POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Towards Real-Time Inference: A Fusion of Pose Estimation and Object Tracking

Laurea Magistrale in Computer Science and Engineering

Author: Greta Corti

Advisor: Prof. Matteo Matteucci

Co-advisors: Francesco Lattari Simone Mentasti Riccardo Santambrogio

Academic year: 2022-2023

1. Introduction

In the dynamic realm of computer vision, object pose estimation models have demonstrated exceptional accuracy in capturing and predicting the spatial orientations of various objects. Despite their exceptional capabilities, integrating these models seamlessly into real-time scenarios, especially on resource-constrained systems, such as wearable devices, presents an impressive challenge due to their demanding computational nature. The issue lies in the complex computational processes that are integral to these sophisticated models. Recognizing detailed spatial information often comes with a high parameter count, resulting in computationally intensive processes that become a bottleneck, introducing delay. This issue is crucial especially in resource-constrained environments and in real-time contexts that require rapid decision-making or interaction. The importance of investigating how to overcome this challenge is particularly pronounced in wearable applications, such as augmented reality glasses, where rapid and precise spatial information is paramount.

Starting from this problem, the work proposed addresses this challenge through a multi-faceted approach, introducing a novel framework that unifies pose estimation and object tracking. The primary objective is to significantly enhance inference speed, making real-time applications on wearable devices not just a possibility but a reality.

The key contributions of this work encompass a comprehensive quantitative analysis of diverse embedded platforms, the introduction of an innovative framework, and the development of a flexible, lightweight deep-learning-based network dedicated to object tracking. The strategic integration of pose estimation and object tracking is exemplified through (SwiftTrack), an advanced model designed to ensure high-speed and precise pose estimation tailored for wearable devices.

As we delve into the executive summary, the following pages will provide a detailed exploration of the challenges addressed, methodologies employed, and the compelling outcomes achieved through the amalgamation of pose estimation and object tracking. This pioneering approach not only pushes the boundaries of computational efficiency but also opens new possibilities for the seamless integration of sophisticated computer vision models into real-world, resource constrained environments.

2. Proposed Method

The method proposed involves the integration of a pose estimation algorithm and an object tracking algorithm, leveraging the strengths of both neural networks. While pose estimation algorithms excel in accuracy, they often lag in speed during inference. On the other hand, object tracking models are designed for real-time applications with high inference speed but may compromise accuracy over time. Our innovative model, represented in Figure 1, strategically utilizes the pose estimation neural network at a lower frequency for accurate initial pose estimation. Simultaneously, at a higher frequency, it employs the object tracking model, called *SwiftTrack*, to track the object's pose in subsequent frames. This dynamic interplay ensures that the object tracking network continuously benefits from the pose estimation network's accurate predictions, avoiding delays and providing real-time performance.

GDR-Net [4] is chosen as the pose estimation algorithm, with a new developed object tracking model, SwiftTrack, discussed in detail later. The proposed framework's flexibility allows for the incorporation of various pose estimation models, provided they output both the object's 2D bounding box and 3D pose.

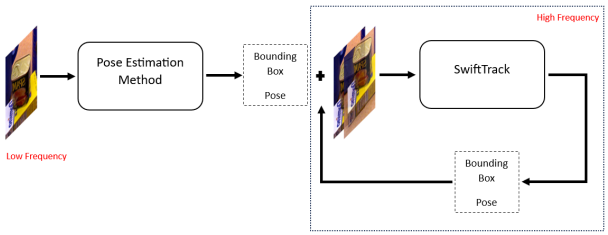

Figure 1: Structure of the proposed framework.

