A Proposal of FPGA-based Low Cost and Power Efficient Autonomous Fruit Harvester

Kumar Nilay†, Swarnabha Mandal†, Yash Agarwal †, Rishabh Gupta†,
Sumeet Kumar†, Poojan Shah†, Sombit Dey†, Manthan Patel†, Annanya†

Abstract—**In this paper, we present a power-efficient and low-cost prototype of a robotic harvester which employs multiple subsystems such as fruit detection, odometry, localization, proficient manipulation through computer vision, deep learning and a novel end-effector design. Fruit Plucking is performed using an end effector, and 3-degree of freedom(DOF) arm (made out of the integration of two linear actuators and a rotating platform) consolidated with a 4-wheeled differential drive mobile platform. Effective implementation of the visual processing is executed on the FPGA Fabric of the Xilinx PYNQ-Z2 Board, which accelerates Deep Neural Networks (DNNs) with improved Latency and Energy Efficiency as compared to a CPU or GPU based implementation.**

Index Terms—**autonomous vehicle; autonomous fruit plucking; control system; object detection**

I. Introduction

The horticulture industry remains heavily manual labour intensive and is profoundly affected by labour costs. The growing and changing demands need to be met by the agriculture industry that is facing labour shortages and rising costs for labour work. Farmers are increasingly relying on technology to make their farms more productive. The significant setbacks they face is due to the exorbitant prices of the machines in the market. The shift of agricultural tasks to automation ha improved every phase of the agricultural process, which includes pre-harvest, to harvest and post-harvest steps. The significant contribution of our work is:-

- Development of a power-efficient bot which is enabled by FPGA, a low-power embedded system which shows optimization in both power consumption and performance in Deep Neural Networks (DNNs) applications.
- The biggest challenges for the cost-effective robotic-fruit plucker are detection, i.e. determining the location of each fruit and detaching them from the plant without harming the plant in the absence of costly sensors and processing unit.
- We demonstrate a simple and effective vision-based algorithm for fruit, a 3D localization, and grasp selection method.

The remainder of this work will cover our contributions to accomplish the challenging task: design of a mobile robot (Sec. 3); Embedded Architecture (Sec. 4); Navigation and Controls (Sec. 5); object detection and visual servoing (Sec. 6); and finally we conclude in (Sec. 7).

†Denotes equal contribution

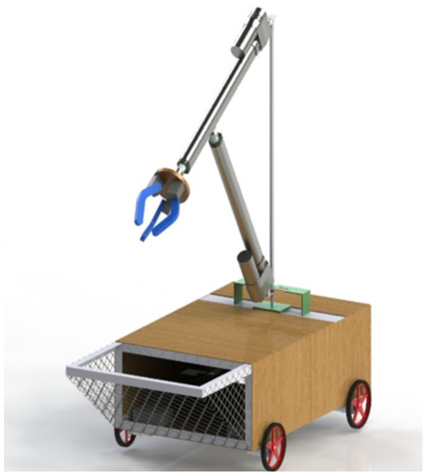

Fig. 1: A SolidWorks model of robot

II. Related Work

There has been quality research on the design and development of autonomous vehicle control system, fruit plucking and obstacle avoidance strategy.

In [1]similar visual servoing work is done where the bot is used to find a tool panel on a field, pick an appropriate wrench from the panel, and therewith operate a valve stem, where 3D point clouds acquired from a laser scanner are used along with Support Vector Machines (SVM) + Histogram of Oriented Gradients (HOG) object detector. In an earlier research project, a similar design of robotic arm (joint manipulator) was proposed [2]. The manipulator has been tested to avoid obstacles by curve fitting function and multiple DOF.

In [3], a feature-based approach for image-based fruit segmentation is proposed. The algorithm is tested for multi-class image segmentation for apple. The experiments shown in the paper classifies output to provide reliable apple yield estimation. In another paper [4], an autonomous robot attempted to harvest sweet pepper using a blade attached to the end-effector.

