Comparative Analysis of Heuristic Techniques for Vehicles movement detection

- **BTP CODE : B24SMP03**

OUR TEAM

Dr.S Manipriya

P.Rupesh chowdary

S20210010173

O. Khadhar Basha

S20210010164

S.V.S Apparao

S20210010208

Research Paper

- Drafted the research work done during the BTP tenure
- Submitted our research paper in CoMSO 2024
- The paper got accepted and presented in CoMSO 2024 , held between November 16th – 18th

4th International Conference on Modeling, Simulation and Optimization

CoMSO 2024

16 – 18 November, 2024

Certificate of Participation

This is to certify that

Dr. Manipriya Sankaranarayanan

from NIT Silchar, India

has participated and presented a technical paper titled

Comparative Evaluation of Vehicle Direction and Motion Detection Methods for Multi-layer Contiguous Virtual Layer (MCVL)

authored by

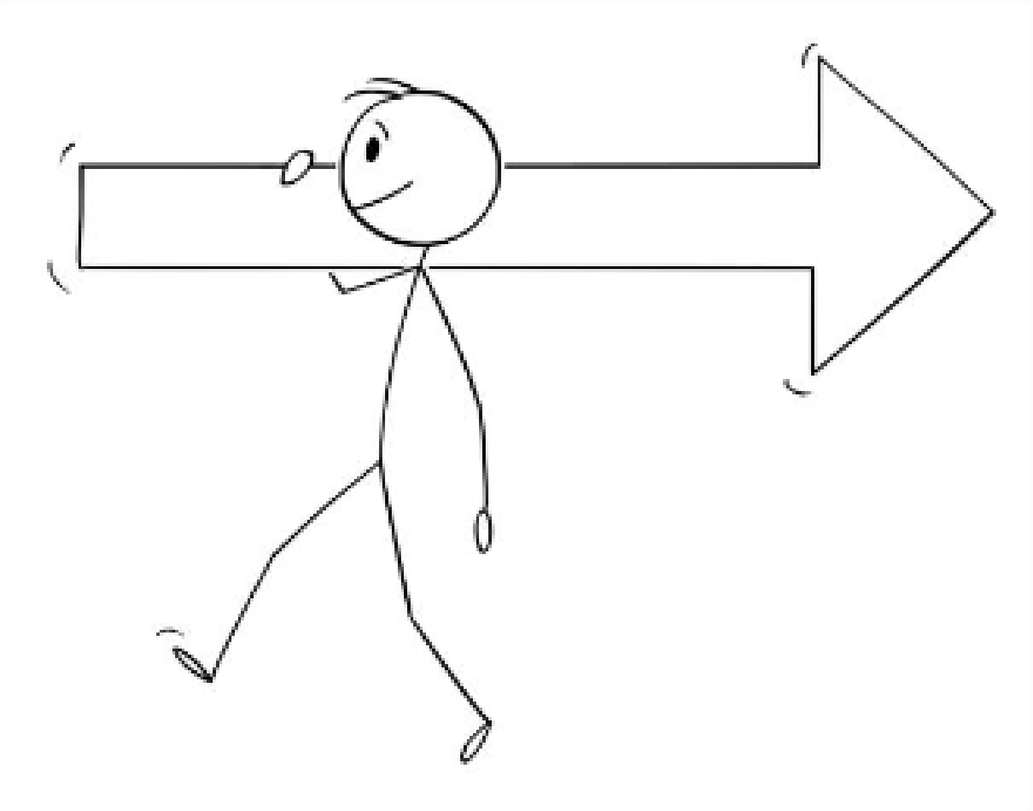
Dr. S. Manipriya, P. Rupesh, S.V.S Apparao and O Khadhar Basha

in 4th International Conference on Modeling, Simulation and Optimization (CoMSO 2024) held at National Institute of Technology Silchar, Assam, India.

