THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Design and Implementation of Object Recognition System Using SoC and FPGA

Truong Cong Vinh
P. G. Scholar, Department of Electronics and Communication, Engineering, Hindustan University, Chennai, India
Dr. Vanaja Shivakumar
Professor & HOD, Department of Electronics and Communication, Engineering, Hindustan University, Chennai, India
M. Rajmohan
Assistant Professor, Department of Electronics and Communication, Hindustan University, Chennai, India

Abstract:
Object tracking and recognition have a wide range of applications. Recent years, many projects have been done in this field. It is easy to see that its application plays an important role in medical and biomedical image processing, geographical information, industry image analysis, earth science, or satellite based military. Feature-based algorithms like Speeded Up Robust Features (SURF), and SIFT (Scale-invariant feature transform) are well suitable for such operations [3]. Among these, Oriented fast and Rotated BRIEF (ORB) has been proved to achieve optimal results. ORB is based on the well-known fast key point detector concept Binary Robust Independent Elementary Features (BRIEF) descriptor [2]. Nearest neighbour matching algorithms that are Fast Library for Approximate nearest Neighbours (FLANN) [1] which is known as one of the best algorithms for features matching. Simulation has been carried out on Window platform using Opencv-Python. This project also provides an idea to implement ORB algorithm on SoC (ARM–BCM2835) to increase the execution speed and reduce the complexity in system design. By applying Opencv-Python, which provides a rich library for image processing applications, time consuming and the complication in developing software can be reduce remarkably. Also, the proposed object recognition system is empowered with Altera FPGA CYCLONE III which is believed to handle all control signals to and from external devices.

1. Introduction

Object recognition is one of the most fascinating abilities that are far from difficult for humans to process. Only with a simple glance of an object, even a child is able to identify and tell about that object and its characteristics. Moreover, humans can easily generalize from observing objects to recognizing objects that they have seen them the first time. A good evidence for this is kids can easily generalize the concept of table, chair or car after observing just a few examples. However, it is significantly difficult to design a vision system that meets the cognitive capabilities of human beings or a system that can identify and tell exactly about the object that is being observed. The problems come from the factors such as pose of an object in front of the camera, light variation, and so on. The recognition process is conducted by matching features of a test object against model objects. The process consists of three main key stages which are image acquisition, image analysis, processing and decision. Image acquisition is the first component in the system, and its task is to capture video signal and give output signal to the next component of the system. The output signal is known as video signal which is measured by frames/second (f/s). The second component is called processing unit which can be a microcontroller, DSP, FPGA or even a SoC. This unit is responsible for analysis, processing frames (images). In detail, the processing unit will perform operations such as image compression, image enhancement, image restoration, morphological, segmentation, and object recognition.

It has been shown that when designing object recognition systems, engineers are facing with many problems in system design and software development. This comes from the requirements of the capacity of the memory both data memory and program memory, and the desired speed as well. Besides, the task of developing software is not a pice of cake in this case. Luckily, the problem of hardware can be overcomed by using the SoC BCM2835 and the workload in software development can be eased with the help of linux and Opencv-Python.

2. Object Recognition System

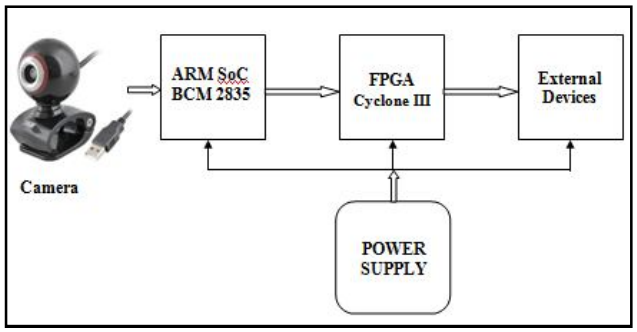

Figure1: Object Recognition System

The objective of this work is to implement the object recognition system on the SoC - ARM BCM2835 and FPGA – Cyclone III. This is a good combination in hardware. Where the SoC is handling all the tasks from capturing video sequence to object recognition, and FPGA is responsible for giving control signals to external devices. ARM BCM2835 running with the speed of 800MHz can be a good choice for handling hue task like image processing. Besides, the FPGA – Cyclone III is also known as a high performance chip. Moreover, the ORB algorithm and FLANN are applied, This is a good reason for a belief in a system with high accuracy and high performance. The main elements in the system are indicated as in Fig.1.

