ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-5, 1386 – 1391

AN EMPIRICAL AND RECURSIVE INNER CENTER METHOD FOR HUMAN MOTION TRACKING

D.Neelima[1], Mudunuri Harshita[2]

[1] *Assistant Professor, Dept of CSE, Jawaharlal Nehru Technological University, Kakinada, India*
[2] *M.Tech student, Dept of CSE, Jawaharlal Nehru Technological University, Kakinada, India.*

Abstract

We propose RIC algorithm using Human Model Point Selection(HMPS) and Logarithmic data point search(LDPS) which reduces long computations of closest point in model points .It is much more faster than RIC using K-D tree approach . It process in the neighbouring closest points in the upper layer in the model points for finding the closest points in the lower layer and it accelerates the search by visiting the data points in the model. We apply this in Human Motion Tracking and much faster it works on moving objects.

Index Terms—*articulated tracking, human pose tracking, human motion, physical simulation, physics-based priors, Bayesian filtering, particle filtering.*

----------***----------

1. INTRODUCTION

Recursive Inner Centre (**RIC**) is an algorithm employed to minimize the difference between two clouds of points. RIC is often used to reconstruct 2D or 3D surfaces from different scans, to localize robots and achieve optimal path planning (especially when wheel odometry is unreliable due to slippery terrain), to co-register bone models, etc.The algorithm[1][2] is conceptually simple and is commonly used in real-time. It iteratively revises the transformation (translation, rotation) needed to minimize the distance between the points of two raw scans. Inputs: points from two raw scans, initial estimation of the transformation, criteria for stopping the iteration. Output: refined transformation. Essentially the algorithm steps are :

1. Associate points by the nearest neighbour criteria.
2. Estimate transformation parameters using a mean square cost function.
3. Transform the points using the estimated parameters.
4. Iterate (re-associate the points and so on).

The main algorithm drawback is that it is prone to accumulative errors, which can lead to the mapping algorithm failure.

1.1 Basic RIC Algorithm:

In its simplest form, the RIC algorithm iterates two steps. Beginning with an initial estimate of the registration parameters, a0, the algorithm forms a sequence of estimates ak which progressively reduce the error E(a). Each iteration of the algorithm comprises the following two steps, labelled C and T:

C. Compute *correspondences*, _: Set

$$\phi(i) = \underset{j\in\{1\ldots N_m\}}{\operatorname{argmin}} \ \epsilon^2\left(\left|\, \mathbf{m}_j - T(\mathbf{a}_k;\mathbf{d}_i)\,\right|\right) \qquad i = 1\ldots N_d$$

So that m_(i) is the closest model point to the datum di transformed by the current estimate ak.

T. Update *transformation*, a

$$\mathbf{a}\text{: Set } \mathbf{a}_{k+1} = \operatorname{argmin}_{\mathbf{a}} \sum_{i=1}^{N_d} \epsilon^2\left(\left|\, \mathbf{m}_{\phi(i)} - T(\mathbf{a};\mathbf{d}_i)\,\right|\right)$$

In many common cases, step T can be performed in closed form, or via well understood, numerically stable, procedures such as singular value decomposition. It is easy to see that both steps must reduce the error, and that the error is bounded below. Thus convergence to a local minimum is guaranteed. Furthermore, it is straightforward to discern a termination criterion: when the set of correspondences does not change in step C, the value of ak+1 will be set equal to ak in the T step, so no further change is possible.

The main drawback of the RIC algorithm is that it requires a huge amount of computational time to find the corresponding closest points between the model and the data set. This time increases explosively as the number of points in the model and data sets increases. There are several ways to speed up the RIC algorithm, which can be divided into two approaches. The first approach is The first approach is to reduce the number of iterations. That is accelerated RIC [3]which updated the motion parameters using linear or parabolic extrapolation to

reduce the number of iterations. The second approach is to reduce the search time for finding the corresponding closest points. That is K-D tree to decrease the search time for finding the corresponding closest points.

2. RELATED WORK

2.1.1 Accelerated RIC:

RIC algorithms are used to align range images of an object taken from different viewpoints into a common coordinate system (Fig.1). The general structure of an RIC-algorithm is illustrated in Fig.2.