2.1. SwiftTrack

SwiftTrack is a novel object tracking model designed to estimate frame poses at an impressive speed of approximately 250 frames per second. SwiftTrack stands out for its innovative approach, overcoming size-complexity trade-offs and ensuring both speed and precision in pose estimation. Given two RGB images, one at time $t-1$ and another at time t, the network directly outputs the relative transformation needed to obtain the new 3D pose and object bounding box from the initial pose and bounding box of the object. The network uses two cropped portions for each frame: the first crop focuses on the object's previous location, the second on its current location, both padded for contextual information, and then resized to 64x64 pixels. The initial crop helps the network recognize and position the object, while the second aids in analyzing the broader search region. The two cropped and scaled RGB images are concatenated into a six-channel tensor input, as shown in Figure 2. The backbone network, based on FlowNet Simple architecture [1], employs convolutional layers to extract features related to spatial and temporal variations. The processed features are used to predict the optical flow vectors and subsequently undergo further processing to predict object transformation parameters, including center, bounding box dimensions, translation vector, and rotation matrix of the object in the new image.

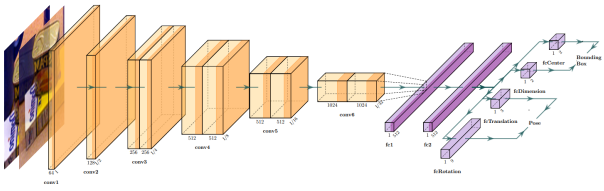

Figure 2: SwiftTrack architecture.

To streamline the complexity and optimize the performance of the neural network, a disentangled representation is employed, where individual components of relative transformations are isolated to simplify the learning process. The proposed method, inspired by Li et al.[2]'s work, involves decoupling the estimation of relative rotation and translation. This is achieved by adjusting the center of rotation and using axes parallel to the camera frame. Given the absence of depth information, it is quite difficult to obtain the relative 3D translation directly from 2D images. Instead of directly predicting the 3D vector, the network is trained to regress changes in the object's 2D positioning v_x and v_y while accounting for scale modifications v_z, using these three formulas:

$$\begin{aligned} v_x &= f_x\left(\frac{x_f}{z_f} - \frac{x_i}{z_i}\right), \\ v_y &= f_y\left(\frac{y_f}{z_f} - \frac{y_i}{z_i}\right), \\ v_z &= log\left(\frac{z_i}{z_f}\right), \end{aligned} \tag{1}$$

where f_x and f_y are the focal lengths of the camera.
Rotation estimation is approached using the gnomonic projection, with a specific implementation of the Cayley transform for rotation matrices. This methodology focuses on small rotations and provides advantages in terms of numerical stability.
In conclusion, the proposed representation of relative transformation offers several benefits, including independence of translation estimation from rotation, utilization of basic translation and scaling variables, and the elimination of the need for prior knowledge about the object.
The network outputs four elements for object localization: the center and dimensions of the bounding box, the translation vector, and the rotation matrix at time t. The total network loss is defined as a combination of losses for these elements, considering L1 distance for measurements. To address the challenge of balancing losses with different scales, the losses are merged into two components:

$$L_{network} = \alpha L_{bbox} + \beta L_{pose}, \tag{2}$$

where L_{bbox} is the bounding box loss and L_{pose} is the pose loss. The bounding box loss is calculated using mean absolute error between predicted and target bounding boxes. For the unified pose regression loss, a geometric reprojection error is employed, measuring the mean distance between 2D projections of 3D points based on estimated and ground truth poses. This approach ensures a comprehensive evaluation of object localization and orientation.

3. Experimental Setup and Evaluation Methods

In this section, we will explain the system setup used to conduct the experiments and investigate the effective ability of our system in predicting 2D bounding boxes and 3D poses. Before that, the dataset creation process is depicted.