3. Techniques Used

To detect an object, a data base is required. In this case, the data base is model images. Normally, the data base contains hundred to thousand of images. The data base should be fulfill the aspects of the object such as the object in different angles, light conditions, distances from the image to the camera, etc. From the data base, features of the model images need to be found and calculated. This has been done with the help of ORB (Oriented FAST and Rotated BRIEF) algorithm. The video signals will be captured by the camera, and the of Features Detector & Descriptor algorithm SIFT is also applied on each frame to find and calculate the features. After finding the features of both model images and frames, the matching algorithm will match the features between model images and the the features from frames and give the last decision whether the object is matched or not.

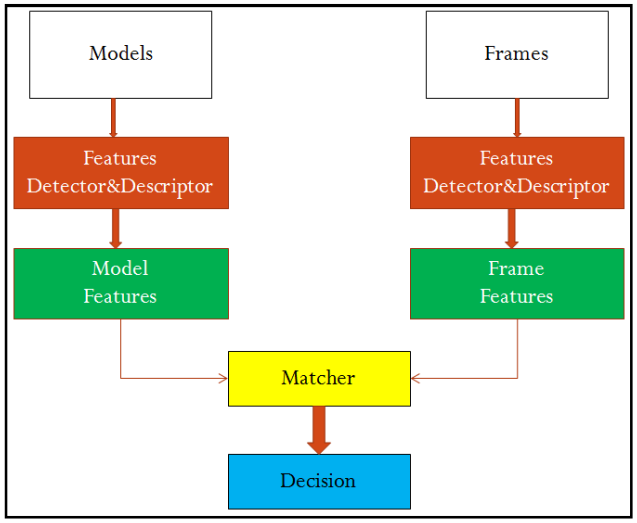

Figure 2: Steps to Recognize an object

- Step 1: Features detection and description is performed by Oriented Fast and Rotated BRIEF (ORB) Algorithm ORB is a simple method but it is effective in measuring corner orientation, and the centroid of the intensity. The moments can be defined as:

$$m_{pq} = \sum_{k=0}^{n} x^p y^q I(x, y) \quad (1)$$

The centroid can be found by using the following formula

$$C = \left(\frac{m10}{m00}, \frac{m01}{m00}\right) \quad (2)$$

A vector is constructed from the corner's center, O, to the centroid OC. The orientation of the patch is defined as:

$$\theta = \text{atan2}(m01, m10) \quad (3)$$

where, atan2 is the quadrant-aware version of arctan.

3.1. The algorithm

1. Run each test against all training patches.
2. Order the tests by their distance from a mean of 0.5, forming the vector T.
3. Greedy search:
   a. Put the first test into the result vector R and remove it from T.
   b. Take the next test from T, and compare it against all tests in R. If its absolute correlation is greater than a threshold, discard it; else add it to R.
   c. Repeat the previous step until there are 256 tests in R. If there are fewer than 256, raise the threshold and try again.

- Step 2. Matching features is done with the help of scalable Nearest Neighbor Algorithm

The nearest neighbor that performs the searching operations in a metric space can be defined as follows:
Consider a given set points p={p_1,p_2,…….p_n} in a metric space M and a query point q € M , to search the element NN(q,P) ϵ P that is the closest to q with respect to a metric distance MxM →R:

$$NN(q,P)=argmin_{x\epsilon P} d(q,x) \quad (4)$$

The K-nearest neighbor search more formally in the following manner:

KNN(q,P,K)=A (5)

where A is a set that satisfies the following conditions:

|A| =K, $A \subseteq P$ (6)

$$\forall x \in A,\ y \in P\text{-}A,\ d(q,x) \leq d(q,y)$$

The radius nearest neighbor search can be defined as follows:

RNN(q,P,R) = {p∈ P,d(q,p)<R} (7)

Radius K-nearest neighbor (RKNN) search is a combination of K-nearest neighbor search and radius search, where a limit can be placed on the number of points that the radius search should return:

RKNN(q,P,K,R) = A (8)

such that:

|A| ≤ K, $A \subseteq P$ (9)

$\forall x \in A$, $y \in$ P-A, d(q,x) < d(q,x) R and d(q,x) ≤ d(q,y) (10)

Algorithm Description:

The k-means tree is built by clustering the data points at each level into K separate regions using k-means partitioning, and then applying the same method recursively to the points in each region. The recursion ends when the number of points in a region is smaller than K.