Figure 1: range images from different viewpoints have to be transformed into the same coordinate system.

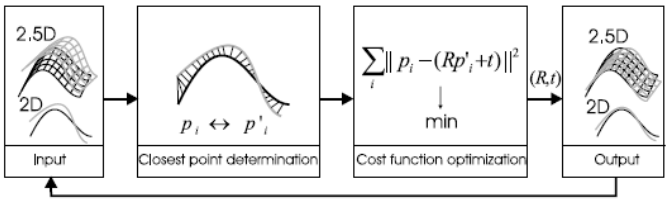

Figure 2: The general structure of an RIC algorithm. The output of each iteration is the input for the next iteration

To find the rotation and translation between two range images of overlapping regions in a first step closet points between the two data sets are determined. In a second step a cost function depending on the distances of closet points is minimized with respect to the six rotation and translation parameters. In a final step the resulting transformation is applied to the respective data set, so that both data sets come closer to each other. These steps are iterated until convergence. Usually, the cost function used in an RIC algorithm is a least squares sum of closet points Pi and Pi'.

$$\Phi = \sum_i ||\mathbf{p}_i - (\mathbf{R}\mathbf{p}'_i + \mathbf{t})||^2 \qquad (1)$$

With the translation vector t and the rotation matrix R. R and t resulting from optimization of this cost function are the maximum likelihood solutions to the problem of aligning the two data sets under the assumptions that the closet points are corresponding points and data noise is Gaussian distributed.

The point with a small neighbourhood can be best localized in its normal direction n, and the localization accuracies in its directions e1,e2 of minimal and maximal curvatures are proportional to the respective extremal curvatures. Therefore it makes sense to introduce for each data point in the cost function (1) three confidence values that take into account the different localization accuracies in each of these directions. Before introducing these confidence values we rewrite (1) with the help of the difference vector.

$$\mathbf{d}_i = \mathbf{p}_i - (\mathbf{R}\mathbf{p}'_i + \mathbf{t}) \qquad (2)$$

as

$$\Phi = \sum_i \mathbf{d}_i^2 = \sum_i \big[(\mathbf{d}_i \cdot \mathbf{n}_i)\,\mathbf{n}_i + (\mathbf{d}_i \cdot \mathbf{e}_{1i})\,\mathbf{e}_{1i} + (\mathbf{d}_i \cdot \mathbf{e}_{2i})\,\mathbf{e}_{2i} \big]^2 \qquad (3)$$

Where in (3) we represented di in the local coordinate system of pi (Fig.3)

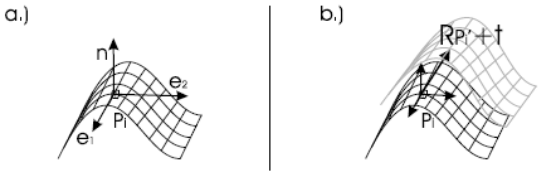

Figure 3: a) The normal and the directions of minimal and maximal curvatures define a local coordinate system in each data point. b) The difference vector of Pi,to the transformed closet point Pi' in the second data set can be represented in the local coordinate system of Pi.

Now we are able to introduce the point-wise confidence values c_{1i},c_{2i} and c_{3i},

$$\tilde{\Phi} = \sum_i \big[c_{1i}\,(\mathbf{d}_i \cdot \mathbf{n}_i)\,\mathbf{n}_i + c_{2i}\,(\mathbf{d}_i \cdot \mathbf{e}_{1i})\,\mathbf{e}_{1i} + c_{3i}\,(\mathbf{d}_i \cdot \mathbf{e}_{2i})\,\mathbf{e}_{2i} \big]^2. \qquad (4)$$

The confidence values for the tangential directions against the confidential value for the normal direction , i.e. $c_{1i}=1$ $c_{2i}=c_{3i}=0$. Experiments with the cost function (4) confirm theoretical results that we need less iterations of the RIC until convergence: on average 1/10 of the iterations are needed compared to the standard cost function (1).

2.1.2 K-D Tree:

a ***k*-d tree** (short for *k-dimensional tree*) is a space-partitioning data structure for organizing points in a *k*-dimensional space. *k*-d trees are a useful data structure for several applications, such as searches involving a multidimensional search key (e.g. range searches and nearest neighbour searches). *k*-d trees are a special case of binary space partitioning trees.

