Embedded Low Power Controller for Autonomous Landing of Small UAVs using Neural Network

Ahmad Din[1, 2], Basilio Bona[1], Joel Morrissette[3], Moazzam Hussain[1,3], Massimo Violante[1], Fawad Naseem[2]

[1]Politecnico di Torino,
Turin, Italy
[2]COMSATS Institute of Information Technology,
Islamabad, Pakistan

[3] Intel Corporation,
5200 NE Elam Young Parkway,
Hillsboro OR, USA

***Abstract*— we present real-time, stereo vision based autonomous landing system for small Unmanned Aerial Vehicles (UAV) onto an unknown landing target. The paper describes the algorithms and design of FPGA based co-processor implementing Artificial Neural Network (ANN) to implement real time object tracking, 3D position estimation using Visual Odometry(VO), Horizontal displacement and Euclidean distance from landing target. This approach doesn't require any explicit marker or landing target, it estimates attitude, track safe landing area, and compute distance and horizontal displacement form landing target. Experimental results show suitability of the real-time stereo vision landing approach using FPGA for tracking, that doesn't require any explicit landing marker.**

***Index Terms*— UAV, autonomous landing, neural network, FPGA implementation, real time object recognition, adaptive learning**

I. INTRODUCTION

Landing is the most critical and accident prone phase of the UAV flight. Remotely controlled landing requires an experienced pilot, because of limited situation awareness and lack of realism. Pilot relies on video taken by onboard cameras. Most of the accidents during landing are due to human factors. Therefore, automated landing is preferable, as it increases accuracy and do not suffer from fatigue. For Tactical UAVs, normally highly precise and expensive external aid is used for automatic landing, which is normally not available for small UAVs due to cost, target environment and complexity. Vision based automatic landing has got attention because it is passive, less expensive, light weighted and low power consuming.

In this work, we have proposed a novel framework for autonomous landing of Micro UAVs and Quad-rotors. We evaluate the building blocks and define factors affecting the performance. An evaluation on the suitability of neural networks for object detection and tracking during the landing phase of an autonomous Micro UAV has also been presented. Then we present FPGA (Field Programmable Gate Array) based real time architecture for neural network, and we evaluate the chip area requirements and performance numbers of the architecture.

Most of the Vision based landing systems detect known visual markers and compute relative pose estimation of the vehicle. In [1] and [7], "H"-shaped landing pad is detected and tracked for landing. black and white squared patterns used by [2] for rotary UAV landing. Another work presented in [3] used vision controller for pose estimation and elevated the autonomous landing using "chessboard"-shaped pattern. In [4], authors have designed a landing target with concentric white rings on a black background, it is designed in a way that it can be detected form both high and low heights. But the autonomous landing approach using known markers is not useful in situations like safe emergency landing or to determine area to deliver payload etc [16].

Another approach for autonomous landing technique is to go for without explicit marker or landing target, in such scheme, the system tracks safe landing area, estimate attitude and, measure distance of UAV from and landing pad using stereo camera. [26] calculates only height, while [5] and [6] have proposed algorithms to find safe landing area using monocular camera, but fell short of proposing full autonomous landing solution.

FPGA is widely used in UAVs for control [10] [11] [14] and, visual tracking and Object recognition [8] [9]. An efficient algorithm for object detection and tracking is implemented on FPGA for real time performance by [12]. In [13] FPGA is used for 3D perception and path planning. [15] Presents FPGA based hardware architecture for vision processing on UAV, but according to our knowledge no one reported FPGA based landing solution.

II. OVERALL FRAMEWORK OUTLINE

The basic idea of the paper is to track the landing area identified by the safe landing area module, and measure the Euclidean and horizontal displacement between landing target and UAV. Visual representation of the Problem is shown in fig. 2. Proposed method is implemented using C/C++ and Matlab® based on OpenCV, and landing area tracking is implemented in FPGA. For FPGA Development, data path design is accompanied by extensive validation on practical scenarios in the RTL simulations, then an FPGA realization of the proposed architecture is performed to be able to evaluate the area and power constraints of the system. The algorithmic level development and evaluation was initially performed using Matlab, Later an equivalent FPGA architecture is designed in Verilog HDL while using Matlab as a golden model for hardware verification (fig. 5). The overall architecture has been synthesized on Xilinx XC2V1000 FPGA and area/power analysis and computational capability of the proposed