3.1.1. Algorithm 1: Building the Priority Search k-means Tree

Input: features dataset D, branching factor K, maximum iterations I_{max}, center section algorithm to use C_{alg}
Output: K-means tree
if |D| < K then
Create leaf node with the points in D
else
P ← select K points from D using the C_{alg} algorithm
converged ← false
iterations ← 0
while not converged and iterations < I_{max} do
C ← cluster the points in D around nearest centers P
P_{new} ← means of cluster in C
if P = P_{new} then
Converged ← true
end if
P ← P_{new}
iterations ← iterations + 1
end while
for each cluster C_i ∈ C do
create non-leaf node with center P_i
recursively apply algorithm to the points in C_i
end for
end if

3.1.2. Algorithm 2: Searching the Priority Search k-means Tree

Input: k-means tree T, branching factor K, query point Q, maximum number of point to examine L, number of neighbors K
Output: K-nearest approximate neighbor of query point
Procedure search K means tree(T,Q,L,K)
count ← 0

PQ ← empty priority queue
R ← empty priority queue
Call traverse K means tree (T, PQ, R, count, Q)
while PQ not empty and count < L do
N ← top of PQ
Call traverse K means tree (N, PQ, R, count, Q)
end while
return K top points from R
Procedure search K means tree(N,Q,L,K)
if node N is a leaf node then
search all points in N and add them to R
count ← count + |N|
else
C ← child node of N
C_q ← closest node of C to query Q
C_p ← C\C_q
add all nodes in C_p to PQ
call traverse tree (C_q, PQ, R)
end if

3.1.3. Algorithm 3: Building One Hierarchical Clustering Tree

Input: feature dataset D
Output: hierarchical clustering tree
Parameters: branching factor K, maximum leaf size S_L
if size of D < S_L then
create leaf node with the points in D
else
P ← select K points at random from D
C ← cluster the points in D around nearest center P
for each cluster $C_i \in C$ do
create non-leaf node with center P_i
recursively apply the algorithm to the points in C_i
end for
end if

4. Results and Discussion

The simulation of features detector has been done by using Opencv-Python on Window platform. The simulation of Features detector is done by applying ORB algorithm the simulation result is shown as the following figures.

