Deep Learning-based localization of 3D indications from multiple view X-Ray Imaging

Julian Betancur[1], Ihssane Ghalas[1], Kartik Viswanathan[2], Ba-Khuong Dang[2], and Luca Oriani[2]

[1] SAFRAN FRANCE
[2] ENSAI

Abstract. X-ray imaging allows the inspection of metallic turbine blades during production. Around a dozen X-ray projections are studied by trained operators to identify manufacturing defects and/or inaccuracies inside the blade in a non-destructive way. The operator can also infer the localization of the indication in a 3D model of the part because multiple projections are available. The goal of the project is to evaluate the feasibility of using deep learning approaches to localize 3D indications of an object from few X-ray projections. Well known approaches for 2D object detection have been studied to detect indications in Xrays such as YOLO or EfficientDet. We aim to evaluate state-of-the-art 3D detection architectures to localize 3D indications from multiple view X-rays while exploiting the knowledge of the acquisition geometry[9]. Our implementation source code and pretrained model are available at https://github.com/katzkid/MV2Dbasedlifting.

1 Introduction

1.1 State of the Art

Purpose of the Project in an Industrial Context Using X-Rays The aim of this project is to explore the feasibility of automating 3D localization of indications (defects or inaccuracies) in turbine blade models from multiple X-ray projections. Currently, such tasks are performed by operators manually examining a dozen X-ray projections to identify manufacturing defects. The automation of this task can significantly reduce cost while maintaining high-quality standards in production. The challenge lies in accurately deriving 3D information from limited 2D projections, necessitating advanced algorithms and the use of multi-view geometry.

Objectives The project is developed on the basis of the following objectives:

1. **Developing an Enhanced Model:** Modify and adapt the existing MV2D codebase, which is currently optimized for the NuScenes autonomous driving dataset, to process X-ray images of medical datasets.
2. **Implementing a 3D Localization Pipeline:** Incorporate multi-view geometry and X-ray projection techniques to enable effective localization of defects.

3. **Performance Optimization and Validation:** Evaluate and optimize the modified pipeline using public medical datasets (e.g., LIDC dataset) while aligning the approach with Safran's proprietary industrial requirements.

Datasets will include publicly available medical data. The LIDC dataset serves as a proxy for testing and validating the methodology.

Industrial Scenario Safran, a leading aerospace company, faces challenges in automating defect detection in turbine blades using multi-view X-ray imaging. This project addresses the automation of this process using deep learning-based approaches.

Automating defect detection and localization ensures cost-efficiency, consistency, and scalability in production while maintaining stringent quality standards. Multi-view 3D object detection, combined with domain-specific adaptations, is crucial for tackling Safran's unique challenges in non-destructive testing.

1.2 State-of-the-Art 2D to 3D Lifting Algorithms

Introduction to Methods Recent advancements in computer vision have introduced sophisticated 2D-to-3D lifting methods that enable 3D object detection from multi-view 2D images. Key approaches include:

1. **DETR3D:** This method projects fixed object queries into 2D image spaces using camera parameters, enabling the sampling of features for 3D bounding box predictions. While effective in simple environments, fixed queries may cause false positives or missed detections in dynamic or cluttered settings.
2. **PETR:** PETR encodes 3D positional data directly into 2D image features, bypassing explicit 2D-to-3D transformations. This approach enhances efficiency and scalability but may lack fine-grained adaptability.

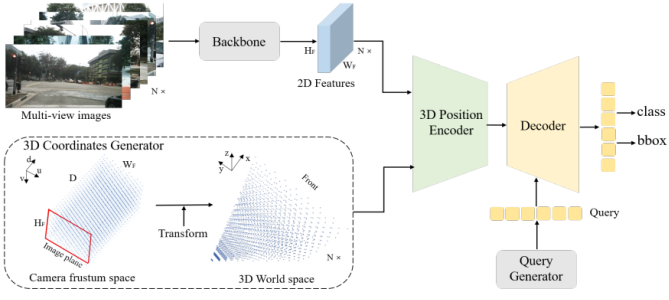


Fig. 1: Illustration of PETR: Encoding 3D positional data directly into 2D image features to enhance efficiency and scalability.

3. **Bird's-Eye-View (BEV):** BEV-based methods transform multi-view images into a top-down representation for easier 3D interpretation. While useful in large-scale scenes, these methods are computationally intensive.

Overview of MV2D MV2D improves upon these methods by dynamically generating object queries based on 2D detections. This approach ensures better precision and recall, particularly for sparse and small features, by combining semantic richness from 2D detections with geometric constraints. Key features include:

- **Dynamic Query Generation:** Queries are derived from 2D bounding boxes, enabling adaptive detection.
- **Sparse Cross Attention:** Reduces noise by focusing on regions relevant to detected objects.
- **Enhanced Accuracy:** Integration of 3D spatial data improves localization while reducing computational overhead.

Why Choose MV2D? MV2D was selected for its ability to exploit the semantic depth of 2D detectors for accurate 3D localization. Its dynamic query mechanism, tailored for sparse feature aggregation, aligns well with the project's requirements for real-time defect localization in X-ray projections. Moreover, its reliance on existing 2D detection advancements ensures continued improvements as the field evolves.

Performance Analysis The two tables provide a comparative analysis of various 3D object detection methods on the nuScenes dataset for both validation and test sets.

Method	Backbone	Resolution	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
DETR3D [41]	ResNet-50	1600 × 900	0.373	0.302	0.811	0.282	0.493	0.979	0.212
PETR [26]	ResNet-50	1408 × 512	0.403	0.339	0.748	0.273	0.539	0.907	0.203
PETRv2 [27]	ResNet-50	1600 × 640	0.494	0.398	0.690	0.273	0.467	0.424	0.195
MV2D-S‡	ResNet-50	1408 × 512	0.440	0.398	0.665	0.269	0.507	0.946	0.203
MV2D-T‡	ResNet-50	1600 × 640	**0.546**	**0.459**	**0.613**	**0.265**	**0.388**	**0.385**	**0.179**
FCOS3D† [39]	ResNet-101	1600 × 900	0.415	0.343	0.725	0.263	0.422	1.292	**0.153**
PGD† [40]	ResNet-101	1600 × 900	0.428	0.369	0.683	**0.260**	0.439	1.268	0.185
DETR3D† [41]	ResNet-101	1600 × 900	0.425	0.346	0.773	0.268	0.383	0.842	0.216
BEVFormer-S† [23]	ResNet-101	1600 × 900	0.448	0.375	0.725	0.272	0.391	0.802	0.200
PETR† [26]	ResNet-101	1600 × 900	0.442	0.370	0.711	0.267	0.383	0.865	0.201
BEVFormer [23]†	ResNet-101	1600 × 900	0.517	0.416	0.673	0.274	0.372	0.394	0.198
PolarFormer [17]†	ResNet-101	1600 × 900	0.528	0.432	0.648	0.270	0.348	0.409	0.201
PETRv2† [27]	ResNet-101	1600 × 640	0.524	0.421	0.681	0.267	0.357	0.377	0.186
MV2D-S‡	ResNet-101	1600 × 640	0.470	0.424	0.654	0.267	0.416	0.888	0.200
MV2D-T‡	ResNet-101	1600 × 640	**0.561**	**0.471**	**0.593**	0.262	**0.340**	**0.368**	0.184

Fig. 2: Validation Set Performance Metrics: Performance metrics such as NuScenes Detection Score (NDS) and Mean Average Precision (mAP) highlight that MV2D-T consistently outperforms other methods with higher NDS and mAP scores, demonstrating superior accuracy and reliability in 3D detection.

