International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Image Reconstruction and Per-pixel Classification

Gondhi Navabharat, Reddy,Sruthi Setlem

Abstract: ***We describe face classification algorithm which can be used for object recognition, pose estimation, tracking and gesture recognition which are useful for human-computer interaction. We make use of depth camera (Creative Interactive Gesture Camera – Kinect®) to acquire the images which gives several advantages when compared over a normal RGB optical camera. In this paper we demonstrate a intermediate parsing scheme, so that an accurate per-pixel classification is used to localize the joints. We make use of an efficient random decision forest to classify the image which in turn helps to estimate the pose. As we employ depth camera to acquire depth image it may contain holes on or around depth map, so we first fill those holes and the classify the image. Simulation results was observed by varying several training parameters of the decision forest. We generally learned an efficient method which stems the basics in the development of pose estimation and tracking. Also we gained an intensive knowledge on Decision forests.***

Keywords: Depth map, Decision Tree, RDF, Per-pixel Classification, Weak learner function, Entropy.

I. INTRODUCTION

In the last few decades, accurate estimation of pose has received a lot of attention in many fields like animation, gaming [19, 2], human-computer interaction, robotics [21], hand tracking systems [3, 20], gesture recognition [17, 18], sign language recognition [22], security systems and many other commercial applications [1, 5, 13]. Despite of extensive research and efforts in the field of hand pose or body pose estimation, it still remains a challenging problem, which is mainly due to the complex nature of human body articulations. The parsed parts are very useful high level features for pose estimation and gesture recognition. Hence, the first step in either object recognition or pose estimation is effective classification of the image into parts also called as Per-Pixel Classification. In our paper we attempt to classify the face into eight different regions and by using Classification forests [16] we train and test the acquired images.

Now let us discuss about some major problem in body pose estimation like the use of optical cameras. In many previous studies optical camera is used as the input [17] which makes it more difficult in discriminating the face parts in the cluttered background. As the face is quite homogenous in color, it will make the processing steps more complex.

Manuscript received on March 15, 2020.
Revised Manuscript received on March 24, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Gondhi Navabharat Reddy*, Assistant professor, ECE, Vignan Institute of Technolgy and science(VITS).
Sruthi Setlem, Assistant professor ECE.

To overcome this problem we make use of a high speed depth sensor which has greatly simplified the task of parsing by providing several advantages over colour camera such as it can work in low light conditions, help remove ambiguity in scale, colour and texture, and also resolve silhouette ambiguities. The use of depth camera also helps in the pre-processing steps of our algorithm by simplifying the task of background elimination. However, the depth image from Kinect has some holes due to reflection and occlusion on or around object boundaries. The depth image provided by Kinect has resolution up to 640×480 and the color image has resolution up to 1280×960. There are several hole filling approaches are proposed, based on inpainting [23]. Matyunin et al. [24] proposed method uses motion information to enhance the temporal stability of depth image and used color information of corresponding objects to fill hole areas. Yang et al. [25] used depth information of neighboring pixels to fill hole regions. Face parsing scheme can be seen as a classification problem, hence we thought of making use of an efficient thus highly successful Random Decision Forest (RDF) [7, 12] to solve our classification problem.

This paper is based on the classification forest which has been a core framework in the development of commercially successful Microsoft Kinect [2] gaming system for real time tracking of human body. We are employing an identical classification forest to estimate the body pose by parsing. The aim of our research is to design a system which is robust and computationally efficient. Towards our aim this paper presents an algorithm for estimating pose by parsing using RDF which will classify the image into 8 different regions, C={background, forehead, eyes, nose, mouth, chin, ears, cheeks} [11].

Figure 1. The labelled distribution for different body parts

Given a depth image obtained from an kinect® depth camera we wish to say which part each pixel belongs to. This is a typical job for a classification forest. In our algorithm we have considered 8 different classes as mentioned earlier. The unit of computation here is a single image pixel and its depth feature, x coordinate and y coordinate.