The *k*-d tree is a binary tree in which every node is a k-dimensional point. Every non-leaf node can be thought of as implicitly generating a splitting hyper plane that divides the space into two parts, known as half-spaces. Points to the left of this hyper plane represent the left sub tree of that node and points right of the hyper plane are represented by the right sub tree. The hyper plane direction is chosen in the following way: every node in the tree is associated with one of the k-dimensions, with the hyper plane perpendicular to that dimension's axis. So, for example, if for a particular split the "x" axis is chosen, all points in the sub tree with a smaller "x" value than the node will appear in the left sub tree and all points with larger "x" value will be in the right sub tree. In such a case, the hyper plane would be set by the x-value of the point, and its normal would be the unit x-axis.

Construction

Since there are many possible ways to choose axis-aligned splitting planes, there are many different ways to construct *k*-d trees. The canonical method of *k*-d tree construction has the following constraints: 1.As one moves down the tree, one cycles through the axes used to select the splitting planes. (For example, in a 3-dimensional tree, the root would have an *x*-aligned plane, the root's children would both have *y*-aligned planes, the root's grandchildren would all have *z*-aligned planes, the root's great-grandchildren would all have *x*-aligned planes, the root's great-great-grandchildren would all have *y*-aligned planes, and so on.) 2. Points are inserted by selecting the median of the points being put into the subtree, with respect to their coordinates in the axis being used to create the splitting plane. (Note the assumption that we feed the entire set of *n* points into the algorithm up-front.)

This method leads to a balanced *k*-d tree, in which each leaf node is about the same distance from the root. However, balanced trees are not necessarily optimal for all applications. Note also that it is not *required* to select the median point. In that case, the result is simply that there is no guarantee that the tree will be balanced. A simple heuristic to avoid coding a complex linear-time median-finding algorithm, or using an O(*n* log *n*) sort of all *n* points, is to use sort to find the median of a *fixed* number of *randomly* selected points to serve as the splitting plane. In practice, this technique often results in nicely balanced trees.

3. OUR APPROACH

In this paper, we propose a fast RIC algorithm that uses two acceleration techniques: hierarchical model point selection (HMPS) and logarithmic data point search (LDPS), which will be explained later. To evaluate the proposed algorithm, we apply our proposed RIC to the 3-D human body motion tracking problem.

3.1 Hierarchical Model Point Selection (HMPS)

It assumes the existence of a neighbourhood relationship between the two sets of points P and X. The relationship hypothesis is that two neighbours in a data set possess closest points that are neighbours in the other data set.

Formally, the principle of this neighbourhood relationship is exposed in figure 1: given a point **pk** in data set P and its corresponding closest point **xk** in data set X, the closest point **xi** of **p i**, if **pk** belongs to neighbourhood V of **pi**, V(**pi**), is found in the neighbourhood V' of **xk**, V'(**xk**).

The proposed idea towards a faster search is to use good approximations of the closest points instead of exact closest points[4]. The neighbourhood relationship is used to get a first approximation of the closest point and, then, a local search can be performed to refine the result instead of an exhaustive one: if **pi** possesses a neighbour **pk** in data set P, with a known closest point **xk** in data set X, finding the closest point of **pi** can be reduced to searching the closest point in the neighbourhood V' of **xk**, V'(**xk**). It appears that for each point **pi**, the closest point search is performed either in the full set X or only in the neighbourhood V'(**xk**), depending if at least one neighbour of **pi** has already a known closest point. Formally, this closest point search algorithm has therefore a theoretical best case complexity of O(Np). This is better than the best case using k-D tree, O(Np logNx) and lets us expect an overall gain in speed over using a tree search.