MV2D Description The MV2D architecture is organized into the following components:

Method	Backbone	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FCOS3D† [39]	ResNet-101	0.428	0.358	0.690	0.249	0.452	1.434	0.124
PGD† [40]	ResNet-101	0.448	0.386	0.626	**0.245**	0.451	1.509	0.127
BEVFormer† [23]	ResNet-101	0.535	0.445	0.631	0.257	0.405	0.435	0.143
PolarFormer† [17]	ResNet-101	0.543	0.457	0.612	0.257	0.392	0.467	0.129
PETRv2† [27]	ResNet-101	0.553	0.456	0.601	0.249	0.391	**0.382**	0.123
MV2D-T ‡	ResNet-101	**0.573**	**0.483**	**0.567**	0.249	**0.359**	0.395	**0.116**
BEVDet [16]	Swin-Base	0.488	0.424	0.524	0.242	0.373	0.950	0.148
M2BEV‡ [42]	ResNeXt-101	0.474	0.429	0.583	0.254	0.376	1.053	0.190
DETR3D§ [41]	V2-99	0.479	0.412	0.641	0.255	0.394	0.845	0.133
BEVDet4D[15]	Swin-Base	0.569	0.451	**0.511**	**0.241**	0.386	**0.301**	0.121
BEVFormer§ [23]	V2-99	0.569	0.481	0.582	0.256	0.375	0.378	0.126
PolarFormer§ [17]	V2-99	0.572	0.493	0.556	0.256	0.364	0.440	0.127
PETRv2§ [26]	V2-99	0.582	0.490	0.561	0.243	0.361	0.343	**0.120**
MV2D-T§	V2-99	**0.596**	**0.511**	0.525	0.243	**0.357**	0.357	**0.120**

Fig. 3: Test Set Generalization Metrics: The test set results further validate MV2D-T's generalization capabilities, achieving advanced performance across key metrics like localization accuracy (mATE), orientation accuracy (mAOE), and velocity prediction (mAVE). These results underscore MV2D-T's robustness and efficiency, making it a standout model for multi-view 3D detection.

2D Detection and Feature Extraction Multi-view X-ray images are processed using a Mask R-CNN-based 2D detector, which identifies and extracts object regions in each projection. Mask R-CNN is well-suited for X-ray analysis due to its ability to detect small, localized anomalies within cluttered visual environments.

Dynamic Query Generation 2D detections serve as the basis for generating dynamic object queries. By incorporating camera parameters and 2D bounding box information, these queries are dynamically tailored to specific regions of interest, enhancing adaptability and precision.

Feature Aggregation A sparse cross-attention module aggregates features from detected regions, minimizing interference from irrelevant areas. This module enhances efficiency and focuses computational resources on critical regions.

3D Bounding Box Prediction The refined object queries are passed through a transformer decoder and a feed-forward network to predict 3D bounding boxes. These predictions integrate 2D detection results and spatial embeddings, ensuring accurate localization of anomalies.

2 Dataset

In this project, we use open LIDC-IDRI database [3] that is available from the NCI Cancer Imaging Archive [4]. TCIA is a comprehensive resource that provides

[3] LIDC-IDRI Dataset: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI

[4] TCIA Collections: https://www.cancerimagingarchive.net

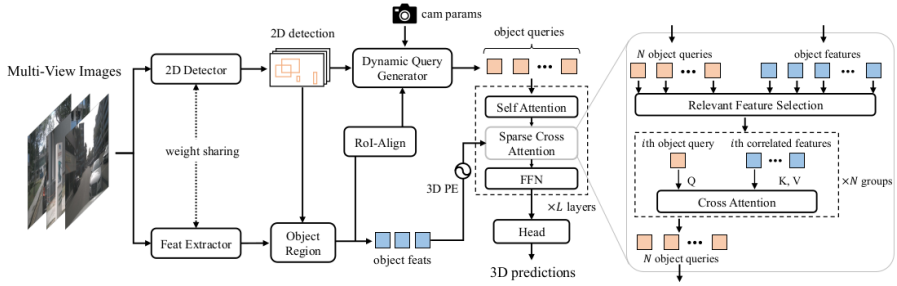


Fig. 4: Graphical representation of the MV2D architecture, showcasing the interaction between 2D detection, dynamic query generation, feature aggregation, and 3D bounding box prediction.

a broad range of medical imaging datasets, including lung cancer image data. Lung Image Database Consortium (LIDC) is a subset of TCIA that focuses on lung cancer detection. The LIDC dataset serves as a benchmark for several machine learning and deep learning tasks, including nodule detection, segmentation, classification, and prognosis modeling [1]. The original dataset contains CT scan for more than 1000 patients in DICOM format, with 3D annotations for ground truth bounding boxes of nodules. In this project, our input data consists of 791 items, with 632 items used for training (80%) and 159 items used for testing (20%).

2.1 Scans and Annotations from DICOM

The LIDC-IDRI dataset is complex, as it includes multiple annotations per nodule from different radiologists, stored in a structured DICOM format. We use `pylidc` to access and process information from LIDC, as it provides high-level API for querying, visualizing, and analyzing nodule annotations [5] [3].

3D CT Scan class: CT scan is provided in the DICOM format. Each patient scan is stored as a sequence of 2D slices that together represent a 3D volume with the shape (H, D, W), representing the number of slices along (x, y, z) axes respectively. In order to convert from voxel representation to physical dimension (in mm), we use the `slice_thickness` and `pixel_spacing` attributes of DICOM standard format. Slice thickness measure the nominal spacing between adjacent slices from the center-to-center of each slice. Pixel spacings are encoded as the physical distance between the centers of each two-dimensional pixel, specified by a numeric pair [6].

Annotation: Annotations for each patient is given as a segmentation mark up for each slice if nodules are present. A nodule has many contours, each of which refers to the contour drawn for nodule in each scan slice. For each slice, we

[5] `pylidc package`: https://pylidc.github.io/tuts/scan.html
[6] DICOM Image plane attributes: https://dicom.innolitics.com/ciods/rt-dose/image-plane

will have cluster annotations, because each nodule is annotated at a maximum of 4 doctors, and each doctors have annotated the malignancy of each nodule in the scale of 1 to 5. The annotations are store as slice objects that can be used to index into the image volume corresponding to the bounding box. In particular, 1 nodule class is represented as a matrix $A \in \mathbb{R}^{3\times 2}$, where $A_{i,:}$ denotes the start and end indices of the nodule along the i-axis. This matrix serves as the ground truth for the 3D bounding box of the nodule. Since slices are indexed using the CT local coordinate system, with $(0, 0, 0)$ at the top-left corner, we translate the bounding box so that the origin $(0, 0, 0)$ is located at the center of the volume. This translation is performed to account for variations in CT scan measurements, and is done by subtracting CT center mass from the coordinate of bounding box. Following the convention of `NuScence` dataset in MV2D [9], ground truth 3D boxes are stored as `[x, y, z, dx, dy, dz]`. Figure 5 shows example of a slice with corresponding annotations.

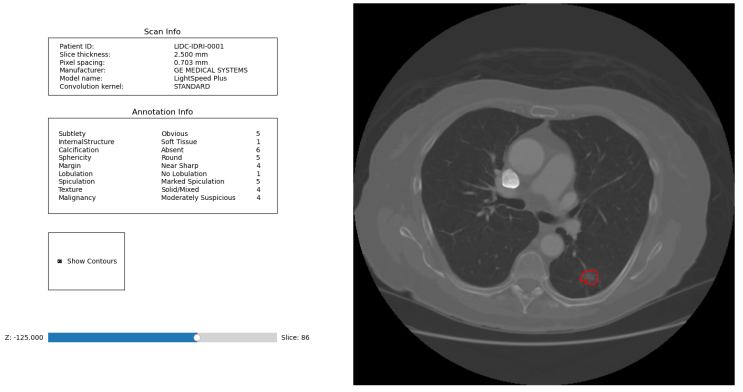


Fig. 5: Example of CT scan for patient LIDC-IDRI-0001. This shows slice index 86 along the Z axis. Using matrix indexing, this can be queried by `scan[:, :, 86]`. The slice has 1 annotation, and it's properties judged by doctor. The subtlety of the nodule is the highest (5), which means the nodule is easy to detect.

2.2 Data formatting

MV2D, like most implementations in the MMDetection framework, is trained on standard datasets such as `nuScenes`, which is not suitable for our application. Specifically, `nuScenes` contains data in a LiDAR-based 3D system, accompanied by point cloud files for training and inference. Consequently, it is necessary to rebuild our database and data class to support our training and testing pipeline. In this section, we present our method for preparing the dataset to adhere to the standard structure of the MMDetection framework. Generally, MMDetection requires two database information as inputs for the model: one containing ground

truth 3D bounding boxes and another for 2D annotations. Figure 7 shows the detail schema of annotation structures for LIDC dataset that we adapt from MV2D.