978-0-7695-4927-9/12 $26.00 © 2012 IEEE
DOI 10.1109/FIT.2012.42

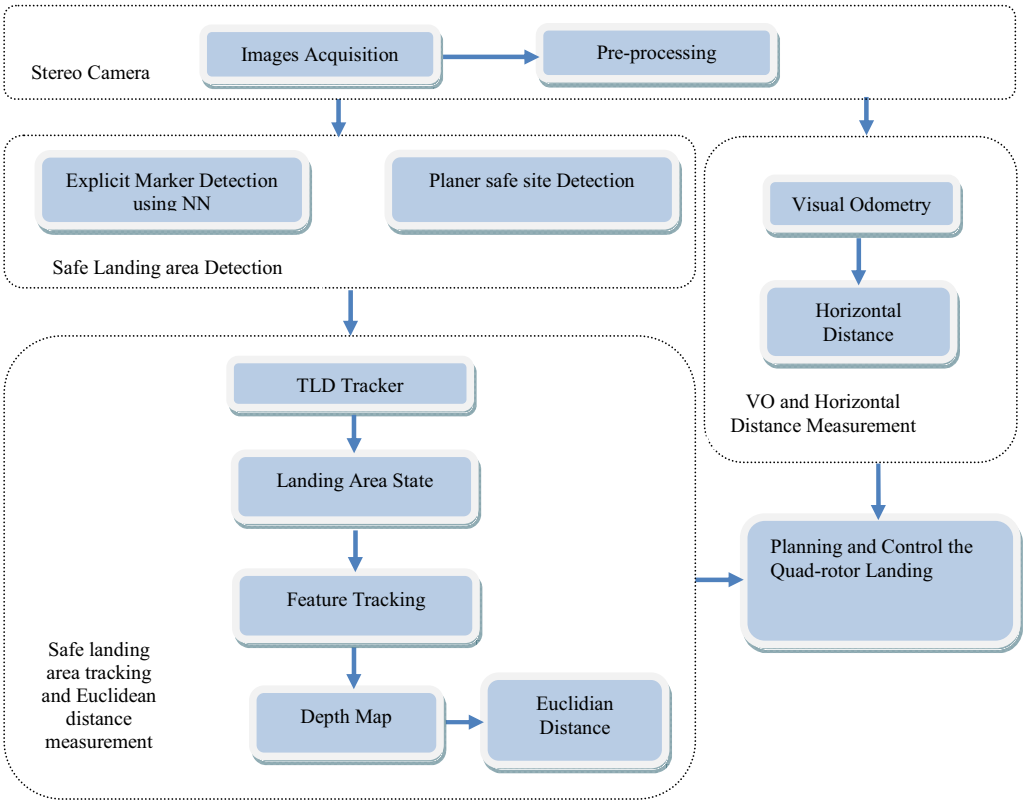

Figure 1: Algorithm for landing small UAV

architecture is presented in section IV. The real time matching capability of the FPGA architecture clearly indicates the usefulness of the framework. The prototyping board used as a proof of concept is equipped with XC2V1000 FPGA with onboard camera link based camera interface. The choice of a low end FPGA is deliberate to prove the limited area requirements of the proposed approach and its suitability towards low power implementation.

Algorithm for proposed method is shown in fig. 1. In first step stereo camera grabs the images, and left image is taken as a reference. In a second step there are two operations. a). Visual Odometry which estimates the relative position and rotation of the UAV and b). Tracking Landing target, it includes Safe land detection module, and safe landing area tracking and Euclidean distance measurement module, which is tracking safe landing area and measures distance between landing target and UAV.

At Visual Odometry module shown in fig. 3, image gain correction, calibration and image rectification is performed in calibration and pre-processing step. Then key-points are

