3D Multi-modal Multi-object Tracking via Machine Learning and Analytic Collision Risk Calculation for Autonomous Vehicles Navigation: The Geometric and Appearance Multi-object Tracking Module

Spathoulas Dimitrios
Department of Industrial Engineering and Management
Democritus University of Thrace

CONTENTS

LIST OF FIGURES

3D Multi-modal Multi-object Tracking via Machine Learning and Analytic Collision Risk Calculation for Autonomous Vehicles Navigation: The Geometric and Appearance Multi-object Tracking Module

***Abstract*—Moving towards the world of the future, terms such as machine vision, perception, and machine learning have managed to become integral components of the present, not only in the broader field of robotics but across all sciences. Concepts like recognition, sensation, perception, and attention have never before had such a direct and profound association with anything other than human cognition. This boundary, though still in early stages, is beginning to blur as humanity advances toward the creation and development of true intelligence.**

The present work examines and employs abstract patterns to first represent them in dimensions comprehensible to humans and then as a means of perception for an autonomous agent. The goal is for the agent, through artificial intelligence methods, to "see" and "perceive" objects and the environment around it, and even to "sense" the imminent danger of collision, the same way a human does unconsciously. All of this is set in one of the most complex and attention-demanding environments humans face: that of driving.

One of the most challenging real-world datasets in the field of autonomous navigation is utilized. Seven sensors are employed, one LiDAR and six cameras, to provide complete coverage of the autonomous vehicle's surrounding environment.

Overall, four distinct and interconnected modules are developed: the feature extraction module,[1] [2] the manifold learning module, the multi-object tracking module[3], and the risk detection module [4]. The feature extraction module extracts geometric features from an innovative 3D detector based on point clouds, projects the 3D detections onto the corresponding camera, and extracts appearance features using a renowned 2D detector and segmentor. In the manifold learning module, these high-dimensional features are represented through linear and non-linear methods. The multi-object tracking module employs these features as input for tracking using machine learning techniques. Finally, the risk detection module utilizes the tracking data to perform an analytical collision risk assessment. All modules were developed with a focus on xi real-time application.

I. Introduction

The tracking framework is adopted from [1]. For each sampling at time *t*, 3D detection is carried out to identify targets of interest surrounding the vehicle (**A**). The Kalman filter predicts the states of existing tracks from time *t-1* to *t* (**B**). Subsequently, the detections and tracks are fed into the data association unit (**C**), which outputs three distinct sets: matched tracks and detections ($\mathbf{D_{match}}$ / $\mathbf{T_{match}}$), unmatched tracks ($\mathbf{T_{unmatch}}$), and unmatched detections ($\mathbf{D_{unmatch}}$). The tracks are then updated (**D**) in the Kalman filter update step. In the final step of the architecture, unmatched entities enter the birth and death memory. There, it is decided whether the detections will be initialized as new tracks ($\mathbf{T_{new}}$) or if certain tracks will be terminated ($\mathbf{T_{lost}}$).

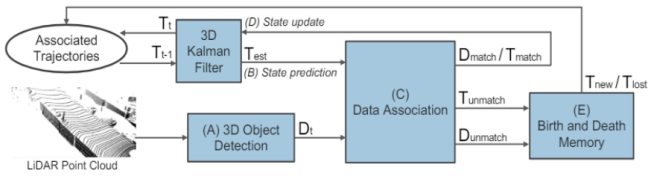

Fig. 1. Base Tracking Architecture

Data association is performed using an affinity matrix between detections and tracks. The assignment of observations to corresponding tracks is achieved by solving the bipartite graph matching problem on the affinity matrix, which can be optimally solved in polynomial time using the Hungarian algorithm. A track is terminated if it has not been updated for a certain number of consecutive algorithm steps. In the simplest approach, new tracks are initialized for every unmatched target.

II. Geometric Appearance Multi-object Tracking

The implementation heavily relied on a recent TRI-Stanford publication [2].