Information about 3D ground truth is collected into a `pickle` file, based on `nuScence` database from MV2D. This is a list of information for each patient, here we present some important keys that will be used in our code:

- metadatas of CT scan: `sample_id`, `lidc_id_ref` which is cross-reference with the original LIDC dataset.
- Camera infos: dictionary with 10 keys corresponding to 10 camera positions. The `data_path` points to our 2D reconstructed Xray images. In here, we also store the information about our data parameter (intrinsic matrix). Each camera is uniquely recognized by a data token of the form: `patientIDcamXX`.
- Information about ground truth boxes: `gt_boxes` $\in \mathbb{R}^{b\times 7}$, with b is the number of nodules present in patient's CT scan. Each box is represented by 7 parameters: `[x, y, z, dx, dy, dz, yaw]`. For our dataset, the `yaw` rotation is 0. Similarly, we have an array of ground truth label `gt_names` $\in \mathbb{R}^{1\times b}$ that assigns class for b nodules. Our project is a binary classification task with only 1 class: `nodule`.
- `valid_flag` is used in nuScence to filter out noisy data or invalid objects. We include this attribute in our dataset for completeness only, all our nodules are valid.

2.3 X-ray reconstruction

Experiment setup We use `DiffDRR` [2] to digitally reconstructed radiograph (DRR) to generate Xray 2D images from 3D CT scan [7]. `DiffDRR` constructs synthetic X-ray images by using Siddon's method, an exact algorithm for calculating the radiologic path of an X-ray through a volume, as a series of vectorized tensor operations. Each X-ray path originates from the source and travels to the detector, symmetrically positioned around the CT object. Based on this, our experimental setup is as follows: we virtually place 10 cameras, evenly spaced around the 3D volume. All cameras and image planes are aligned with the center of the CT volume. This setup corresponds to a sequential rotation around the Z-axis and a translation of $300mm$ from the CT origin. Each rotation is given by $rot_i = 36°i$ with $i \in [0..9]$ is the index of camera. Figure 6 illustrates 3D view of our experiment setup, which shows 3 different positions of camera: `cam00` ($\theta = 0°$), `cam05` ($\theta = 180°$), and `cam09` ($\theta = 324°$). The source to detector distance is $300(mm)$, which corresponds to $600mm$ C-Arm in real world Xray machine.

Coordinate transformation MV2D leverages image features and projected 3D points to generate 3D position-aware features (building upon PETR [6]), thus coordinate transformation plays an important role in how to train the model.

[7] `DiffDRR`: https://github.com/eigenvivek/DiffDRR

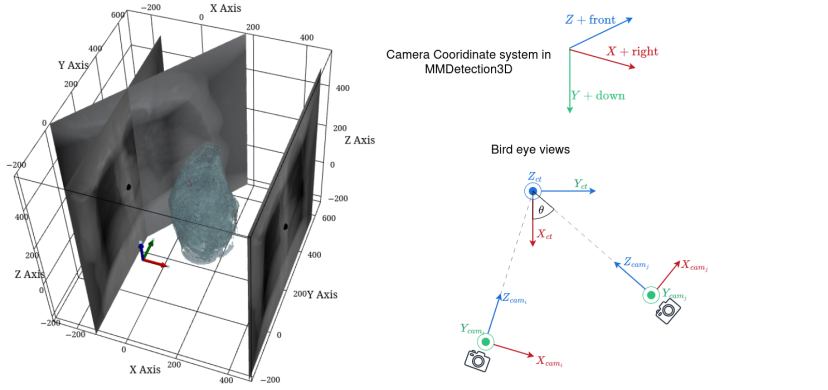


Fig. 6: Visualization of X-ray reconstruction setup. Left: 3D view of different camera positions (`cam00`, `cam05`, `cam09`). Right: Camera coordinate system defined in MMDetecion engine: y-axis points to the ground, the x-axis points to the right, and the z-axis points to the front. Below is the bird eye view of our setup, which illustrates the coordinate system transformation from CT center coordinate to camera local coordinate, using rotation angle θ

MMDetection3D uses three different coordinate systems. According to their documentation, the existence of different coordinate systems in the society of 3D object detection is necessary, because for various 3D data collection devices, such as LiDAR, depth camera, etc., the coordinate systems are not consistent, and different 3D datasets also follow different data formats. In partiuclar, `nuScence` uses several systems in their data: LiDAR, ego, sensor, local camera and global world coordinates. Therefore, it is crucial to align our coordinate system with the `nuScenes` and MMDetection3D frameworks. In this section, we present our method for performing 3D coordinate alignment to adapt to these frameworks.

We first denote coordinate systems that are used in our setup:

- CT centered coordinate (ct): this is our world coordinate, with (0, 0, 0) is at the center of CT volume and is the same for all patients. Our ground truth bounding box lives in this coordinate. CT coordinate has Z axis points upwards, Y axis to the right and X axis towards the camera. (Right-hand coordinate system).
- Camera local coordinate ($\text{cam}_i \quad i = 0..9$): this is local coordinate of each camera, which we follow the definition of camera system from MMdetection3D. The positive direction of the y-axis points to the ground, the positive direction of the x-axis points to the right, and the positive direction of the z-axis points to the front (towards object and DRR). Figure 6 illustrates the 3D view of camera coordinate system in MMDection3D. For each camera position, its local coordinate is realized by doing rotations and translations from CT coordinate. In particular, for a given camera cam_i :

- The rotation matrix is defined as:

$$\mathbf{R}^{i}_{\text{cam}} = \mathbf{R}_i \cdot \mathbf{R}^{\text{global}}_{z} \cdot \mathbf{R}^{\text{global}}_{x},$$

where:
  - $\mathbf{R}^{\text{global}}_{z}$ is the same for all cameras and represents a rotation of $\frac{\pi}{2}$ radians (90 degrees) in the counter-clockwise direction, aligning the x-axis to the right.
  - $\mathbf{R}^{\text{global}}_{x}$ is used to rotate the y-axis in the clockwise direction, pointing it downward for all camera positions.
  - Finally, $\mathbf{R}_i = 36^\circ \times i$ is applied to represent the changing position of the camera, where i corresponds to the camera index.

$$R_z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow R = R_x \cdot R_z = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

This corresponds to the camera at 0°.

$$C = \begin{bmatrix} 0 \\ 0 \\ 300 \end{bmatrix}$$

General form of a transformation from World to Camera is given as

$$p_c = R(p_w - C)$$

where C is the centre of the camera space w.r.t the world coordinates. The overall transformation is given as:

$$T = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Now lets take into account the 10 cameras around the object each 36° apart. The camera center as a function of θ is given as:

$$C(\theta) = \begin{bmatrix} r\cos(\theta) \\ r\cos(\theta) \\ 0 \end{bmatrix}$$

- The Z coordinate of the camera space is given as:

$$Z_c(\theta) = \frac{0 - C(\theta)}{||C(\theta)||} = - \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{bmatrix}$$

- The X coordinate of the camera space is given as the cross product of Z_c and the world up coordinate Z:

$$X_c(\theta) = Z_C(\theta) \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \\ 0 \end{bmatrix}$$

- Finally $Y_c(\theta)$ is obtained as cross-product of $Z_c(\theta)$ and $X_c(\theta)$

$$Y_c(\theta) = Z_c(\theta) \times X_c(\theta) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- The overall transformation in θ is given as:

$$T(\theta) = \begin{bmatrix} R(\theta) & -R(\theta)C(\theta) \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -\cos(\theta) & \sin(\theta) & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We should note that this rotation matrix transforms a point from CT coordinate to camera local coordinate. To map a point from camera coordinate back to global coordinate, we can use the inverse of R^i_{cam}, and since for rotation matrix, the inverse of matrix is its transpose, we have:

$$T^i_{ct}(\text{cam} \rightarrow \text{global}) = (T^i_{cam})^{-1} = (T^i_{cam})^T \quad = \begin{bmatrix} -\sin(\theta) & 0 & -\cos(\theta) & 0 \\ \cos\theta & 0 & \sin\theta & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & r & 1 \end{bmatrix}$$

