Implementing Image Registration Algorithms on Reconfigurable Computer

Miaoqing Huang[1]
Olivier Serres[1] Tarek El-Ghazawi[1] Greg Newby[2]

[1]Department of Electrical and Computer Engineering
The George Washington University

[2]Arctic Region Supercomputing Center
University of Alaska Fairbanks

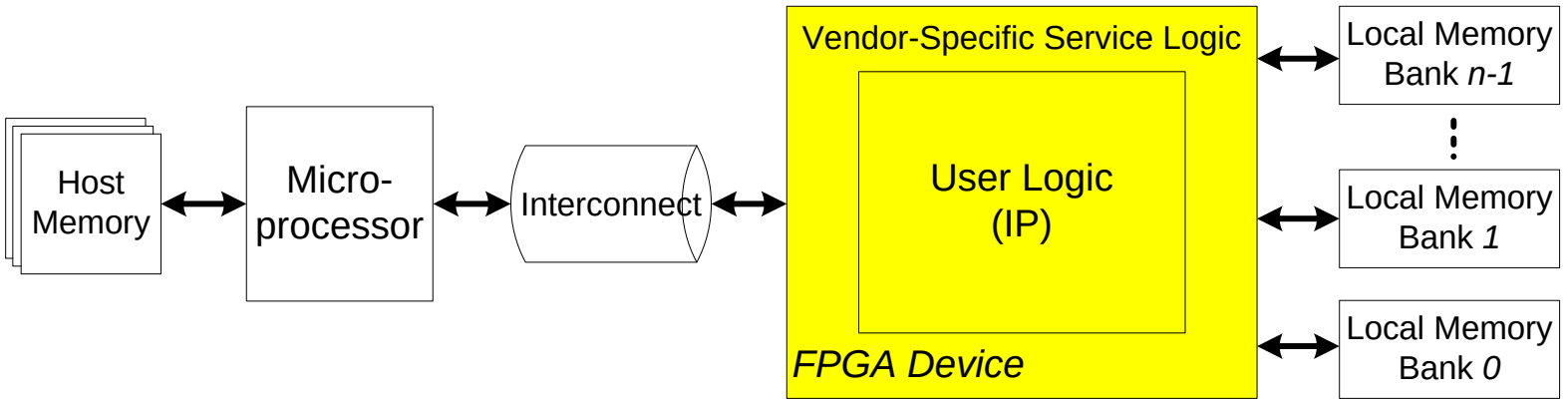
MAPLD 2008

Outline

Outline

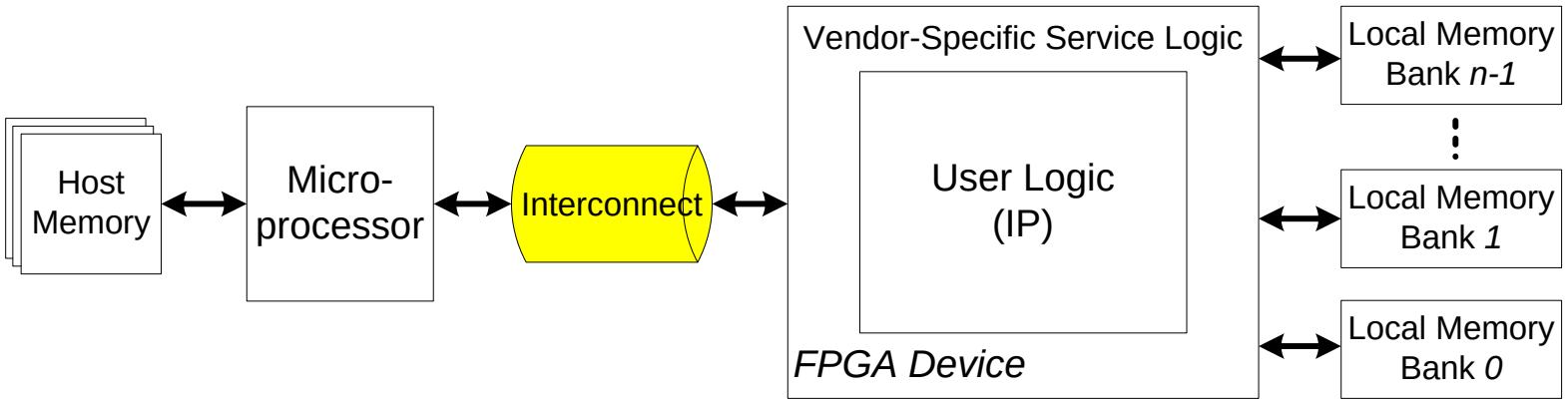
Outline

Generic Hardware Architecture of Modern HPRC



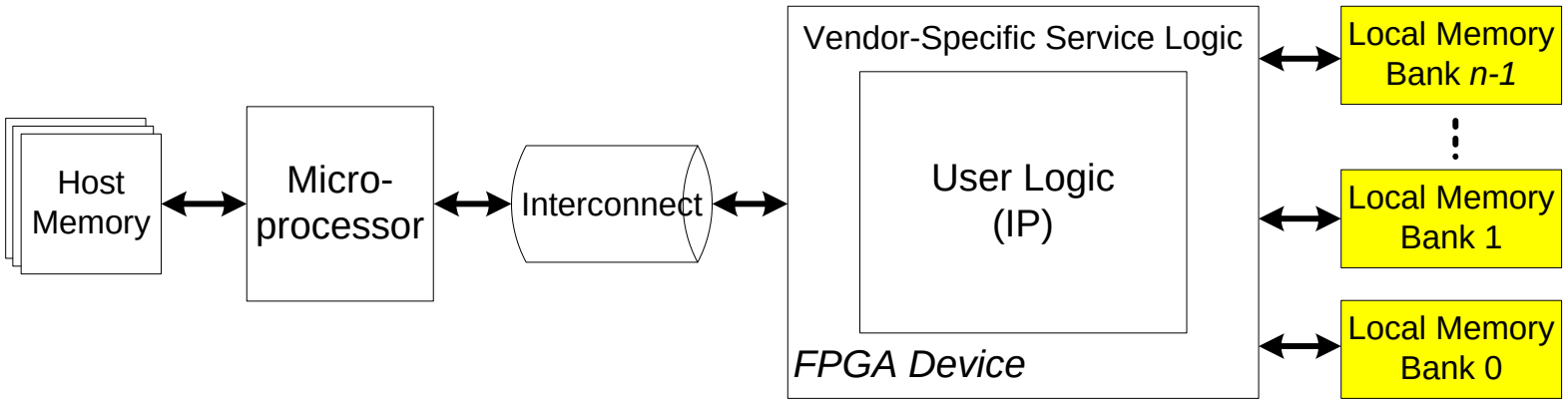
- **Reconfigurable Hardware Accelerator (e.g. FPGA) is treated as co-processor to microprocessor (μP)**

Generic Hardware Architecture of Modern HPRC



