed of $19\frac{m}{s}$ during landing. Both UAVs have electrical propulsion.

To verify the performance of the altitude, airspeed and course controller we performed several flight experiments where the controller gains were tuned. The altitude controller was able to control the altitude within $\pm 1m$ around the desired value during straight forward flight and even during turns with up to $45°$ roll angle as shown in Fig. 10. The altitude and speed controller allow flights with high dynamics as shown in Fig. 10 where the altitude was changed in steps of $\pm 120m$.

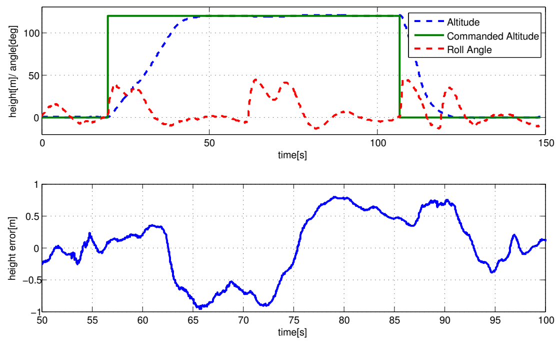

Fig. 10. The altitude controller performance in flight

We successfully tested the automatic landing without the aid of the vision system measuring the desired touch down position $\vec{X}_{land}$ and runway heading χ_{runway}. Using a high precision differential GPS it was possible to land both UAVs on specified positions. In our experiments we found that for a safe landing it is essential that all control loops are active until the UAV is on the ground. E.g. in [10] a flare maneuver is described where the engine is set to idle and this can lead to problems. In one of our automatic landing experiments we decreased the gains of the airspeed controller. These gain settings worked well during level flight, approach and glide but resulted in a loss of control during the flare maneuver. The airspeed dropped during flare and the airspeed controller did not reacted fast enough as shown in Fig.11. After the airspeed dropped below $12\frac{m}{s}$ the rate of decent increases rapidly and the resulting hard touch down with up to $2.5g$ from low altitude damaged the landing gear, propeller and caused minor damage to the fuselage.

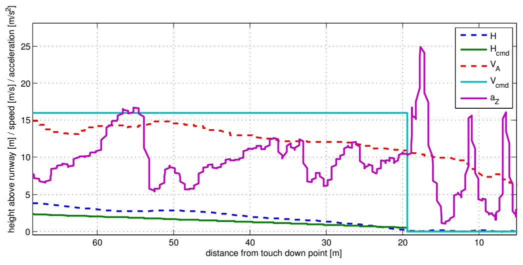

Fig. 11. Hard landing due to slow airspeed controller response

Several flight experiments have been performed to record image data of the runway from different view angles, altitudes and in different weather conditions. The recorded image data were used to test and adapt the runway detection algorithm offline. Examples of a successfully detected runway are shown in Fig. 7.

The presented automatic landing system was extensively tested with active vision system detecting and tracking the runway.

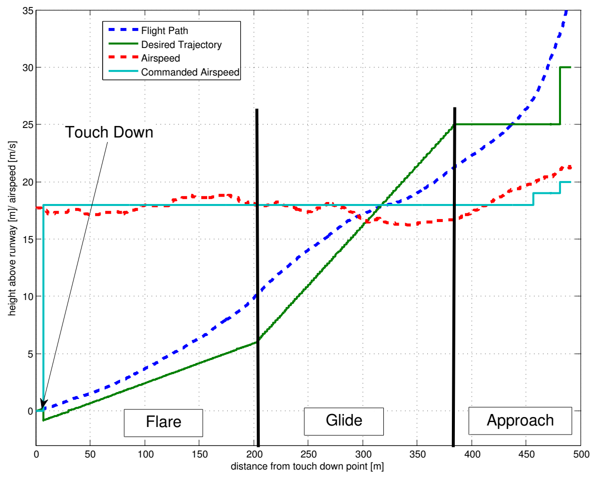

Fig. 12. Landing phase, altitude and speed profile during an automatic landing

The experiments were conducted as follows: During automatic waypoint following at an altitude of $140m$ above the runway, position and orientation of the runway were successfully detected by the vision system. The $\vec{X}_{land}$ and χ_{land} were calculated and transmitted to the *Landing Trajectory* block. The resulting landing trajectory had to be confirmed by the operator in the ground station. An example of the resulting flight path is shown in Fig. 6 and 12. The runway used for flight experiments is approximately $1km$ long, $20m$ wide and is surrounded by grass. As can be seen in Fig. 6 and 12, the system follows the calculated trajectory in all phases of the landing. At the end the UAV meets the predefined touch down point. The vision module was able to detect the runway and to track it down to an altitude of $5m$.

VI. Conclusions

We presented a multi-sensor automatic landing system by combining visual runway detection and GPS guided approach. The two main parts of the system are high precision aircraft controller and vision based runway detection algorithm. In experiments we demonstrated that the runway detection algorithm works in different weather and daylight conditions. This algorithm uses the maximum information contained in the image by processing a large portion of the image for runway detection. This makes it robust against partial occlusion, reflections or shadows which would be hard to overcome with edge or line detection as well as with other feature based approaches. The system was tested with different UAVs under different environmental conditions. The developed algorithms for trajectory generation and high performance control allow a precise landing even under not optimal wind conditions. The presented system for automatic landing can be easily adapted to other UAVs. Future work will be devoted to automatic landings on a mobile ground vehicle using the presented system.

References

[1] "Israel Aerospace Industries," http://www.iai.co.il.
[2] "General Atomics Aeronautical," http://www.ga-asi.com/.
[3] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *Robotics and automation, 2002. Proceedings. ICRA'02. IEEE international conference on*, vol. 3. IEEE, 2002, pp. 2799–2804.
[4] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 1720–1727.
[5] D. Mellinger, M. Shomin, and V. Kumar, "Control of quadrotors for robust perching and landing," in *Proceedings of International Powered Lift Conference,(Philadelphia, PA)*, vol. 1, 2010.
[6] R. Cory and R. Tedrake, "Experiments in fixed-wing uav perching," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2008.
[7] E. Dickmanns and F.-R. Schell, "Autonomous landing of airplanes by dynamic machine vision," in *Applications of Computer Vision, Proceedings, 1992., IEEE Workshop on*, Nov-2 Dec, pp. 172–179.
[8] A. Miller, M. Shah, and D. Harper, "Landing a uav on a runway using image registration," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May, pp. 182–187.
[9] T. F. Gonçalves, J. R. Azinheira, and P. Rives, "Vision-based autonomous approach and landing for an aircraft using a direct visual tracking method," in *Proc. Int. Conf. on Informatics in Control, Automation and Robotics*, 2009, pp. 94–101.
[10] P. Riseborough, "Automatic take-off and landing control for small uavs," in *Control Conference, 2004. 5th Asian*, vol. 2, July, pp. 754–762 Vol.2.
[11] M. Bernard, K. Kondak, and G. Hommel, "Framework for development and test of embedded flight control software for autonomous small size helicopters," in *International Workshop on Embedded Systems: Modeling, Technology and Applications*, 2006.
[12] J. Shang and Z. Shi, "Vision-based runway recognition for uav autonomous landing," *International Journal of Computer Science and Network Security*, vol. 7, no. 3, pp. 112–117, 2007.