– Camera intrinsic and extrinsic matrix: camera global position can be fully encoded using intrinsic and extrinsic matrix, which we can build from previous rotation and translation matrix. Intrinsic parameter reflects the internal characteristics of a camera, thus it will be the same for all of our virtual camera. By using matrix multiplication of extrinsic and intrinsic parameters, we can combine transformation from global to camera, and then from camera to image plane. The mapping of any point from global coordinate to pixel position is done with homogeneous coordinate system.

$$\text{Intrinsic:} \quad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Extrinsic:} \quad \begin{bmatrix} -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -\cos(\theta) & \sin(\theta) & 0 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{ct2cam transformation: } \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\text{ct2img} = \text{lidar2img} = \begin{bmatrix} K_{3\times3} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{3\times1} & 1 \end{bmatrix}$$

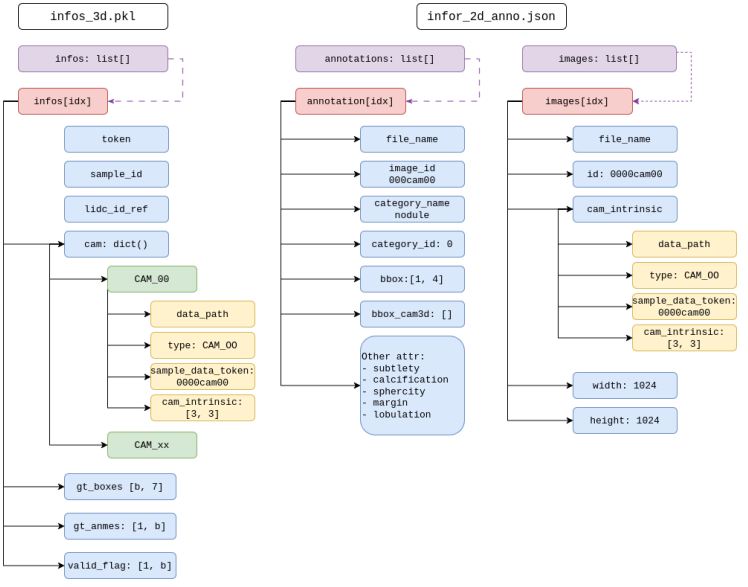


Fig. 7: Schema of information for ground truth database. Information about 3D bounding boxes follows `nuScence` data converter (left), while information about 2D annotation follows `COCO` data format (right).

Information about 2D annotations: we follow COCO annotation format [8], as suggested by MMDetection3D. The annotations are stored using `JSON`, and consists of 2 lists: the annotations list and the images list. A pair of image and annotation is linked together using `image_id` key in the annotations list. We should note that since we only have 1 object class in our classification task, our category is constructed as a singular list. In general, we do not declare the non-object background (normal) class. MV2D uses the Hungarian loss for loss calculation and back propagation, which automatically matches the background to the `null` object.

[8] COCO data format: https://cocodataset.org/#format-data

- Annotation item: each object instance annotation contains a series of fields, including `image_id` to match annotation with image and `file_name` points to the corresponding image that contains the current box. The categories field of the annotation structure stores the mapping of category id to category. An enclosing bounding box is provided with $\texttt{bbox} \in \mathbb{R}^{1\times4}$. Box coordinates are measured from the top left image corner and are 0-indexed. Finally, we also store some characteristics of the nodule extracted from LIDC dataset, such as: subtlety, margin, lobulation, etc...
- Image item: each image contains not only it's attributes but also intrinsic parameter of camera that capture it.

3 Training configurations and metrics

3.1 Data processing pipeline

All images in the dataset have 1024x1024 pixels. Pixel resolution of the images is 0.78125 mm. The images are in `.png` format (RGB, 24 bits). Pixel intensity has been codified in 8 bits. Our data is processed by customized MMEngine DataClass and encapsulated in a DataContainer structure, designed to be fully compatible with PyTorch's DataLoader. We apply photo metric distortion to image sequentially, with a transformation probability of 0.5. Our transformation are: random brightness, random contrast (mode 0), random Gamma Correction, random Gaussian noise, random Gaussian blur, random salt and pepper noise, and random vignetting effect.

3.2 Training hyperparameters

We train our model using AdamW with initial learning rate is 2×10^{-4}. Learning rate has a scheduling policy CosineAnnealing, where the learning rate decreases following a cosine function from the initial value to a minimum value, which is 10^{-3} of the initial learning rate. Additionally, learning rate has a warm up policy for the first 500 iterations, starting at $\frac{1}{3}$ of base learning rate. Our model is trained with 24 epochs.

4 Implementation Details

4.1 2D Detections

Feature extraction : Our input data to the model is $I \in R^{1\times10\times3\times1024\times1024}$, which represents 10 views images for 1 patient, each image has 1024×1024 with 3 channels color [9]. First, we use backbone network (ResNet-50) to extract the feature maps, followed by Feature Pyramid Network (FPN) to capture feature maps at different scales. Our result is a list of different levels of feature maps

[9] our image has 3 duplicated color channels, technically we can train the model using only 1 channel

$\{\mathcal{F}_i \quad i = 1...5\}$, with $\mathcal{F} = \{\mathbf{F}_v \in R^{C \times H^f \times W^f}\}$. In our experiment, we use 5 levels of feature map, each has the number of output channel is $C = 256$, and our FPN is built to produce feature maps with downsample stride $\{8, 16, 32, 64, 128\}$

Region proposal : Using this feature map, a list of potential object locations is generate by Region Proposal Network (RPN), which uses anchor boxes that are centered at each location of the feature map. For each image, we receive a result of $Proposal_i \in R^{N \times 5} \quad i : 1...10$, where N is the number of potential bounding boxes. The last dimension with 5 elements represents $(tl_x, tl_y, br_x, br_y, score)$ - top-left, top-right, bottom-left, bottom-right coordinates of the box and classification score to determine if the anchor contains an object or not. Non-Maximum Suppression (NMS) is used to remove redundant or overlapping bounding boxes. We impose some parameters to control the proposal for anchor boxes, namely: maximum number of region per image is 1000, Intersection over Union (IoU) threshold for NMS is 0.6.

2D Detections is calculated based on the upstream features and the proposal list, using Faster-RCNN as base detector. Our result is a set of 2D object bounding boxes $\mathcal{B} = \{\mathbf{B}_v, \quad v = 1...10\}$ where $\mathbf{B} \in R^{M_v \times 6}$ represents the predicted 2D bounding boxes in the v-th image and M_v is the number of detected boxes for each image. 6 elements in the last dimension of $\mathbf{B}_v$ are $(x_1, y_1, x_2, y_2, \text{score}, \text{label}_{id})$, correspond to 4 coordinates of bounding boxes in image plane (pixel), the classification score and the predicted object label respectively [10]. Similar to RPN layer, there are hyperparameters that we use to control the selection of the boxes: the score threshold for a box to be select is 0.05, the maximum number of box per image is 100, and the IoU threshold for NMS is 0.7. Finally, another FPN is applied to the detection feature map (5 levels), we only process and extract the middle layer (level 2) to be used in 3D detection module.

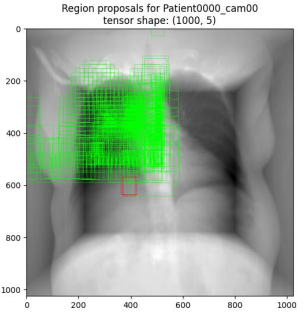


Fig. 8: Original X-Ray image with region proposals (in green).

2D loss functions is taken directly from base detector Faster R-CNN [7], and is defined as:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda \frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*).$$

[10] Our problem is a binary classification, so our label id will be 0 = nodule for all predicted boxes

Here, i is the index of an anchor and p_i is the predicted probability (our score) of anchor i being an object. p_i^* is the ground-truth label, an indicator function that return 1 if the anchor is positive and 0 if the anchor is negative (background). t_i and t_i^* are vectors representing the 4 coordinates of the predicted bounding box and ground-truth box respectively. The classification loss L_{cls} is log loss over two classes (object vs. not object). The regression loss L_{reg} is the L_1 robust loss function. Both terms are normalized by N_{cls} and N_{reg}, which are the size of batch and the number of anchor points.

