Presentazione argomento di Tesi

Alessandro Sciarrillo

Introduzione problema Bioretics

Bioretics offre un prodotto software che viene eseguito da macchine per la selezione della frutta e include l'utilizzo dell'algoritmo Marching Squares (MS). L'argomento proposto dall'azienda è quello di ridurre i tempi di esecuzione di MS che è utilizzato nella fase di segmentazione dei difetti.

Le macchine per la selezione gestiscono un flusso di circa 10 frutti al secondo; ne consegue che ci sia approssimativamente 0.1s a disposizione per ogni frutto. Quindi un'implementazione parallela dell'algoritmo MS, che riesca a ottenere un speedup anche solo di 1.1 sarebbe considerato un risultato positivo per l'azienda.

All'interno delle macchine vengono scattate immagini dei frutti che sono poi elaborate da una CNN che a sua volta restituisce un tensore WxHxC, dove ogni canale rappresenta una classe (esempio: picciolo, ammaccatura, muffa). I canali vengono poi passati singolarmente al MS che definisce i contorni di ogni classe. Il risultato del processo è visibile dall'Immagine 1 dove si possono notare le varie classi delimitate da colori differenti.

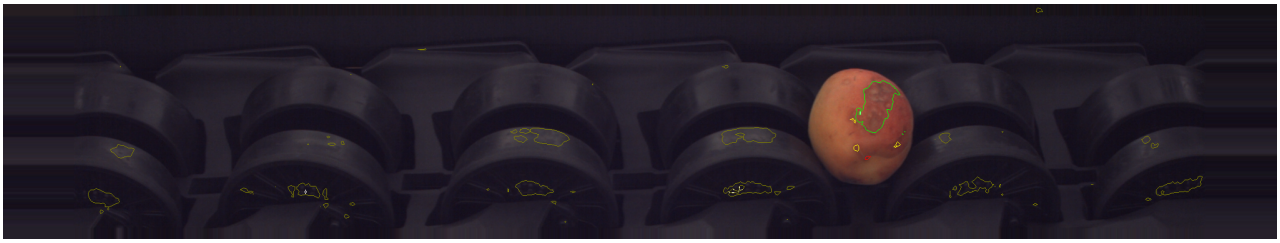

Immagine 1: Risultato ottenuto in seguito all'utilizzo dell'algoritmo marching squares.

Implementazione utilizzata attualmente

L'implementazione di MS che è attualmente utilizzata dall'azienda deriva dalla libreria scikit-image [1] che offre una versione seriale dell'algoritmo. Il metodo della libreria che viene chiamato è scritto in Python [2] ma la parte principale è stata scritta in Cython [3].

Il codice in Cython deve essere compilato prima di essere eseguito, nel complesso però riduce i tempi di esecuzione rispetto all'equivalente in Python. Bisogna quindi considerare che il tempo totale di esecuzione di MS è stato già in parte ridotto dagli autori della libreria.

Potenziali strumenti per la parallelizzazione

Attualmente sono state individuate alcune strade da poter seguire per parallelizzare l'algoritmo MS:

1. Utilizzare il compilatore nvc++ con i flag per la compilazione parallela sul codice C++ derivato dal Cython [4] della libreria scikit-image.
2. Sfruttare le API CUDA per Python [5] per scrivere manualmente i kernel CUDA e gestire l'uso delle memorie. Si potrebbe implementare usando direttamente le API fornite da Nvidia oppure appoggiandosi alla libreria Numba [6].

Riferimenti

[1] https://scikit-image.org/

[2] https://github.com/scikit-image/scikit-image/blob/main/skimage/measure/_find_contours.py

[3] https://github.com/scikit-image/scikit-image/blob/main/skimage/measure/_find_contours_cy.pyx

[4] https://developer.nvidia.com/blog/accelerating-python-on-gpus-with-nvc-and-cython/

[5] https://nvidia.github.io/cuda-python/overview.html

[6] https://numba.readthedocs.io/en/stable/cuda/index.html