Methods of scene graph and text generation for the creation of assembly instructions

Main Track

XY

Abstract. In many production processes of consumer products assembly instructions are often manually generated. This assembly instructions are relevant for planning and for running a production process. In this paper we show an approach of generating a visual graph and text outputs with scene graph generation techniques to describe assembly instructions. The output of a state of the art Mask-RCNN model is processed to a Siamese Network which is combining the output information. Finally there is applied a Graph network structure to improve the model and to integrate context information. We compare the results to a transformer network, which is as well producing a scene graph representation.

Keywords: Assembly Instructions · Scene Graph Generation · Mask-RCNN · Graph Convolutional Network · VisionToTextTransformer · Visual Relationship Detection

1 Introduction

Scene graph generation is used for visual scene understanding, so that scenes in pictures can be described with objects and their relation to each other. Objects are described as nodes and their relation as edges. The last years there were several approaches to solve this task. The classical approach uses a Mask-RCNN or Faster R-CNN backbone to generate input for the actual scene generating network. In this network semantic and visual information is processed in many examples through MLP models to generate as output the relation of objects and to characterize the relation. At the end of the classical approach there is often a message passing structure integrated for example in using Graph Neural Networks to integrate contextual information for improving the model. The GNNs integrate information of neighbouring nodes in a graph.

Like in many other fields transformer architectures are used to solve different kind of machine learning problems. So as well in the context of scene graph generation transformer technology is used to solve this task. In this approach instead of a Mask RCNN structure there is a transformer structure like DETR used. The output is then processed again in a further transformer architecture to make predictions about objects and their relations to each other.

This paper shows a classical approach to solve the practical problem of generating assembly instructions with scene graph generation techniques. One objective is to show a relatively easy solution with out of the box algorithms, because

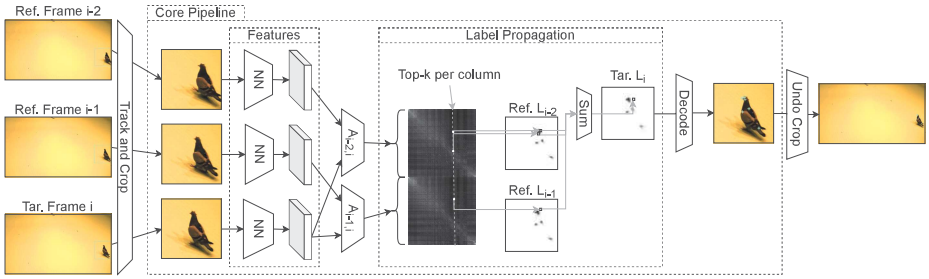

Fig. 1. *Overview of the joint tracking and label propagation pipeline, showing pigeon keypoint tracking as an example.* The core pipeline of our method uses deep neural network features to compute affinity matrices between a target frame and one or multiple reference frames. Labels for the target frame are computed by propagating the reference labels based on the combined affinity matrices. Finally, the predicted labels are decoded to coordinate values. The proposed extended pipeline shown outside the dashed box is necessary for tracking small objects. Here, an object tracker is used to locate the object of interest in each frame. This allows us to crop a region around the object and perform keypoint propagation at a much higher resolution, after which we undo the coordinate transform of the cropping for the decoded keypoints.

research paper codes are often not so easy to handle in the implementation. So as backbone network there is a Mask-RCNN architecture of MMdetection used to generate bounding boxes, segmentation masks and class prediction of objects. Then there are state of the art MLP CNNs used to model the Siamese network and finally with the library of Pytorch Geometric there is a applied a GNN architecture at the end. To compare our results to the transformer approach the RelTR algorithm is applied to the same data set as the classical architecture. Another objective is to show how to solve the problem with less semantic information needed. Semantic information can be an important input information for scene graph generation, because relations and the characterization of them can be derived out of semantic information. Our approach tries to put a focus stronger on geometric information of the assembly parts to make semantic information less necessary. A challenge is the application of the algorithm having only a small data set, which makes generalization difficult. We use the technique of XY to make to algorithm working on a relatively small training data.