4.2 Dynamic Object Query

MV2D is built on top of PETR [6] with one major improvement: it introduces the concept of dynamic query generation. One drawback of existing object lifting models is requirement of dense fix-length object queries ditributed in 3D space to ensure sufficient recall of object. Dynamic query generator uses 2D objects guided feature and can adapt to the image feature maps and camera parameters to select and recall the object without noise or distraction.

This is done in 2 steps:

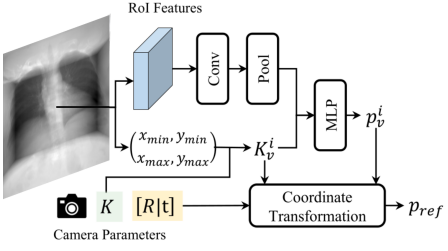


Fig. 9: Dynamic object query generator. Adapt from MV2D paper [9]

Step 1: Mask R-CNN [4] is the 2D object detector. It contains the RPN (Region proposal network). RPN is a neural network that generates regions of interest (RoIs) based on the identification of potential objects in the image. Each region of interest is the bounding box and each bounding box is classified. Starting from the last processed feature maps $\mathbf{F}_v \in R^{256\times32\times32}$ $v \in [1, 10]$ and a list of 2D bounding boxes $\mathbf{B}_v \in R^{M_v\times4}$, MV2D models uses Roi-Align from Mask R-CNN [4] which extracts feature maps from the RoIs. In our case each bounding box corresponds to 7×7 RoI align feature map having 512 channels. $\mathcal{O} = \{\mathbf{O}_v\}$ is a set of all RoI feature from all 10 camera images, our RoI feature is obtained by:

$$\mathbf{O}_v = \text{RoI-Align}(\mathbf{F}_v, \mathbf{B}_v) \in R^{H^{roi}\times W^{roi}(7\times7)}$$

Step 2: Camera Frustum Coordinate position encoding and 3D Positional Encoding

The object feature RoI is processed by a simple Convolution and Pooling network to learn about local representation of the object. After this, it contains sufficient information to infer the object center location in image space.

This step creates an intermediate set of feature maps where the camera intrinsic params are positionally encoded with the RoI feature maps, and we obtain

the 3D camera frustum coordinate space Figure 10. For each bounding box in an image the neural network learns the representation of the object location. p_v^i is the co-ordinate space that is generated from the co-ordinate space of the intermediate updated feature maps. This is the camera frustum coordinate positional encoding where we encoded the 2D features/ bounding boxes over to the 3D camera space.

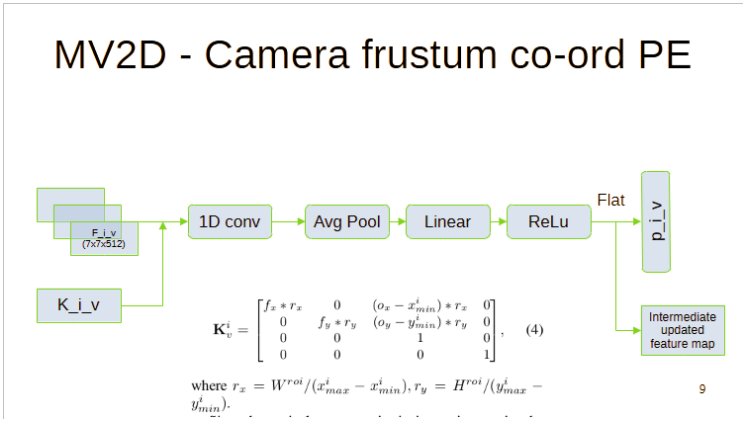


Fig. 10: Camera frustum Positional Encoding

The downside is we miss geometric information about the object, as the object region is rescaling into fixed sized RoI. To compensate for this, we include camera parameters (extrinsic and intrinsic matrices as explained in section...) to obtain the reference point in the world coordinate space $\mathbf{p}^i_{ref}$ of each object in global space. By this transformation, MV2D generates a set of dynamic reference points $\mathcal{P}_{ref} = \{\mathbf{P}_{ref,v} \in R^{M_v \times 3} \quad v = 1...10\}$ guided by 2D bounding boxes, object features and camera parameters. These reference points can be seen as encoded location of object in global space.

These $\mathcal{P}_{ref}$ points are normalized based on the point cloud dimensions of the scene and then positionally encoded using `SinePositionalEncoding3D`. Linear transformation of these PE values form the dynamic object queries. This is a major improvement over the PETR [6] architecture.

4.3 3D object detection

The transformer in the MV2D is a decoder transformer. But there are serious improvements made to the transformer decoder compared to the implementation in DETR. The dynamic object queries is processed through a DETR [8] transformer decoder with position embedding from PETR [6] (as explained in the previous section). MV2D relies heavily on a "Dynamic Query Generator" that produces object queries conditioned on the output of a 2D object detector. This

generator utilizes pre-extracted image features, 2D detection boxes, and camera parameters to create the queries, suggesting that these queries already encapsulate significant information from the input images, potentially eliminating the need for a separate encoder.

Decoder with Sparse Cross Attention The decoder takes 2 inputs:

- 1.**Object Queries:** Set of learned positional embedding that represent potential objects in an image.
- 2.**Intermediate features:** Multi-view intermediate feature maps from the dynamic query generator stage.

The decoder uses **cross-attention** to attend to relevant regions in the feature map for each object query. We use 6 layers of transformer decoder, each layer composed of a self attention layer, followed by a cross attention layer. Both are multi-headed attention layer with 8 heads. Finally we apply a feed-forward network (FFN) with 2048 hidden dimensions. Each object query produces a prediction (class label and bounding box).

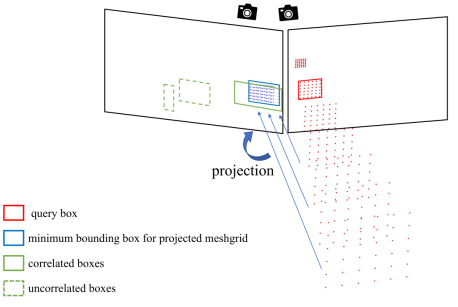


Fig. 11: Illustration of relevant region selection. Figure from MV2D [9]

Relevant Object Feature Selection MV2D tries to improve the existing models by filtering out irrelevant features before putting them to the transformer (i.e making the input *sparse*). Instead of calculating key-value pair for all 2D bounding boxes from 10 images, the model associates each bounding box only with its relevant boxes from other views by relevant object selection method.

For a given 2D bounding box $\mathbf{b}_v^i$ of v-th image as the target object, MV2D creates a 3D meshgrid by projecting it to several parallel planes in global space. Denotes $\mathbf{G} \in R^{7\times7\times D\times4}$ the meshgrid for each RoI, each point from the RoI will transform into homogeneous coordinate $\mathbf{g_{x,y,z}} = (x \times dx, y \times dy, dz, 1 \in G$, where (x, y) is the coordinate in image plane, $d_z \in \mathbf{D} = \{d_0, ..., d_{D-1}\}$ is a predefined set of distance from image - which can be seen as the depth value of object.

The meshgrid $\mathbf{G}$ is then projected into other image planes $\{w \in [1, 10] \quad w \neq v\}$. The whole process can be seen as a transformation $T_{v\to w}$ of 1 point from v-th image to w-th image, using camera extrinsic and global space as an intermediate.

The relevant selection is done using two 2 methods: **top IoU** selects boxes that have the highest intersection-over-union with target box, and **all overlapped** selects all the boxes that have the overlap with target box.

3D loss and overall loss functions: In the implementation of MV2D, the 3D object detection loss consists of two parts: a focal loss [5] for the class labels

and a L_1 loss for the bounding box parameters. Normally, the number of predict boxes M^* is larger than the number of ground-truth box M for each image. To account for this difference, we follow the method of DETR3D [8], we pad the set of ground truth boxes with ∅s (empty object) up to the number M^*. The matching between prediction and ground-truth is done using Hungarian algorithm via bipartite matching, so we establish a 1-to-1 matching.

