بسمه تعالی

دانشکده مهندسی کامپیوتر

بینایی کامپیوتر

نام استاد: دکتر محمدی

تمرین هفتم

نام دانشجو: آرمان حیدری

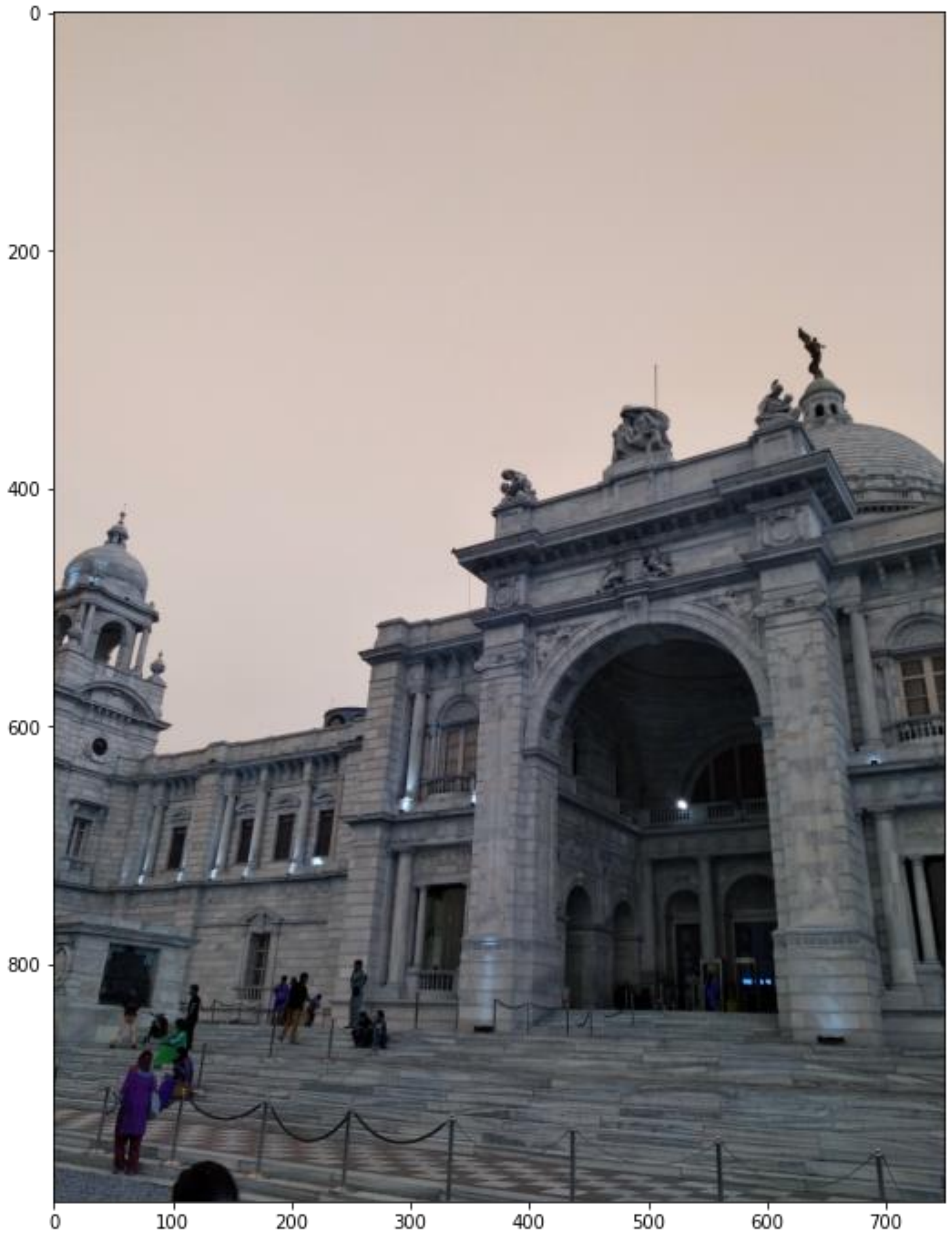
شماره دانشجویی: ۹۷۵۲۱۲۵۲

آبان ۱۴۰۱

فهرست

پاسخ سوال اول

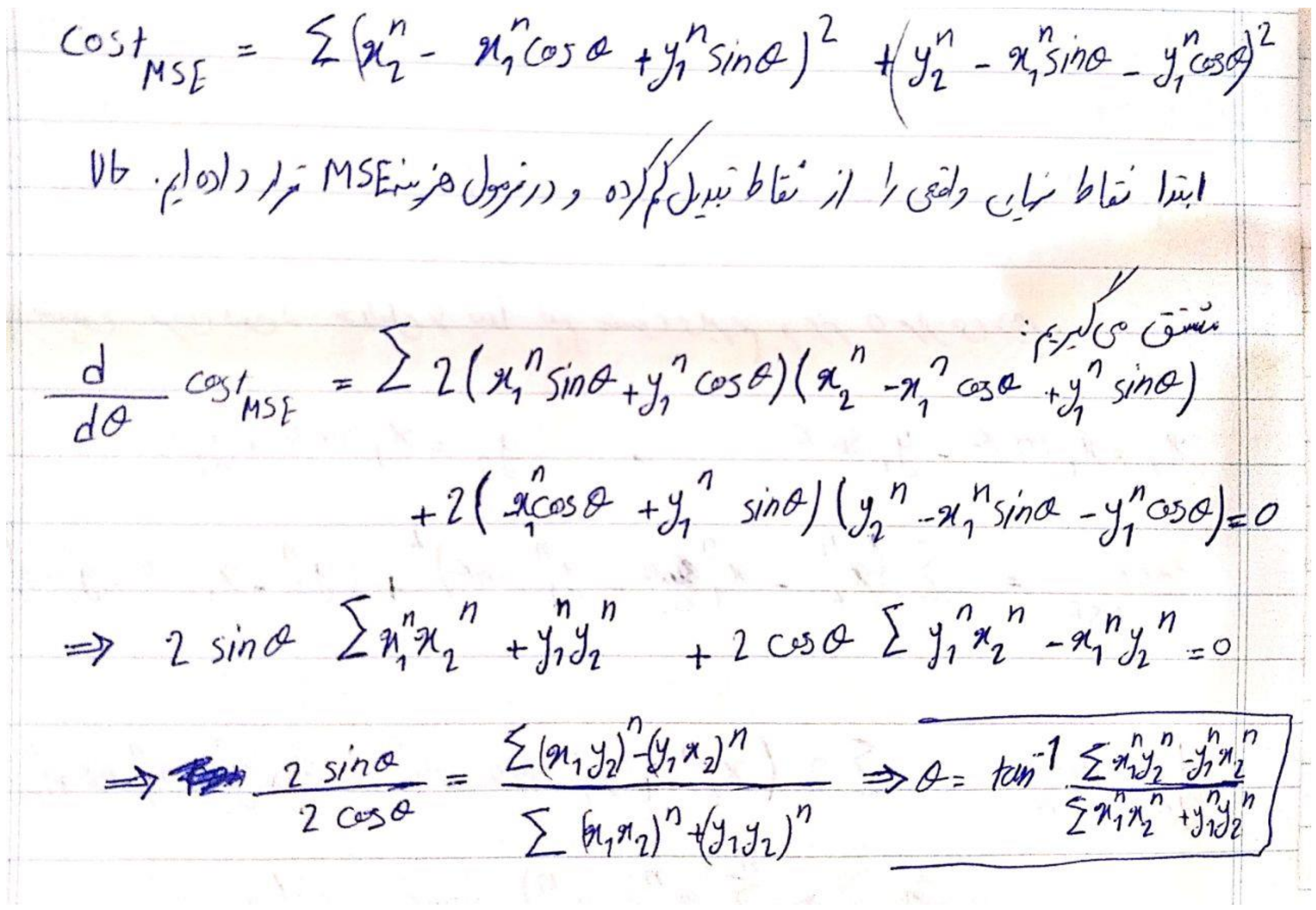
قسمت های خواسته شده در نوتبوک پیاده سازی شده اند. نتیجه نهایی:



منابع

- https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/

پاسخ سوال دوم



$$\text{cost}_{MSE} = \sum (x_2^n - x_1^n\cos\theta + y_1^n\sin\theta)^2 + (y_2^n - x_1^n\sin\theta - y_1^n\cos\theta)^2$$

ابتدا نقاط نهایی واقعی را از نقاط تبدیل کم کرده و در فرمول هزینه MSE قرار داده ایم. Vb

مشتق می گیریم:

$$\frac{d}{d\theta}\text{cost}_{MSE} = \sum 2(x_1^n\sin\theta + y_1^n\cos\theta)(x_2^n - x_1^n\cos\theta + y_1^n\sin\theta)$$

$$+ 2(x_1^n\cos\theta + y_1^n\sin\theta)(y_2^n - x_1^n\sin\theta - y_1^n\cos\theta) = 0$$

$$\Rightarrow 2\sin\theta \sum x_1^n x_2^n + y_1^n y_2^n + 2\cos\theta \sum y_1^n x_2^n - x_1^n y_2^n = 0$$

$$\Rightarrow \frac{2\sin\theta}{2\cos\theta} = \frac{\sum (x_1 y_2)^n - (y_1 x_2)^n}{\sum (x_1 x_2)^n + (y_1 y_2)^n} \Rightarrow \boxed{\theta = \tan^{-1}\frac{\sum x_1^n y_2^n - y_1^n x_2^n}{\sum x_1^n x_2^n + y_1^n y_2^n}}$$

