ChatGPT

AI-Based Single-Camera Tracker Assignment for Brain Surgery: State of the Art and Architectural Proposals

Introduction

In neurosurgery and other minimally invasive procedures, accurate tracking of a camera's position relative to the tissue is essential. Traditionally, stereoscopic tracking uses two cameras; their lines of sight intersect to localize a point in three-dimensional (3-D) space. The project under study aims to achieve similar tracking accuracy using only a single camera by placing triangular markers ("trackers") on the tissue. Each tracker is a triangle with four lights: two lights on side *a* and one light on each of sides *b* and *c*. There are 27 possible configurations (3 possible positions for the pair of lights on side *a* × 3 positions for the light on side *b* × 3 positions for the light on side *c*). A tracker can be encoded by three digits (e.g., `1 0 1`), where the first digit indicates the missing light on side *a* and the next two digits indicate the positions of the lights on sides *b* and *c*.

A single-camera system sees only a collection of lights in each frame; some trackers may be outside the field of view or partially occluded. The neural network (NN) must assign each visible light to a tracker (including cases with false positives), estimate the tracker's configuration, and output the confidence and 2-D location of each detected tracker. The challenges include occlusions, variable numbers of trackers, variable lighting, and extraneous points.

This report surveys related works, analyses suitable input representations, explores NN architectures for variable-size inputs/outputs, and proposes an architecture and output encoding for the described problem. Citations from connected sources are provided to support key claims.

1 Related Work

1.1 Marker-Based Tracking with a Single Camera

Geometric methods and marker design. Several earlier approaches tackle single-camera tracking by designing markers that are easy to recognize and then solving for pose using geometric constraints. A monocular head-tracking system uses multiple reflective markers and exploits geometric heuristics to reduce the correspondence search. It notes that a 3-D pose can be recovered from the 2-D projections of at least four non-colinear points when correspondences are known; however, ambiguous correspondences make the problem NP-hard. The authors reduce the combinatorial search by partitioning marker points into sets called **anchor sets**, using size differences among markers, and applying convexity constraints [1] [2]. Their approach is rule-based and uses geometry rather than deep learning.

Wearable skin markers (WSM). Another relevant system is a real-time motion-analysis platform for rehabilitation that uses *wearable skin markers* composed of equilateral triangles arranged as hexagons [3]. The authors avoid radially symmetric patterns to ensure unique recognition and generate 384 distinct marker IDs by combining two adjacent hexagons with unique binary patterns [4]. They

determine the physical size of the triangles based on camera resolution and distance so that each triangle occupies at least 80 pixels, enabling reliable recognition [5]. The recognition algorithm uses adaptive thresholding, contour recognition, and scanning of binary patterns to determine the marker ID [6]. While the WSM work offers insights into marker design and ID encoding, it does not address variable numbers of markers or deep-learning-based assignment.

1.2 Object Detection and Assignment Models

The problem resembles object detection and assignment: given a set of light points (similar to objects), the network must cluster them into trackers (objects) and identify their configuration (class). Two families of models are relevant.

Anchor-based detectors (e.g., YOLO). YOLO and its successors treat detection as a regression problem. The image is divided into a grid, and for each grid cell a convolutional neural network (CNN) predicts a fixed number of bounding boxes and class probabilities. Each box includes coordinates, width/height (or offsets), and a confidence score. YOLO's original version uses 24 convolutional layers followed by two fully connected layers [7]. A known limitation is that each grid cell can only predict a limited number of boxes and one class, which causes localization errors and difficulty detecting small or overlapping objects [8]. Later versions incorporate anchor boxes, batch normalization and multi-scale predictions to improve detection of different object sizes [9].

Set-based detectors (e.g., DETR). The Detection Transformer (DETR) formulates object detection as a set prediction problem. A transformer encoder–decoder produces a fixed-size set of object queries that interact with image features through cross-attention. During training, a bipartite matching (Hungarian algorithm) between predictions and ground-truth objects minimizes a set-based loss, eliminating the need for non-maximum suppression [10]. This approach ensures one prediction corresponds to each ground-truth object and gracefully handles variable numbers of objects by using a fixed number of queries padded with "no object" predictions. A precursor, DeepMultiBox (2014), also generated a small set of bounding boxes and solved an assignment problem to train them, enabling cross-class generalization [11].