Denote σ as the set of 1-to-1 assignment between ground truth $y(c, b)$ and prediction $\hat{y}(\hat{c}, \hat{b})$, the 3D loss for classification and regression is:

$$\mathcal{L} = \lambda_{cls3d} \times \mathcal{L}_{cls3d} + \mathcal{L}_{reg3d}$$

where λ_{cls3d} is hyperparameter controls the balance between classification loss and bounding box regression loss. We set $\lambda_{cls3d} = 0.25$ in our experiment.

The overall loss function of MV2D is a combination of 2D and 3D loss:

$$\mathcal{L} = \mathcal{L}_{2d} + \lambda_{3d} \times \mathcal{L}_{3d}$$

with hyperparameters λ_{3d} controls the weight balance between 3D detection and 2D detections losses. We set this term to 0.1 in our experiment.

4.4 MV2D model description (implementation)

The MV2D architecture is a sophisticated multi-view 2D object detection model that combines:

- A Faster R-CNN-like backbone with deformable convolutions.
- A transformer-based decoder for refining predictions.
- Positional encoding and cross-attention mechanisms for better feature fusion.
- ROI-based detection with advanced loss functions and proposal matching.

This architecture is well-suited for tasks requiring accurate 3D object detection from 2D multi-view inputs, such as autonomous driving or robotics.

Model type

- `type='MV2D'`: The model is a multi-view 2D object detection network. It leverages 2D image features from multiple views to predict 3D bounding boxes for objects.

Grid Mask

- `use_grid_mask`: This module applies a grid mask to the input images to improve generalization and prevent overfitting. It includes:
  - `use_h` and `use_w`: Enable masking along height and width dimensions.
  - `rotate`: Rotates the grid mask.
  - `ratio_range`: Controls the density of the mask.
  - `mode` and `prob`: Define how and when the mask is applied.
  - `interv_ratio`: Adjusts the interval between masked regions.

Base Detector

- `base_detector`: We use MaskRCNN [4] as our base detector. We follow MV2D [9] by initializing the base detector with checkpoint of the 2D detectors on nuImages. The backbone for feature extraction is ResNet50, pretrained weight is imported from TorchVision. Faster R-CNN [7] is used for 2D object detections. The backbone of the model is based on a Faster R-CNN-like architecture with the following modifications:
  - **Backbone:**
    - `with_cp`: Determines whether checkpointing is used to save memory.
    - `dcn`: Deformable Convolution (DCNv2) is applied in specific stages (`stage_with_dcn`) to handle geometric transformations effectively.
  - **Neck:**
    - `type='FPN'`: A Feature Pyramid Network (FPN) is employed to extract multi-scale features.
    - `in_channels` and `out_channels`: Define the number of input and output channels for the FPN.
    - `start_level` and `end_level`: Specify the range of feature levels to utilize (in this case, only `p4` with a downsampling rate of 16 is used).

ROI Head

- `roi_head`: The Region of Interest (ROI) head processes proposals from the base detector and refines them into final predictions.
- `MV2DSHead`: Multi-View 2D Segmentation Head
  - `bbox_roi_extractor`: Extracts features from the ROI using RoIAlign.
  - `bbox_head`
    - `CrossAttentionBoxHead`: Cross attention for feature fusion
    - `num_classes`: Binary classification for LIDC
    - `group_reg_dims`: Groups regression dimensions for better localization.
    - `transformer`: PETRTransformerDecoder with 6 layers for refining predictions.
    - `bbox_coder`: Encodes and decodes bounding boxes using NMSFreeCoder.
    - `loss_cls and loss_bbox`: Define the classification (Focal Loss) and regression (L1 Loss) losses.

Query Generator `query_generator` queries the transformer decoder.

- `with_avg_pool`: Option for average pooling
- `num_shared_convs and num_shared_fcs`: Define the number of shared convolutional and fully connected layers.
- `in_channels and fc_out_channels`: Specify input and output channels.
- `extra_encoding`: Adds additional encoding layers for richer feature representation.

Positional Encoding `pe:` Add positional encoding to the features.

- `SinepositionalEncoding3D:` Uses sine-based encoding for 3D coordinates.
- `strides:` Define the stride for encoding.
- `depth_num:` Number of depth bins for encoding.
- `with_fpe:` Use feature pyramid encoding.

Box Correlation `box_correlation:` Correlates proposals across different views.

- `correlation_mode:` Defines how correlations are matched. 'topk_matched:1:0.0:0.0'

5 Result analysis

Our experiment is based on ResNet-50 backbone, 1024×1024 input resolution with 24 epochs. The 3D bounding box predictions on the test dataset are not convincing, so in this section we provide detailed analysis of the model's architecture step by step. The 3D prediction relies heavily on the result of 2D predictions and the feature maps from the backbone network. In MV2D paper with nuScence dataset, the author also makes argument that 2D object detectors can predict convincing 2D object proposals, the detected object bounding boxes imply which region contains the most distinctive information about an object, and RoI features contain sufficient object appearance information to infer the object center locations in image [9], our assumption is that the failure with LIDC dataset comes from 2D detection phase. The model produces only 3D object predictions without 2D bound boxes, so we run a modified version without the 3D RoI head and the transformer part. Figure 12 shows the predict bounding boxes for Patient0000.

We observe that our model fails to recognize the correct position of nodules in most of the images, with some predictions are quite closed to the ground truth (`cam05`, `cam07`, `cam08`), while others are really far from ground truth (`cam00`, `cam01`, `cam03`). Figure 13a shows that our predictions cluster around the left side of the image, where there is more detail than on the right side. In Figure 13b where our ground truth is located on the left side, we have better results with 2D predictions are closer to the ground truth, and the box that has score 1 seems to match the ground truth, but we still do not have matching for box location and size. In general, Figure 12 suggests that our model mostly focuses on region that has more details in the inputs, and it fails to recognize nodules that are located in blurry region. This comes from the nature of X-Ray image, where sometime the details are not so easy to be captured. Starting from this observation, we trace back the forward pass process to investigate the problem.

5.1 Regional proposal

Faster R-CNN is a two stage detector that generate RoI and set of bounding boxes from the regional proposal network (RPN). Figure 14 shows the region proposals for Patient0000, and we observe that our RPN detections are almost