- Reconfigurable Hardware Accelerator (e.g. FPGA) is treated as co-processor to microprocessor (μP)
- **FPGA is connected to μP through high-speed interconnect**
  - **SGI RC100: NUMAlink4, 3.2GB/s**
  - **Cray XD1: HyperTransport, 1.6GB/s**

Generic Hardware Architecture of Modern HPRC

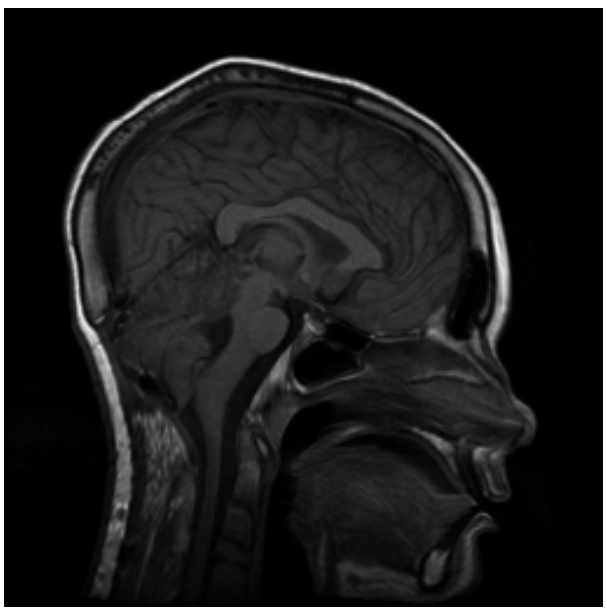


- Reconfigurable Hardware Accelerator (e.g. FPGA) is treated as co-processor to microprocessor (μP)
- FPGA is connected to μP through high-speed interconnect
  - SGI RC100: NUMAlink4, 3.2GB/s
  - Cray XD1: HyperTransport, 1.6GB/s
- **FPGA is connected into multiple local memory banks**
  - **User logic can access multiple banks in parallel**