Initially we have to construct a tree structure based on training images and weak learner function along with its parameters which will be explained in the following sections. The RDF uses simple depth comparison features which gives 3D translation invariance while maintaining high efficiency in computation [4]. The forest has to build in such way that it should be able to run on any consumer hardware in real-time leaving sufficient memory to allow any other complex application and graphics to run in parallel [10].

In this paper we validate the use of RDF classifier for parsing scheme without exploiting temporal constraints. We investigate the effect of various training parameters and specify their importance. We begin with hole filling algorithm in Section II, introduction to Random Decision Forest in Section III, then we describe our method in Section IV and finally the simulation results along with conclusion is described in Section V and VI respectively.

II. HOLE FILLING ALGORITHM

For classification we consider x coordinate, y coordinate and depth value at that pixel, Now if there are any holes on the depth map then at that pixel depth value will be zero and if we classify then we may not get the correct classification so it is necessary to fill all the depth holes and use the filled image for classification. Depth holes are usually caused by reflection from black objects like glasses etc. and occlusions and steps for depth hole filling is as follows.

1) Apply Gaussian filter to remove noise from the image.
2) Extract the edge maps from the image using sobel operator.
3) Because the depth holes are usually caused by reflections or occlusions the information for depth hole filling will be around hole region boundary.
4) Expand the depth hole region boundary by using a 5×5 diamond structured element.
5) Find histograms on the depth hole region boundary by
$$h^L(d_k) = n_k^L, 0 \le d_k \le 255$$
6) The depth value with number of pixels greater than 9% of the total number of pixels in the expanded hole region boundary is considered as one dominant depth value of a depth hole region.
7) Threshold depth value of a depth hole region is determined by average of all dominant depth values.
8) The median of the dominant depth values which are greater than threshold depth value is used to fill all pixels in hole region.

This filled image is used for classification and the algorithm is explained in next sections.

III. RANDOM DECISION FOREST MODEL

The popularity of decision forests [8] is mostly due to their recent success in classification tasks. Decision forests [14] are used to solve problems related to the analysis of complex data such as text, photographs, videos and medical images that can be further categorized into small set of machine learning tasks: Classification, regression, density estimation, manifold learning, semi-supervised learning and active learning. A decision tree is a set of questions organized in a hierarchical manner and represented graphically as a tree which helps to split complex problems into simple ones. Figure 2. gives a general tree structure.

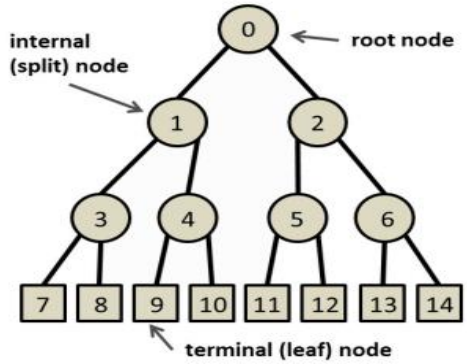
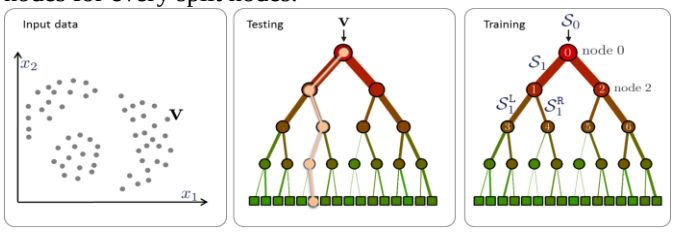


Figure 2. A general tree structure

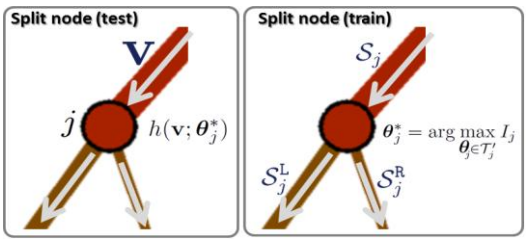
A tree is a kind of graph in which the data structure is made of a collection of edges and nodes organized in a hierarchical manner. Nodes are subdivided into terminal/leaf nodes and internal/split nodes. The first node is root node which is divided into split nodes and ends up at the leaf nodes. Here we have constructed a binary tree i.e., two child nodes for every split nodes.