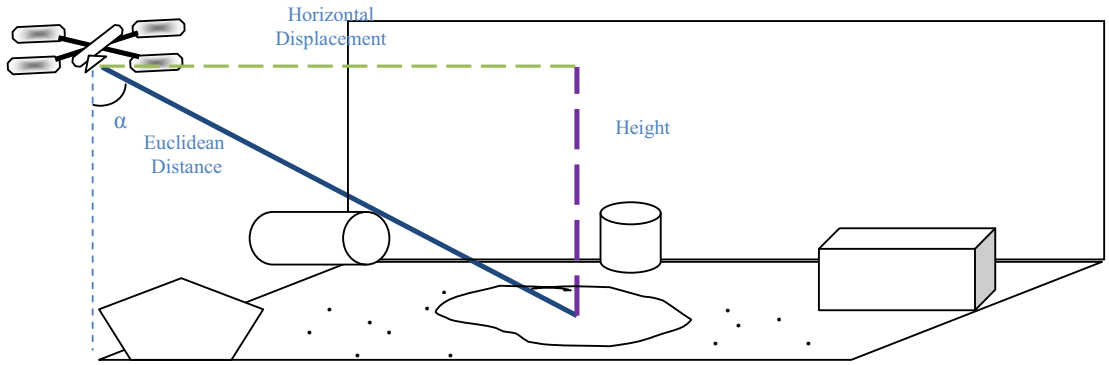

Figure 2: Vision based landing of small UAVs

extracted from left image using FAST [21], though there are many algorithms to perform feature extractions like SIFT, SURF etc. Since our algorithm is works in real-time i.e. 30fps, therefore FAST is best choice because it is fast and detects a lot key-points. Then, features/keypoints detected in left image are mapped/matched with corresponding features in right image. The feature in left image is searched in certain disparity limit like 16 ×16 pixels square box in right image. Normalized correlation is computed between feature in left image and all potential features in that box using Zero normalized correlation (ZNCC), a pair with largest correlations is considered as a matched pair.

Next step is tracking location of key-point from frame to frame. Kanade Lucas Tomasi (KLT) tracker is used to track because it is apply affine distortion model to key-points which helps performs better in large changes in appearance. Then, disparity is calculated, which is horizontal displacement between a particular feature in left and right image, that cross-pond same point in the real world. At Triangulation step, 3d points are calculated using disparity map. Outliers are rejected using RANSAC and Rotation and translation is estimated using 3point algorithm. In the end drift is reduced using bundle adjustment. The height calculated by the Visual Odometry Pose estimation is used to calculate the horizontal displacement between landing target and UAV.

At the Safe landing area detection Module, goal is to find an appropriate site for landing, which should fulfill following conditions [5].

1. The landing area is a flat and horizontal plane, without obstacle (like trees, boxes, rocks, tables, people etc.)
2. The landing area is big enough for UAV to land
3. The landing area must have sufficient features that it can be tracked

Normally, Explicit is marker is used for landing and normally the area where marker is place fulfill above given conditions for it is beneficial to search if there is any marker, the marker is searched using Neural network, but this marker is not used for landing rather just to find safe landing site. Other module is searching for safe site detection in parallel, If Neural network fails to detect any explicit marker then this module finds appropriate planner area for landing according to above mentioned criteria. Safe Landing area detection is part of ongoing research and we assume that safe area is already detected and it is a bounding box around that safe area is input into TLD tracker in tracking and Euclidean distance measurement Module. Landing area tracking is implemented in FPGA and explained in Section III.

At the, tracking and Euclidean distance measurement module, the TLD tracker [16] is tracking safe landing area from frame to frame. The TLD tracker has the uncanny ability of performing robust, long term tracking of unknown objects at much less computational cost. The TLD tracker divides the task of tracking in three steps of Tracking-Modeling-Detection[16]. The output of the TLD tracker is a bounding box representing a safe landing area is called "Landing Area State". This module also adjusts the size of the this bounding box, maximum size of this bounding box is defined, if the box size grows larger than threshold then it creates another box at the centre of that bounding box, which is equal less than threshold, It is important because when altitude of UAV decreases the size of safe Landing target will increase in size (when the distance decreases), therefore we target the middle of that landing target when UAV is approaching for landing. So the output of this step is a bounding box whose size is less than certain threshold. TLD tracker will track this bounding box instead of previously it was tracking.

In feature tracking module, features are detected, matched and tracked, in the state of landing area. The detail of this step is explained in Visual Odometry section. Depth map is calculated based on set of features tracked in previous step. Using depth map and geometry of camera, 3D position (X, Y, Z) of the point can be calculated. Camera model and parameter bare calculated during camera calibration. Using this information disparity can be turned into distance in the world coordinate as shown in fig. 3. L and R are two cameras of stereo setup and f is focal length of cameras. B is the distance between two cameras, it is called baseline. Projected coordinate of P on the left and right are denoted by $(x_r\ ,\ \ y_r)$ and $(x_l\ \ ,\ \ y_l)$. As the left is reference system, so the right camera

