CONVOLUTIONAL CRF FOR INSTANCE SEGMENTATION AND SENSOR FUSION - SMART PARKING SYSTEM

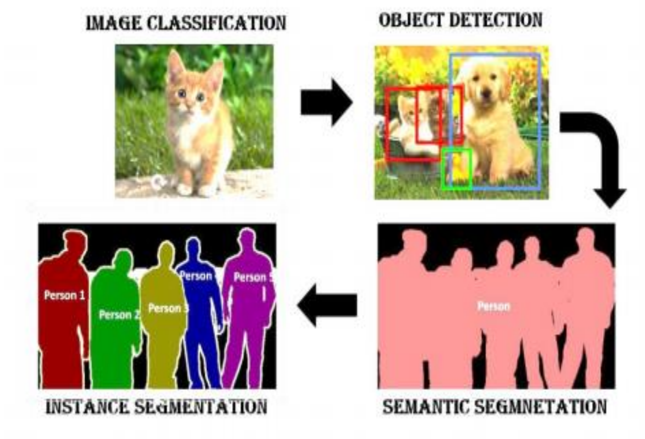
JERRIN BRIGHT
SUPERVISOR: Wei-Tyng Hong
Vellore Institute of Technology, Chennai, India

Abstract: In this paper, a robust system for autonomous parking system is proposed using convolution and sensor fusion algorithms. The glimpse of the paper is we will use instance segmentation along with Convolutional Conditional Random Fields via a RGBD camera (i.e., Kinect) and fuse the sensor using a Lidar. Thus, even when the Camera fails, we will use the Lidars to estimate and navigate accordingly thereby preventing the system from total failure. This sensor fusion has been done using Extended Kalman Filter in this paper. Adding CRF to the segmentation process increased the efficiency of the system by 14.75% to the initial setup and Lidar fusion also increased the performance abruptly thereby resulting with an average of 94.8% robustness in the system.
Keywords: CRF, Segmentation, Convolution, Image processing, EKF, Kinect, Fusion.

1. INTRODUCTION

In an averagely crowded country, parking vehicles is one of the toughest and an activity to be considered requiring more dexterous driving. It is observed an average of 25% of road accidents are caused as a result of bad parking practices. A robust autonomous parking system is thus proposed in this paper that will enhance the driver experience making him comfortable and completely safe and secure. This system proposed works precisely despite weather conditions, heavy lightings and vision deceptions like shadow and glares. We will use RGBD camera as our prime sensor to detect the obstacle using segmentation techniques and motion using techniques like dead reckoning which will be discussed in the Xth section. 3D Lidar will be used as a fail safe in case the RGBD Camera fails or gives false positives. Thus, Lidars will be induced with the measurements and estimates of the RGBD Camera using the sensor fusion technique with the help of Extended Kalman Filters. The EKF consists essentially of 3 steps- Predict, Measurement and Update. We will look in depth of the EKF algorithm in the Xth section. Thus, before going into the core research outputs, in section X to X, we will discuss the basics of AI including what and how segmentation (custom) can be done and also explaining what Conditional Random Fields are and why we chose Convolutional CRF over other CRF types and end the X section with Sensor fusion, Extended Kalman filters and also why EKF in particular for sensor fusion amidst other new algorithms.

2. INSTANCE SEGMENTATION



In **Image classification**, it takes an image as an input and outputs the classification label of that image with some metric (probability, loss, accuracy, etc.). For Example: An image of a cat can be classified as a class label "cat" or an image of Dog can be classified as a class label "dog" with some probability.

In **object detection**, the bounding boxes are always rectangular. So, it does not help with determining the shape of objects if the object contains the curvature part. Object detection cannot accurately estimate some measurements such as the area of an object, perimeter of an object from image.

Instance Segmentation: Identifying the boundaries of the object and label their pixel with different colors. Instance segmentation can detect objects within the input image, isolate them from the background, and also it takes a step further and can detect each individual object within a cluster of similar objects, drawing the boundaries for each of them. Thus, it can not only differentiate a group of individual species but the number of individuals resulting in the species. That is, in the example image mentioned below, in semantic segmentation, we were able to say there are many goats but can't differentiate each and every goat individually. But in instance segmentation, we are able to say there are 3 different goats standing together. This is simply what instance segmentation does. Instance segmentation is the latest deep learning technique adapted after image recognition, object detection, and semantic segmentation.

