TREIZIÈME COLLOQUE GRETSI - JUAN-LES-PINS DU 16 AU 20 SEPTEMBRE 1991

TRANSFORMATIONS GEOMETRIQUES POUR LE PROCESSEUR LIGNE SYMPATI 2

Jean-Paul CARRARA*, Jean-Luc BASILLE*
Jean-François LARUE**, Didier JUVIN**

*IRIT : 118 route de Narbonne 31062 TOULOUSE CEDEX Tél : 61-55-63-08 Fax : 61-55-62-58
**C.E.A. D.E.I.N. Saclay 91191 GIF SUR YVETTE CEDEX Tél : 69-08-56-26

RÉSUMÉ

Pour exécuter des transformations géométriques sur le Processeur Ligne SYMPATI 2 de type SIMD [1] nous exploitons ses possibilités d'accès en ligne ou en colonne. Ces possibilités nous permettent de décomposer les transformations d'images en deux étapes, l'une s'exécutant en laissant les x constants, l'autre en laissant les y constants. Cette décomposition nous permet d'obtenir des temps d'exécution très intéressants si on les rapproche de ceux que l'on peut obtenir avec un opérateur cablé.

ABSTRACT

In order to execute geometric transforms on the SIMD type Line Processor SYMPATI 2 we take advantage of its access capabilities for processing the image either in a row by row or in a column by column fashion. These capabilities allow us to decompose image transforms into two stages. The first one processes a x-constant transform, the second one a y-constant transform. This decomposition makes it possible to obtain very attractive processing times in comparison with those provided by wired operators

1. INTRODUCTION.

Les applications vidéo utilisent de plus en plus d'outils de traitement ou de manipulation d'images. Pour des raisons de temps d'exécution, ces outils sont le plus souvent cablés. Nous voulons montrer que certains de ces outils peuvent être mis en œuvre sur une structure programmable tout en conservant un bon temps d'exécution. Les transformations géométriques constituent un exemple de ces outils opérant au niveau du pixel et pour lesquelles l'utilisation d'un Processeur Ligne se révèle avantageuse. Nous commencerons par rappeler le problème des tranformations géométriques, puis nous donnerons quelques éléments d'implantation sur une architecture de type Processor Array et nous aborderons les algorithmes qui nous ont permis la mise en œuvre des tranformations géométriques que nous illustrerons par des exemples.

2. LES TRANSFORMATIONS GEOMETRIQUES

2.1 Les problèmes rencontrés.

Nous allons rappeler le problème des transformations géométriques. Une transformation géométrique peut être définie comme une application fournissant un point (X,Y) = H(x,y) à partir du point (x,y) où X = f(x,y) et Y = g(x,y).
Les transformations que nous traiterons seront définies bijectivement dans un espace continu. Deux problèmes apparaissent dans un espace discret :

- la modification d'échelle
- le sens de la mise en correspondance.

Le problème de la modification d'échelle apparaît avec les transformations géométriques qui ne sont pas des isométries, c'est à dire lorsque l'échelle de la grille est modifiée. Il se peut alors qu'un point (X,Y) de l'image transformée ait plusieurs antécédents voire aucun suivant le sens de variation de l'échelle de la grille. Nous devons nous assurer d'une part que la valeur affectée au pixel de la grille de l'image transformée est la valeur moyenne de la fonction image en un cercle centré sur le pixel considéré et d'autre part que deux cercles adjacents ne se chevauchent pas trop.
Le second problème provient de la correspondance entre chaque point transformé (X,Y) et la discrétisation initiale de l'image et réciproquement entre chaque point (x,y) et la discrétisation finale de l'image. Cela revient à dire qu'un point (X,Y) de l'image transformée n'a pas nécessairement d'antécédent dans l'image initiale et réciproquemement. De ces deux points de vue découlent deux approches pour déterminer l'image transformée.

2.2 Les deux approches.

La première approche consiste en la distribution des points (x,y) de la grille initiale sur la grille transformée. La valeur assignée au point (X,Y) de l'image transformée est la somme des valeurs respectivement distribuées. Le nombre de ces valeurs est compris entre 3 et 5.