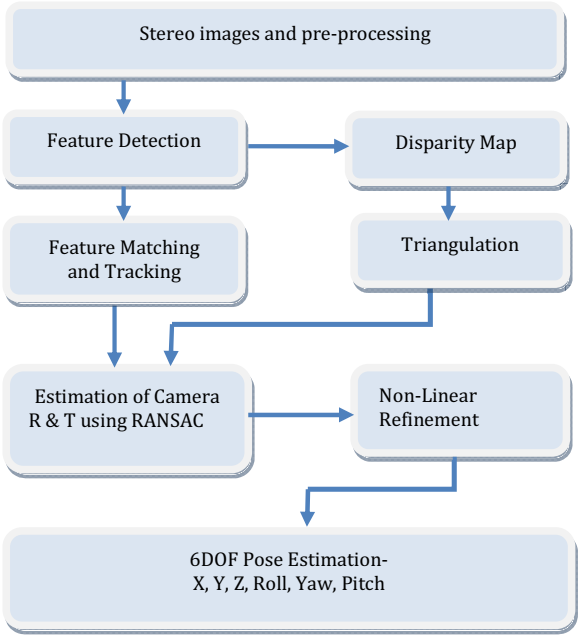

Figure 3: Stereo Visual Odometry (VO) pipeline

is rotated and translated with respect to left camera based on fundamental matrix [F], and then equations are.

$$\frac{B}{Z} = \frac{T + x_l - x_r}{Z - f} \qquad (1)$$

$$Z = \frac{fB}{x_l - x_r} \qquad (2)$$

Where $(x_l - x_r)$ is disparity d.
Then,

$$Z = \frac{fB}{d} \qquad (3)$$

As, we are using low cost and low quality camera, so some of the distances will be erroneous and will give constant value irrespective of the real distance from Landing state. We have filter out these erroneous distance by averaging and equation that gives actual distance using average distance and know distance from object.

$$Ed = \frac{(\sum_{i=1}^{n} Z_i)}{n}$$

Where, Ed is an average Euclidean distance of all points and n is total number of disparities found within the Safe land state. Using epipolar constraint, possible two-dimensional search in stereo images can be reduced to a one dimensional line. Horizontal displacement is calculated using trigonometric relations. It is calculated using pose estimation data give by

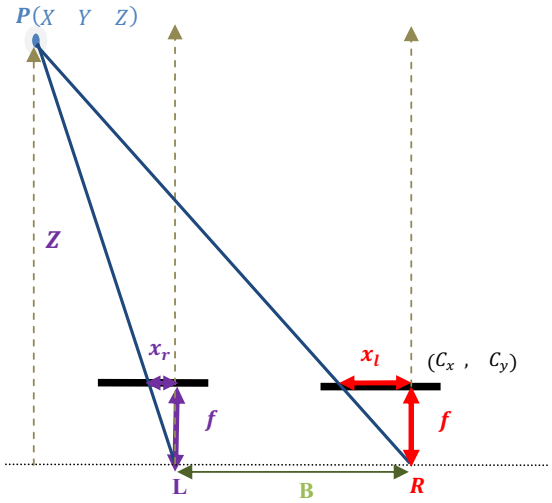

Figure 4: Distance estimation.

Visual Odometry (VO) and IMU (Inertial Measurement Unit). As height and Euclidean distance is know so horizontal displacement is calculated using simple trigonometric relations. It can also be calculated using height and angle α between height and Euclidean distance, this angle can be measured using IMU. Another option is, using angle α and Euclidean distance, height can be calculated, and then horizontal displacement is calculated using height and Euclidean distance. This method is more accurate than other methods, because uncertainty in Euclidean distance decreases, when UAV is approaching Landing target. Horizontal displacement may contains certain errors, but gives us useful estimation to move toward the landing target.

III. Algorithmic Dissection Of FPGA Based Tracking

The algorithmic level building blocks of automatic identifier and classifier are as follows:

4.1- Image Acquisitioing and pre-processing:

Images are acquired in real time from onboard camera based on camera link interface. The image is transmitted from the camera in the form of high speed LVDS interface and onboard decoder (National electronics DS90288) that translates the serial interface to a compatible format where the data is available on every positive edge of data. The FPGA pre-processing block is responsible for acquisitioning and storage of input frame in the memory and to perform set of operations to reduce noise and clutter. This block extracts features from the input frame using Hough transform based feature extraction logic. The UAV is assumed to have down-looking or side looking camera to be attached to the processing platform. [23] uses a front looking camera for obstacle avoidance and object recognition. If needed, the acquired image is pre-processed to generate a scaled and north-aligned image that is in-compatibility with the satellite training data. The added cost of computing affine transformation on the image helps in improving the results for the classification process.