Dr. Biplab Das
Organizing Chair
CoMSO 2024

Prof. Rajat Gupta
General Chair
CoMSO 2024

Recap

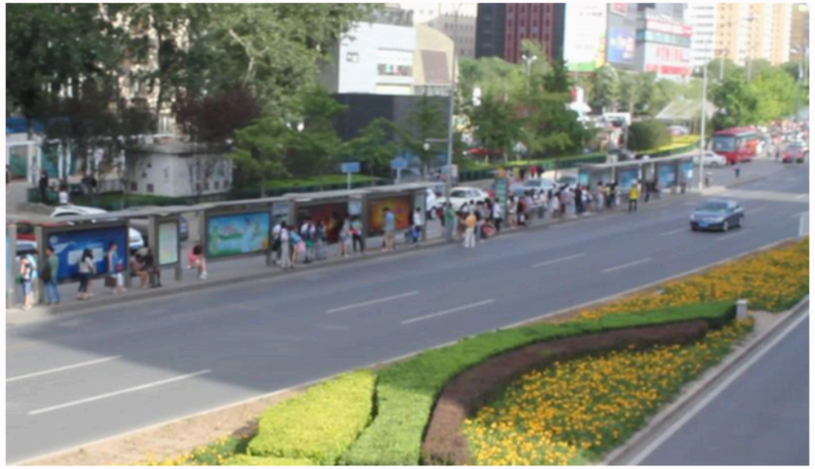


01 Dataset collection and Literature Survey

02 Exploring ideas, working on implemetations and results

03 Comparative study by analysing space and time

Dataset



Video Pre-processing

Road Surface Segmentation

Gaussian Blur

Noise Reduction

Frame Differencing

Thresholding

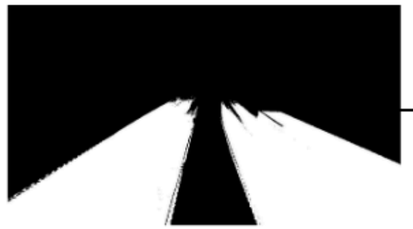
Background Subraction

Using cv2.createBackground SubtractorMOG2

Bit-wise AND

∩

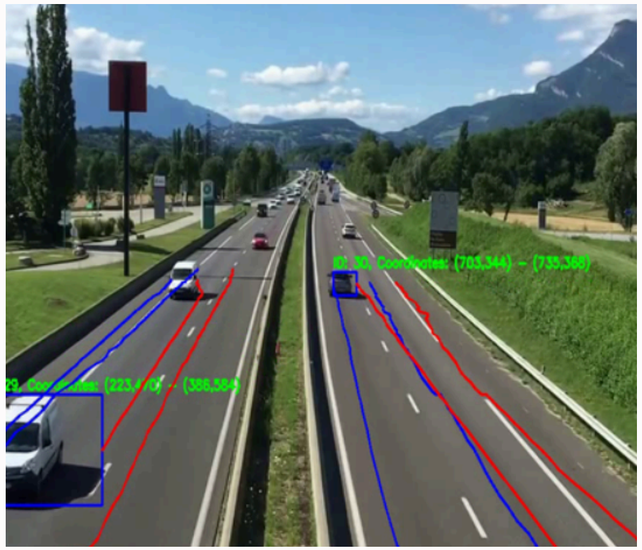
Σ



Binary mask of Road Surface

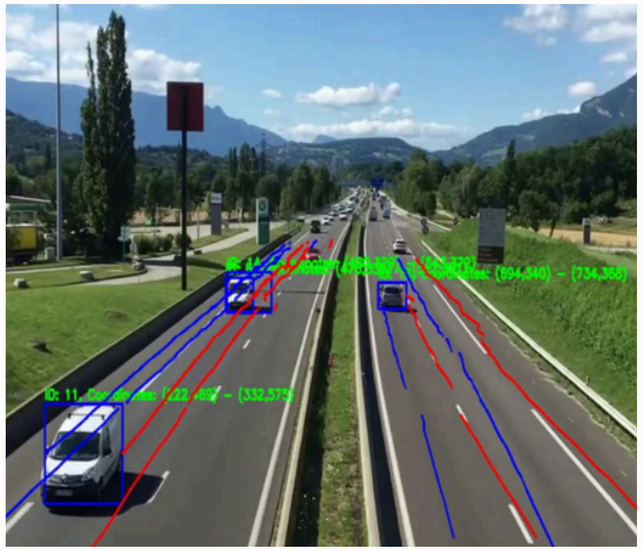
Adaptive Algorithms

Adaptive Blob Tracking



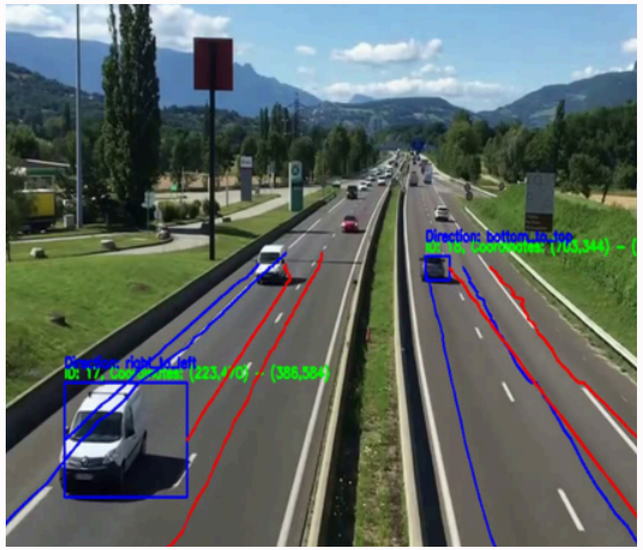
This method detects moving vehicles by identifying regions of connected pixels (blobs) that differ from the background. It is effective in stationary camera setups where objects stand out from the background.