Semantic Segmentation: Labeling each pixel in the image (including background) with different colors based on their category class or class label. The above-discussed techniques can be utilized in many fields such as:

- **Driver-less Cars:** Object Recognition is used for detecting road signs, other vehicles, etc.
- **Medical Image Processing:** Object Recognition and Image Processing techniques can help detect disease more accurately. For Example, Google AI for breast cancer detection detects more accurately than doctors.
- **Surveillance and Security:** such as Face Recognition, Object Tracking, Action Recognition, etc.

We will thus be using Semantic Segmentation for our efficient and robust autonomous parking system. The new deep learning technique created after instance segmentation is Panoptic segmentation which is a combination of both semantic and instance segmentation. This section of the paper is split into 5 steps for ease of the readers. They are detailed in the coming:

I. Installations

Essential installations for the working of the project Python libraries/ packages. The required libraries are imported in this section. Some of the important libraries in this project are NumPy, Shutil, TensorFlow.

II. Dataset

Creating datasets and stacking in proper directories are essential for the segmentation. Structure of the Zip file for the dataset to be custom trained:

- Train Directory – Will consist of the JPG images and Annotations of each JPG images obtained to train.
- Validation Directory – Will consist of the JPG images and Annotations of each JPG images obtained to validate.

These annotations for both training and validation images can be built using various software like LabelIMG, VGG Image Annotator, etc. Thus, the structure of the dataset has to be clearly defined and drafted first.

III. Training

The dataset extracted in the previous section is trained in this section.

IV. Inference

The inference part of the code is further split into the following steps:

- Initialization of the root directory of the project.
- Import Mask R-CNN to find a local version of the library.
- The directory to save logs and the trained model.
- Run detection on one image at a time instead of pushing all images at the same time to increase precision.
- Device to load the neural network on. Useful if you're training a model on the same machine, in which case use CPU and leave the GPU for training.
- Inspect the model in training or inference modes
  values: 'inference' or 'training'
- TODO: code for 'training' test mode not ready yet
- Return a Matplotlib Axes array for visualizations in the notebook. Provide a central point to control graph sizes. Adjust the size attribute to control how big to render images
- Load the validation dataset from the directory structured to train the model.
- Must call before using the dataset, prepares the model using **prepare ().**
- Create a model in inference mode with MaskRCNN. RCNN stands for Region-based Convolutional Neural Network.
- Loading weights using **load_weights()**
- Weights were constantly changing the visualization, so reloaded the visualization alone instead of the notebook.

V. Testing

Testing of the model trained earlier is done in this section. Run object detection with **detect ()** and store in the results variable.

In this section, we discussed image segmentation with the help of examples, and also custom segmentation is explained alongside. The image segregation for training and validation alongside the annotations created by software like LabelIMG and VGG Image Annotations is the first step. Followed by training and testing of the model. Some applications of image segmentation are automatic traffic control, biometrics, an inspection of electronic components and chips, etc. Thus, they are very efficient and rapidly developing technique.

CONDITIONAL RANDOM FIELDS

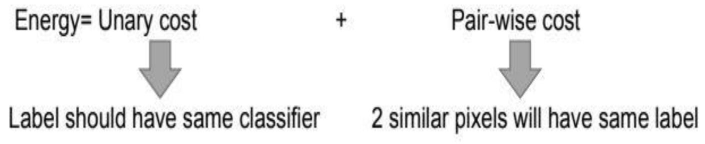
Probabilistic framework for labelling and segmenting structured data is the basic idea of Conditional Random Fields. It tries to model the relationship between labels.
For example,

- Nearby pixels more likely to have same label
- Pixels with similar color tend to have same label

- Pixels above aero-plane are more likely to be sky/cloud than a person

In other words, CRF simply refines result by iterations. It is iterated with Convolutional Neural Networks to further enhance the segmenting task. CRFs on their own are notoriously slow and hard to optimize. It is two times slower than CNNs. So therefore, training in CRFs makes it dreadful for research activities. Therefore, we will be using combinations of Neural network with CRFs to make it ideal for our conventions.
Pixel level labelling tasks, such as semantic segmentations, play a central role in image understanding. Recent approaches have attempted to harness the capabilities of deep learning techniques for image recognition to tackle pixel-level labelling tasks. One central issue in this methodology is the limited capacity of deep learning techniques to delineate visual objects. To solve this problem, we introduce a new form of convolutional neural network that combines the strengths of Convolutional Neural Networks (CNNs) and Conditional Random Fields (CRFs)-based probabilistic graphical modelling.