The worst case complexity corresponds to the conventional full search discussed so far. A good suggestion here is to use a tree search, instead of an exhaustive search, when searching the full set X - other potential ideas would be to use a temporal cache or a heuristic method like 3 steps algorithm, in range images -. Using this method, it is interesting to note that the neighbour search algorithm acceleration worst case is basically equal to the tree search O(Np logNx), as long as the cost of the local search is smaller than a tree search of course. The algorithm is shown below:

1) Prepare a model point set $X = \{(x(i), y(j)) \mid i = 0\ to\ N\ by\ 1,\ j = 0\ to\ N\ by\ 1\}$ and a data point set Y, where the model point set has a size of $(N+1) \times (N+1)$.
2) Select the model points of the first layer as $X_1 = \{(x(i), y(j)) \mid i = 0\ to\ N\ by\ N,\ j = 0\ to\ N\ by\ N\}$.
3) Find the closest data points $C(X_1, Y)$ corresponding to the model points X_1 using any search method, where C is the closest point operator.
4) For $n = 2, 3, \cdots, L$, where n and L are the layer index and the number of layers, respectively
   a) Select the model points of the nth layer as $X_n = \{(x(i), y(j)) \mid i = 0\ to\ N\ by\ \frac{N}{2^{n-1}},\ j = 0\ to\ N\ by\ \frac{N}{2^{n-1}}\} - \bigcup_{m=1}^{n-1} X_m$.
   b) For each model point $\mathbf{x} \in X_n$
      i) Obtain four distinct neighboring model points $N_4(\mathbf{x})$ of the model point $\mathbf{x}$ among the model points in the model set $\bigcup_{m=1}^{n-1} X_m$.
      ii) Take the four closest data points $C(N_4(\mathbf{x}), Y)$ corresponding to $N_4(\mathbf{x})$.
      iii) Construct the search window $S(N_4(\mathbf{x}))$ for finding the closest data point corresponding to the model point $\mathbf{x}$, which includes all data points within the rectangle obtained by the four closest data points $C(N_4(\mathbf{x}), Y)$.
      iv) Find the closest data point $C(\mathbf{x}, S(N_4(\mathbf{x})))$ of the model point $\mathbf{x}$.

3.2 Logarithmic Data Point Search (LDPS)

LDPS assumes that the data points are smoothly and continuously distributed. One simple way to reduce the search time to find the closest data points is to scan the data points as small as possible. To meet this goal, we propose to use a 2-D logarithmic search method that does not scan all data points to find a closest data point corresponding to a model point. The algorithm is shown below:

1) Initially, the search window has a size of $(2W+1) \times (2W+1)$ that covers data points whose positions are $\{(p, q) \mid p = -W\ to\ W\ by\ 1,\ q = -W\ to\ W\ by\ 1\}$.
2) Take nine data points $\mathbf{Y}_9 = \{\mathbf{y}_1, \cdots, \mathbf{y}_9\}$ that are located at $(0,0)$, $(0,d_1)$,$(0,-d_1)$, $(-d_1,0)$, $(d_1,0)$, (d_1,d_1), $(d_1,-d_1)$, $(-d_1,d_1)$, and $(-d_1,-d_1)$, where $d_1 = 2^{k-1}$ is the distance between two adjacent data points and $k = \lceil log_2 W \rceil$.
3) Obtain the best data point $\mathbf{y}^*$ whose distance metric between a model point $\mathbf{x}$ and nine data points $\mathbf{Y}_9$ is minimum as $\mathbf{y}^* = \min_{\mathbf{y} \in \mathbf{Y}_9} \mid \mathbf{x} - \mathbf{y} \mid^2$.
4) Move the center position of the search window to the best data point $\mathbf{y}^*$, reduce the window size to $W = \frac{W}{2}$, then take the nine data points that are $\mathbf{y}^*$ and the eight perimeter points at distance $d_2 = \frac{d_1}{2}$.
5) Repeat the steps 3) and 4) until $d_k = 1$, where the eight perimeter search locations are space by one point, and the best data point $\mathbf{y}^*$ at the kth iteration is the closest data point.

3.3 3d Human Body Model

The advantages of using a 3D model for human motion tracking are that reasonable kinematics constraints can be easily enforced and high level applications of tracking results such as animation or action analysis can also be easily performed. We use simple geometric models, such as sphere, cylinder, and cuboid, to construct the 3D human model used in our algorithm, which is shown in Figure 4. Several figure parameters are used to control the physical dimensions of different body parts, and we assume they are known in advance. These parameters can be assigned manually or estimated automatically by some initialization procedures. Besides given figure parameters, there are also motion parameters that describe human postures, and these motion parameters constitute the unknown state that we want to estimate during the tracking process.