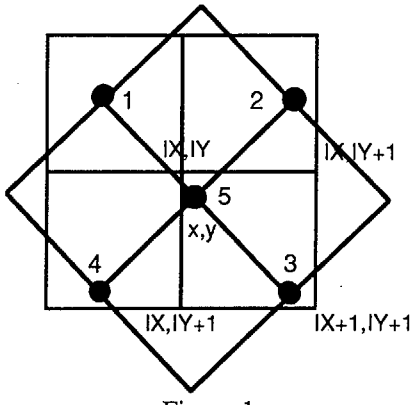

Figure 1.

Dans la figure 1 la distribution de la valeur du pixel (x,y) est effectuée sur les quatre pixels (IX,IY), (IX+1,IY), (IX,IY+1), (IX+1,IY+1), où IX est la partie entière de X=f(x,y) et IY la partie entière de Y=g(x,y). Cette distribution s'effectue de fait sur les quatre coins du carré de coordonnées entières auquel appartient le point (X,Y). On remarque dans cet exemple que la valeur assignée au point (IX,IY) est la somme pondérée de cinq valeurs provenant des pixels de l'image d'origine.
Cette méthode est applicable aux transformations non isométriques en prenant quelques précautions. Si l'échelle de la grille est diminuée alors chaque point de la grille d'arrivée reçoit une fraction de la valeur de plusieurs pixels de l'image initiale. Une normalisation est alors suffisante. A l'opposé, si l'échelle de la grille est augmentée, une répartition simple comme celle vue en figure 1 n'est pas suffisante et de nombreux points de la grille transformée n'auront aucune valeur affectée. Une répartition plus complexe peut éliminer ce problème mais elle nécessite beaucoup plus de traitements.
La deuxième approche détermine directement la valeur des points transformés. L'antécédent (x,y) d'un point (X,Y) de la grille transformée appartient nécessairement au carré (Ix,Iy), (Ix+1,Iy), (Ix,Iy+1), (Ix+1,Iy+1). On remarque qu'il est nécessaire que f et g soient injectives. Mais, il se peut que la fonction f (respectivement g) ne soit pas surjective. Cela signifie que l'antécédent se situe en dehors de l'image initiale. Dans ce cas, ces points de l'image finale sans antécédent se verront assigner une valeur par défaut correspondant au "fond" de la scène.

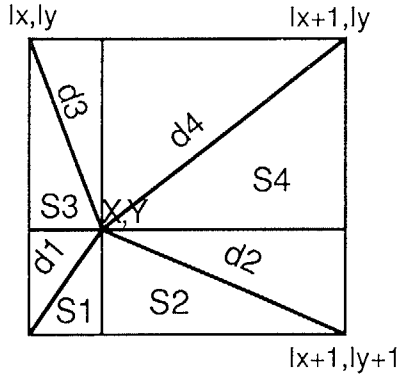

Figure 2.

La valeur assignée au point (X,Y) sera déterminée à partir des valeurs des 4 points précités considérant (X,Y) comme leur barycentre. Le meilleur jeu de coefficients semble être la distance euclidienne (d_i) entre (X,Y) et chacun des 4 points. Mais, dans un but de simplification des calculs, il a été montré [2] que les surfaces (s_i) des 4 rectangles définis par (X,Y), comme montré en figure 2, fournit également un bon ensemble de coefficients.
La deuxième méthode est de loin la meilleure. C'est celle que nous avons choisie et nous allons voir comment la paralléliser.

3. MISE EN ŒUVRE SUR UN PROCESSOR ARRAY.

3.1. Sur un Processor Array bidimensionnel.

La mise en œuvre des transformations géométriques se heurte, sur les processeurs bidimensionnels, au problème de la transmission des données : le chemin pour atteindre (X,Y) à partir du point initial n'est pas régulier. En conséquence, la méthode doit être décomposée en plusieurs étapes successives, en utilisant par exemple l'algorithme de WEIMAN [3],[4]. Une telle décomposition dépend des paramètres de la transformation : le coefficient de zoom par exemple. Le temps d'exécution devient variable, et, pour certaines valeurs, ce temps d'exécution devient très important [5]

3.2. Sur un Processeur Ligne.

Un Processeur Ligne est un Processor Array mono-dimensionnel. Le Processeur Ligne SYMPATI 2 permet d'accéder indifféremment à une ligne ou à une colonne. Il permet de réorganiser les données de manière à les traiter dans un mode tabulé où le calcul d'adresse est effectué à l'intérieur du Processeur Elémentaire. Une transformation géométrique peut être effectuée en deux étapes selon l'une des 2 décompositions duales suivantes :

- soit on débute par une transformation laissant les y invariants puis on termine par une transformation laissant les X invariants :

(x,y) ---> (X=f(x,y),y) ---> (X,Y=g(x,y)),

- soit on débute par une transformation à x constant et on termine par une transformation à Y constant :

(x,y) ---> (x,Y=g(x,y)) ---> (X=f(x,y),Y).