CNNs yield a coarse prediction on pixel-labeled tasks. CRFs improve the result by accounting for the contextual information in the image. Therefore, learning the whole pipeline end-to-end significantly improves the results. Pixel-wise labelling can be done as an energy minimization problem


Energy= Unary cost
+
Pair-wise cost
Label should have same classifier
2 similar pixels will have same label

If unaries are poor, inference with pairwise potentials can’t help. Detection potentials are used to overcome cases where unaries are poor. Curtailing Energy upsurges Accuracy of the labelling tasks. Energy can be condensed using the Bilateral filters- Intensity Filter & Spatial Filter.

STEPS INVOLVED

Feature Functions

The feature function represents a set of features. Features should be binary valued. The feature function should return an array of features, i.e., binary values. Though feature values should be binary, using a real numbered data type avoids unnecessary conversions, example returning {0.0, 1.0} instead of {0, 1} or {false, true}. The following parameters are passed to the feature function:
- yp label of observation x[t-1]
- yt label of observation x[t]
- x array of all observations
- t position of current observation

The observation at t=1 has no predecessor, therefore no yp. Most CRF papers and implementations use a special START label as value of yp at t=1. However, since the labels can be of any type, a special START label would either require a type union or a user defined start label passed to the Sequence constructor. Both solutions aren't really nice. Therefore, we use a different approach: two methods of the feature function:
- method (yt, x, t) called for t=1

- method (yp, yt, x, t) called for t>1

Parameter Estimation

After you defined your feature function, you want to use labeled training data for parameter estimation. The training data consists of one or more sequences of observations with corresponding sequences of desired labels. Parameter estimation is done by maximizing the loglikelihood function. The CRF package doesn't provide a function optimization algorithm.

Sequence Labelling

The process of labelling the sequences are as follows:

1. Import the CRF Library
2. Include the files required to be evaluated
3. Assign the estimated weight parameters using the sequences
4. Load the data [CSV/VOC file]
5. Remove sequences used for the weight estimations
6. Create sequences
7. Call the function ‘label’ using the sequences to get the labels
8. Compare the True and Predicted label.

CONVOLUTIONAL CONDITIONAL RANDOM FIELDS

Let’s assume two pixels ‘I’ and ‘J’, and the distance between their midpoints as D. Also, we shall define a term k which is called Filter size. Filter size refers to the size of the filter, 5 means a 5*5 kernel. Bigger the kernel less will be the noise, but the edges will blur out and also bigger images mean more undesirable artifacts as well. Filtering is typically done during convolution.

Now we can say that, ‘I’ and ‘J’ are conditionally independent if D (I, J)>K. This is a very strong assumption. Therefore, we can say that the convolutional CRF is a supplement of FullCRF with a conditional independence assumption. This assumption, thus reduces the complexity of the CRF pairwise potential abruptly. These assumptions thereby make convolutional CRF very efficient.

TRAINING VIA CONVCRF

Method 1 [Two-Stage Training Strategy]

We will be using the pre-trained Pascal VOC DATASET 2012 rather than the large COCO dataset. We have to label the desired part of all the images as the Ground-Truth using software like LabelIMG or LabelBox and then save in VOC format [XML File]. Later, send 10% of data to fine-tune the internal CRF parameters and remaining data will be sent to train the unary CNN. This part of training is called the Two-Stage Training Strategy. So, in the First stage, the unary CNN model is trained and its parameters are fixed. Then, in the Second stage, internal CRF parameters are augmented with respect to the unary CNN predictions.