Main distributions of this paper are:

- To show how to implement state of the art algorithms of the classical approach
- To build a scene graph model relying less on semantic data
- To build a scene graph model using a Siamese network
- To use XY to handle the small dataset
- To show a specific graph structure for the practical problem

2 Related Work

3 Approach

The objective of the model is to be able to sufficient generalize to predict correctly an assembly instruction of unknown parts. The model is trained on toy vehicles (train, boat, plane) and should be able to predict the assembly routines of a different vehicle (car), which has similar, but unlearned parts. Building the model there is the MMdetection infrastructure needed for the Mask R-CNN network, which is trained on a dataset containig XY labeled images of the three different toy vehicles. Bounding boxes, shapes and classes of the parts are predicted. The information, if parts have a relation and if so which kind of relation is documented in graph data for training the model. The description of the graph is a crucial point. The three vehicles are combined to one graph, where different ways of modeling the graph are possible like Fig XY shows. The graph can be modeled as a vehicle based graph (Fig XY.1) or as a class based graph (Fig XY.2). We use a combination of both (Fig XY.3) to integrate vehicle specific information but as well the class information. Using an only class based graph would mean that semantic information goes more into the graph setting.

3.1 MASK R-CNN Backbone

As backbone there is used a MASK R-CNN architecture which predicts O objects in I image giving as output B bounding boxes, S shapes and C classes for each O in I. Additionally, the graph structure G is given for training. G is not represented in an adjacency matrix but as an edge list Nx2, where N is the number of nodes that have an edge connection to another node. There are $O(n^2)$ possible connections for each I, but N represents only the existing edges.

3.2 Siamese Network

The Siamese network is modelled in three stages in an EndToEnd structure: first the backbone output B and S is processed to the Siamese network in creating a feature vector x for B and y for S based on Resnet50 with an additional fully connected layer to allow the network to modify the extracted features. C represents the sematic information of the MASK R-CNN output and is converted to W, a word vector. X and y are then processed to individual paths, from which each is constructed in a Siamese manner. The Siamese network expects two inputs, which are processed in the same CNN with the same weights. Describing only the X path this means, that the feature vectors for two connected nodes are used by the network according the graph data G. So there is a X1 and X2 processed to the network. The Siamese architecture should allow the network to learn features comparing two images. Finally, the outputs of X1 and X2 are concatenated to one single vector X3, which is the input for a 2 layer MLP, which task is to predict the actual class of the connection between the nodes.

The concatenation is a direct concatenation function of the vectors from the union box of each object pair. In the example there are 4 classes used ("join", "plug", "screw", "no connection") to describe how the assembly parts can be put together. The 2 layer MLP at the end of the Siamese network combines the vectors X3, Y3 and W to predict the final class of the relation.

3.3 GNN Network

At the end a GNN structure is implemented to enable the learning of contextual information.

3.4 Loss function

4 Applications and Case Studies

The generic way of formulating label propagation enables the propagation of various kinds of information through a video with a single framework. Examples include masks, keypoints and textures. Here, we take a look at mask and keypoint propagation. For each of these tasks, we have certain information in our problem domain $\mathbb{P}$, which we have to map to our label propagation framework. Thus, we have to define an encoding function $E : \mathbb{P} \to [0,1]^{h\times w\times l}$ and a decoder $D : [0,1]^{h\times w\times l} \to \mathbb{P}$.

4.1 Pose Tracking

Encoding keypoints requires some effort to transform the n points $K_{i,j} \in \mathbb{R}^2$, $j = 1,\ldots,n$ for the ith frame into 2D functions. Commonly [15,6,12,30], each keypoint is encoded to a separate layer in the label function, thus $l = n$. The label encoding for the jth keypoint is then a 2D Gaussian with mean $K_{i,j} \odot [w/W, h/H]^T$, where $\odot$ denotes point-wise multiplication. Its standard deviation σ is a hyperparameter and should be chosen according to the feature size. Here, we found we can improve upon some existing implementations. As the label function has the resolution of the feature map, it is often only $\frac{1}{4}$ or even $\frac{1}{8}$ of the frame resolution. Therefore, some inaccuracies in the placement of the Gaussian function can occur if μ is not computed with sub-pixel precision. For example, UVC [15] scales the features to the feature size but then casts the floating point numbers to integers, thus discarding the sub-pixel precision. In contrast, we implement our encoding function in such a way that the peak of the Gaussian does not necessarily lie on an integral pixel position.

Decoding label functions to keypoints requires the approximate peak location for each label function. The intuitive solution of using the location of the maximum is unreliable, as the quality of the result hinges on a single point. Hence, some methods compute the mean or center of gravity of the top k positions to improve the robustness of the solution to outliers. Our method follows the latter approach of using the center of gravity not only for its improved robustness, but