(a) (b) (c)

Figure 3. (a) Data-points in 2D, (b) Testing, (c) Training

Let us consider an example to explain in detail about the working of decision forest given in the Figure 3, where a two dimensional data-points are the input units for computation. During training a decision tree sends all the data-points along with the classified label as a training set 'S_0' into the root node of the tree and moves the training data either left or right by optimizing the objective function(i.e., Information Gain) and each terminal node will have a predictor model. During testing a split junction/node applies a test on the input data point '*v*' and sends it to the appropriate binary branch either to the left child or right child based on the stored learned parameters. This process is repeated until leaf node is reached.



(a) (b)

Figure 4. (a) Split node (testing), (b) Split node (training)

The split node is associated with a weak learner function (or split function, or test function) given by,

$$h(v,\theta_j): F*T \rightarrow \{0,1\}$$

where, F: set of all data feature responses (2D in this example) and T: set of all possible θ_j's.
Training the parameters θ_j of node j involves maximizing the information gain I_j.

$$\theta_j^* = \arg\max_{\theta_j \in T} I_j$$

When training trees it is convenient to think of training points as subsets being associated with different tree branches. In the Figure 4, we can observe the subset representation. In binary trees following properties are applied:

$$S_j = S_j^L \cup S_j^R$$
$$S_j^L \cap S_j^R = 0$$
$$S_j^L = S_{2j+1}$$
$$S_j^R = S_{2j+2}$$

for each split node j.

Tree testing (on-line): Starting at the root node, each split node applies its associated weak learner or split function $h(.,.)$ to 'v' until the leaf node is reached. The leaf node contains a predictor or estimator which associates an output with the input 'v'.

Tree training (offline): The split function stored at the internal nodes are key for the functioning of the tree. One may think of designing these split functions manually, but for more realistic problems it needs to be learned automatically. The optimal split function parameters are chosen such that it maximize the objective function I_j. Tree growing can be stopped at a point when a node contains very few training data points. Hence, at the end of the training phase we obtain:

(i) The optimal split function associated with each node.
(ii) A learned tree structure.
(iii) Different set of training data points at each leaf node.

The basic building blocks of the training objective function are the knowledge of entropy and information gain.

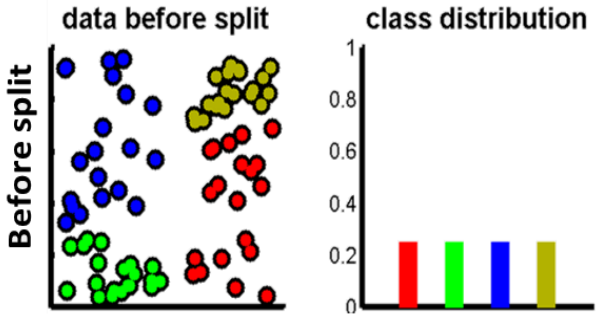


Figure 5. Dataset S before a split.

The empirical distribution over the class label when considered the whole image is found to be uniform as we have exactly same number of points for each class. This says that the entropy of the training data sets is rather high. For a discrete probability distribution we can use the Shannon's entropy given by,

$$H(S) = -\sum_{c \in C} p(c)\log(p(c))$$

where, $p(c)$ gives the probability distribution of the training points within the set S. We know that from information theory, entropy of a system is inversely proportional to the information content of the system. Hence, to have a maximum information gain we should obtain an empirical distribution which has lowest entropy.

After splitting the training set into left and right subsets, the information gain [9] is given by,

$$I = H(S) - \sum_{i \in \{L,R\}} \frac{|S^i|}{|S|} H(S^i)$$

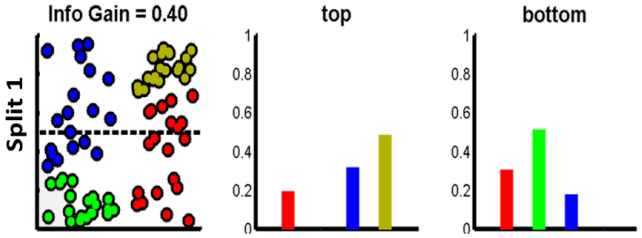


Figure 6. After horizontal split.