Method 2 [End-to-End Learning]

We train the network using a training protocol. Then, we optimize both CRF and CNN jointly, and concurrently introduce a unary loss to counterpoise the vanishing gradient. At the end of each iterations, the internal CRF parameters are tweaked keeping the CNN parameters fixed.

TWO-STAGE VS END-TO-END LEARNING

Two-Stage Training Strategy are better than End-to-End Learning in the following:

- Very flexible.
- Training stages don't interface at all.
- Keeps the system interpretable.
- Efficiently tackles the vanishing gradient problem.
- Overall fast, robust and reliable training

ConvCRF VS FullCRF

In FullCRF, Inference is done via mean field algorithm. Everything is simple in the algorithm but the message passing fragment. The computation is quadratic in the number of pixels and thus unfeasible. Thus, we will use ConvCRF, which makes highly effectual GPU computations and complete feature learning possible. Thus, we will be using Convolutional CRF here.

CONVOLUTIONAL CRF WITH INSTANCE SEGMENTATION

SENSOR FUSION

Combining two or more sensor data in a way that generates a better understanding of the system is called as sensor fusion. Thus, sensor fusion does the sensing and perceiving part for the autonomous systems. That is, it will take multiple sensor readings, combining it to get a better model, so that the system can plan and act efficiently. The major advantages of sensor fusion are:

- Increase in the quality of data by removes constant noises
- It increases reliability (even if one fails, we will get value but not as accurate as before obviously).
- Unmeasurable states can be measured (one camera can't measure the distance between itself and an object, but 2 cameras together can).
- Increasing range (like having many ultrasonic around a car for close distance detection)

Kalman filters are one of the most efficient sensor fusion algorithms used. The advantage of Kalman filters is that the mathematical model of the system is already built in the filter, so fusing of sensors can be done and map quality can be enhanced. Kalman filters will operate in 2 steps:

- Predict
- Measurement
- Update

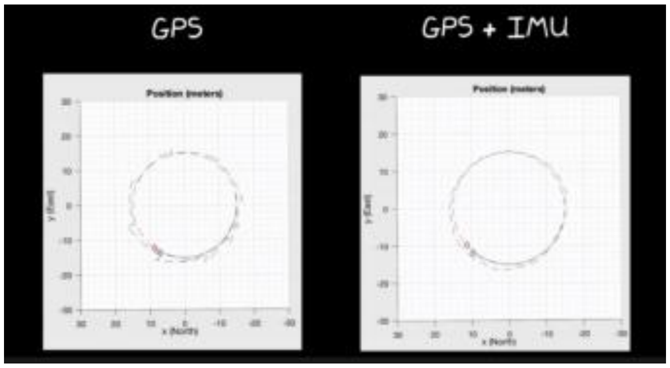
Predict:

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1}$$

Update:

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}$$

Prediction is based on the previous knowledge of the robot and kinematic equations. Measurement is getting the readings form the sensors. Update is updating the value based on the predicted and measured readings. This the ‘predict’ equation uses the position of the previous time-step k-1 together with the motion model to predict the current state x_k. This will be updated in the update equation shown above using the Bayes theorem thus combining the measured and predicted states. Now we will have the posterior distribution. Similarly, we can now consider this posterior as the previous posterior and find the next posterior. Thus, this method is iterated for further measurements. It will first measure using a sensor and then uses a mathematical model to compare it with the measured values from the sensors. Thus, it estimates the state from the confidence attained from the predicted and measured values. This way, the sensor will run asynchronized measurements. IMU is a fusion of accelerometer, magnetometer and gyroscope. IMU is often fused with GPS in drones.



In the image seen above, when we use only GPS, the prediction is drastically running away from the ground truth due to the slow update rate of the GPS. Whereas when IMU is added the prediction will be corrected with the fast update rate of the IMU sensor against that of the GPS. Thus, once the sensor readings converge, we will get a better prediction and thus a better state estimation.

LIDAR CAMERA FUSION

LIDAR has an advantage of sharp accuracy in ranging and thus results in good mapping. Coming to Visual sensors, they provide a very dense map and comparatively cheap and consumes less power. But it also has its cons, including poor depth estimation, small range for stereo-visions, difficult to use RGBD cameras outside due to the lightings. Thus, considering both its pros and cons, a fusion of these two sensors would prove to be very efficient in SLAM.