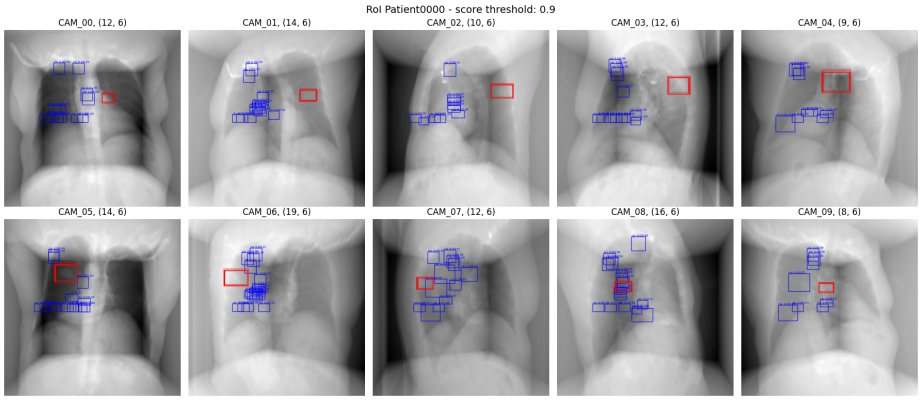


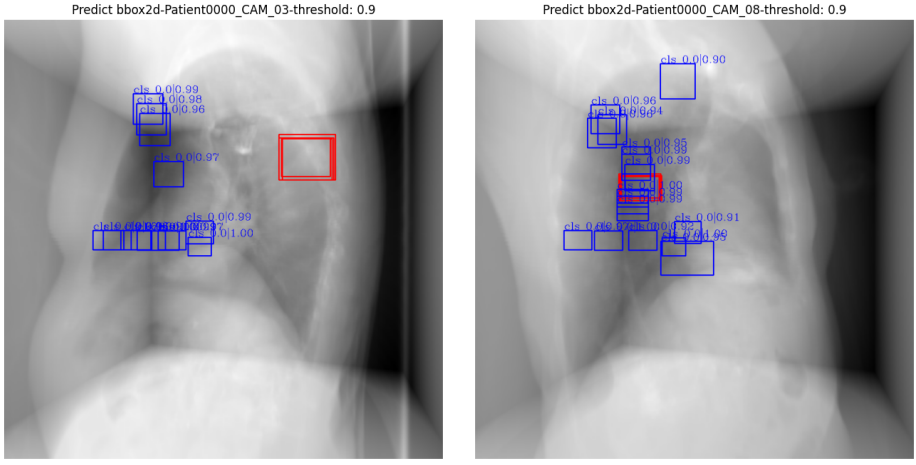
Fig. 12: RoI of Patient0000 with 10 different camera positions. The score threshold is set to 0.9, so the model will only select bounding box that has classification score larger than 0.9. The predictions are tensor of shape `(N, 6)` with `N` is the number of predicted 2D boxes, 6 elements in the last dimension represent `(x, y, dx, dy, class, classification score)`

the same for all images, even though the position of ground truth bounding boxes are different for each camera position. From camera 05 to camera 09, we have proposed regions cover correctly the bounding boxes, but the model fails to cover ground truth boxes from camera 00 to camera 03.

5.2 Feature map extraction

Going back 1 step further, we analyse the performance of feature map extraction from Res-Net 50 backbone. Res-Net50 produces 5 levels of pyramid feature network, each has 256 targeted channels. Theoretically, FPN captures information at multiple scale and enhances the model's ability to detect objects of various sizes (small, medium, large objects). Figure 15 displays the feature maps for camera 0, with all 5 FPN levels. Our deepest layers (level 5) has smallest spatial sizes (8×8) and needs to capture abstract high-level features. The shallower levels (level 0 to 3) have larger sizes, with lowest level 0 has size (128×128), capture fine-grained details. Due to the nature of nodules being relatively small, we can expect that the most distinctive features (to separate nodules from the background) will be captured by the middle levels (levels 1 to 3).

Figure 15a displays the channel with highest activation value at each level, we observe that our model mostly focuses on the top-left hand side of the image, even though the ground truth boxes are not present in this region. In fact, this is true for all other images in our experience. Taking the mean across all 256 channels, we see that level 1 and level 4 have the highest generalization. Figure 15b shows that level 1 activates almost the entirely image, while level 4 activates the central region of the image. The other levels remain focused on the extreme left regions.



(a) CAM03 (b) CAM08

Fig. 13: Detailed bounding boxes prediction for camera 03 and camera 08, with classification scores

Next, we investigate if this is the cause of incorrect proposed regions being located mostly on the left of our input image. We study in detail level 1, and Figure 16 displays the top 10 channels with the highest activation value. We see that even though the average of level 1 covers the whole region of image, top 10 channels from level 1 still focus mainly on the left.

6 Conclusion and Improvements

6.1 Conclusion

The MV2D project is a very promising solution for the open problem at Safran Tech for non-destructive testing of aeronautic machine parts. The 3D lifting philosophy is an extension of 2 older but strongly foundational works of Position Encoded Transformers (PETR) and Detection Transformers (DETR). In theory, MV2D has improved the overall algorithms by introducing the dynamic query generation which reduce the number of queries to only the relevant ones in the feature map. And introducing a *sparsing* of the regions of interest and bounding box candidates based on the overalps of 2D bounding box predictions from individual camera views which in turn reduces the complexity of the encoder and decoder transformer.

In our effort to understand the entire implementation, we were able to correlate our studies in the courses and the algorithm to a very large extent.

– 1. We could understand how the features were being generated and transformed at each step of MV2D architecture.

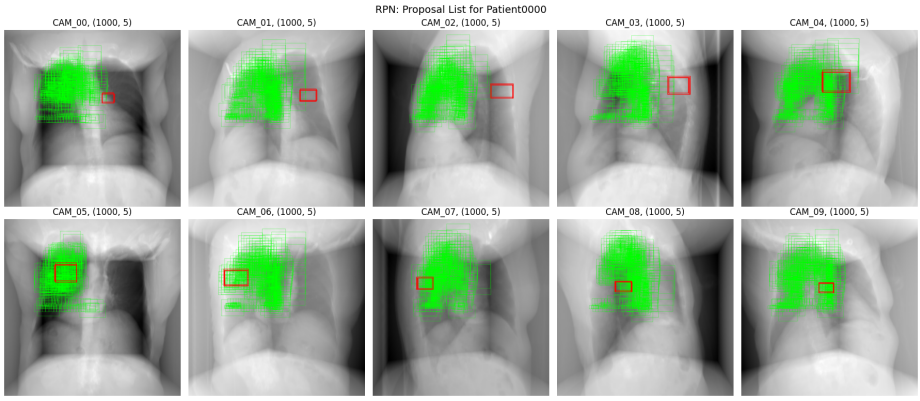


Fig. 14: RPN result for Patient0000 with 10 different camera positions. The predictions are tensor of shape `(N, 5)` with `N` is the number of suggested regions and is controlled by hyper parameter maximum number of regions. We set this to 1000 in our experience.

- 2. We could see the optimization algorithm at work and the calculation of combined losses (classification + regression) in the Hungarian Assigner
- 3. We could see Cosine Annealing algorithm in action where the learning rate adjusts dynamically during training, following a cosine function, to help the model converge more effectively. This method is particularly useful for avoiding local minima and improving generalization.
- 4. We understood the transformer decoder architecture and the mechanisms of self-attention and cross-attention, to get the global features and predicted classification and bounded boxes.

We were introduced to a new framework for implementing vision related neural networks using the MMDetection framework. Though it took us sometime to get comfortable with the framework, it was worth it and we are sure that it will be helpful in our careers.

List of works:

- 1. Modification of the Mask R-CNN implementation to make it run with large number of camera outputs (10 in LIDC in place of 6 in Nuscenes). This was done in order to make the training functional on the SSP Cloud server which has limited GPU memory (16GBs). Finding was provided to Safran, so that they can allocate atleast a 24GB or 32GB tensor card for their purpose, since their use case can have upto 20 cameras.
- 2. Data formatting of the input data to the MV2D model. A large part of the effort was spent to get this done. Now we are clear how the data needs to be correctly passed to the model, in order that it works. Engineers at Safran tech can take our data formatting code and use it as a template for inputting their data.

- 3. We added special data augmentation techniques specially for XRAY data, like Gaussian blurring, Gaussian noise, Salt and Pepper noise, Vignetting and Random Contrast.
- 4. The model and code was modified to take in account the camera parameters (extrinsic and intrinsic) and experimental setup of a scan that resembles of an industrial case. This setup was quite different from the original Nuscenes setup, where the variety of sensors was large and the reference points were in lidar coordinates which needed to be changed entirely into camera coordinates.
- 5. Code available at: git@github.com:katzkid/MV2Dbasedlifting.git

6.2 Improvements

Since the LIDC data was run on the existing MV2D code base that is tuned for automotive data, the results are not exceptional.