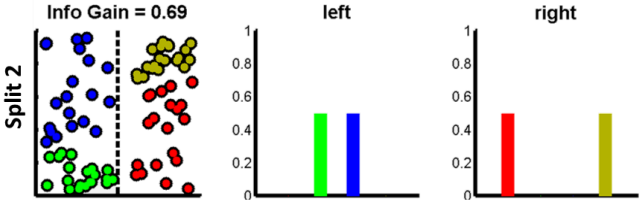


Figure 7. After vertical split.

The example shown in the Figure 6 and 7 shows how we can analyse the information gain as training objective function and used to find optimal weak learner functions. Thus maximizing the information gain helps select the split parameters which produce the highest confidence in the final distribution at the leaf nodes.

Ensemble of Trees: A random decision forest is an ensemble of randomly trained decision trees. As selecting parameters is random in nature we may end in wrongly classified tree if we consider a single tree and ensemble also reduces the noisy tree contributions. The key aspects of the decision forest model is the fact that its component trees are all randomly different from one another. This leads to de-correlation between the individual tree predictions and in turn results in improved generalization and robustness.

$$p(c|v) = \frac{1}{T}\sum_{t=1}^{T} p_t(c|v)$$

IV. PER-PIXEL CLASSIFICATION

In our algorithm, the first step is to acquire the depth image from a depth camera and perform essential preprocessing to obtain the cropped image of the face alone. The Figure 8, shows both the color image and the cropped depth image which will be considered as input image for rest of our algorithm.

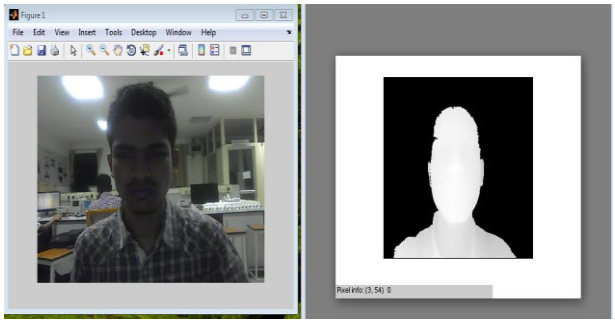


Figure 8. Depth image acquisition.

Training: Each tree in the random forest is trained using fully labeled image for the given equivalent input depth image. We assume that *u* encodes a 2D pixel location in the input image For every pixel *u* there exist a part label $c \in C$, Say a random subset of N=1600 example pixels is taken from the input image to forms a subset S={*u*}. Steps for the construction of tree structure is as follows:

1) For each pixel *u* we have three features, x coordinate, y coordinate and depth at that pixel, consider all pixels S={*u*}.

2) Partition the set into left and right subsets for each split point using weak learner function i.e., S={*u*} into left $S_j^L(\theta)$ and right $S_j^R(\theta)$ subset by each candidate θ, based on the weak learner function given by,

$$h(\mathrm{u};\theta_j) = [\mathrm{f(u)} \geq \tau_j]$$
$$= 1\,;\, S_j^R(\theta)$$
$$= 0;\, S_j^L(\theta)$$

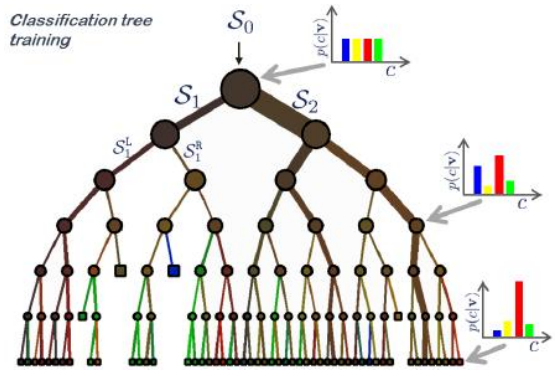


Figure 10. Training tree structure.

3) Compute the optimal θ_j giving the largest information gain.