Adaptive Yolo and Sort



YOLO is a real-time object detection algorithm that classifies and locates vehicles within video frames, while SORT tracks those detected objects across frames, assigning unique IDs to each vehicle for continued tracking .

Adaptive Optical Flow



This method estimates the motion of vehicles by analyzing the displacement of pixel intensities between consecutive frames, making it useful for tracking moving objects based on their movement patterns using lucas kanade optical flow

Comparison of Results

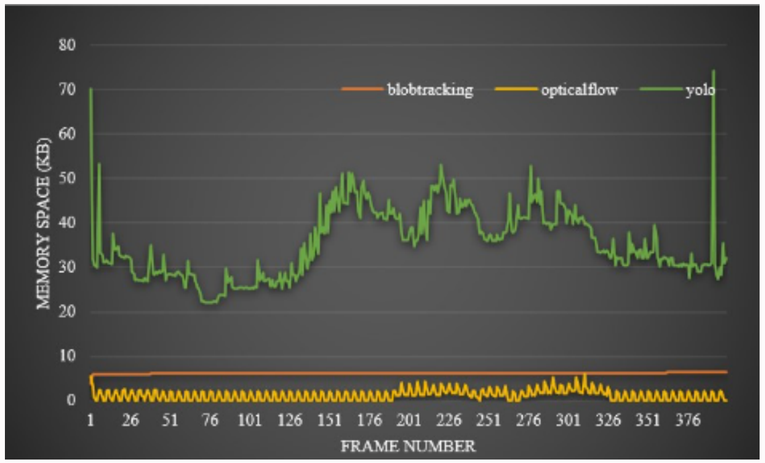
Videos	Adaptive Optical FLow	Adaptive Blob Tracking	Adaptive YOLO and SORT
1.) 	MSE : 1384.38 RMSE : 37.21	MSE : 648.25 RMSE : 25.46	MSE : 594.75 RMSE : 24.39
2.) 	MSE : 2467.25 RMSE : 49.671	MSE : 742.5 RMSE : 27.25	MSE : 3407.75 RMSE : 58.36
3.) 	MSE : 1133.75 RMSE : 33.67	MSE : 2260.05 RMSE : 47.54	MSE : 1889.25 RMSE : 43.46