1.3 Neural Networks for Variable-Size Inputs

A key challenge is processing variable numbers of light points. Point clouds are a similar representation: unordered sets of points with no regular grid. **PointNet** processes point clouds directly by applying a shared multi-layer perceptron (MLP) to each point and then aggregating with a symmetric function (e.g., max pooling). This ensures **permutation invariance** and robustness to input perturbations; the network summarises the input by selecting a few key points [12]. Additional dimensions (e.g., color, intensity) can be appended, and the network is robust to missing or extra points because max pooling selects the most informative features [13].

Set Transformer extends this idea by using **self-attention** to model pairwise interactions among elements in a set. The authors note that set-input problems require permutation invariance and the ability to process inputs of any size. Standard feed-forward networks and RNNs do not satisfy these requirements. The Set Transformer introduces an inducing-point mechanism to reduce the quadratic complexity of self-attention from $O(n^2)$ to $O(n \times m)$, enabling efficient processing of variable-sized sets. Attention enables the model to capture relationships among points (e.g., distances and angles between lights) beyond what simple max pooling can achieve.

1.4 Dynamic Neural Networks and Variable-Size Outputs

Assigning a variable number of trackers implies a variable number of outputs. Modern detectors handle this by producing a fixed maximum number of predictions and using confidence scores to indicate presence or absence. DETR uses a fixed set of queries; predictions matching no ground-truth are treated as “no object” using bipartite matching [10]. This approach avoids non-maximum suppression and yields a one-to-one assignment between predictions and ground-truth objects. Dynamic neural networks that adapt to input size (e.g., with gating or token pruning) exist, but variable-size outputs are typically handled by fixed-size outputs with padding and mask indicators.

2 Input Representation

Four potential input schemes were proposed:

1. **Raw image (synthetic or real) showing lights.** This is the most straightforward: all visible lights are represented in a fixed-size image. Convolutional detectors (e.g., YOLO, DETR) excel at this input.

2. **Fixed-size list of light coordinates.** Each light is represented by its (x, y) coordinates. If there are N visible lights per image, the input is a matrix of shape ($N\text{max} \times 2$), where $N\text{max}$ is the maximum expected number of lights. Missing positions are padded with a sentinel value (e.g., −1). This representation may be processed by an MLP or transformer.

3. **Variable-size set of coordinates.** Each light is represented by its (x, y) coordinate; the number of lights varies across images. This representation is naturally processed by set-based models like PointNet or Set Transformer.

4. **Combination of image and coordinate features.** The network receives both the raw image and the coordinates of detected lights (either fixed-size or variable). Combining modalities may improve performance.

2.1 Handling Variable Inputs

The Set Transformer and PointNet provide principles for handling variable inputs. PointNet uses a shared MLP and max pooling to aggregate features, ensuring invariance to the order and number of points [12]. However, PointNet does not model interactions between points; it might struggle to distinguish different tracker configurations that depend on relative geometry (e.g., two lights on the same triangle edge). The Set Transformer uses self-attention to capture pairwise relations and can process input sets of any size. For our problem, attention would allow the network to compute distances and angles between lights, essential for grouping them into triangles.

Alternatively, one can avoid variable input by padding the coordinate list with a fixed maximum number of lights. This allows use of feed-forward networks but may waste capacity when the number of lights is far less than the maximum. A hybrid approach is to use a transformer with masking; the masked tokens correspond to absent lights.

2.2 Recommended Input Scheme

A **two-stream input** combining the raw image and the variable-size set of detected light coordinates appears promising:

- **Image stream:** A CNN or vision transformer processes the full image. This provides contextual information about brightness, background noise and occlusions.
- **Coordinate stream:** A Set Transformer processes the variable-size set of detected light coordinates (x, y). Because the coordinates themselves contain spatial information, self-attention can model pairwise distances and help assign lights to trackers.
- **Fusion:** The outputs of both streams are fused via concatenation or cross-attention. This combination can exploit detailed visual cues and precise geometric relations.

The alternative schemes have drawbacks: using only images requires the network to infer geometry implicitly, which may be challenging for synthetic lights with minimal texture; using only fixed-size coordinate lists can handle variable numbers of lights but may ignore image context and require padding. The variable-size set representation with attention is more flexible and better at capturing geometric patterns.