پاسخ سوال سوم

الف) با روش luminance عکس رنگی را به سیاه سفید تبدیل میکنیم. ازین رابطه استفاده میکنیم:

$$Y = 0.299R + 0.587G + 0.114B$$

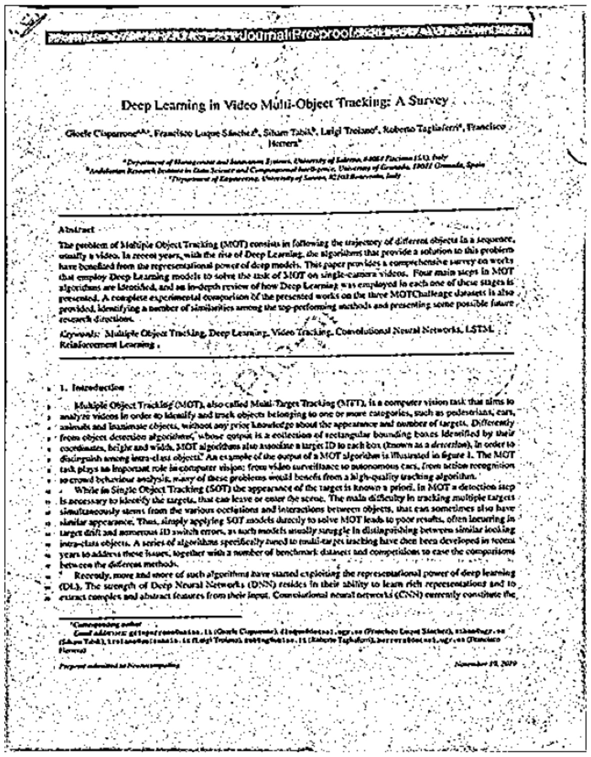
ب) از تابع bilateralFilter استفاده میکنیم که به خوبی میتواند نویز را با حفظ لبه ها حفظ کند. البته سه هایپرپارامتر d و sigmaColor و sigmaSpace را باید مقداردهی کنیم که با کمی تغییر به مقادیر مناسبی رسیدم.

پ) از لبه یاب Canny در cv2 با دو threshold ۵۰ و ۲۲۰ استفاده کرده ام و به خوبی لبه های کاغذ که هدف ماست پیدا شده و البته نوشته ها هم مشخصند.

ت) مطابق توضیحات نوتبوک جلو رفتم و ۴ گوشه به خوبی با findcontour پیدا شد.

ث) مطابق توضیحات نوتبوک جلو رفتم و متوازری الاضلاع به خوبی به مستطیل تبدیل شد.

ج) برنامه CamScanner ممکن است از الگوریتم پیچیده ای برای رسیدگی به موارد مختلف استفاده کند. اما من سعی خواهم کرد یک رویکرد اساسی برای چنین مشکلی را پوشش دهم، ایده اصلی در اینجا Binarization تصویر ورودی داده شده است، یا به طور دقیق تر می توانیم بگوییم Theresholding تصویر داده شده.



Journal Pre-proof

Deep Learning in Video Multi-Object Tracking: A Survey

Gioele Ciaparrone[a,b], Francisco Luque Sánchez[b], Siham Tabik[b], Luigi Troiano[c], Roberto Tagliaferri[a], Francisco Herrera[b]

[a] Department of Management and Innovation Systems, University of Salerno, 84084 Fisciano (SA), Italy
[b] Andalusian Research Institute in Data Science and Computational Intelligence, University of Granada, 18071 Granada, Spain
[c] Department of Engineering, University of Sannio, 82100 Benevento, Italy

Abstract

The problem of Multiple Object Tracking (MOT) consists in following the trajectory of different objects in a sequence, usually a video. In recent years, with the rise of Deep Learning, the algorithms that provide a solution to this problem have benefited from the representational power of deep models. This paper provides a comprehensive survey on works that employ Deep Learning models to solve the task of MOT on single-camera videos. Four main steps in MOT algorithms are identified, and an in-depth review of how Deep Learning was employed in each one of these stages is presented. A complete experimental comparison of the presented works on the three MOTChallenge datasets is also provided, identifying a number of similarities among the top-performing methods and presenting some possible future research directions.