4.2- Detection and Segmentation:

This block extracts the features of the acquired visual sensor data from the background. At this point, the model building of the input information does not take expected matching features. The goal is just to extract features irrespective of their potentiality of being a match. The presumed region of search based on features extracted from pre-processing block is evaluated for its candidacy as belonging to a potential target. The geometric features like corners, line segments are used a yardstick for definition of the target. This block must perform automatic feature extraction autonomously. The point distribution based classification of

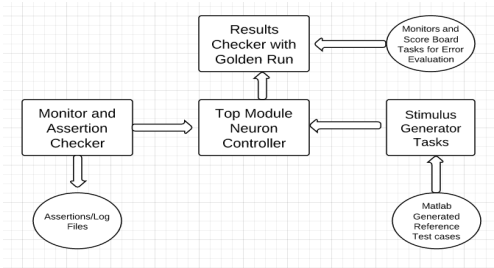

Figure 5: Neural Network RTL Development and Verification Methodology

the density estimation is affected by size of object, feature scaling and data dimensionality. The implementation is based on point model of object surfaces where the point presents an

object surface through a set of points. This model is used for its simplicity in real time implementation using an embedded platform. This model can be used as input to the neural networks based classifier that performs registration of measurement points on the reference points to evaluate the distance of the two signals for real time classification. The detected points are passed through a decimation filter to obtain set of optimal points that may be compatible with the active memory of the neural network where measurement of distance is performed using a non-linear classifier like KNN between the model feature and the measurement point set.

Apparently, the only available criterion for outlier identification is based on distance measurement and type of distance computation plays a pivotal role in this. As shown in the results section, it is obvious that the error definition based on distance cannot be the used as the only criteria of decision due to its non-zero false acceptance rate. It is recommended that a series of recognition may be performed and results may be updated for increased confidence and minimize the distance.

There are a few factors that add to the complexity of object identification involving object variations when viewed from different view angles, Vehicle attitude in 6-DOF, different shadows and lightening conditions that require a robust algorithm for feature extraction and matching, making it even challenging to implement in a real time low power hardware architecture. Our feature extraction relies on low computations and is possible using FAST corner detector [19]. FAST evaluates a set of pixels to classify it as a feature. If a small subset of sample data shows minimal variance, the pixel set is not classified as a feature. Rostern [19] perform 12 tests for 12-pixels to declare that set a feature and only two conclude that a feature does not exist in that. Due to its better response to the high frequency contents of the image (that has higher potential of being a feature), we used FAST with N=9. FAST algorithm relies on a set of if-else conditions and due to highly regular structure of the hardware design; a high level of hardware re-utilization is possible that reduces a lot of chip area. A reference C-implementation of FAST corner detector is available from the website that served as a golden reference of RTL development. The results from feature extraction are applied as a feature template to ANN.

4.3- Classification:

The feature extracted template is applied to classifier block that performs matching with target features. Using the methodology based on ANN, we have eliminated requirements for centroid calculation and many other feature extractions which was practically impossible in a real time system operating in a power constrained environment. The Target templates are pre-built that are based on complete or partial target information and is saved as a knowledge base. Compared to template based matching, the template based matching can be very sensitive to affine moments. On the contrary, if the extracted features are matched based on reliable features, the KNN based classifier gives minimal distance for similar objects and can confidently reject similarities amongst different classes. The neural network performs a series of matching by minimizing the distance between the learnt model features and measurement points.

4.3-B: Hardware Implementation of the Classifier

The mechanics of KNN function Classifier offers enormous computational flexibility and hence offers power and storage advantage over traditional linear search methods. The active memory cells can be loaded with new contents by a series of writes and the network can be armed to match a series of templates just in a fraction of time instances. The matching performance of classifier based architecture remains linear over the range of examples for recognition [22], which is unlike the processor architecture where the computational complexity grows exponentially with the amount of data to be matched primarily due to fixed number of cycles to perform match of the input sequence. The core of the architecture is based on associative memories. The association between input pattern $\{\mu1, \mu2, \mu3, \mu4, \mu5, \text{---} \mu n\}$ and the expected output for each sample $\{\Omega1, \Omega2, \Omega3, \Omega4, \Omega5, \text{----} \Omega n\}$ determine the weights in the neural network that are adjusted during the learning process and define the (mostly) non-linear relationship between input and output pattern. The network can produce associative pattern [17] and desired response pattern during the evaluation of input sequence that is similar to stored sequence.

FPGA based realization of KNN classifier offers high degree of parallelism and flexibility in numeric precision (depending on the application requirement). The weights of the neural network can be easily updated just by simple writes to the elements in the FPGA Block RAM. For very large patterns of neurons, the building blocks may be resource shared while the weights of the neural network may be stored in on-board SRAM.

The building blocks of ANN are as follows:

1- Input/output Data interface (the overall system is designed to be AHB compatible, the data interface is designed to be an AHB slave.