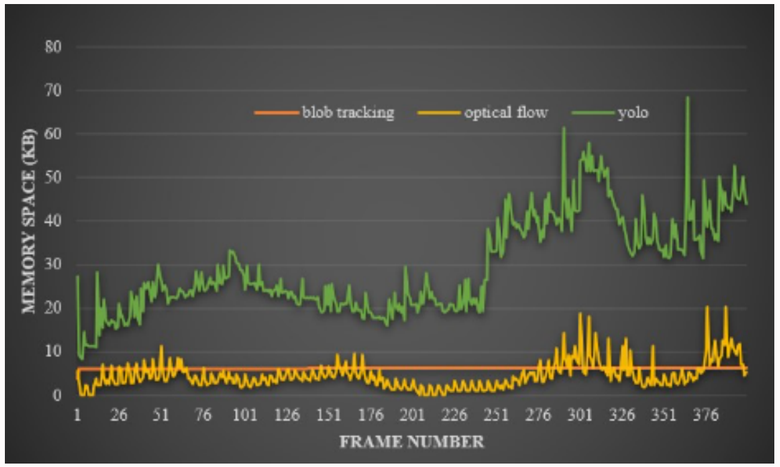
***MSE: Mean Square Error**

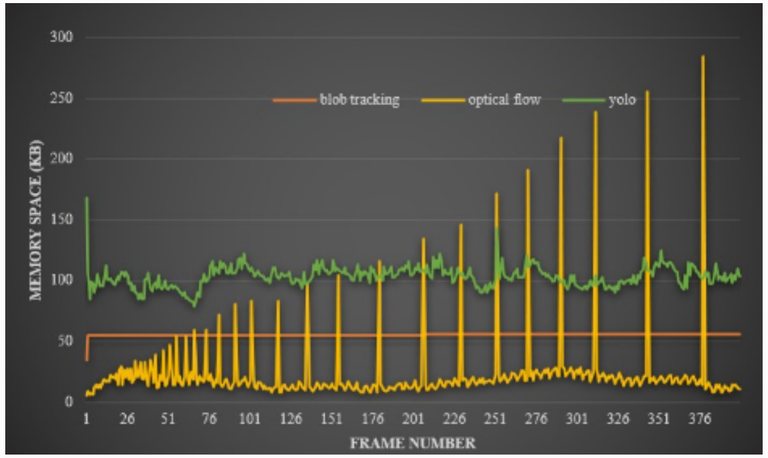
***RMSE: Root Mean Square Error**

Time and Space Analysis

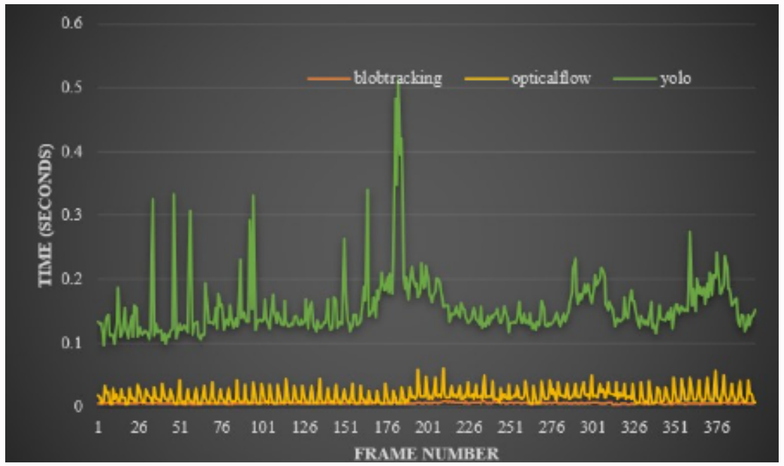
- **Space Analysis :**

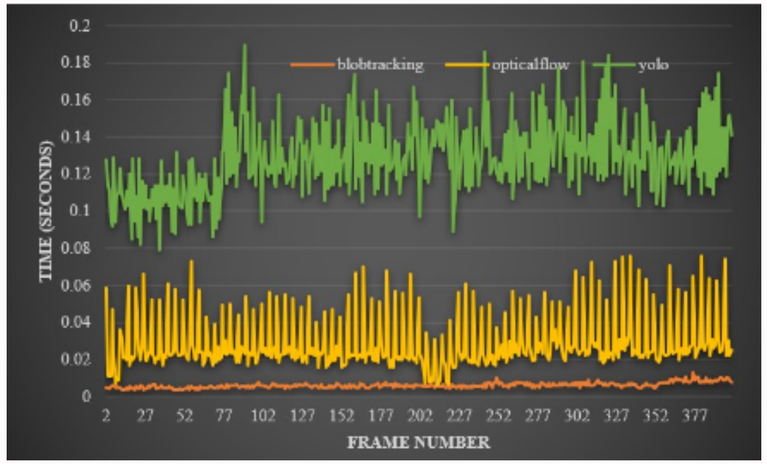


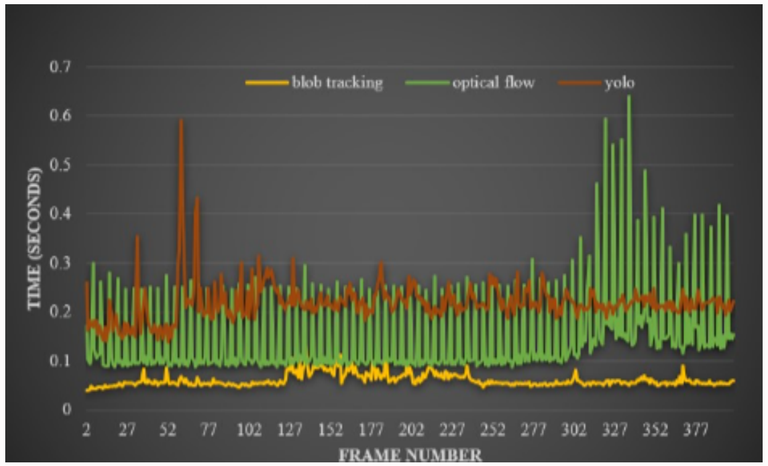


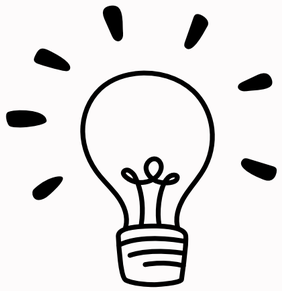


- **Time Analysis :**

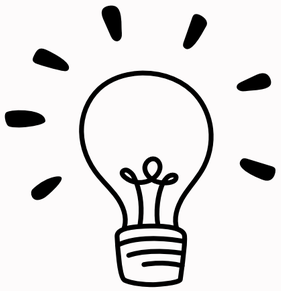








Current Work



Dynamic/Adaptive thresholding

Why Adaptive Thresholding is required ? :

- Manual thresholding uses a fixed threshold for all pixels in the image
- Hence, it cannot deal with images containing, for example, a strong illumination gradient or weak illumination gradient
- Adaptive or Dynamic thresholding changes the threshold dynamically over the image
- This more sophisticated version of thresholding can accommodate changing lighting conditions in the image

Chow and Kaneko Dynamic Thresholding :

- Chow and Kaneko divide an image into an array of overlapping subimages and then find the optimum threshold for each subimage by investigating its histogram
- The image is divided into small overlapping windows of size window_size × window_size. Each window represents a local region around a pixel.
- The threshold for each single pixel is found by interpolating the results of the subimages.
- The threshold T(i,j) for each pixel (i,j) is calculated using the Chow and Kaneko formula:

$$T(i,j) = \mu(i,j) \times \left[1 + k\left(\frac{\sigma(i,j)}{128} - 1\right)\right]$$