Note that training happens by maximizing the information gain for discrete distribution.

$$\theta_j^* = \arg\min_{\theta \in T} \sum_{d \in \{L,R\}} \frac{|S^d(\theta)|}{|S|} I\left(S^d(\theta)\right)$$

where,

$$I(S) = -\sum_c p(c|S) \log p(c|S)$$

4) The steps 2, 3, 4 are repeated for every split node *j* until the leaf node of the tree structure is reached. The depth of the tree can be limited based on the GPU speed and number of input pixels.

5) The resulting leaf nodes will contain the classified distribution for the individual pixels of the image considered.

Testing: The advantage of using Randomized decision trees and forests is that they are fast and effective multiclass classifiers for many tasks [10, 13] and can be implemented efficiently on GPU. Figure 11, illustrates the ensemble of *T* decision trees with split and leaf nodes.

During testing each pixel *u* of a particular test image is applied to the root node of all the trained trees in a decision forest. Based on the defined weak learner function at the split nodes *n*, the pixel *u* traverses a path to the leaf in every other tree of the forest.

$$h(\mathrm{u};\theta_j) = [\mathrm{f(u)} \geq \tau_j]$$

If $h(u;\theta_n)$ gives 0, the pixel *u* path traverse to the left child otherwise to the right child. This repeats until the leaf node is

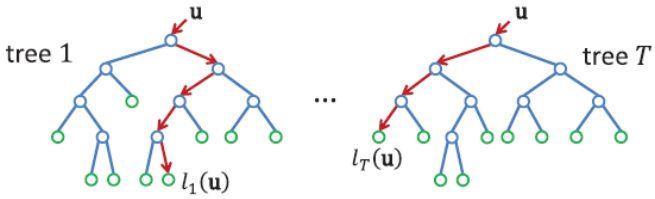


Figure 11. RDF during testing.

reached. Let us use *l(u)* to indicate a particular leaf node is reached for pixel *u*. Our main aim is to predict the label at each pixel in a particular image. This is achieved by storing a distribution $p_l(c)$ over the discrete body part *c* at each leaf *l*. The distribution is averaged for all the trees in the forest to obtain final classification as:

$$p(c|u) = \frac{1}{T} \sum_{l \in l(\mathrm{u})} p_l(c)$$

V. SIMULATION RESULTS AND PERFORMANCE EVALUATION

In this section we describe the simulations performed to evaluate our algorithm on several training parameters and design challenges. We observed through our simulation that the accuracy largely depends on the depth of trees, number of trees, number of features. Note that no kinematic or temporal constraints are employed in our simulation. The obtained ground truth labels at the leaf nodes are pictorially represented.

In our simulation the following training parameters were used. We trained 5 trees in the forest. Each tree was grown to a depth of 20 and we considered 10 images with N=1600 random training pixels per image. During training we considered large number of candidates for split function parameters around 1600*3=4800 at every split node. The simulation was done using MATLAB software tool and the following results were obtained (Figure 13). Let us analyze the performance of our algorithm by varying a number of training parameters such as depth of trees, features and number of trees.

Case 1: The effect of varying number of trees.

Figure 12 (a), shows as the number of trees increases accuracy also increases but it starts to saturate around 4 trees and the qualitative results have shown that more number of trees will reduce noise.

Case 2: The effect of varying depth of trees.

Depth of trees is found to have the most significant effect when compared with the other training parameters as it directly affects the capacity of the classifier. The depth of the tree should be selected such that it does not overfit(i.e., some leaf nodes will not have any prob. of distribution) when large number of input images are considered. We observed that at a depth of 15, we obtained better results with high accuracy gradient for the given set of training images as sown in Figure 12 (d).