3.1. Dataset

The research employs the YCB-Video dataset for training and evaluating a model designed for object tracking. This dataset consists of 92 RGB-D videos capturing 21 objects in real-world and synthetic scenarios. For training, 80 videos are used, with an additional 12 for testing and 15 for validation.
Initially, a large training dataset is created with over 400,000 samples, but due to computational constraints, it is later reduced to 28,954 samples by randomly selecting 100 frames from each video. The dataset includes synthetic images, challenging scenarios, and diverse environmental conditions.
The study identifies a problem in the neural network's behavior during training, where it struggles to recognize object movements between consecutive frames due to minimal pixel shift. To address this, a custom dataset is created, focusing on pairs of images separated by 10 frames, improving the model's ability to track object movements.
For validation, four datasets are established, varying in frame distance (1 frame and 10 frames) and the application of data augmentation. These datasets assess the model's performance in different scenarios, including ideal conditions and situations with augmented input. They are named "*Static Single Frame*" (*SSF*), "*Dynamic Single Frame*" (*DSF*), "*Static 10 Frames*" (*S-10F*), and "*Dynamic 10 Frames*" (*D-10F*).

3.2. Training Setup

The model is trained on the YCB-Video dataset using a network initialized with pre-trained FlowNet-Simple weights. The architecture consists of ten convolutional layers with a stride of 2 in six layers, LeakyReLU non-linearity after each layer, and decreasing filter sizes. The network utilizes a 7x7 kernel for the first layer and 5x5 for the next two, followed by 3x3 from the fourth layer. Feature maps double in the deeper layers, and two fully connected layers with output dimension 512 are followed by four fully connected layers with dimensions 2, 2, 3, and 9, outputting center, dimensions of bounding box, translation vector, and rotation matrix.
The project is implemented in PyTorch with additional libraries. Training batches consist of 512 image pairs, and validation batches have 64 pairs. Training is performed on an NVIDIA GeForce GTX 1080 with 4 CPUs using the Adam optimizer with a learning rate of 10^{-4}. The bounding box increase is fixed at 50%, and input model images have dimensions of 64x64

pixels. Early stopping with a patience of 20 epochs is employed to prevent overfitting, monitoring the validation dataset's loss for image pairs 10 frames apart with no data augmentation. Training halts if the loss fails to decrease further.

4. Results

Different tests are conducted to evaluate the performance and the efficiency of the object tracking model SwiftTrack. One of the most important aspects related to this research is performance. For this reason first of all we have conducted an investigation of how pose estimation algorithms perform on different embedded platforms. Then, we provide an account of the experiments conducted to arrive at the final model. Lastly, we unveil the outcomes derived from out proposed framework to address our problem.

4.1. Preliminary Analysis

The analysis on different embedded devices is conducted utilizing two different pose estimation algorithms: GDR-Net, a pose estimation network built for processing RGB images, and DenseFusion [3], a pose estimation algorithm that utilizes both RGB images and depth information. We examine these two models to evaluate whether the input given to the network has a significant impact on timing and accuracy. The goals of this investigation are:

- Assessing whether there is a significant decrease in performance when using a standard RGB camera compared to an RGB-D camera in a practical application.
- Establishing whether using lower-performing hardware results in a significant performance decrease in inference times on embedded systems. Such hardware, though, is ideal for real-world applications with limited available space.

Both neural networks perform accuracy and inference calculations on the LineMOD dataset. The DenseFusion model achieves the highest average accuracy of 95.23%, achieving an astonishing level of accuracy exceeding 98% in few objects, while GDR-Net has an accuracy of 93.69%. In Table 1, the performance obtained by the two pose estimation networks on different embedeed platform is shown. Using the time library in Python, we can compute the duration required by the models to determine the position for each image in the LineMOD test dataset. It is evident that GDR-Net achieves faster inference times than DenseFusion. Also, as expected, the best timings are obtained using a GPU, and inference times improve whenever a Jetson device with better hardware specifications is used. Finally, it is evident that the inference timings are very similar when using a CPU and a Jetson Nano, so if we perform an inference analysis on one of the two devices, we can get an idea of what the timings will be on the other hardware.

Embedded Platform	**GDR-Net Inference (*fps*)**	**DenseFusion Inference (*fps*)**
4 CPU	9	5
GPU	70	52
Jetson Nano	7	5
Jetson TX2	-	6
Jetson AGX Xavier	25	21

Table 1: Comparison among the inference time obtained in different embedded platforms using two different pose estimation algorithms.