- 1. The Mask R-CNN output is using the pretrained weights of the Nuscenes data set that is based on automobile driving data. Hence it is not representative of XRAY data. Hence the output of the RPN is not satisfactory. A poor RPN output will result in poor queries being generated for the predicted bounding boxes.
- 2. The extrinsic parameters of the cameras was not available. Hence an assumption was made on the position of each camera in the 3D scene. There were some anomalies observed using the derived extrinsic parameters. This is a crucial parameter, since it help generate the $\mathbf{p_{ref}}$ coordinates that are input to the transformer.
- 3. For XRAY images where the features are are more hidden and abstract, we propose exploiting deformable convolution networks and deformable attention networks.This will allow for:
  - Adaptive Sampling: Instead of attending to fixed locations, they learn to focus on deformable regions of interest in the input feature map.
  - Dynamic Receptive Fields: The network dynamically adjusts its receptive field based on the input, allowing it to capture more relevant features.
  - Efficiency: By focusing on fewer, more relevant regions, deformable attention reduces computational overhead compared to global attention mechanisms.
- 4. Refine feature selection and Attention mechanisms: Since XRay images can be noisy, we need to ensure that object queries focus on genuine anomaly structures rather than noise artifacts
- 5. Improve data augmentation to include rotation and flipping

7 Acknowledgements

We sincerely thank Julian Betancur and Ihssane Ghalas from Safran France for their insightful discussions, constructive feedback, and invaluable advice

throughout this project, without which this project would not be possible. Their expertise and guidance were instrumental in addressing key challenges. We deeply appreciate their time, effort, and commitment to advancing this project. We are thankful for the LIDC public data set whose research was funded by in part by the National Institutes of Health, National Cancer Institute, Cancer Imaging Program.

References

[1] K. Vino Aishwarya and A. Asuntha. "A Survey on Comparative Study of Lung Nodules Applying Machine Learning and Deep Learning Techniques". In: *Multimedia Tools and Applications* (Aug. 20, 2024). ISSN: 1573-7721. DOI: 10.1007/s11042-024-20009-0. URL: https://doi.org/10.1007/s11042-024-20009-0.

[2] Vivek Gopalakrishnan and Polina Golland. "Fast auto-differentiable digitally reconstructed radiographs for solving inverse problems in intraoperative imaging". In: *Workshop on Clinical Image-Based Procedures*. Springer. 2022, pp. 1–11.

[3] Matthew C. Hancock and Jerry F. Magnan. "Lung Nodule Malignancy Classification Using Only Radiologist-Quantified Image Features as Inputs to Statistical Learning Algorithms: Probing the Lung Image Database Consortium Dataset with Two Statistical Learning Methods". In: *Journal of Medical Imaging* 3.4 (Dec. 8, 2016), p. 044504. ISSN: 2329-4302. DOI: 10.1117/1.JMI.3.4.044504. URL: http://medicalimaging.spiedigitallibrary.org/article.aspx?doi=10.1117/1.JMI.3.4.044504.

[4] Kaiming He et al. *Mask R-CNN*. Jan. 24, 2018. DOI: 10.48550/arXiv.1703.06870. arXiv: 1703.06870 [cs]. URL: http://arxiv.org/abs/1703.06870.

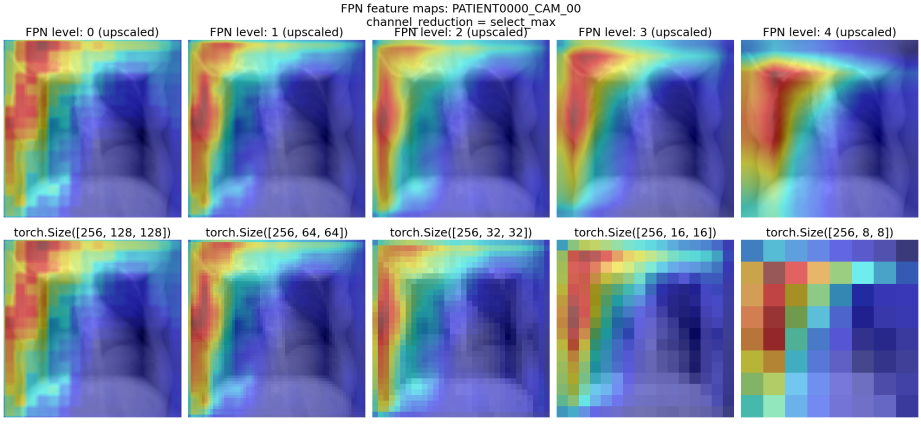
[5] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. Feb. 7, 2018. DOI: 10.48550/arXiv.1708.02002. arXiv: 1708.02002 [cs]. URL: http://arxiv.org/abs/1708.02002.

[6] Yingfei Liu et al. *PETRv2: A Unified Framework for 3D Perception from Multi-Camera Images*. Nov. 14, 2022. arXiv: 2206.01256 [cs]. URL: http://arxiv.org/abs/2206.01256.

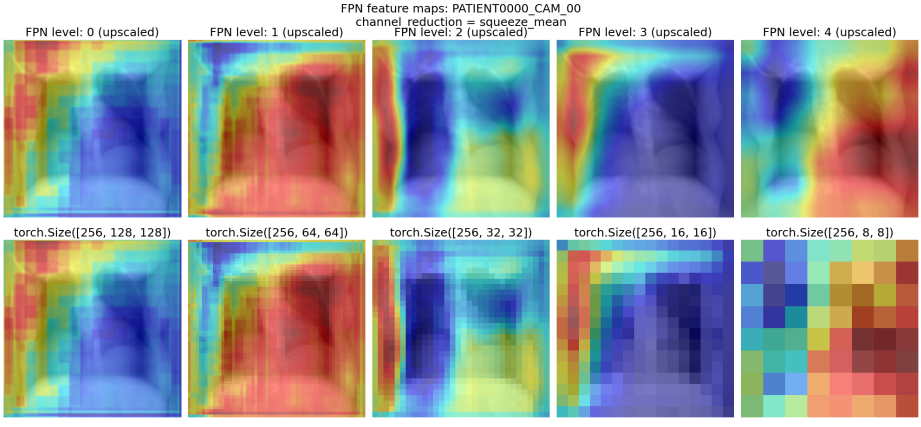
[7] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Jan. 6, 2016. DOI: 10.48550/arXiv.1506.01497. arXiv: 1506.01497 [cs]. URL: http://arxiv.org/abs/1506.01497.

[8] Yue Wang et al. *DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries*. Oct. 13, 2021. DOI: 10.48550/arXiv.2110.06922. arXiv: 2110.06922. URL: http://arxiv.org/abs/2110.06922.

[9] Zitian Wang et al. *Object as Query: Lifting Any 2D Object Detector to 3D Detection*. Nov. 6, 2023. DOI: 10.48550/arXiv.2301.02364. arXiv: 2301.02364. URL: http://arxiv.org/abs/2301.02364.



(a) Channel reduction with "select_max"



(b) Channel reduction with "squeeze_mean"

Fig. 15: Feature map result for camera 00, with 5 levels of FPN. Each level has 256 channels, with decreasing feature size. We use different channel reduction methods to summarize activation values. Figure 15a selects the channel that has highest activation value, while Figure 15b calculate the mean of activation value across all 256 channel. Feature map is plot with original image overlaid, using `COLORMAP_JET` gradient. Red color indicates highest activation value, and blue color represents low value. The image size (1024×1024) is different from the feature size, so we upscale the feature map to match the image size. For each channel reduction method, the bottom row shows the feature map in its original size, while the upper row shows the upscaled version.

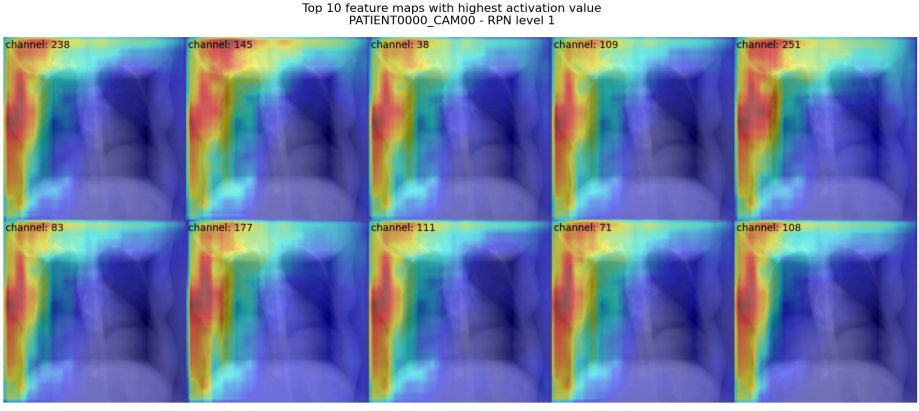


Fig. 16: Top 10 channels with highest activation value from RPN level 1. Feature maps are up-scaled to match original size of image.