$\mu(i,j)$: Local mean

$\sigma(i,j)$: Local standard deviation

- If I(i,j)>T(i,j) the pixel is set to 255 (white, foreground).
- Otherwise, it is set to 0 (black, background)

Local Adaptive Thresholding :

- This method statistically examines the intensity values of the local neighborhood of each pixel.
- .The size of the neighborhood has to be large enough to cover sufficient foreground and background pixels, otherwise a poor threshold is chosen. (Window size: k x k)
- Within the local region, a Gaussian kernel is applied. A Gaussian kernel assigns more weight to pixels closer to the center of the region and less weight to pixels farther away.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where, $G(x,y)$: Weight for the pixel at (x,y) relative to the center

σ: Standard deviation

- The weighted sum of pixel intensities within the local region is computed using the Gaussian weights. This gives the local threshold (T) for the pixel.
- If I(i,j)>T the pixel is set to 255 (white, foreground).
- Otherwise, it is set to 0 (black, background)

Otsu's Thresholding :

- Compute the histogram of the grayscale image and the histogram represents the frequency of each intensity value (0–255 for 8-bit images).
- Normalize the histogram to calculate probabilities P(i) of each intensity i
- For each threshold t, Divide the histogram into two classes :

  1.) Class 1 (Background): Intensities [0,t] 2.) Class 2 (Foreground): Intensities [t+1,255]