A. Visual-spatial and State Vectors

The output of the feature extraction module consists of the state vector (relative to the global frame), the geometric feature vector, the appearance feature vector, and the camera encoder, separated for each object class in each frame of each scene in the selected nuScenes subset:

$$\mathbf{O}_{s,t,c} = \{\mathbf{x}_{s,t,c}, \mathbf{g}_{s,t,c}, \mathbf{a}_{s,t,c}, \mathbf{d}_{s,t,c}, \tau_{s,t,c}\} \quad (1)$$

[1]https://github.com/DimSpathoulas/PointCloud_Feature_Extractor
[2]https://github.com/DimSpathoulas/2D_FEATURE_EXTRACTOR
[3]https://github.com/DimSpathoulas/GeomApp_3MOT
[4]https://github.com/DimSpathoulas/Collision_Risk_Calculation

Where:

- $\mathbf{O}_{s,t,c}$ represents the object in scene s, at timestamp t, and of class c.
- $\mathbf{x}_{s,t,c} \in R^7$ is the state vector in global coordinates.
- $\mathbf{g}_{s,t,c} \in R^{512\times3\times3}$ represents the geometric features.
- $\mathbf{a}_{s,t,c} \in R^{1024}$ represents the appearance features.
- $\mathbf{d}_{s,t,c} \in R^6$ is the camera decoder vector.
- $\tau_{s,t,c} \in R$ is the detection threshold.

B. 3D Kalman Filter

The state vector of each object at time t is:

$$\mathbf{s}_t = (x, y, z, a, l, w, h, dx, dy, dz, da)^T \quad (2)$$

Here x, y, z denote the target's position in the reference frame, a the orientation, l, w, h the dimensions of the bounding box, dx, dy, dz, da is the change in x, y, z, a between two consecutive detections.

A dynamic model with constant linear and angular velocity is assumed. The prediction step is written as:

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{t+1} &= \mathbf{A}\boldsymbol{\mu}_t \\ \hat{\boldsymbol{\Sigma}}_{t+1} &= \mathbf{A}\boldsymbol{\Sigma}_t\mathbf{A}^T + \mathbf{Q} \quad (3)\end{aligned}$$

The observation model takes the form:

$$\begin{aligned}\hat{\mathbf{o}}_{t+1} &= \mathbf{H}\hat{\boldsymbol{\mu}}_{t+1} \\ \mathbf{S}_{t+1} &= \mathbf{H}\hat{\boldsymbol{\Sigma}}_{t+1}\mathbf{H}^T + \mathbf{R} \quad (4)\end{aligned}$$

where the observation matrix $\mathbf{H}_{7\times11} = [\mathbf{I} \quad \mathbf{0}]$.

The correction step after detection-to-track association is formulated as:

$$\begin{aligned}\mathbf{K}_{t+1} &= \hat{\boldsymbol{\Sigma}}_{t+1}\mathbf{H}^T\mathbf{S}_{t+1}^{-1} \\ \boldsymbol{\mu}_{t+1} &= \hat{\boldsymbol{\mu}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{o}_{t+1} - \hat{\mathbf{o}}_{t+1}) \\ \boldsymbol{\Sigma}_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\hat{\boldsymbol{\Sigma}}_{t+1} \quad (5)\end{aligned}$$

where:

- $\mathbf{o}_{t+1}$ represents the observations.
- $\hat{\mathbf{o}}_{t+1}$ represents the predictions.

C. Covariance Matrices Estimation

Instead of initializing the covariance matrices $\mathbf{Q}$, $\mathbf{R}$, and $\boldsymbol{\Sigma}_0$ empirically, a method is proposed to estimate them based on the variances of the actual measurements from the training dataset—to avoid bias in the noise during inference—and the measurements from the detector [3]. For the matrix $\mathbf{Q}$, the positions of the actual measurements over three consecutive frames were used—second-order derivative, i.e., acceleration—to extract the variance of the linear and angular velocity:

$$\begin{aligned}Q_{xx} &= \text{Var}\left((x_{t+1}^{[m]} - x_t^{[m]}) - (x_t^{[m]} - x_{t-1}^{[m]})\right) \\ Q_{yy} &= \text{Var}\left((y_{t+1}^{[m]} - y_t^{[m]}) - (y_t^{[m]} - y_{t-1}^{[m]})\right) \\ Q_{zz} &= \text{Var}\left((z_{t+1}^{[m]} - z_t^{[m]}) - (z_t^{[m]} - z_{t-1}^{[m]})\right) \\ Q_{aa} &= \text{Var}\left((a_{t+1}^{[m]} - a_t^{[m]}) - (a_t^{[m]} - a_{t-1}^{[m]})\right) \quad (6)\end{aligned}$$