La détermination de la valeur à affecter au point (X,Y) nécessite la valeur de son antécédent (x,y). Nous avons déjà vu que f et g devaient être des applications injectives. Ceci entraîne nécessairement l'injectivité de la fonction H qui évalue le point (X,Y) à partir du point (x,y). Ces hypothèses sont suffisantes mais sont elles nécessaires?
Si nous considérons la fonction f_y comme une restriction de la fonction f à la ligne y c'est à dire :

$$f_y(x) = f(x,y) = X ,$$

la fonction f_y doit être injective afin de connaître la valeur de l'antécédent (x,y) du point (X,y).

On peut remarquer que l'injectivité de f implique celle de f_y. La réciproque n'étant pas nécessairement vraie.

En effet si nous prenons l'exemple d'une fonction linéaire f(x,y) = a∗x + b∗y où a et b sont constantes, b # 0, nous pouvons trouver (x1,y1) # ((x2,y2) tels que f(x1,y1) = f(x2,y2). Il suffit pour cela de considérer x2 # x1 et $y2 = \frac{y1 + (x1-x2) * a}{b}$.
Par contre f_y , restriction de f à la ligne y, est injective. En effet : a∗x1 + b∗y = a∗x2 + b∗y implique x1 = x2 si a # 0.

Dans la première décomposition, l'injectivité de H, ainsi que celle de f_y, est impérative. Il en va de même dans la deuxième décomposition pour H et g_x, où g_x est la restriction de la fonction g à la colonne x c'est à dire :

$$g_x(y)=g(x,y).$$

Le choix parmi ces deux décompositions sera donc dicté par l'injectivité de l'une des deux restricitions f_y ou g_x.

Par exemple les rotations de $k*\frac{\pi}{2}$ (où k appartient à N) ne peuvent être décomposées selon l'une des deux procédures. La rotation de $\frac{\pi}{2}$ de centre (0,0) peut s'exprimer de la façon suivante :

$$f(x,y) = -y$$
$$g(x,y) = x$$

De la sorte $f_y(x)$ est constante et donc non injective. Il en va de même pour $g_x(y)$.

Bien sur, de telles rotations peuvent être effectuées sur un Processeur Ligne. C'est d'une manière directe que l'on obtiendra le point (X,Y) en redistribuant les points (x,y).

Un autre exemple pourrait être le suivant :

$$(x,y) \text{ ----> } H(x,y) = (X,Y) \text{ où}$$
$$X = f(x,y) = a*x + b*y$$
$$Y = g(x,y) = x$$

Ici $f_y(x)$ est injective comme nous l'avons vu précédemment. Mais $g_x(y)$ n'est pas injective. H l'est-elle?

$$(X1,Y1) = (X2,Y2) \text{ implique}$$
$$a*x1 + b*y1 = a*x2 + b*y2$$
$$\text{et } x1 = x2.$$

On en déduit $b*y1 = b*y2$ et $y1 = y2$ si $b \neq 0$.

Donc si (X1,Y1) = (X2,Y2) alors (x1,y1) = (x2,y2) et H est injective

4. MISE EN ŒUVRE DE DEUX TRANSFORMATIONS SUR SYMPATI 2.

4.1. La machine SYMPATI 2.

SYMPATI 2, SYstème MultiProcesseur Adapté au Traitement d'Images, est un Processeur Ligne conçu pour le traitement d'images mais avec lequel nous pouvons mettre en oeuvre d'autres types d'applications. Ainsi nous avons conçu une architecture composée de 32 PEs dans la version de base et pouvant comporter jusqu'à 128 P.E.s. Chaque Processeur Elémentaire adresse 32 K octets de mémoire, ce qui fait 1 Méga octet de mémoire pour la version de base. La machine est constituée d'une carte pour l'unité de contrôle et d'une, deux ou quatre cartes pour les Processeurs Elémentaires.