3 Architecture Design

3.1 Overview

The goal is to assign lights to trackers and estimate each tracker's configuration and centroid. The architecture must handle variable numbers of lights and produce a fixed maximum number of tracker predictions, each with a confidence score, configuration, and position. The key components include:

1. **Feature extraction from the image and light coordinates** (two-stream design).
2. **Interaction modelling to group lights into trackers.**
3. **Set-based prediction with assignment loss** to output a fixed number of tracker predictions.

3.2 Feature Extraction

Image stream: A CNN or vision transformer (e.g., ResNet, Swin Transformer) processes the input image. The output is a feature map or a global feature vector. A lightweight CNN may suffice because the images consist mainly of discrete bright spots on dark background.

Coordinate stream: A Set Transformer processes the set of light coordinates. Each light is represented as a vector including `(x, y)` and optionally additional attributes (e.g., brightness, confidence of detection). The Set Transformer's self-attention layers compute relations between lights (e.g., pairwise distances) and output a latent representation of each light. By using an inducing-point version, the complexity remains linear with respect to the number of lights.

Fusion: The two streams can be fused using cross-attention: the coordinate features attend to the image features, or vice versa. Alternatively, the global image feature can be concatenated to each light's representation before further processing.

3.3 Tracker Assignment Module

After obtaining fused features for each light, the next step is to group lights belonging to the same tracker. Several approaches are possible:

1. **Graph-based grouping:** Model the lights as nodes in a graph and predict edges representing whether two lights belong to the same tracker. Graph Neural Networks (GNNs) or transformer layers can compute pairwise affinities. A clustering algorithm or graph matching can then group lights.

2. **Object-query based detection (DETR-style):** Introduce a fixed number of learnable **tracker queries**. These queries attend to all light features via transformer decoder layers. Each query outputs a prediction representing a tracker: a confidence score, configuration (three integers from 0 to 2), and a 2-D centroid. During training, a Hungarian assignment matches the predicted trackers to ground-truth trackers, similar to DETR [10]. Predictions that do not match any ground truth are treated as "no tracker". This design naturally handles variable numbers of trackers and avoids non-maximum suppression.

3. **Modified YOLO-like detection:** Divide the image into a grid and predict a fixed number of tracker candidates per cell. Each candidate outputs the tracker configuration, position relative to the cell, and confidence. Because the trackers' positions are not restricted to grid cells and may overlap, this design may require many anchor boxes or complex heuristics to assign lights, making training difficult when the number of trackers is variable. YOLO's grid-based limitation (each cell can predict only a few objects) leads to localization errors for small or overlapping objects [8].

Recommendation: The DETR-style **object-query mechanism** is well-suited for the problem. A set of M queries (where $M \geq$ the maximum number of trackers expected) can produce predictions for trackers. Each query is free to attend to any light; the bipartite matching ensures that each tracker is predicted once. This design supports overlapping trackers and avoids the grid limitations of YOLO. The cross-attention between queries and light features implicitly learns to group lights belonging to the same tracker.

3.4 Output Encoding and Losses

Each tracker query outputs:

- A **confidence score** (probability that the query corresponds to a real tracker).
- **Configuration probabilities:** a 27-class softmax representing the 3×3×3 combinations; or three separate softmaxes of length 3 indicating the absence position on side a and the light positions on sides b and c.
- **Centroid coordinates** `(cx, cy)` in normalized image space (0–1). This can be regressed with a smooth L1 loss.
- **Optional: Light assignment mask**: a vector of length equal to the number of detected lights, indicating whether each light belongs to the tracker. This could use a sigmoid per light. However, this increases complexity; grouping may be implicit via attention weights.

Losses: During training, predictions are matched to ground truth via the Hungarian algorithm [10]. The loss includes a classification loss for configuration, a regression loss for centroid, and a binary cross-entropy loss for confidence. To encourage correct grouping, one may add an auxiliary loss on the

attention weights or predicted assignments (if explicit). Alternatively, treat the attention weights as soft assignments and use them only implicitly.