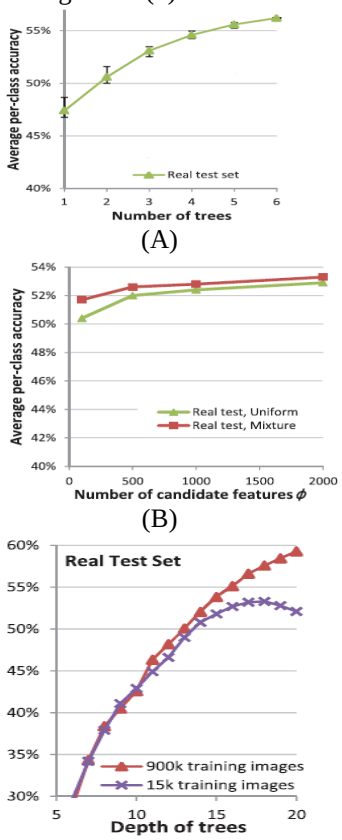


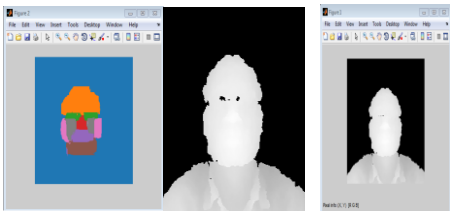
Figure 12. Variation in accuracy based on training parameters.

Case 3: The effect of varying candidate features

Figure 12(c) shows the effect of varying candidate feature offsets during training respectively. We observed that as the number of features are increased, accuracy also increases, but by increasing number of features processing required at each split node also increases i.e., $\theta \in T$. So we have considered 4800 features for the evaluation of our results which gives good accuracy.

Figure 13, shows the final per-pixel classified output (3rd row) , training input(2nd row) along with its depth input (1st row). We have successfully parsed the image into eight different classes using RDF classifier which is very essential in pose estimation problem.

Figure 14, shows results for more images and as we considered only 1600 pixels from each image(65536 pixels) since processing increases if we consider more pixels for training we could not get accurate classification for test images.



a) b) c)

Figure 13. a) Original Depth image, b) Reconstructed image and c) Per-pixel classified result.

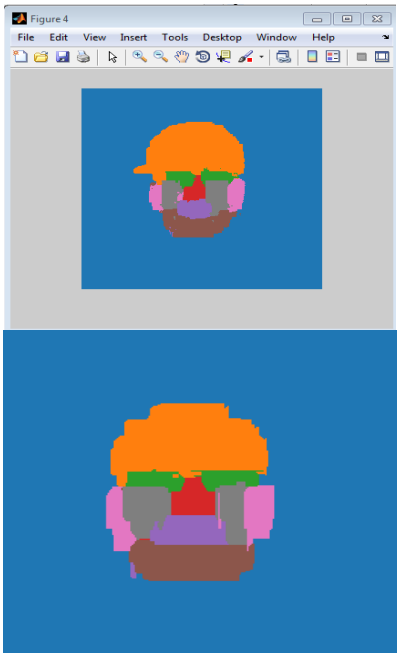


Figure 14. Per-pixel classified result for different testing data

VI. CONCLUSION

In this paper we have proposed a intermediate approach to estimate the pose by parsing a single depth image. We learned the position distribution of each labelled part and used in combination with the RDF classifier to form an ensemble of classifiers for per-pixel classification. Simulation results demonstrate that our system is capable of delivering fast and accurate estimations. This was possible by carefully selecting the training parameters and design specifications. In future work, we plan to detect faces and joints of human body and also exploit temporal constraints. In this paper we have obtained the visualization of the face in 2D image, thus our next step is to reproject this into world space (3D image) and generate reliable proposals to obtain the position of the 3D skeletal joints [15]. We wish to implement our design in real time using C++ open CV and also extend our research to estimate human body poses and many other articulated objects.

REFERENCES

1. V. Ganapati, D. Koller, C. Plagemann, and S. Thrun, ”Real-Time Identification and Localization of Body parts from Depth Images,” proc. IEEE Int’l conf. Robotics and Automation, 2010.
2. Microsoft Corp., Redmond, Wash., “Kinect.”
3. R. Wang and J. Popovic, “Real-Time Hand-Tracking with a colour Glove,” proc. ACM Siggraph, 2009.