2- Distance Calculation (we used Euclidean norm for distance calculations). The hardware building of square root implementation is accomplished by square root approximation, that exponentially reduce the overall computation as compared to Cordic or Newton Raphson Method of square root core from the Xilinx[24]. Factors like numeric precision requirements can significantly affect the overall chip area. Floating point IEEE 754 presentation serves as reference expected numerical results and the hardware realization is based on fixed point arbitrary formats, where the size of the multiplier is reduced by dropping bits at input patterns and possibility of transforming a multiplier to nearest 2n to achieve it as a simple logic. In order to accomplish an optimized implementation, we used fixed point C-implementation of the Golden model for quantization noise analysis and defined optimum word length that has minimal effect on the system performance while offering significant area reduction.

3- Control Block: It is responsible for control of data flow in the real time application to the distance calculation block, fetching required data and updating weights. The length of feature vector is parameterized and this can be varied from 8 to 1024, based on the requirements of the application and available chip area on the FPGA, the neurons with a fixed depth of feature vector can be generated. For the sake of our implementation, we have implemented a group of 8-Neurons each with 128-Byte feature (implemented as a byte wide data path in FPGA to process gray scale data). The chip area is reported in the results section. The Building blocks are shown in Fig. 6.

As shown in fig. 6, above, the overall Neural network in FPGA is implemented as a set of 8-Neurons. These 8-neurons are constituted by 128-byte weights. Byte wide data storage for weights is chose to simplify the design and it compatibility with Gray Scale images. The length of the critical path in FPGA approaches to 85ns in XC2V1000 FPGA and limits the frequency of operation of the neural network to 11 MHz. This frequency is low as compared to the other design components yet is meets the performance number requirements due to parallel nature of the architecture with wider data paths. The control block manages the data to be applied to the network during the learning process and the results are read and sorted in the control block to define a match with the given data set. The KNN classifier primarily is based on presentation from Duda Hart[20]. The K-Nearset Neighbor classification rule is typically assumed to based on density estimation, yet the classifiers are very popular due to they simplicity of design and implementation and interpretation. The KNN based network allows all the neurons to fire for a sequence and relevant distance can be sorted to obtain the closest match. This may increase the overall complexity when dealing with massively

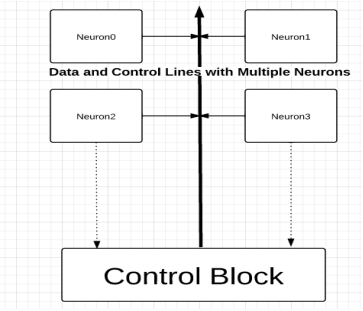

Figure 6: Arrangement of Neurons and Controller on a shared bus

large data sets with many neurons. The learning is performed in incremental steps. For each new applied pattern, the distance measurement defines whether to learn the pattern or not. Sufficient learning is mandatory before field deployment. Gaps in the space can result in a situation where model can never be discovered from the input pattern and set is simply rejected due to high distance values. The probability of such situation is pretty high due to the fact that we can see some false positives due to noise during image segmentation or feature extraction.

IV. RESULTS AND DISCUSSION

Stereo camera is calibrated using Matlab® toolbox for camera calibration and minrou camera calibration software, accuracy of distance measurement depends on calibration, large size of chessboard increases the accuracy.

Safe Land area is tracked using FPGA, in the preliminary set of experiments, we learnt 8-feature vectors corresponding to a landing stripe that could be recognized in real time by the ANN based matching engine. This requires the implementation of 8-KNN based neurons in the FPGA. The 8-neurons are allocated in the FPGA. Each extracted feature is matched with all set of neurons and the distance is calculated in real time. If the distance lies within the acceptable threshold, that would be an indicative of acceptable match. Depending upon the view angle, and vehicle orientation, more training data can be helpful. The probability of getting false acceptance is inversely proportional to the number of allocated neurons.

Matching efficiency with crude level of feature extraction is around 71%. Out of 5000 Patterns, we were able to match around 3478 with FAR of 8%. More training data can help improve the performance.

The block diagram of data flow and corresponding sample images are shown in fig. 6:

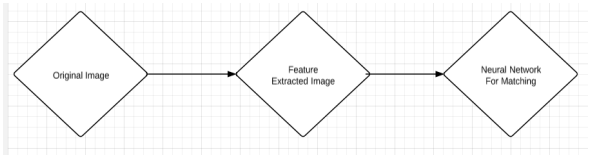

Figure 7: Image operations

Figure 8: a). Sample Original Image b). Features from FAST are marked

From the block diagram, it is clear that autonomous landing based on neural network and feature extraction capitalizes the edges of the runway as clear features and neural network look for similar pattern in the test image.