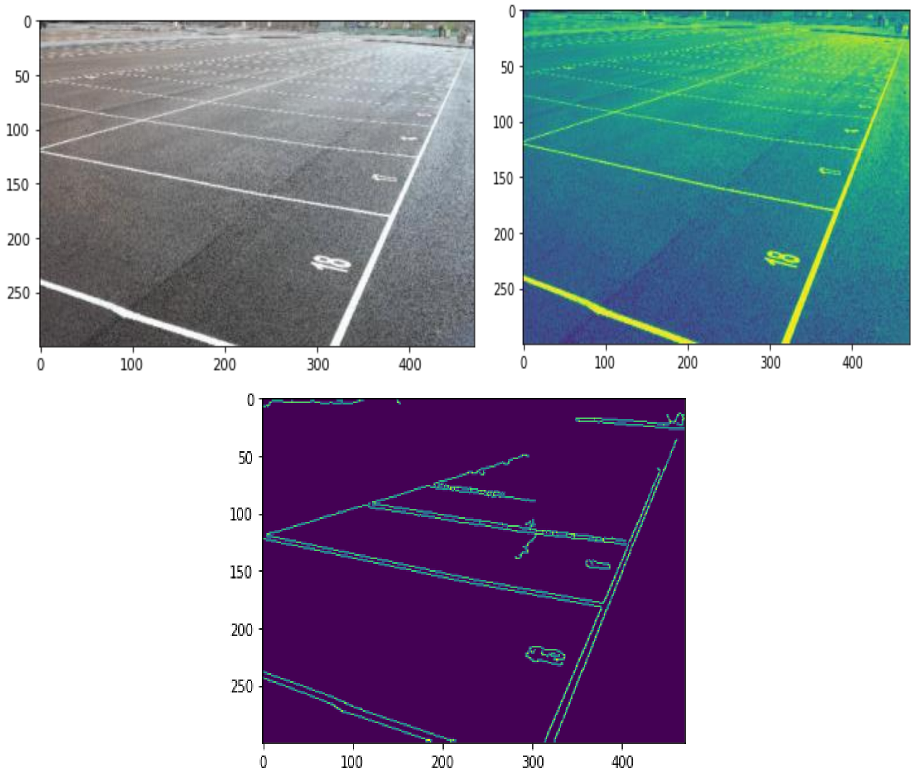
Two important considerations when it comes to fusion of sensors are the calibration of the sensors to be fused and the approaches/ architecture of fusing the sensors. One of the approaches of the fusion is:

- Camera resolution is much higher than a lidars but less depth information. Thus, a gaussian process regression was performed to interpolate the missing depth values using the lidar. Thus, Lidars where used to initiate the features declared in the images.
- When the visual tracking is unsuccessful, the Lidar pose can be used to localize the point clouds data of the RGBD camera to build the 3D map.

A few sensor fusion methods are:
- Kalman filtering
- Optimization based methods

Kalman filtering is again divided into 2 types:
- Loosely coupled (fuse processed information's from individual sensors)
- Tightly coupled (fuse raw measurements directly)

Optimization-based methods are mostly tight-coupled. The advantages of using Kalman filter for sensor fusion are:
- Sensor data fusion can be easily implemented using Kalman filters as it already has mathematical models inbuilt.
- Increases the on-line estimation by reducing the noise and bias.



We compared the performance of lane detection with Hough lines and gaussian blurring to that of convolutional neural networking detection. For good performance of the canny detectors, we will have to mask the desired region of interest (ROI) which varies from one vehicle to another. Moreover, those filters and edge detectors tend to perform relatively poorly in high steep areas. Also, with real-life observations using monocular cameras shadows, glares and rapid movement of the vehicle tend to result in poor frame clarity resulting in inaccurate detection. Thus, we implemented neural networks for lane detection tending to solve the above-mentioned issues and thereby increasing the robustness and accuracy of the system.

So, comparing the performance of lane detection using Hough lines, Gaussian filters and that of lane detection using convolutional neural networks, detection via CNN proved to be more robust and consistent than compared to other lane detection techniques.