4. J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman,"Real-Time Human pose estimation from single depth Image," proc. IEEE conf. Computer Vision and Pattern Recognition, 2011.
5. [5] J. Gall and V. Lempitsky,"Class-Specific Hough Forests for Object Detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.
6. V. Lepetit, P. Lagger, and P. Fua, "Randomized Trees for Real-Time Keypoint Recognition," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp.2:775-781, 2005.
7. A. Criminisi, J. Shotton, and E. Konukoglu, "Decision Forests: A Unified Framework," Foundations and trends in computer Graphics and Vision, vol. 7, pp.81-227, 2012.
8. S. Nowozin, "Improved Information Gain Estimates for Decision Tree Induction, " proc. Int'l Conf. Machine Learning,2012.
9. T. Sharp, "Implementing Decision Trees and Forests on a GPU," Proc. European Conf. Computer Vision, 2008.
10. J. Winn and J. Shotton, "The Layout Consistent Random Field for Recognizing and Segmenting Partially Occluded Objects," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2006.
11. G. Rogez, J. Rihan, S. Ramalingam, C. Orrite, and P. Torr, "Randomized Trees for Human Pose Detection," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2008.
12. S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pp. 509-522, Apr. 2002.
13. A. Montillo, J. Shotton, J. Winn, J. Iglesias, D. Metaxas, and A. Criminisi, "Entangled Decision Forests and Their Application for Semantic Segmentation of CT Images," Proc. 22nd Int'l Conf. Information Processing in Medical Imaging, 2011.
14. D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 603-619, May 2002.
15. Shotton, Jamie, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. "Efficient Human Pose Estimation from Single Depth Images", IEEE Transactions on Pattern Analysis and Machine Intelligence, Dec 2013.
16. J. P. Wachs, M. Kolsch, H. Stern and Y. Edan, Vision-Based Hand Gesture Applications, in Communications of the ACM, vol. 54, no. 2, pp. 60-71, Feb. 2011.
17. Z. Ren, J. Yuan, J. Meng and Z. Zhang, Robust Part-Based Hand Gesture Recognition Using Kinect Sensor, in IEEE Trans. Multimedia, vol. 15, no. 5, pp. 1110-1120, Aug. 2013.
18. M. K. Bhuyan, V. V. Ramaraju and Y. Iwahori, Hand gesture recognition and animation for local hand motions, in Int. J. Mach. Learn. & Cyber., Mar. 2013.
19. C. Keskin, F. Kirac, Y. E. Kara and L. Akarun, Hand Pose Estimation and Hand Shape Classification Using Multi-layered Randomized Decision Forests, in ECCV 2012.
20. A. Gustus, G. Stillfried, J. Visser, H. Jorntell, and P. van der Smagt. Human hand modelling: kinematics, dynamics, applications. Biological Cybernetics, 106(11-12):741–755, 2012.
21. H. Wang, M. C. Leu, C. Oz, American Sign Language Recognition Using Multi-dimensional Hidden Markov Models, in Journal of Information Science and Engineering, 2006.
22. F. Qi et al., "Structure guided fusion for depth map inpainting," Pattern Recognition Letters, vol. 34, no. 1, pp. 70-76, Jan. 2013.
23. S. Matyunin, D. Vatolin, Y. Berdnikov, and M. Smirnov, "Temporal filtering for depth maps generated by Kinect depth camera," in Proc. Of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 2012, pp. 797-800.
24. N. E. Yang, Y. G. Kim, and R. H. Park, "Depth hole filling using the depth distribution of neighboring regions of depth holes in the Kinect sensor," in Proc. of IEEE Int. Conf. on Signal Processing, Communication, and Computing, 2012, pp. 658-661.

ABOUT AUTHORTS

Navabharat Reddy, Currently his working has a Assitant professor in vignan institute of technology and science he completed his masters in communication engineering at VIT Vellore his having industrial experience in VLSI and his research interests in electronics that covers Image processing,Very large scale integration and Embedded Systemms ,Communications wireless and Antennas.

Sruthi Setlem, Curently she is working has a software developer in MNC she completed her Masters in communication engineering at VIT Vellore she did her internship from ISRO. She is having Industrial experience as well apart from Teaching her current research interest in electronics.