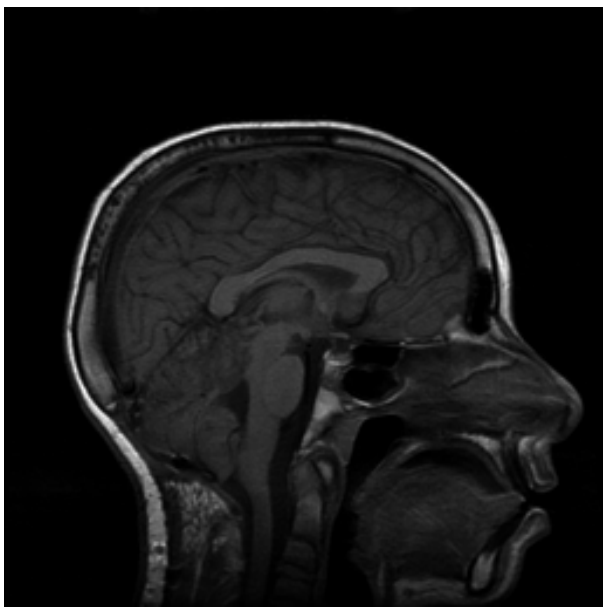
Outline

What is Image Registration?

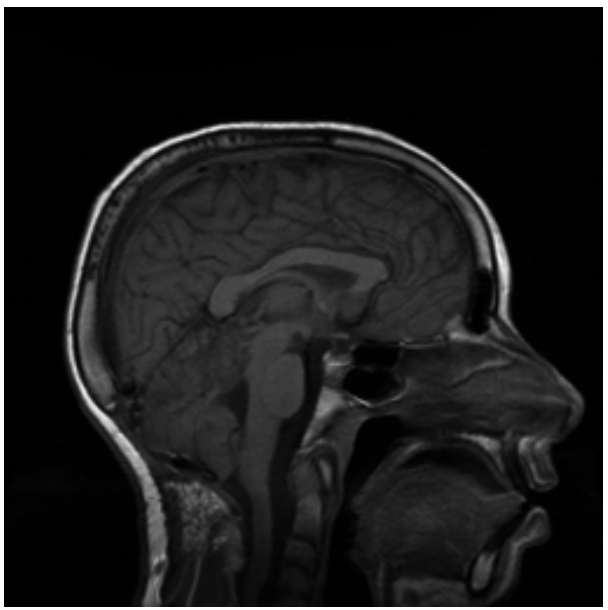
Image registration is a fundamental task in image processing used to match two or more pictures taken at different times, from different sensors, or from different viewpoints.



source



target



registered source

Image Registration as Transformation between Images

Mapping two images, the reference image and the test image, both spatially and with respect to intensity