Different cycle times (fruit per second) and fruit picking efficiency (per cent of fruits reachable) was measured [5] using the kinematic model of the robot and dynamics of the manipulators respectively. Attempts have been made to reduce the cost of the robot by using low-cost stereo vision camera combined with a robotic arm [6]. Microsoft Kinect Sensor has been used in [7] for characterization of various small-structured vegetation structures.

In [8], different approaches are used for the accurate detection of fruits and to pluck them using a mechanical gripper tool. A monocular camera and a stepper motor are used in [9] for generating the depth maps of the environment.

III. Mechanical System

A. Design Overview

Robot's mechanical system consists of chassis, actuators and a 3-DOF robotic arm with an end-effector as shown in Fig 1. The chassis structure provides mounting points and protection for sensors and actuators. Assemblies are designed in SolidWorks, simulated using ANSYS finite element analysis (FEA) software.

B. Drive Mechanism

Locomotion of the robot is achieved by a four-wheel differential drive which can take a zero radius turn about its centre of mass with some drift. The bot is driven by four high torque DC geared motors of the specifications- 30 RPM, 12V and 60 kg-cm torque with metal gears having a shaft diameter of 6mm, length of 30mm.

C. Robotic Arm with end effector

The prototype is such that the robot can pluck fruits at different heights according to the requirement, as shown in Fig. 2. [10] The arm has twisted revolute, prismatic joints mechanism [2] to change the pitch and yaw angle and extend the reach of the arm. The arm extension velocity is constant, i.e. 1.4 cm/sec and the maximum extension achieved is 40 cm. The payload capacity of the actuator is 200 N. It provides a plucking height range from 44cm to 150cm from the ground. The bot uses a 3-D printed three finger gripper which offers the sufficient amount of grip on the fruits having the diameter in the range 4cm to 9cm. The pressure sensor is attached at the centre of the fingers so that the pressure applied does not cross the safety limit, which otherwise would crush the fruit.

IV. Embedded Architecture

A. Fast And Power Efficient Processor

The PYNQ Z2 ZYNQ works on 650MHz ARM Cortex-A9 dual-core processor. Overlay is used to accelerate the Neural Network on the FPGA fabric.

It is desirable to minimize the energy spent per image classification, which corresponds to maximizing the FPS per watt when many images are to be classified. Atmega 2560 is used for low-level interfacing of actuators and sensors.

TABLE I: Sensor specification

Microsoft Kinect	Depth stream range: 0.4–5m, Field of View(FoV): 43°vertical × 57°horizontal, 30 fps
Logitech C270	Field of View (FOV): 60°, Frame Rate: 30fps @ 640×480
Razor IMU	ITG-3200 (MEMS triple-axis gyro), ADXL345 (triple-axis accelerometer), and HMC5883L (triple-axis magnetometer) - nine degrees of inertial measurement
Origin GPS	NMEA protocol, Position determination accuracy: 2.5m, Supply Voltage: 1.8V
Rotary Encoder	2 Channel Quadrature Encoder having 2000 Counts Per Revolution
Pressure Sensor	1.75×1.5" sensing area, Force in the range 10g–10Kg