4.2. SwiftTrack Experiments

In this section, we thoroughly analyze the results of our experiments to gain a comprehensive understanding of the factors that affect our model's performance.

4.2.1 Comparing Loss Changes

We conduct a comparative analysis of the use of four losses versus two losses to determine the subtle effects of different loss functions on our model's overall performance. As shown in Figure 2, the model outputs four components, thus the most straightforward method for creating the loss function of the model is to add up the L1 losses of each component (*Multifaceted Loss*). The varying nature of the components causes the singular loss contributions to affect the total loss in different ways, thus to ensure equal training of all components, the network loss must be balanced. Determine the optimal coefficients for the Multifaceted Loss is very difficult, therefore to simplify our objective, we seek a loss representation for all network outputs with fewer components. We consolidate the center and size of the bounding box into a single vector that signifies the top-left and bottom-right corners of the bounding box, while we opt to compute the 3D dimensions of the object's bounding box in the

Experiment		Video 2	
Loss	**Weights**	**IoU (%)**	**ADD (mm)**
L_{mul}	$\alpha = 1$ $\beta = 4.75$ $\gamma = 1495$ $\delta = 1200$	81.72	27.04
L_{ess}	$\theta = 0.17$	85.38	23.24

Table 2: Comparison between Multifaceted Loss and Essential Loss models and various balancing configurations on video 2, each composed of 1000 frames and with ground truth updates every 30 frames.

real world by combining the translation vector and the rotation matrix obtaining the *Essential Loss*. Due to the shift from four to two components for the total loss, we only need to tune one coefficient now, making the loss balancing process easier. In addition, this change has brought to an increase in accuracy on both bounding box and pose estimation as shown in Table 2.

4.2.2 Augmenting Training Data

As a crucial aspect of model training, data augmentation plays a pivotal role in enhancing the robustness and adaptability of our model against spatial variations. We opt to solely use translation as a form of data augmentation, as it mirrors the most accurate depiction of the real scenario where the prediction at instant $t-1$ deviates from the object's center point represented in the image at instant t. The process of applying this transformation is not straightforward as SwiftTrack operates on inputs with varying dimensionality: the 2D bounding box in the image plane and the 3D pose in the real world. The approach for data augmentation involves retaining the original image pairs and translating the 3D object into world coordinates. Consequently, we derive a new position in the 2D image plane, which is further utilized to select and crop the two input images. The experiments presented in Table 3 are executed with consistent hyperparameter configurations, with the sole distinction being the inclusion of data augmentation in the training dataset. It is evident that data augmentation substantially enhances the accuracy of the model's predictions, making it notably more robust.

Experiment	Video 2	
Data Augmentation	**IoU (%)**	**ADD (mm)**
No	78.19	26.75
Yes	87.12	25.79

Table 3: Comparison between models trained with and without the use of data augmentation in the training dataset on video 2, each composed of 1000 frames and with ground truth updates every 30 frames.

4.2.3 Multi Task Learning

Multi-task learning (*MTL*) is a paradigm in machine learning that differs from the traditional approach of training models for singular objectives. Instead of solely focusing a model's proficiency in one task, MTL involves simultaneously mastering multiple, frequently related, tasks. The model specializes in two tasks: estimating the 2D bounding box in the image and estimating the 3D pose in the object's world. Model performance is highly sensitive to weight selection and this can significantly impact the results. Tuning weight hyper-parameters can be a costly and time-consuming process, often requiring several days for each trial. Therefore, we attempt to apply MTL techniques to find the optimal weights and evaluate if there are substantial improvements in the estimation accuracy of our tasks compared to manual loss balancing. As shown in Table 4, we mainly utilize three multi-task learning: **Random Loss Weighting**, **Homoscedastic Uncertainty**, and **Conflict-Averse Gradient Descent**. Experiments with Uncertainty and RLW show a decrease in the IoU score, but have distinct behaviours considering the ADD metric. RLW exhibits improvement, while Uncertainty experiences a deterioration in pose estimation. This suggests that RLW is better able to balance the losses compared to the model with Uncertainty. Finally, the experiment with CAGrad demonstrates improvements in bounding box estimation but shows a decline in pose estimation.