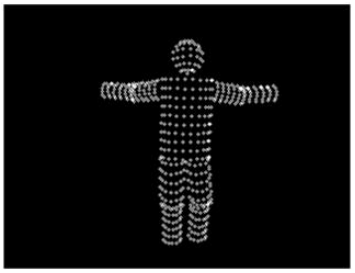

Figure 4: Human Body Model

Each limb is modeled by a 3D cylinder or a 3D sphere that is linked by the universal joint, where each 3D cylinder is parameterized with the major radius *rmajor*, minor radius *rminor*, and height *h* and each 3D sphere is parameterized with only the radius *rsphere*. The 3D points on the 3D cylinder or 3D sphere surface are represented by $X = [(x1, y1, z1)T, ..., (xN, yN, zN)T]$ in the camera-centered 3D coordinate system, where N is the number of points, the 2D image point $\mathbf{u} = (ux, uy)$ corresponding to a 3D point $\mathbf{x} = (x, y, z)T$, and pixel coordinate $\mathbf{u} = (ux, uy)T$ is obtained by projecting $\mathbf{x}$ onto the 2D image coordinate under the perspective projection as

$\mathbf{u} = p(\mathbf{x}) = f/z\ [x\ y]^T$, (3)

where f is the focal length. The motion of each limb is represented by the 3D motion parameter vector □ that includes three rotation parameters (*wx*,*wy*,*wz*) and three translation parameters (*tx*, *ty*, *tz*). When a point on the cylinder or sphere surface is represented by $\mathbf{x} = (x, y, z)T$, the point is transformed by the motion vector Θ as

$M(\mathbf{x}; \Theta) = R\mathbf{x} + T$, (4)

where *M* is a rigid transformation function that is represented by a 3D rotation matrix $R \in \mathbf{R}3\times3$, which is a function of three rotation parameters (*wx*,*wy*, *wz*), and a 3D translation vector $T \in \mathbf{R}3\times1 = (tx, ty, tz)$. Since it is very complicated and difficult to parameterize the 3D rotation matrix by the Euler representation

3.4 3-D Motion Parameter Vector and Combined Fitting

It tracks the human body motion when the human body model is near the data initially. To remove this limitation and track a a rapidly moving human body robustly, we propose a novel fitting algorithm that it combines the particle filter algorithm into the RIC registration algorithm. The detailed explanation on how the particle filter works is given below. The particle filter consists of two models: the observation model and the state transition model. For the observation model, we take a matching function that approximates the observation likelihood in terms of the normal vectors and the existences of the 3D articulation data at the positions projected by the model points. For the state transition model, we use the motion history information to track the fast moving limbs. The observation model describes the relationship between the input image and the human model at the current state using the observation likelihood. Often, a matching function can be used to approximate the observation likelihood [6] because it requires the minimal computational effort. In this work, we define the matching function as

$$P(I_t \mid \theta^p_{i,t}) \propto \exp(-\frac{1}{2}(\frac{1}{N}\sum_{i=1}^{N}(1 - \mathbf{n}_m(\mathbf{x}_i)\cdot\mathbf{n}_d(u_i)) + \frac{1}{N}\sum_{i=1}^{N}(1 - I_b(u_i)))), \quad (5)$$

where $\mathbf{n}m(\mathbf{x}i)$ and $\mathbf{n}d(ui)$ are the surface normal vector of each limb at the position $\mathbf{x}i$ and the surface normal vector of the 3D articulation data at the position ui that corresponds to the projected position of the $\mathbf{x}i$, respectively, and $Ib(ui)$ is the binary valued function whether the 3D articulation data exists or not at the position ui. The first term is used to represent the directional similarity of the normal vectors between the model and the data, and the second term is used to represent how much the model and the data are overlapped. The state transition model describes the dynamics of the moving limbs. Often, it is represented by the zero-velocity model, but it is not appreciate to track the limb like the arm because it moves fast. To solve this problem, the state transition model should be adaptive to the fast movement of the limb. In this work, we assume that the dynamics of the moving limbs obey the constant velocity model. Then, the velocity $\mathbf{v}t-1$ of the ith limb at the time $t-1$ can be computed by