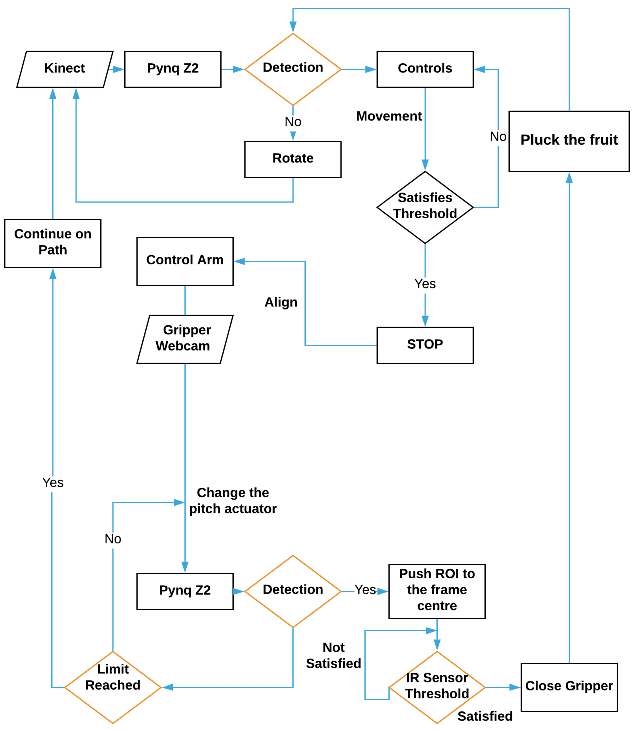

Fig. 2: Overview of plucking algorithm

V. Navigation And Controls

A. Localization

Data from multiple sensors are used to estimate the state of the bot using a sensor fusion technique. The Localization module fuses the data from the encoders(one on each wheel), IMU and GPS using Extended Kalman filter (EKF) [11] to estimate the state vector. defined as:

$$X = [R,\ \dot{R}]^T$$

where $R = [x,\ y,\ z,\ \psi,\ \theta,\ \phi]$ and $x,\ y,\ z$ are the spatial coordinates and $\psi,\ \theta,\ \phi$ are rotation angles along x, y and z-axis respectively. EKF is often employed to estimate the states of the object with high accuracy even in the presence of sensor noise. We have used the EKF implementation available in robot localization, a ROS package, which takes the sensors data and measurement uncertainty to estimates the state vector of the vehicle. More detailed information on this can be found in Thrun et al [12]. Fig 3 displays the path traversed by the robot and Fig 4 shows the closed loop error of the estimate from the robot localisation.

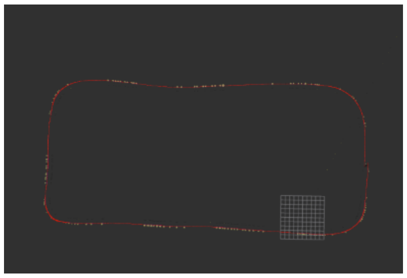
Fig. 3: Localisation track and loop closure

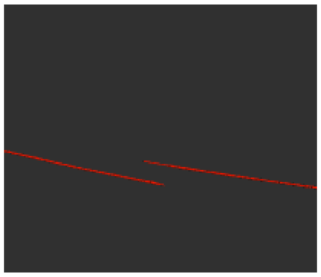
Fig. 4: Loop closure error

B. Path Tracking

The robot maneuvers on a predefined path, specific to the farm, which is produced by manually driving the robot before deploying it in autonomous mode. The geometric path is generated, taking into consideration the estimated state space of the robot and the plantation rows.

1) Pure Pursuit: Pure pursuit tracking algorithm [13] has been used for tracking the predefined path. The real pursuit method is the most common geometrical method used for path tracking. It determines a goal point (g_x, g_y) on the path based on a fixed look-ahead distance from the centre of the differential drive mechanism, and calculates the radius of curvature of the arc that connects these two points. By applying the law of sines in Fig. 5

$$\frac{l_d}{\sin(2\alpha)} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)}$$

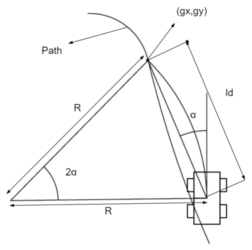

Fig. 5: Pure Pursuit

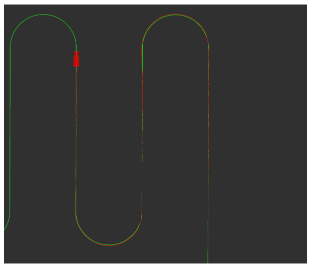
Fig. 6: Predefined Path

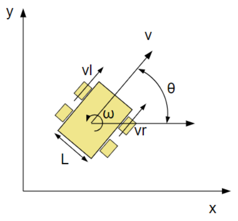

Fig. 7: Differential Kinematics

$$\frac{l_d}{2\sin(\alpha)\cos(\alpha)} = \frac{R}{\cos(\alpha)}$$

$$\frac{l_d}{\sin(\alpha)} = 2R$$

2) Differential Drive Kinematics: Using the value of R and the longitudinal velocity V, ω of the bot is calculated. Since the bot has a differential drive, the given longitudinal velocity and ω is converted into individual velocities of the left and right motors (Fig. 7)

$$\omega = \frac{V}{R}$$

$$V_r = \omega(R + \frac{l}{2})$$

$$V_l = \omega(R - \frac{l}{2})$$

Where l is the distance between the left and right wheels.