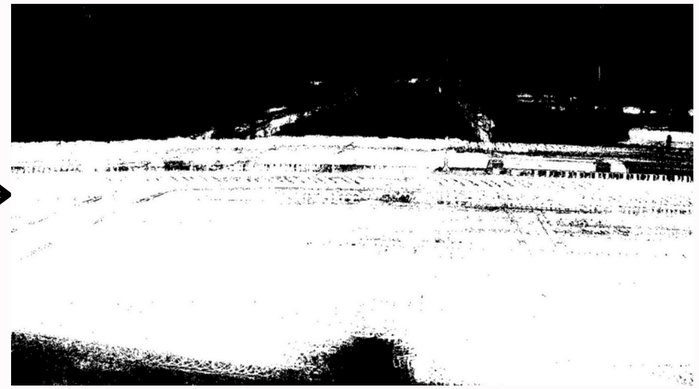
- Compute the probabilities of each class : $\omega_1(t) = \sum_{i=0}^{t} P(i)$ $\omega_2(t) = \sum_{i=t+1}^{255} P(i)$
- Calculate the mean intensity of each class: $\mu_1(t) = \frac{\sum_{i=0}^{t} i \cdot P(i)}{\omega_1(t)}$ $\mu_2(t) = \frac{\sum_{i=t+1}^{255} i \cdot P(i)}{\omega_2(t)}$
- Calculate the between-class variance : $\sigma^2_{\text{inter}}(t) = \omega_1(t) \cdot \omega_2(t) \cdot [\mu_1(t) - \mu_2(t)]^2$
- Identify the threshold that maximizes the between-class variance: $t^* = \arg\max_t \sigma^2_{\text{inter}}(t)$

Dynamic Thresholding Example

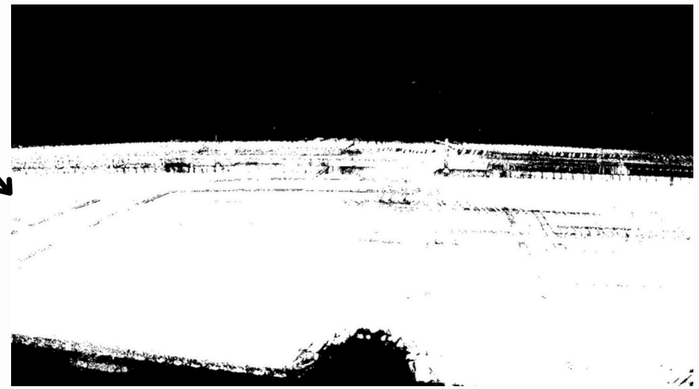
- Threshold Value : 40

- **Ground Truth (Manual Thresholding)**

- **Chow and Kaneko adaptive thresholding**
- Accuracy : 93.363%



- **Adaptive local thresholding**
- Accuracy : 95.343%



- **Otsu's thresholding**
- Accuracy : 99.181%

Accuracies of Road surface segmentation on different videos

Videos	Chow and Kaneko adaptive thresholding	Adaptive local thresholding	Otsu's Thresholding
1.) 	74.164%	97.706 %	99.3588%
2.) 	96.029%	96.133%	98.523%
3.) 	90.751%	92.4%	98.949%

Contd.

Analysis of Results :

- **Otsu's Thresholding** consistently achieves the highest accuracy across the videos.
- **Adaptive Local Thresholding** shows better accuracy than Chow and Kaneko but slightly lower than Otsu's.
- **Chow and Kaneko Adaptive Thresholding** has the lowest performance among the three methods.
- **Otsu's Thresholding** is more robust and performs well across diverse scenarios, making it a reliable choice for road surface segmentation.
- **Adaptive Local Thresholding** can be a good balance between computational efficiency and accuracy for moderately challenging videos.

Conclusion

- Our **Bachelor's Thesis Project (BTP)** research work presents and compares three heuristic methods—based on **optical flow, blob tracking, and YOLO sort** algorithms—for detecting vehicle movement in traffic videos.
- The work begins with Video pre-processing using **Road Surface segmentation** and our implementation includes both **Manual** and **dynamic thresholding**.
- Unlike existing research that relies heavily on deep learning, which has notable drawbacks, this study proposes adaptive approaches as alternatives and evaluates them using metrics like **Mean Square Error (MSE)** and **Root Mean Square Error (RMSE)**.
- Additionally, the **time and space analysis** helps to understand the algorithms computational capacity and system requirements to run these algorithms.

THANK YOU

- **BTP CODE : B24SMP03**