Figure 3: ID card

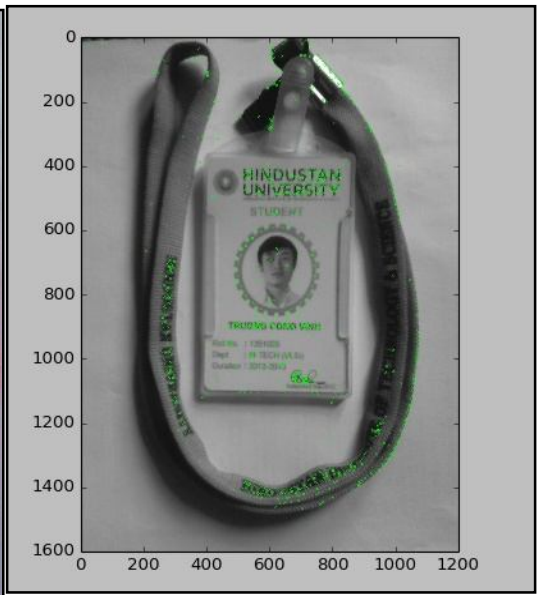

Figure 4: ID card with features detected

Figure 5: Model image the KIT

Figure 6: Frame with the appearance of the KIT

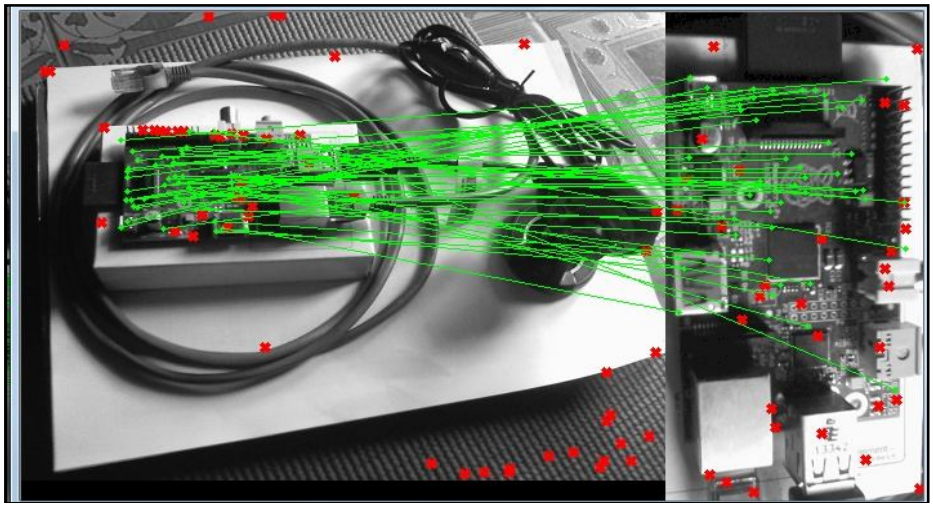

Figure 7: Features matching between object in data base and object in front of the camera

The lines indicate the matching features between the model image which is the KIT (on the left right side) and the KIT in the frame.

4.1. The Synthetic Test Set with Added Gaussian Noise of 10

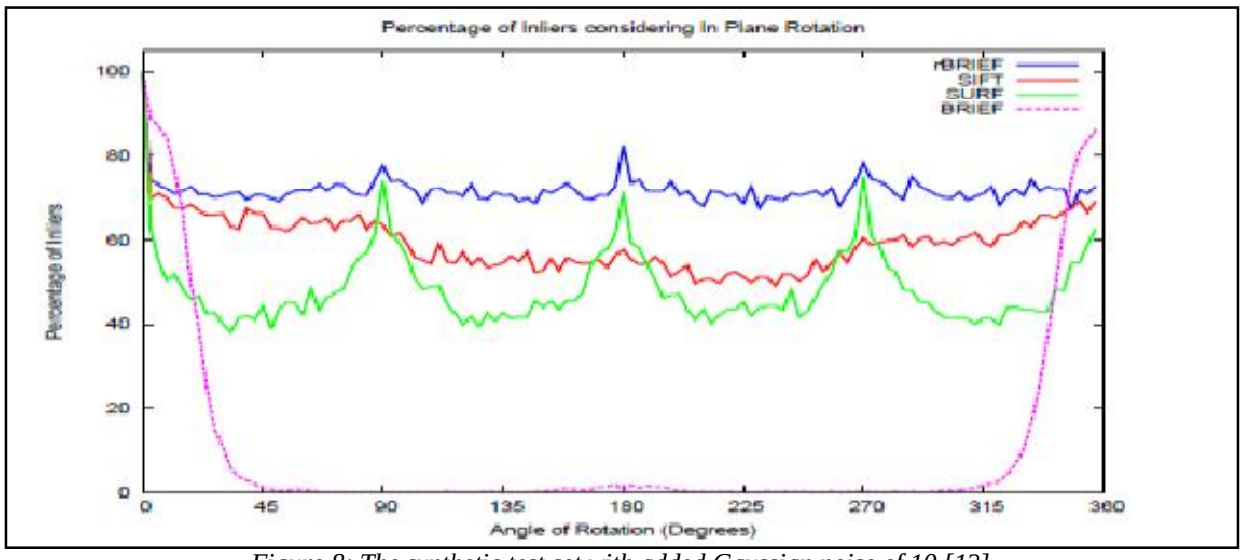

Figure 8: The synthetic test set with added Gaussian noise of 10 [12]