A PID controller [14] has been implemented for either side to achieve the desired velocity as calculated above. The independent implementation of the two controllers ensures efficient path tracking. In the PID loop, we continuously monitor the angle between the heading vector of the bot and the position vector of the fruit w.r.t the CG of the robot. Once this angle is in a specific range of about 60, the velocity of the bot is set to zero to stall the movement of the bot on the predefined trajectory, and the control is switched over to the robotic arm controller for fruit plucking.

C. Robotic Arm Control

Assuming the position of the fruit (x, y, z) w.r.t. the base of the arm and A being the total length of a linear actuator then using eq (1) we can identify whether the fruit is within the reach of the gripper or not.

$$A_{min}{}^2 < (x - A \times sin(\alpha_z) \times sin(\alpha_y))^2 + (y - A \times sin(\alpha_z) \times cos(\alpha_y))^2 + (z - A \times cos(\alpha_z) - 65)^2 < A_{max}{}^2 \quad (1)$$

The position of the nearest fruit is used to determine the target pitch and yaw angle of the robotic arm, which is calculated from the data of Kinect.

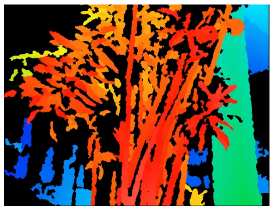
Fig. 8: Disparity Map

Fig. 9: RGB Image

VI. Fruit Detection and Visual Servoing

Detection of fruits and finding their real-world coordinates is the fundamental task necessary for fruit plucking. The key elements for succeeding in this task are correct detection of fruit, precise positioning of the detected fruits, and reliable visual servoing.

As the main focus of our work is to keep the bot as cost and power efficient as possible, we have used Microsoft Kinect for generating the disparity map and hence generating precise coordinates of the objects with respect to the camera frame.

A. Binary Neural Network

A Binary Neural Network (BNN) [15], runs on the FPGA [16] via an overlay to classify any kind of subject (fruit) present in the image. A sliding window detector is then used to determine the position of the subject by giving the centre coordinate of the fruit detected in the frame. BNN architecture was trained on fruit classes of MS COCO [17] data-set (95.8% classification accuracy). Images were reduced to 64×64 images with 24 bits/pixel to feed into BNN which contains six 3×3 convolutional layers, three 2×2 max pooling layers and three fully connected layers with 512 neurons each.

B. Determination of Global Coordinates of Detected Fruits

The Kinect has a monocular camera and two IR cameras that enable us to determine the disparity map (Figure 8) of our frame under consideration. Kinect is mounted at the front part of the robot, which constantly takes video feed at 30 frames per second(FPS), which is then transferred to the PYNQ-Z2 for further processing. The previously mentioned BNN [15] based detector gives the position of all detected subjects in the frame in the form of bounding boxes. Location of our subjects in terms of coordinates with respect to the centre of the camera is obtained through a simple transformations from the Kinect's raw disparity data.