The FPGA based architecture allows efficient control over the cycle count of the application. In our architecture, we are able to achieve the completion of weight adjustment per byte of input data (for a new pattern during learning phase) in 31-clock cycles. The FPGA area report for 16-KNN based neurons is shown in Table 1 (on XC2V1000FG456 FPGA). The are requirement of these systems is strictly dependant on the requirement of numeric precision. Higher precision requirement can significantly increase chip area requirement that make it impossible to fit inside small FPGA. The square root implementation based on approximation has its own limitations and it can result in huge errors for large numbers.

For smaller numbers, however, like pixels gray scale, these limitations don't disturb the results. The Proposed framework has shown a lot of tolerance for scale variations when the feature distance is computed based on Absolute Norm. The variations in other affine moments like rotation limit the performance of the framework and may increase False Rejection if the movements are larger than 25 degrees in either direction.

Building Blocks of Architecture	FPGA Slice Utilization	BRAM Utilization	Top Level Pin Count	FlipFlops	FPGA Multiplier Blocks
8-KNN Neurons	3149/5120 (61%)	128Kbits/720Kbits (16%)	36/328 (11%)	4532/160K (2.8%)	8/40(20%)
Control Block	49/5120 (1%)	0%	--	26/160K (0.01%)	0%
Search and Sort Block	259/5120 (5%)	0%	--	431/160K(0.27%)	0%
FAST (Feature extraction)	1127/5120 (22%)	36Kbits/720Kbits	--	1127/5120 (22%)	36Kbits/720Kbits

Table 1. FPGA area for 16-KNN based neurons

The tracker detects the safe Landing area from frame to frame and output of the module is the bounding box as shown in fig. 9a, as UAV approaches target for landing and size of the target increases, when the size of the bounding box increases from Threshold size the Landing area module automatically decreases the size of the safe Landing area as shown in fig. 9b. Visual Odometry module is working in real time at 30 fps; timing required to execute each module on intel® core i5 is shown in Table 2. drift is reduced using key-frame selection and nonlinear refinement.

Step	Time
Image acquisition and pre-processing	3ms
Feature matching (500 features)	17ms
RANSAC	4ms
Refinement	1ms

Table 2: Running times for Visual Odometry Modules

Distance estimation experiments show that, different baseline measures different distances, and with wider baseline longer Euclidean distances can be measured form landing area with certain acceptable threshold error. When baseline is 0.06 m and distance is 0.6m the error is less than 0.05cm and for 1m error is 0.2m, for baseline 0.1m error is less than .0.5m for 1m and less than 0.1m for 2m. 0.3m baseline can be used to measure up to 6m with an acceptable error threshold.

V. CONCLUSION AND FUTURE WORK

In this paper novel stereo vision based landing system without any explicit maker or known helipad, is presented. Vision based system allows to indentify safe landing area , track safe landing state using FPAGA based tracker, UAV pose estimation and distance estimation between UAV and landing target. The software components mandate efficient coding to meet real time deadline while safe landing state tracking is primarily implemented in FPGA .This work describes a design and evaluation of a custom FPGA implementation of a KNN based classifier tacking Landing state. This evaluation can be used as a yardstick for further development. The caveats of feature extraction with limited available computational power can be resolved by working on low power yet better feature extraction and training data. The preliminary results presented are encouraging and we are working on further investigation and extension of our experiments. Our ongoing research includes automatic safe land detection using SLAM and semantic map. We are still developing FPGA realization of

Figure 9: a). Landing area state, b). After redefining Landing area state

some of the building blocks in this architecture, otherwise implemented using software modules. Integration in a Run time framework [25] with the given IP sets can help envision a complete framework for intelligent robotics. Such run time framework can help utilize low cost FPGA in implementing complex architectures using real time context switching and can be used as an enabling technology for rapid development of electronics for intelligent robotics.

REFERENCES

[1] Srikanth Saripalli, James F. Montgomer and Gaurav S. Sukhatme, "Vision-based Autonomous Landing of an Unmanned Aerial Vehicle," IEEE International Conference on Robotics and Automation (ICRA), , 2002; pp. 2799--2804

[2] Courtney S. Sharp Orriid Shakernia S. Shankar Sastry, "A Vision System for Landing an Unmanned Aerial Vehicle," IEEE International Conference on Robotics; 8. Automation,. May 2001; pp. 1720—1727

[3] C. De Wagter and J.A. Mulder, "Towards Vision-Based UAV Situation Awareness," AIAA Guidance, Navigation, and Control Conference, San Francisco, California, Aug 2005

[4] Lange, S., Sünderhauf, N., Protzel, P. A Vision Based Onboard Approach for Landing and Position Control of an Autonomous Multirotor UAV in GPS-Denied Environments. Presented at the Proceedings of the International Conference on Advanced Robotics, Munich, Germany, 22–26 June 2009; pp. 1-6.