Keywords: Multiple Object Tracking, Deep Learning, Video Tracking, Convolutional Neural Networks, LSTM, Reinforcement Learning

1. Introduction

Multiple Object Tracking (MOT), also called Multi-Target Tracking (MTT), is a computer vision task that aims to analyze videos in order to identify and track objects belonging to one or more categories, such as pedestrians, cars, animals and inanimate objects, without any prior knowledge about the appearance and number of targets. Differently from object detection algorithms, whose output is a collection of rectangular bounding boxes identified by their coordinates, height and width, MOT algorithms also associate a target ID to each box (known as a *detection*), in order to distinguish among intra-class objects. An example of the output of a MOT algorithm is illustrated in figure 1. The MOT task plays an important role in computer vision: from video surveillance to autonomous cars, from action recognition to crowd behaviour analysis, many of these problems would benefit from a high-quality tracking algorithm.

While in Single Object Tracking (SOT) the appearance of the target is known a priori, in MOT a detection step is necessary to identify the targets, that can leave or enter the scene. The main difficulty in tracking multiple targets simultaneously stems from the various occlusions and interactions between objects, that can sometimes also have similar appearance. Thus, simply applying SOT models directly to solve MOT leads to poor results, often incurring in target drift and numerous ID switch errors, as such models usually struggle in distinguishing between similar looking intra-class objects. A series of algorithms specifically tuned to multi-target tracking have then been developed in recent years to address these issues, together with a number of benchmark datasets and competitions to ease the comparisons between the different methods.

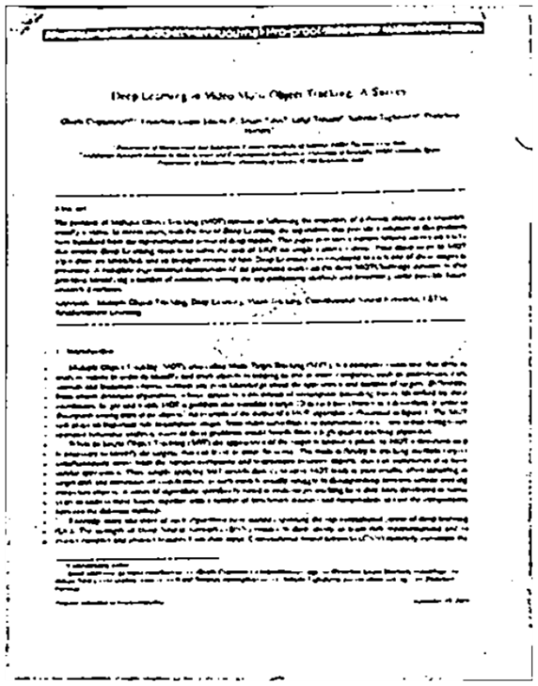
Recently, more and more of such algorithms have started exploiting the representational power of deep learning (DL). The strength of Deep Neural Networks (DNN) resides in their ability to learn rich representations and to extract complex and abstract features from their input. Convolutional neural networks (CNN) currently constitute the

*Corresponding author
Email addresses: [illegible] (Gioele Ciaparrone), [illegible] (Francisco Luque Sánchez), [illegible] (Siham Tabik), [illegible] (Luigi Troiano), [illegible] (Roberto Tagliaferri), [illegible] (Francisco Herrera)

Preprint submitted to Neurocomputing *November 19, 2019*

اگر یک تصویر دارای شرایط نوری متفاوت در مناطق مختلف باشد. در آن صورت، adaptive thresholding می تواند کمک کند. در اینجا، الگوریتم threshold یک پیکسل را بر اساس ناحیه کوچکی در اطراف آن تعیین می کند. بنابراین آستانه های متفاوتی را برای مناطق مختلف یک تصویر دریافت می کنیم که نتایج بهتری برای تصاویر با روشنایی متفاوت می دهد. و وضوح تصویر زیاد شده است.

اما چون این قسمت نویز نمک و فلفل زیادی را در تصویر ایجاد کرده است، سعی در حذف آن با فیلتر median هم داشتم که نتایج خوب شود:


Deep Learning in Video Multi-Object Tracking: A Survey

[illegible]

منابع

- https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html
- https://www.programcreek.com/python/example/89328/cv2.approxPolyDP
- https://stackoverflow.com/questions/56828644/215assertion-failed-npoints-0-depth-cv-32f-depth-cv-32s-in
- https://medium.com/analytics-vidhya/opencv-findcontours-detailed-guide-692ee19eeb18
- https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html
- https://theailearner.com/tag/cv2-getperspectivetransform/
- https://stackoverflow.com/questions/9808601/is-getperspectivetransform-broken-in-opencv-python2-wrapper
- https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html
- https://stackoverflow.com/questions/32913157/how-to-get-magic-color-effect-like-cam-scanner-using-opencv
- https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html