1) Getting Object Position w.r.t Kinect: The Kinect is mounted on the bot at a vertical inclination of 15 degrees. The following transformations have been used to determine the position the detected objects with respect to the plane of the inclined camera.

$$z = b \times f/(1/8 \times (doff - kd)) \tag{2}$$

$$x = z \times (u - w/2) \times 1/f \tag{3}$$

$$y = z \times (v - h/2) \times 1/f \tag{4}$$

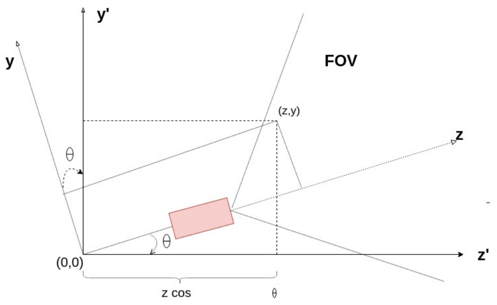

Fig. 10: Angle of Inclination

Where *(x,y,z)* denotes the position of the object with respect to the camera at an inclined plane of projection.

kd : Kinect's raw disparity
f : focal length of camera
doff : offset value particular to a given Kinect device, typically equal to *1090*.
b : baseline constant of Kinect= 7.5 cm.
(u,v) : position of center of detected object w.r.t screen.
(w,h) : resolution parameters of Kinect.
The factor 1/8 appears because the values of *kd* are in 1/8 pixel units.

2) Accounting the Inclination and Determination of Global Coordinates: Considering the inclination of the camera, we are now required to determine the position of the detected object taking into account the tilt of the camera. Then, a simple trigonometric transformation allows us to determine the absolute global position of the object:

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} z' \\ y' \end{bmatrix}$$

Where *(x',y',z')* represent the projected coordinates after accounting the inclination.
Consider the centre of the camera to be *m* ahead and *n* above the centre of the bot. Through the localization module we can determine the position of the centre of the robot with good accuracy. Suppose the position is *(a,b,c)*.

$$X_w = a + x' \tag{5}$$

$$Y_w = b + n + y' \tag{6}$$

$$Z_w = c + m + z' \tag{7}$$

where (X_w, Y_w, Z_w) are the global coordinates of the detected objects.

VII. Results and Discussions

- The localization module was tested for the loop as shown in (Fig 3). Loop closure error was found to be around 0.65m Fig. 4).
- The path tracking algorithm was tested on a predefined path (Fig 6), and the cross-track error was plotted against distance(m) which is shown in (Fig 11). A maximum deviation of 0.3m was observed from the path and the average deviation was around 0.2m.

TABLE II: Robot's Plan Of Action

Fruit Selection	Among the detected candidates, the nearest one is brought to the lower threshold value of the reach of the end effector and its coordinates are stored. Then the bot stops and stores the world coordinates of all the candidates within the upper threshold in a queue. Then bot proceeds to pluck the nearest fruit by precise alignment of the arm with the nearest fruit.
Fruit Tracking	Once the nearest fruit is selected, it is approached by the gripper using visual servoing. Our method uses image-based visual servoing (IBVS) control technique, similar to the method used in [1].
IBVS and Neural Network	IBVS is used to align the gripper with the fruit before grasping. During alignment, proportional control is used along with the BNN model [14] to ensure alignment.
Fruit Alignment	Once the fruit is aligned at the center of the frame, the linear actuator extends the end effector and approaches the fruit.
Fruit Plucking	As soon as the fruit touches the pressure sensor attached on the gripper, then based on the threshold value, the gripper closes and plucks the fruit by retracing of the linear actuator.

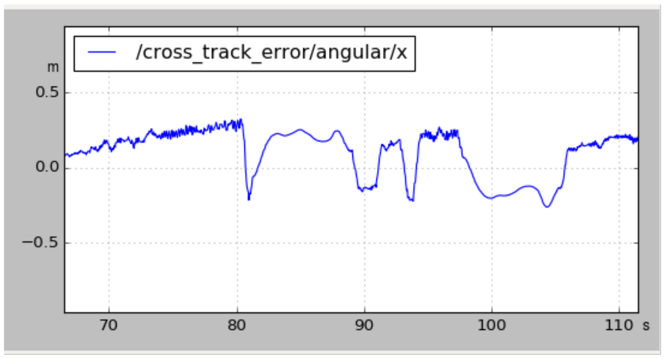

Fig. 11: Cross track error

- The accuracy on a testing set consisting of real-time images was observed to be 95.8%.
- The current implementation resulted in roughly 30 FPS output.
- The time to pluck one fruit ranges from 40-45 seconds on an average.
- The robot was tested for 15 test runs in the self constructed farm having 28 fruits. An average of 18.67 fruits were plucked with a standard deviation of 3.79 fruits.