The matrix $\mathbf{R}$ was constructed using the centers of the detector's output and the corresponding ground truth values, i.e., the deviation between them. The matrix $\boldsymbol{\Sigma}_0$ is equal to $\mathbf{R}$:

$$\begin{aligned}R_{xx} &= \text{Var}\left(Dx_t^{[k]} - Gx_t^{[k]}\right) \\ R_{yy} &= \text{Var}\left(Dy_t^{[k]} - Gy_t^{[k]}\right) \\ R_{zz} &= \text{Var}\left(Dz_t^{[k]} - Gz_t^{[k]}\right) \\ R_{aa} &= \text{Var}\left(Da_t^{[k]} - Ga_t^{[k]}\right) \\ R_{ll} &= \text{Var}\left(Dl_t^{[k]} - Gl_t^{[k]}\right) \\ R_{ww} &= \text{Var}\left(Dw_t^{[k]} - Gw_t^{[k]}\right) \\ R_{hh} &= \text{Var}\left(Dh_t^{[k]} - Gh_t^{[k]}\right) \quad (7)\end{aligned}$$

where:

$$\mathbf{D}_t^{[k]} = \left(Dx_t^{[k]}, Dy_t^{[k]}, Dz_t^{[k]}, Da_t^{[k]}, Dl_t^{[k]}, Dw_t^{[k]}, Dh_t^{[k]}\right) \quad (8)$$

$$\mathbf{G}_t^{[k]} = \left(Gx_t^{[k]}, Gy_t^{[k]}, Gz_t^{[k]}, Ga_t^{[k]}, Gl_t^{[k]}, Gw_t^{[k]}, Gh_t^{[k]}\right) \quad (9)$$

Here, $(\mathbf{D}_t^{[k]}, \mathbf{G}_t^{[k]})$ represents the pair of the k-th detection and ground truth measurements at time t.

D. Feature Fusion Unit

The purpose of this unit is twofold: to transform the dimensions of the appearance features to match those of the geometric features and to align their vectors in such a way that their pairwise addition becomes meaningful:

$$\mathbf{F}_{\text{fused}} = \mathbf{G}_1(\mathbf{F}_{\text{2D}}) + \mathbf{F}_{\text{3D}} \quad (10)$$

$\mathbf{F}_{\text{2D}} \in R^{N\times(1024+6)}$ represents the Mask R-CNN features along with the camera encoder. $\mathbf{F}_{\text{3D}} \in R^{N\times512\times3\times3}$ represents the CenterPoint features for the N objects of the class. $\mathbf{G}_1(\cdot)$ is a fully connected multi-layer perceptron (MLP) with a hidden layer of size 1536, a Rectified Linear Unit (ReLU) activation, and an output layer of dimension 4608, which is reshaped into a tensor of size $512 \times 3 \times 3$. $\mathbf{F}_{\text{fused}} \in R^{N\times512\times3\times3}$ represents the fused features of the detections.

For each match, the fused features are blended with the corresponding track features through a simple factor:

$$\mathbf{F}_t^{\text{tracks}} = \alpha\mathbf{F}_t^{\text{fused}} + (1-\alpha)\mathbf{F}_{t-1}^{\text{tracks}} \quad (11)$$

E. Deep Affinity Similarity Matrix

1) Feature Map: From $\mathbf{F}_{\text{det}}^{\text{fused}} \in R^{N\times512\times3\times3}$ and $\mathbf{F}_{\text{trk}}^{\text{fused}} \in R^{M\times512\times3\times3}$, the input matrix of the feature distance unit is constructed as $\mathbf{F}_{d\times t}^{\text{fused}} \in R^{N\times M\times1024\times3\times3}$. This forms an $N \times M$ map where, at each position, the concatenated features are located, i.e., channels of size $R^{1024\times3\times3}$ for each combination. Thus, in the first row and first column, the features of the first detection are concatenated with those of the first track; in the first row and second column, the features of the first detection and the second track are concatenated, and so on.

