OpenCL - Comparació de patrons

Març 2017

Resum

Aquesta pràctica té com a objectiu implementar un algorisme de comparació de patrons (*pattern matching*) a imatges. Aquest algorisme té una gran càrrega computacional i per això s'aprofitarà la capacitat multiprocessadora de les targes gràfiques per accelerar la computació.

1 El problema

La comparació de patrons és un algorisme que s'utilitza sovint per tal de trobar determinats objectes o patrons en una imatge. Sigui un patró $P(m, n)$ de mida 16×16 (que és el que s'utilitzarà en aquesta pràctica) i una imatge de nivell de gris $I(m, n)$ de dimensions $M \times N$, on podeu suposar que la imatge és més gran que el patró a buscar ($M > 16$, $N > 16$). Per conveni, a $I(m, n)$ la primera coordenada fa referència a la columna, mentre que la segona fa referència a la fila. Suposarem que l'origen de coordenades es troba a la cantonada superior esquerra d'aquests,tant pel patró $P(m, m)$ com per la imatge $I(m, n)$,

Donat un punt (m_0, n_0) a la imatge I, volem saber la similitud del tros 16×16 de la imatge amb el patró. Hi ha diverses formes de computar la similitud. Una de les formes més senzilles és utilitzar l'error quadràtic mig

$$E(m, n) = \frac{1}{16 \times 16} \sum_{i=0}^{15} \sum_{j=0}^{15} (I(m + j, n + i) - P(j, i))^2 \tag{1}$$

Aquesta equació ens dóna un valor numèric pel punt (m, n). Si el valor és petit (en comparació amb altres valors que es puguin obtenir) la similitud del tros 16×16 de la imatge amb el patró és gran. Si el valor és gran, hi ha gran dissimilitud.

L'objectiu és buscar la posició (m_0, n_0) de la imatge en què el patró P s'assembla més al que hi ha a la imatge I en aquella posició. En altres paraules, es tracta de buscar la posició (m_0, n_0) en què el valor $E(m_0, n_0)$ sigui petit (en comparció amb totes les possibles posicions del patró). Per fer-ho es computa primer $E(m, n)$ per a tots els pixels de la imatge I. A continuació es busca la posició $E(m_0, n_0)$ que té el valor més petit (és a dir, on la similitud amb el patró és més gran).

2 La pràctica

L'objectiu d'aquesta pràctica se centra en implementar el primer pas, la computació de $E(m, n)$ per a tots els píxels de la imatge, mitjançant OpenCL. El segon pas, la cerca de la posició (m_0, n_0) que té el valor més petit de E, es realitzarà a una CPU. La imatge $I(m, n)$ podrà tenir una mida qualsevol (superior a la del patró) mentre que el patró tindrà una mida de 16×16 píxels.

2.1 Implementació en C

Es proporciona una implementació base realitzada en C i que us servirà per a realitzar comparació de temps d'execució amb la implementació OpenCL. Per fer-ho cal tenir en compte que l'equació (1) s'ha d'implementar de forma equivalent tant a la CPU com a OpenCL. En particular, a l'equació (1) cal implementar "un sumatori d'elements al quadrat": implementar el quadrat d'un valor x es pot fer com $y(x) = x \cdot x$ o $y(x) = pow(x, 2.0)$. Perquè els resultats dels temps d'execució siguin comparables cal que la implementació en C i en OpenCL facin servir la mateixa implementació.

Els píxels propers a la vora inferior o dreta de imatge requereixen utilitzar píxels que caiguin "fora" de la imatge. Per evitar problemes hi ha dues solucions: i) comprovar si el píxel $(m_0 + j, n_0 + i)$ de la imatge cau fora de la imatge (mitjançant un "if") i procedir corresponentment, ii) abans d'aplicar l'equació (1), "engrandir" la imatge amb 15 píxels per la vora inferior i dreta per evitar haver d'implementar la primera solució. L'equació (1) s'aplica al rang de píxels abans d'engrandir la imatge. En aquesta pràctica s'ha implementat la 2a opció.