[5] P. J. Garcia-Pardo, Gaurav S. Sukhatme, and James F. Montgomery, "Towards Vision-Based Safe Landing for an Autonomous Helicopter," In Robotics and Autonomous Systems, Vol. 38, No. 1, pp. 19-29, 2001.

[6] Sébastien Bosch, Simon Lacroix, Fernando Caballero: Autonomous Detection of Safe Landing Areas for an UAV from Monocular Images. IROS 2006: 5522-5527

[7] Zeng Fucen, Shi Haiqing, Wang Hong, "The Object Recognition and Adaptive Threshold Selection in the Vision System for Landing an Unmanned Aerial Vehicle," International Conference on Information and Automation June 22 -25, 2009, Zhuhai/Macau, China; pp. 117-122

[8] Spencer Fowers, Dah-Jye Lee, Beau Tippetts, Kirt D.sss Lillywhite, Aaron Dennis, and James Archibald "Vision Aided Stabilization and the Development of a Quad-Rotor Micro UAV" 2007 IEEE Int. Symp. on Computational Intelligence in Robotics and Automation, pp. 143-148, June 20-23, 2007

[9] Franz Andert, Florian Adolf: Online world modeling and path planning for an unmanned helicopter. Auton. Robots 27(3): 147-164 (2009)

[10] Dragan V.Lazic ,Waqar Shahid, "FPGA Based Longitudinal and Lateral Controller Implementation for a Small UAV," World Academy of Science, Engineering and Technology 70 2010

[11] W. Alvis, S. Murthy, K. Valavanis, W. Moreno, S. Katkoori, FPGA Based Flexible Autopilot Platform for Unmanned Systems, Control & Automation, 2007. MED '07. Mediterranean Conference on, pp. 1- 9

[12] Beau J. Tippetts, Spencer G. Fowers, Kirt D. Lillywhite, Dah-Jye Lee, James K. Archibald, "FPGA Implementation of a Feature Detection and Tracking Algorithm for Real-time Applications". ISVC 2007, pp. 682-691

[13] Allaire F.C.J., Mohamed Tarbouchi, Gilles Labonté, Giovanni Fusina, "FPGA implementation of genetic algorithm for UAV real-time path planning." J. Intell. Robot Syst. 54, 495{510 (2009)

[14] Ta-Ming Shih, Ho-Chung Chang, "FPGA based hardware in the loop test platform of small size UAV." CIRA 2009: 551-556

[15] Ehsan, S. and McDonald-Maier, K. D., "On-Board Vision Processing for Small UAVs: Time to Rethink Strategy," NASA/ESA Conference on Adaptive Hardware and Systems (AHS), California, USA, July 2009

[16] Sébastien Bosch, Simon Lacroix, Fernando Caballero, "Autonomous Detection of Safe Landing Areas for an UAV from Monocular Images", IROS 2006: 5522-5527

[17] Amos R. Omondi, Jagath C. Rajapakse, "FPGA Implementations of Neural Networks" Publisher: Springer ISBN: 0387284850

[18] Omer Cetin, Sefer Kurnza, Okayay Kaynak, "Fuzzy Logic Based Approach to Design of Autonomous Landing System for Unmanned Aerial Vehicles" International Journal on Robotics Systems 61 Page(s) 239-250

[19] Edward Rosten and Tom Drummond, "Fusing points and lines for high performance tracking". IEEE International Conference on Computer Vision 2005, Page(s): 1508-1511

[20] Richard E Duda, Hart, "Pattern Classification, 2nd Edition" ISBN-10: 0471056693

[21] Edward Rosten, Tom Drummond, "Fusing points and lines for high performance tracking.". IEEE International Conference on Computer Vision, October 2005, pp. 1508-1511

[22] www.cognimem.com

[23] Sai Prashanth, Arvind K Sujeeth, Ashutosh Saxena, "Autonomous Indoor Helicopter Flight Using Single Onboard Camera". International Conference on Robotics and Automation, ICRA 2011Df

[24] www.xilinx.com

[25] Hussain,M, Din Ahmad, Violante M, Bona B, "An adaptively reconfigurable framework for intelligent robotics", IEEE/ASME International Conference on Advanced Intelligent Mechatronics- AIM 2011 Page(s) 996-1002

[26] Kuo-Hsien Hsia , Shao-Fan Lien, "Height Estimation via Stereo Vision System for Unmanned Helicopter Autonomous Landing" , 2010 International Symposium on Computer, Communication, Control and Automation, pp 257 – 260