VIII. Future Work

The developed prototype has performed up to expectations in trials. We further aim to develop this prototype into a robust agricultural robot which could reproduce trial results in Indian fields. We target to implement changes such as track drive instead of differential drive, Soft grippers instead of Hard grippers and addition of extra modules like Pesticide Spraying, Seeding. We also aim to implement several changes in the design of robotic manipulator and platform, which would lead to the addition of extra DOF. Hence we would be able to pluck fruits like apples, guavas.

References

[1] J. Carius, M. Wermelinger, B. Rajasekaran, K. Holtmann, and M. Hutter. *Autonomous Mission with a Mobile Manipulator. A Solution to the MBZIRC*, pages 559–573. 01 2018.

[2] Z. De-An, L. Jidong, J. Wei, Z. Ying, and C. Yu. "Design and control of an apple harvesting robot". *Biosystems Engineering - BIOSYST ENG*, 110:112–122, 10 2011.

[3] C. Hung, J. Underwood, J. Nieto, and S. Sukkarieh. *A Feature Learning Based Approach for Automated Fruit Yield Estimation*, pages 485–498. Springer International Publishing, Cham, 2015.

[4] C. Lehnert, A. English, C. Mccool, A. Tow, and T. Perez. "Autonomous sweet pepper harvesting for protected cropping systems". *IEEE Robotics and Automation Letters*, PP:1–1, 01 2017.

[5] R. Arikapudi, A. Durand-Petiteville, and S. Vougioukas. "Model-based assessment of robotic fruit harvesting cycle times". *American Society of Agricultural and Biological Engineers Annual International Meeting 2014, ASABE 2014*, 7:5098–5104, 01 2014.

[6] D. Font, T. Pallej, M. Tresanchez, D. Runcan, J. Moreno, D. Martinez, M. Teixido, and J. Palacn. "A proposal for automatic fruit harvesting by combining a low cost stereovision camera and a robotic arm". *Sensors (Basel, Switzerland)*, 14:11557–11579, 07 2014.

[7] G. Azzari, M. Goulden, and R. Rusu. "Rapid characterization of vegetation structure with a microsoft kinect sensor". *Sensors (Basel, Switzerland)*, 13:2384–98, 02 2013.

[8] Z. De-An, L. Jidong, J. Wei, Z. Ying, and C. Yu. "Design and control of an apple harvesting robot". *Biosystems Engineering - BIOSYST ENG*, 110:112–122, 10 2011.

[9] B. Billiot, F. Cointault, L. Journaux, J. Simon, and P. Gouton. "3d image acquisition system based on shape from focus technique". *Sensors (Basel, Switzerland)*, 13:5040–5053, 04 2013.

[10] H. Yaguchi, K. Nagahama, T. Hasegawa, and M. Inaba. Development of an autonomous tomato harvesting robot with rotational plucking gripper. pages 652–657, 10 2016.

[11] T. Moore and D. Stouch. "A generalized extended kalman filter implementation for the robot operating system". In *IAS*, 2014.

[12] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[13] J. M. Snider. "Automatic steering methods for autonomous automobile path tracking". Technical Report CMU-RI-TR-09-08, Carnegie Mellon University, Pittsburgh, PA, February 2009.

[14] C S. Shijin and K Udayakumar. "Speed control of wheeled mobile robots using pid with dynamic and kinematic modelling". pages 1–7, 03 2017.

[15] M. Courbariaux and Y. Bengio. "BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1". *CoRR*, abs/1602.02830, 2016.

[16] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. A. Vissers. FINN: "A framework for fast, scalable binarized neural network inference". *CoRR*, abs/1612.07119, 2016.

[17] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common objects in context". In *ECCV*, 2014.