2.2 Implementació en OpenCL

Hi ha diverses formes per realitzar una implementació OpenCL del codi. Es proposen aquestes:

1. Traslladar la implementació en C directament a OpenCL sense tenir en compte l'estructura d'interna de warps, work-groups, etc d'OpenCL. Es recomana començar per aquí fent proves com les següents

   (a) Un work-item per a cada pixel de la imatge (abans d'engrandir-la).

   (b) Un work-item per a cada fila de la imatge (abans d'engrandir-la).

2. Una implementació en què es tingui en compte l'estructura d'interna de warps, work-groups, etc d'OpenCL. La imatge es dividirà en blocs de 32×32 píxels, veure figura 1. Cada bloc de la imatge serà processat per un bloc de fils (*work-group*) i cada bloc de fils estarà format per un conjunt de fils (*work-items*). Tingueu en compte que es possible que la imatge no es pugui dividir exactament en blocs de 32×32. En aquest cas tot aquells fils que caiguin fora del suport de la imatge no faran res.

   Com s'ha comentat abans, cada bloc de la imatge es processarà per un *work-group*. Cada *work-group* estarà format per 32×8 fils. Això dóna un

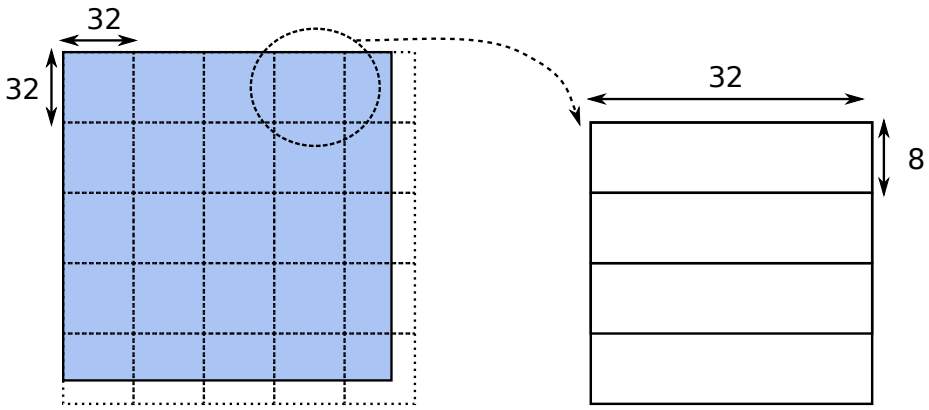

Figura 1: Esquema de la implementació de l'algorisme de la comparació de patrons a una imatge.

total de 256 fils per *work-group*. Per a cada bloc de la imatge, per tant, un *work-item* aplicarà (1) per a 4 posicions diferents. La figura 1 proposa com organitzar els *work-items* perquè es realitzin tots els càlculs per al bloc de la imatge.

Per a l'exemple de la figura 1, observar que la imatge es pot cobrir amb un total de 5×5 blocs de 32×32. Això vol dir que a l'eix horitzontal es crearan un total de 5×32 *work-items*, mentre que a l'eix vertical es crearan un total de 5×8 *work-items*. Els *work-groups* tindran una mida de 32×8.

Per a la implementació d'aquesta part, es proposa

(a) Una implementació que no faci servir memòria local

(b) Una implementació que faci servir memòria local (opcional)

3 Què s'ha d'entregar ?

En aquesta secció es detalla el que s'ha d'entregar:

- El fitxer que entregueu s'ha d'anomenar `P2_Cognom1Cognom2.tar.gz` (o .zip, o .rar, etc). `Cognom1` correspon al primer cognom del primer membre del grup, mentre que `Cognom2` correspon al primer cognom del segon membre del grup. Dintre d'aquest fitxer hi ha d'haver dues carpetes: `src`, que contindrà el codi font, i `doc`, que contindrà un petit document explicatiu.

- La carpeta `src` contindrà tot el codi font. Aquesta carpeta ha d'incloure tots els fitxers necessaris per compilar i generar l'executable a les màquines de l'aula IA. Es demana que les funcions estiguin comentades per facilitar la lectura del codi (no cal seguir estil Doxygen).

- La carpeta `doc` ha de contenir un petit PDF que expliqui els següents punts:

    1. La forma de compilar el codi
    2. Explicació de les implementacions realitzades.
    3. Comparació dels temps d'execució de les diverses parts.

La part del codi computa un 50% de la nota, mentre que el document un 50%.