2) Affinity Combination Unit: The matrix $\mathbf{D}_{\text{feat}} \in R^{N \times M}$ aims to express the concatenated features of each target-track pair as distances (a single value) in the range $[0, 1]$, where 0 represents a perfect match:

$$\mathbf{D}_{\text{feat}} = \mathbf{G}_2(\mathbf{F}_{\text{det}}^{\text{fused}}, \mathbf{F}_{\text{trk}}^{\text{fused}}) \quad (12)$$

Here, $\mathbf{G}_2(\cdot)$ consists of a convolutional layer with a kernel size of 3×3, stride 1, padding 0, and output dimension 256. The output is fed into a fully connected multi-layer perceptron (MLP) with ReLU activation functions and a hidden layer of dimension 128. This network cannot be replaced with a simple metric, such as cosine similarity.

F. Loss Fuction

The network's loss function is:

$$L_{\text{dist}} = \text{BCE}(\mathbf{D}_{\text{feat}}, \mathbf{K}) \quad (13)$$

The supervision matching indicator matrix $\mathbf{K}$ represents the true associations between targets and tracks based on the dataset. It has the same dimensions as the affinity matrix, with each element corresponding to a detection-track pair in $\mathbf{D}_{\text{feat}}$. Essentially, it is a binary mask. It is constructed based on two criteria:
1. **Proximity**: The distance between the detected object and the corresponding ground truth object, as well as the distance between the track and the ground truth object from the previous frame, must be less than a fixed threshold (2 meters).
2. **Identity**: The two corresponding ground truth objects (current and previous) must have the same instance token, which is a unique identifier that remains constant throughout the lifetime of the ground truth object in the dataset.
Thus, for each detection-track pair, $\mathbf{K}$ will have the value 0 if and only if:

1) The distance between the detected object and the corresponding ground truth object is less than the threshold.
2) The distance between the track and the corresponding ground truth object from the previous frame is also less than the threshold.
3) The instance token of the ground truth object corresponding to the current detection is the same as the instance token of the ground truth object corresponding to the track from the previous frame.

If any of these conditions are not met, the corresponding entry in the matrix will have the value 1. In $\mathbf{K}$, there can be at most one zero per row and per column, meaning at most one match for each target and at most one match for each track.

The unit is connected to the feature fusion unit so that the gradient flow based on the loss function $\mathbf{G}_2$ passes through to $\mathbf{G}_1$. In reality, the loss function considers 1 as correct and 0 as incorrect, so the unit does not learn to calculate good distances for correct matches but rather to make incorrect matches as bad as possible to ensure the non-matching of incorrect pairs.
A major issue is that $\mathbf{K}$ is a sparse matrix. For example, in the best case, if it consists of 100 elements (i.e., 10 detections and 10 tracks), the maximum number of zeros is 10. This issue is addressed using the focal loss technique, which reduces the significance of non-matches by down-weighting them:

$$FL(p_t) = -\alpha \cdot (1 - p_t)^{\gamma} \cdot \log(p_t) \quad (14)$$

where α is the ratio between the number of negative and positive associations in $\mathbf{K}$, γ is a hyperparameter controlling the focus and p_t represents the probabilities of the correct classes. In the current modeling, the value 1 is the negative class, so the imbalance equation does not penalize class 1 as in the general form of the loss function but instead reduces its significance to indirectly penalize the 0 samples, which is symmetrical.