Experiment	Video 2	
Balancer	**IoU (%)**	**ADD (mm)**
No Balance	87.12	25.79
RLW	86.78	23.18
Uncertainty	87.09	49.15
CAGrad	88.46	43.93

Table 4: Comparison between models with different Multi-Task Learning algorithms on video 2, each composed of 1000 frames and with ground truth updates every 30 frames.

4.3. GDR-Net and SwiftTrack

We unveil the outcomes derived from the proposed framework expounded in Section 2. Our findings distinctly showcase that the integration of an object tracking algorithm significantly contributes to the improvement of pose estimation method inference times, rendering it well-suited for real-time applications. The fundamental architecture of our proposal involves employing GDR-Net, the selected pose estimation model, at a low frequency, while SwiftTrack assumes the role of predicting every frame at a higher frequency. As demonstrated in Table 5, SwiftTrack is significantly faster in inference compared to GDR-Net, showcasing its remarkable effectiveness in estimating tasks in real-time. Thanks to this capability, it can be utilized for the majority of the time, while GDR-Net is employed at a lower frequency to enhance pose prediction intermittently. An example of 2D bounding box and pose estimation on a frame is shown in Figure 3.

	GDR-Net	**SwiftTrack**
Average Inference (*fps*)	70	256

Table 5: Time inference of GDR-Net and SwiftTrack calculated using GPU.

5. Conclusions

Our investigation into embedded platforms reveals that the NVIDIA Jetson Nano, despite its economical and compact nature, performs similarly to a CPU. This discovery implies that inference analysis on either device can reliably estimate timings on the counterpart hardware. The study underscores the significance of hardware considerations in optimizing performance across embedded platforms. SwiftTrack, despite its simple structure, excels in accurately predicting object bounding boxes and 3D poses using only RGB images. Operating at an impressive speed of 256 frames per second, it proves effective for real-time applications. The model's versatility is highlighted by its ability to swap backbones, underscoring its adaptability to varying computational demands without compromising core strengths. In conclusion, our study establishes the effectiveness of the novel framework composed of the GDR-Net method and the object tracking algorithm for seamless real-time operation. The framework's flexibility accommodates various pose estimation methods, enhancing real-time processing across diverse applications. This adaptability opens avenues for widespread utilization. Future research could explore alternative datasets beyond YCB-Video, experiment with Recurrent Neural Networks for improved prediction accuracy, and fortify the object tracking model against challenging scenarios such as changes in light and high occlusion.

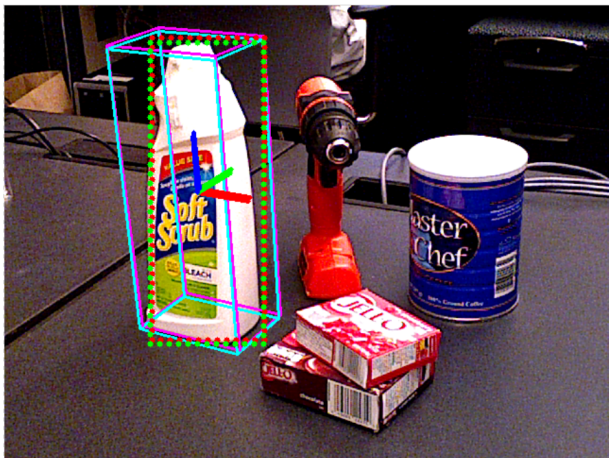

Figure 3: Example of bounding box and pose estimation of SwiftTrack. Ground truth for the 2D bounding box and object pose is denoted by green and light blue, while the model predictions are indicated by red and pink.

References

[1] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt,

Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[2] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[3] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion, 2019.

[4] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation, 2021.