♦ Le Processeur Elémentaire est composé de deux parties : une partie adressage et une partie calcul. La partie adressage est composée elle même de deux modules :

- le module de calcul d'adresse évalue l'adresse mémoire du pixel qui va être traité à partir des coordonnées transmises par l'unité de contrôle. Il est également possible de fonctionner dans un mode tabulé pour des cas particuliers tels la multirésolution, la redistribution des données.
- le module de masquage compare les coordonnées du pixel avec la fenêtre définie par l'utilisateur et masque, si nécessaire, les traitements. Il est ainsi possible de limiter l'algorithme à une région de l'image.

♦ La partie calcul est quant à elle beaucoup plus classique. Elle possède une U.A.L 16 bits pour les opérations classiques (logiques, arithmétiques ainsi qu'un multiplicateur 8x8) ainsi qu'un scratch-pad de registres. Elle inclut également quelques composants spécifiques : un réseau d'interconnexion permettant de définir le chemin de données interne, et 4 indicateurs de travail permettant l'exécution de séquences conditionnelles.

♦ Les Processeurs Elémentaires ont été intégrés en technologie standard CMOS 1,25 µm. Chaque boitier possède 224 broches et renferme 4 P.E.s ce qui représente environ 85000 transistors.

♦ Les P.E.s sont linéairement interconnectés mais chaque P.E. est également connecté aux bancs mémoire de ses P.E.s adjacents. Ceci permet l'adressage d'un pixel quelconque de la fenêtre 3x3 en seulement un cycle. En outre cela permet une accélération de l'accès à de plus grands voisinages.

♦ L'unité de contrôle est constituée classiquemet d'un pipe-line à quatre étages utilisé pour envoyer les différents paramètres aux P.E.s : les coordonnées du segment, la fonction à exécuter, les indicateurs à utiliser, les registres considérés, les chemins de données. De plus, elle fournit la possibilité d'émuler un Processor Array bidimensionnel, c'est à dire qu'elle se charge du balayage de l'image à la place de l'utilisateur.

4.2. La transformation d'échelle : zoom.

Deux catégories de zoom sont à considérer suivant que leur facteur d'échelle est supérieur ou inférieur à 1.

- Dans le premier cas plusieurs points de l'image d'arrivée auront leurs antécédents dans le même carré de la grille de l'image de départ. Il n'y a donc aucun problème quel que soit le rapport de réduction.
- Dans le deuxième cas deux points adjacents peuvent avoir leurs antécédents dans deux carrés non adjacents. De l'information va alors être perdue car certains points de la grille initiale ne seront jamais pris en compte. La transformation de rapport **r** peut être décomposée en 2 autres : d'abord un zoom de facteur d'échelle, de rapport **r1** qui est une puissance entière de deux puis un second zoom de facteur d'échelle **r2** tels que :

$$r = r1 \times r2$$
$$r = 2^{\log_2(r)}$$
$$r = 2^{E(\log_2(r))} * 2^{D(\log_2(r))}$$

où E(x) est la partie entière de x et D(x) sa partie décimale.

Le premier rapport de zoom vaut donc $2^{E(\log_2(r))}$ et le second vaut $2^{D(\log_2(r))}$. Comme dans le cas des rotations de $k*\frac{\pi}{2}$ l'image intermédiaire est obtenue (sur un Processeur Ligne) par une détermination directe (3.2).

4.3 La rotation.

Quelle que soit la valeur θ de l'angle de la rotation, nous pouvons toujours l'exprimer comme ceci :

$$\theta = k*\frac{\pi}{2} + (\theta - k*\frac{\pi}{2})$$

où $k \in N$ et $\theta - k*\frac{\pi}{2} \in [-\frac{\pi}{4}, +\frac{\pi}{4}]$.

Une rotation peut être vue comme la composition de deux affinités :

$$X = a*x - b*y + c$$
$$Y = b*x + a*y + d$$

où $a = \sin(\theta)$, $b = \cos(\theta)$ et (c,d) représente les coordonnées de l'origine du nouveau repère.

Dans le cas de la première approche (figure 3) la première affinité (1) ne pose pas de problème car x et y sont connus pour le calcul de X. Par contre pour la deuxième transformation (2), le calcul de Y nécessite la valeur de x qui n'est plus directement disponible.

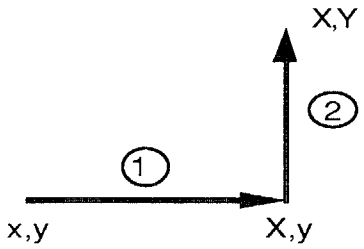

Figure 3.