G. Assignment

Let $\mathbf{D}$ be the affinity matrix of m detections and n tracks:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & \cdots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \cdots & d_{2n} \\ d_{31} & d_{32} & d_{33} & \cdots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & d_{m3} & \cdots & d_{mn} \end{bmatrix} \quad (15)$$

The matching algorithm first flattens $\mathbf{D}$ and then sorts it in ascending order:

$$\mathbf{a} = [d_{11}, d_{12}, \ldots, d_{1n}, d_{21}, d_{22}, \ldots, d_{mn}] \quad (16)$$

$$\mathbf{a}_{\sigma} = [a_{\sigma(1)}, a_{\sigma(2)}, \ldots, a_{\sigma(mn)}] \quad (17)$$

where $\sigma(i)$ represents the original position of the i-th smallest element in $\mathbf{a}$. Next, a matrix is constructed with the pairs of positions from the original matrix in ascending order:

$$(i, j) = \left(\left\lfloor \frac{\sigma(i)}{n} \right\rfloor, \sigma(i) \mod n \right) \quad (18)$$

Thus:

$$\mathbf{A} = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_v \end{bmatrix} \quad (19)$$

where $s_0 = (i, j)$ such that $\min(d_{ij})$ in $\mathbf{D}$ and $s_v = (i, j)$ is the position of $\max(d_{ij})$ and $v \in [0, mn \times mn]$. The matrix $\mathbf{A}$ contains the indices of all combinations of $\mathbf{D}$ in ascending order based on the affinity function d. Then, through a traversal of the rows of $\mathbf{A}$, $M \leq m$ detections are assigned to $N \leq n$ tracks in a one-to-one manner, which is why this technique is considered greedy. The output is a list of the positions of the associations in the matrix $\mathbf{D}$.
Tracks and detections that do not appear in the assignment list are considered unmatched, while those that do appear form the matched list. If the distance d_{ij} of the association $s_p(i, j)$ is greater than an acceptance threshold T, then the track j and detection i are also considered unmatched, as they are farther apart than acceptable. Therefore, two lists are created: a list of unmatched tracks and a list of unmatched detections.
For unmatched detections, new tracks are initialized, or dummy (temporary) tracks are initialized, after a number of steps, they are incorporated as real tracks. Tracks that have not been matched after several steps are deleted.

H. Training and Architecture

For each epoch, the first frame of the respective scene is called. Iteratively, each frame is fed into the unit until the last frame is processed, at which point the first frame of the next scene is called, and so on, until all scenes in the split are passed. Each frame serves as a key for the output dictionary of the feature extractor. In each frame, there are N_i, with $i \in [0, 7)$, detections, each accompanied by its corresponding information. Thus, the tracking unit is called a maximum of 7 times per frame. If the epoch is the final plus one, inference data is introduced, and the system transitions to the inference phase.

The networks are trained for all classes together, meaning there is no separate network for each class. This ensures that the weights generalize the pattern they need to learn. All of this is implemented in a general class (after extensive experimentation) that contains all the networks and their corresponding functions, along with the global tracking unit. The result is a complex architecture that, however, reduces the volume of the algorithm (from approximately 4500 lines to 1700). This is achieved by requiring specific objects (e.g., track lists) to be dynamic—able to be deleted from memory after each scene while changing at each frame—alongside static objects (e.g., acceptance thresholds, position switches), without disrupting the operation of the neural networks during both training and inference.

The Adam optimizer was used with an initial learning rate of 0.001 for 10 epochs, with the checkpoint set at the frame level for a relative smooth transition. The system operates with CUDA support. After the 7th epoch, the system transitions from using Mahalanobis distance to $\mathbf{K}$ for the association process during training, ensuring a smooth transition and optimal results.

III. Results

Although the blending factor is crucial for the network, it does not appear to significantly differentiate the results. For a feature-level analysis, the results are compared for random features, camera-only features, LiDAR-only features with cosine affinity, and the network. As confirmed by the literature, geometric features are more powerful than appearance features [2]. The affinity computation network proves to be highly effective, as it can encapsulate more nuanced feature interactions, thereby generating superior results compared to using only the cosine metric. It is concluded that the models, techniques, and tools used can identify patterns even in controlled noise, despite the results being nearly negligible. All model results are evaluated on the validation split of the NuScenes dataset. The agregated and Per-Class Geometric and Appearance Metrics on NuScenes Validation Split are shown below.

Metric	AMOTA	AMOTP	RECALL
Aggregated	0.404	1.234	0.488

These results were obtained at a stringent detection threshold of 57%, indicating significant potential for enhanced performance at lower thresholds.