3.5 Training Considerations

- **Synthetic data generation:** The project can use synthetic images where lights are placed according to the 27 configurations. To improve realism, add noise, occlusions, brightness variations and random extra lights. Synthetic augmentation helps the network generalize to real data. As real images are collected, they can be annotated and used for fine-tuning.

- **Maximum number of trackers:** Set *M* (number of queries) to the maximum number expected in any frame. Predictions above the actual number will be matched to “no object” with no loss incurred.

- **Data augmentation:** Randomly drop some lights or occlude them; vary intensities and backgrounds. This will train the network to handle missing or partial trackers.

- **Evaluation metrics:** Use precision and recall of correctly identified trackers, configuration accuracy, and centroid error. Because trackers may overlap, evaluate assignment quality separately.

4 Alternative Approaches and Research Directions

4.1 Variable-Size Input without Deep Learning

Earlier works solve the assignment problem with heuristics. In monocular head tracking, the authors reduce complexity by using anchor sets and geometric constraints to narrow the search space, allowing combinatorial matching of points [2]. Such rule-based approaches could be used as a baseline or integrated with deep learning as post-processing.

4.2 Graph Neural Networks (GNNs)

Instead of transformer queries, one could build a graph where each light is a node and edges represent potential membership in the same tracker. A GNN can learn to predict edge probabilities; clustering or graph matching can then group lights into trackers. GNNs are naturally suited for variable-sized inputs and can model local geometry. They may be easier to train when the number of trackers is small. However, generating the clusters and predicting configurations requires additional components.

4.3 Few-Shot or Meta-Learning

Because each tracker has 27 configurations, the problem resembles classification of structured patterns. Meta-learning or few-shot learning methods could be used to quickly adapt to new tracker shapes or lighting conditions. Prototypical networks operate on sets and compute class prototypes; they can handle variable numbers of examples and classes. However, the pattern space here is small and may not need meta-learning.

4.4 Learning with Unknown Number of Outputs

Dynamic neural networks explore adaptive computation, but variable-size outputs are typically handled via fixed-size predictions with padding and mask indicators, as seen in DETR [10]. Some works propose

dynamic computational graphs or token pruning, but they seldom change the number of output queries. The bipartite matching approach remains the most practical solution.

5 Conclusion and Recommendations

This report surveyed related works and modern architectures for the problem of assigning lights to triangular trackers with a single camera. Key insights include:

- Single-camera tracking has been addressed using geometric heuristics and uniquely designed markers. However, these methods rely on known correspondences or robust marker IDs [1] [4].
- Processing variable numbers of input points requires models that respect permutation invariance. PointNet and Set Transformer provide two frameworks: the former uses max pooling to aggregate features [12], while the latter employs self-attention to capture interactions and can handle any input size.
- Object detection models like YOLO treat detection as regression on a grid and may struggle with multiple overlapping objects or small objects [8]. Set-based approaches like DETR use transformer queries and Hungarian matching to produce a fixed set of detections with end-to-end training [10].

Given these insights, a two-stream model combining image features and coordinate-based features processed by a Set Transformer, with a DETR-style assignment module, is recommended. This design can handle variable numbers of lights and trackers, model geometric relations among lights, and output a fixed number of tracker predictions with confidence, configuration and position. Such an architecture leverages advances in set-based neural networks and object detection while being tailored to the unique aspects of the tracker assignment problem.

---

[1] [2] Wide-Angle, Monocular Head Tracking using Passive Markers | bioRxiv
https://www.biorxiv.org/content/10.1101/2021.09.01.458583v1.full

[3] [4] [5] [6] Real-Time Motion Analysis System Using Low-Cost Web Cameras and Wearable Skin Markers - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC8802714/

[7] [8] [9] YOLO Evolution: Transforming Object Detection 2015-2024
https://viso.ai/computer-vision/yolo-explained/

[10] Rethinking Transformer-Based Set Prediction for Object Detection
https://openaccess.thecvf.com/content/ICCV2021/papers/Sun_Rethinking_Transformer-Based_Set_Prediction_for_Object_Detection_ICCV_2021_paper.pdf

[11] Scalable Object Detection using Deep Neural Networks
https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Erhan_Scalable_Object_Detection_2014_CVPR_paper.pdf

[12] [13] 1612.00593.pdf
https://arxiv.org/pdf/1612.00593.pdf