Dans le cas de la deuxième approche (figure 4), il s'agit de déterminer la valeur du point (X,y) à partir du point (x,y) par une première affinité (1) puis du point (X,Y) par une deuxième affinité (2). Dans ce cas, on travaille à partir des équations suivantes :

$$x = a*(X - c) + b*(Y - d)$$
$$y = -b*(X - c) + a*(Y - d)$$

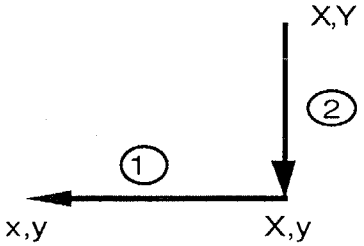

Figure 4.

Dans ce cas c'est lors de la première transformation que Y n'est pas encore connu. On doit donc le déterminer en utilisant la fonction inverse de la deuxième affinité (2).
Quelle que soit la méthode, on s'aperçoit qu'il existe toujours une étape pour laquelle soit une donnée n'est plus disponible, soit une donnée n'est pas encore disponible. Nous avons opté pour la deuxième approche qui ne laisse aucun pixel non affecté dans l'image d'arrivée.
Pour être mise en œuvre sur notre Processeur Ligne, la méthode nécessite des outils qui permettent la réorganisation de l'image en mode tabulé ou hélicoïdal. Le tableau 1 montre les temps d'exécution que l'on a obtenu pour tous les algorithmes élémentaires nécessaires à la réalisation d'une transformation.

Algorithme élémentaire	Temps d'exécution
Hélicoïdal --> colonne tabulée	1.44 ms
Hélicoïdal --> ligne tabulée	1.44 ms
Colonne tabulée --> hélicoïdal	1.44 ms
Ligne tabulée --> hélicoïdal	1.95 ms
Zoom sur les coordonnées en x	3.63 ms
Zoom sur les coordonnées en y	3.63 ms
Affinité sur les coordonnées en x (rotation)	4.19 ms
Affinité sur les coordonnées en y (rotation)	3.88 ms

Tableau 1 : Temps d'exécution pour une image de 256x256 pixels sur un Processeur Ligne de 32 P.E.s

Le tableau 2 résume les temps d'exécution des deux transformations qui nous ont servi de support d'étude.

Transformation	Temps d'exécution
Rotation	14,34 ms
Zoom	13,53 ms

Tableau 2 : Temps d'exécution global pour une image de 256x256 pixels sur un Processeur Ligne de 32 P.E.s.

5.Conclusion.

Par les transformations zoom et rotation nous avons voulu montrer que la mise en œuvre de tels traitements sur notre architecture ne posait aucun problème. Les transformations que nous avons étudiées ne sont pas les seules que l'on puisse exécuter sur un Processeur Ligne, mais elles en constituent deux des exemples les plus significatifs. Notre méthode, si on respecte les contraintes définies au paragraphe 2, permet la réalisation de n'importe quelle anamorphose de l'image.
On remarque que les temps d'exécution qui sont obtenus sur la machine de base sont très intéressants si on les compare à ceux obtenus avec certains opérateurs cablés. L'avantage de notre structure est qu'elle est programmable. On peut donc mettre en œuvre des applications complètes.

7.BIBLIOGRAPHIE.

[1] JUVIN D., BASILLE J.-L., ESSAFI H., LATIL J-Y : "SYMPATI 2, a 1,5 D Processor array for image application." Eusipco Signal Processing IV : Theories and applications, North-Holland, pp 311-314, 1988.

[2] BASILLE J-L : "Traitement d'images numériques, application : système inter-actif de caryotypie", Thèse de Docteur Ingénieur, Université Paul Sabatier, Toulouse, 1976.

[3] C.F.R. WEIMAN : "Highly parallel Digitized Geometric Transformations Without Matrix Multiplication", International Joint Conference on Parallel Processing, pp.1-10, 1976.

[4] C.F.R. WEIMAN : "Continuous Anti-aliased Rotation and Zoom of Raster Images", Computer Graphics Vol. 14(3), pp.286-293, Juillet 1980.

[5] K.A. CLARKE and H. H-S. Ip : "A Parallel Implementation of Geometric Transformations", Image Processing Group, Department of Physics and Astronomy, University College London, Report No. 82/5, March 1982