Class	AMOTA	AMOTP	RECALL
bicycle	0.239	1.342	0.313
bus	0.615	0.942	0.696
car	0.462	1.148	0.541
motorcycle	0.367	1.284	0.464
pedestrian	0.509	1.068	0.622
trailer	0.318	1.444	0.369
truck	0.314	1.410	0.411

IV. Conclusions

A. Feature Extraction and Manifold Learning Modules

Among the tested dimensionality reduction techniques, UMAP proves to be the best choice for this specific dataset. Linear techniques are not sufficiently representative, primarily due to the high variability of the data (six different cameras, complex dense geometries). Among simple metrics, cosine similarity emerges as the best option for preserving structure. It is observed that the extracted features are highly sensitive to the stochasticity of the sensors; even for the same object, there are small but noticeable differences in the feature vectors.

The majority of false detections in the data without an acceptance threshold is overwhelming. The sensitivity to threshold tuning shows that an acceptance range between 0.5 and 0.6 strikes a balance between a significant increase in false detections and a reduction in missed detections. The voxel size visibly affects the morphology of objects, making the choice of it's size a critical factor for the accuracy of representation and the subsequent use of the features.

Finally, analysis via SVD shows that, within reasonable limits, it can retain the most important features, both geometric and appearance-based. However, its use is primarily effective for compression and processing speed rather than absolute structure preservation.

B. Geometric and Appearance Multi-object Tracking Module

The simple cosine metric is not sufficiently representative for computing distances between features. The blending factor is critical in the implementation, even if it does not significantly differentiate the results when used. The Deep Affinity Similarity Matrix aims to learn the intrinsic distances of the object's manifolds in different angles. The manifolds of the training and validation datasets are similar since the tracking system operates effectively. Geometric features are the ones that enhance the system the most. Appearance features derive their primary strength from color. Only through normalization, geometric and appearance features can be meaningfully combined within the feature fusion unit. Linear dimensionality reduction techniques, such as SVD, can approximate the final results but not fully replicate them. The system produces consistent results across multiple iterations, indicating its robustness. The tracking unit can be enhanced beyond the pseudo-metric Mahalanobis distance if properly merged with the feature matrix, as the two operate on different scales.

V. Future Work

A. Feature Extraction and Manifold Learning Modules

The feature extraction module can extract useful and meaningful geometric and appearance patterns. The grid region is of major importance in terms of information volume and complexity, especially for large objects or even the risk of missing small objects. The next level of features has a dimension of 64, and although it has been used in experiments, more refined methods could allow it to contain more spatially compressed information. Appearance features are generally weaker than geometric features. In extreme cases, such as objects moving very quickly across different fields of view and cameras, the feature vector of the same track may have almost no relation except for color.

A faster algorithm for constructing final information vectors could be developed using multi-processing methods, with the autonomous repository[5] serving as a base. The manifold learning module provides relatively stable output curves, although the cosine distance metric, as demonstrated by the tracking module, may not be ideal for accurate mapping. As future research, the use of more modern methods, such as variational autoencoders or encoder-decoder architectures with loss functions inspired by non-linear techniques (e.g., UMAP), would be more appropriate. A similar study on the training subset, as well as a comparison of the two subsets using edge correlation methods of the generated manifolds, would also be highly reasonable.

B. Geometric and Appearance Multi-object Tracking Module

The module could be improved by fusing the two feature and state affinity matrices, as well as through the track initialization unit. Experiments and architectures have already been implemented in the autonomous repository[6]. All units and modules can operate in air (e.g., UAVs), water (e.g., submarines), and even space applications (e.g., space debris avoidance).

References

[1] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366. IEEE, 2020.

[2] Hsu-kuang Chiu, Jie Li, Rares Ambruş, and Jeannette Bohg. Probabilistic 3d multi-modal, multi-object tracking for autonomous driving. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14227–14233. IEEE, 2021.

[3] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *arXiv preprint arXiv:2001.05673*, 2020.

[5] https://github.com/DimSpathoulas/2D_FEATURE_EXTRACTOR

[6] https://github.com/DimSpathoulas/GeomApp_3MOT