$$\mathbf{v}_{i,t-1} = \boldsymbol{\theta}_{i,t-1} - \boldsymbol{\theta}_{i,t-2}, \quad (6)$$

where $\Theta i,t-1$ and $\Theta i,t-2$ are the 3D motion parameter vectors of the ith limb at the time $t-1$ and $t-2$, respectively. Using above motion history information, the motion velocity of the ith limb at the time t is predicted by the motion velocity of the ith limb at the time $t-1$ as

$$\bar{\mathbf{v}}_{i,t} = v_{t-1}. \quad (7)$$

Thus, we can generate a set of particles using the state transition model as

$$\boldsymbol{\theta}^p_{i,t} = \boldsymbol{\theta}_{i,t-1} + \bar{\mathbf{v}}_{i,t} + \eta^p_{i,t}, \quad (8)$$

where $\eta^p_{i,t}$ is the system noise that obeys the Gaussian distribution. Among the set of particles, one particle whose value of the matching function is the largest is taken as the best particle as

$$\hat{\boldsymbol{\theta}}_{i,t} = \arg\max_{\boldsymbol{\theta}^p_{i,t}} \omega^p_t, \quad (9)$$

where ω^p_t is the normalized weight.

Tracking result using RIC method

CONCLUSION

In the existing approaches has that it can not fit the rapidly moving human body motion. We used two techniques to solve this problem that techniques are: Hierarchical Model Point Selection (HMPS) and Logarithmic Data Point Search (LDPS).These are effective to find the corresponding closest data points because they accelerate the RIC computation by reducing the regions to be searched in both the model set and data set, while degrading the registration performance only slightly. Proposed approach is much more faster 1.48 times than K-D tree approach of RIC algorithm.

REFERENCES

[1] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.
[2] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Image Vis. Comput.*, vol. 10, no. 3, pp. 145–155, 1991.
[3] S. Rusinkiewicz and M. Levoy, "Efficient variants of the RIC algorithm," in *Proc. Int. Conf. 3D Digital Imaging and Modeling*, 2001, pp. 145–152.
[4] D. Liu and T. Chen, "Soft shape context for Recursive Inner Centerregistration," in *Proc. IEEE Int. Conf. Image Processing*, 2004, pp. 1081–1084.
[5] M. Greenspan and M. Yurick, "Approximate k-d tree search for efficient RIC," in *Proc. Fouth Int. Conf. 3-D Digital Imaging and Modeling (3DIM)*, 2003, pp. 442–448.

[6] G. Turk and M. Levoy, "Zippered polygon meshes from range images," *Proc. SIGGRAPH*, pp. 311–318, 1994.
[7] R. Benjemaa and F. Schmitt, "A fast global registration of 3D sampled surfaces using a multi-Z-buffer technique," in *Proc. Int. Conf. Recent Advances in 3-D Digital Imaging and Modeling*, 1997, pp. 113–120.
[8] D. Demirdjian, "Enforcing constraints for human body tracking," in *Workshop on Multiple Object Tracking*, 2003, pp. 102–1093.
[9] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1998, pp. 8–15.
[10] D. Kim and D. Kim,A robust real-time 3d human body motion tracking using the RIC and particle filters, Tech. Rep., POSTECH-IM-LAB 2008–02 2008.
[11] J. Deutscher, A. Davision, and I. Reid. Articulated body motion capture by annealed particle filtering.

BIOGRAPHIES

Mrs. D. Neelima is B.Tech(CSE), M.Tech(CSE) from JNTU Hyderabad and currently pursuing P.hd in JNTU Kakinada , Andhra Pradesh, India. She is working as Assistant professor in Computer Science & Engineering department in Jawaharlal Nehru Technological University Kakinada , Andhra Pradesh, India. She has 4 years of experience in teaching Computer Science and Engineering related subjects. She is a research scholar and her area of interest and research includes Video Image Processing. She has published several Research papers out of which 2 are international Journals and 2 are international conferences. She has guided more than 60 students of Bachelor degree, 20 Students of Master degree in Computer Science and Engineering in their major projects.

I am **Mudunuri Harshita** doing M.Tech in Jawaharlal Nehru Technological University, Kakinada ,A.P.,and interesting research area is Video Image Processing.