4.2. The Inlier Performance vs. Noise

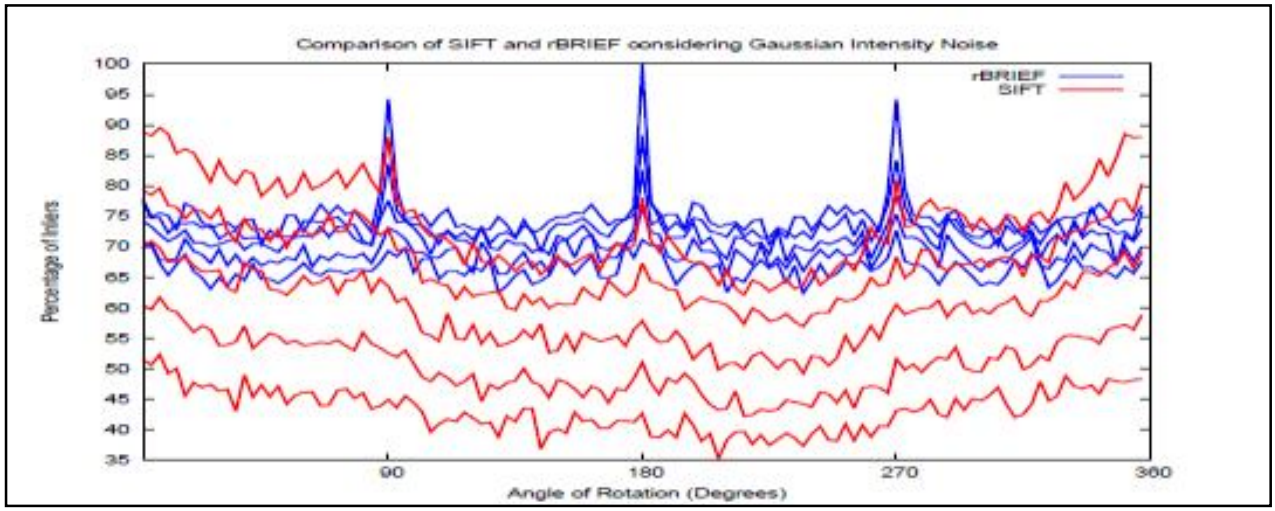

Figure 9: The inlier performance vs. noise, at different noise levels [12]

4.3. Comparison of SIFT, SURF and ORB

Method	Time per frame	Noise immunity	Rotation	Indoor Data Matching	Outdoor Data Matching
SIFT	Low	Moderate	Moderate	Moderate	Moderate
SURF	Moderate	Low	Low	Moderate	Moderate
ORB	High	High	High	Moderate	High

Table 1: Comparison of SIFT, SURF and ORB

5. Conclusion

The goal of the Object Recognition System is creating a system which has ability to track and recognise desired objects in live video signals. The paper aims at providing a final product that in all probability would be a source of potential applications. For example, product defect detection in industry, security system, geographical information, earth science, or satellite based military.

There are two main key steps. The first one is features detection and description which has done by ROB algorithm. As can be seen, the features of the ID card are found as shown in the Fig.4. The second process is matching features which has been done with the help of FLANN algorithm. The result is shown in the Fig.7 where the features of the model image and the image in the frame.

For the future work, the implementation on the SoC BCM 2835 and FPGA should be done.

6. References

i. Scalable Nearest Neighbor Algorithms for High Dimensional Data, IEEE, VOL. 36, No. x, xxxx 2014.
ii. Joseph Howse, OpenCV Computer Vision with Python, Packt Publishing, Inc., 2013.
iii. SIFT Based Approach: Object Recognition and Localization for Pick-and-Place System, ISSN: 2277 128X, Volume 3, Issue 3, March 2013.
iv. G. L. Foresti, Object Recognition And Tracking For Remote Video Surveillance, IEEE Trans. Circuits Syst. Video Technol., 9(7):1045-1062, October 1999.
v. Reza Oji, an Automatic Algorithm for Object Recognition and Detection Based On Asift Keypoints, Signal & Image Processing: An International Journal (SIPIJ) Vol.3, No.5, October 2012.
vi. Hongying Men, Appiah K, Hunter A, Dickinson P, "FPGA implementation of Naive Bayes classifier for visual object recognition," Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on, vol.,no., pp.123,128, 20-25 June2011.
vii. David Katz, Rick Gentile, Fundamentals of Embedded Video Processing, Analog Devices, Inc., 2008.
viii. Karan Gupta, Anjali V. Kulkarni, "Implementation of an Automated Single Camera Object Tracking System Using Frame Differencing and Dynamic Template Matching" 2012 IEEE,vol.,no.,pp.20,80,July20.
ix. James R. Armstrong, Chip –Level Modeling with VHDL, Prentice Hall 1989.
x. Zainalabedin Navbi, VHDL Analysis and Modeling of Digital System, McGraw Hill, Inc., 2nd edition, 1997.
xi. Karim Yaghmour, Jon Master, Gilad Ben-Yossef, Phillipe Gerum, Building Embedded Linux Systems, O'Reilly Media, Inc., 2nd edition, 2009.
xii. Ethan Rublee Vincent Rabaud Kurt Konolige Gary Bradski Willow Garage, Menlo Park, "ORB: an efficient alternative to SIFT or SURF", California{erublee}{vrabaud}{konolig}{bradski}@willo wgarage.com