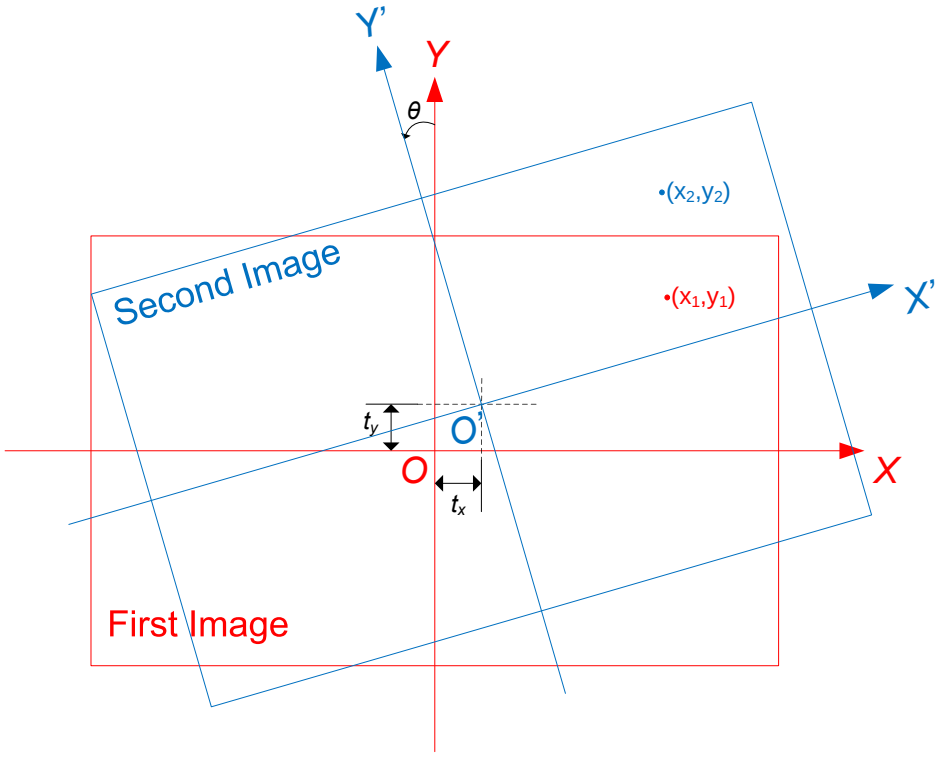
$$I_2(x, y) = g(I_1(f(x, y)))$$

Four components in image registration

1. *a feature space*: the set of characteristics used to match
2. *a search space*: the potential transformations between reference and test images
3. *a search strategy*: decide how to choose next transformation in the search space
4. *a similarity metric*: evaluate the match between reference image and transformed test image

Rigid-body Transformation

Rigid-body transformation [Brown 1992] is a combination of a rotation (θ), a translation (t_x,t_y), and a scale change (s)



$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + s \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

or

$$\overline{p}_2 = \overline{t} + s\mathcal{R}\overline{p}_1$$

Note: the scale change s is assumed to be 1 in this work

[Brown 1992] L. G. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

Outline

Selected Image Registration Mechanism

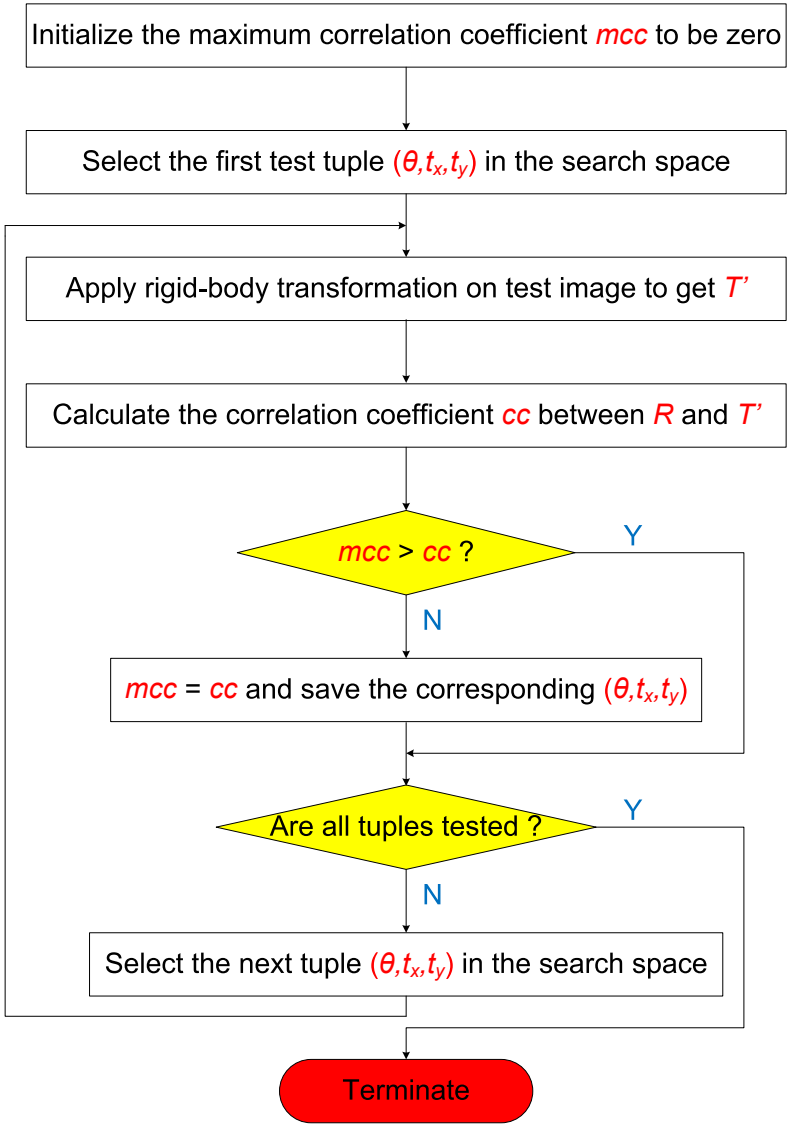
- Transformation: rigid-body transformation
- Search space: $(\Delta\Theta, \Delta X, \Delta Y)$
- Similarity metric: correlation coefficient between reference image R and transformed image T'

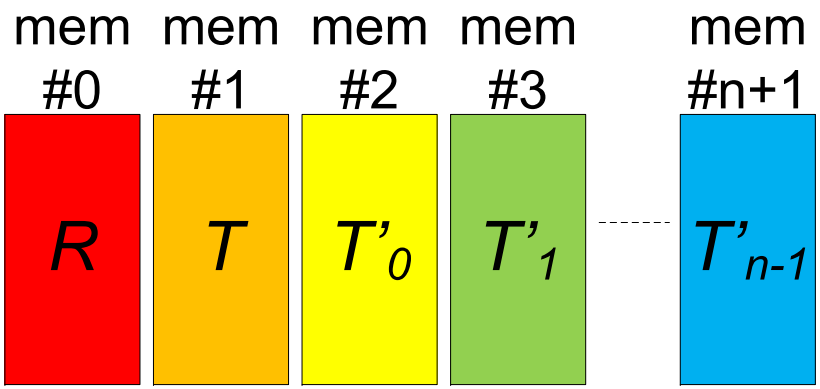
$$\frac{\sum_{x,y}(R(x,y)-\mu_R)(T'(x,y)-\mu_{T'})}{\sqrt{\sum_{x,y}(R(x,y)-\mu_R)^2 \sum_{x,y}(T'(x,y)-\mu_{T'})^2}}$$

- Two search strategies
  1. Exhaustive search algorithm
  2. Discrete Wavelet Transform (DWT)-based search algorithm

Outline

The Algorithm and Advantages of Hardware Implementation



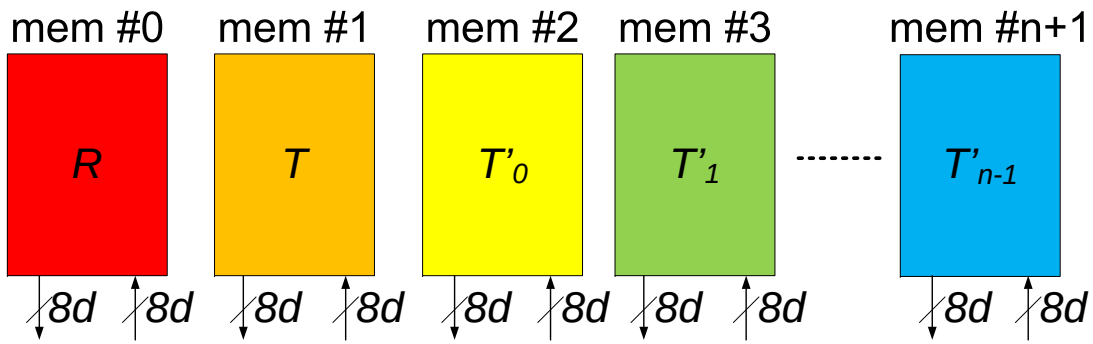


The advantages of hardware implementation

- Multiple (θ, t_x, t_y) tuples can be tested in parallel
  - Limited by available *hardware resource* or *memory banks*
- Pipelined design can process multiple pixels every clock cycle

Calculate the Image Registration Time in Hardware – *Assumptions*

1. Both reference image and test image are in 8-bit gray-scale and consist of m pixels
2. $n+2$ local memory banks with FPGA device
   - One bank $\longrightarrow$ R, one bank $\longrightarrow$ T, n banks $\longrightarrow$ n T's
   - Each bank has independent read and write ports
3. The memory access port is d-byte wide
   - d pixels can be accessed every clock cycle
4. n (θ, t_x, t_y) tuples are processed in parallel
   - Use forward transformation
5. All designs are fully pipelined



Calculate the Image Registration Time in Hardware

Three steps to test one (θ,t_x,t_y) tuple:

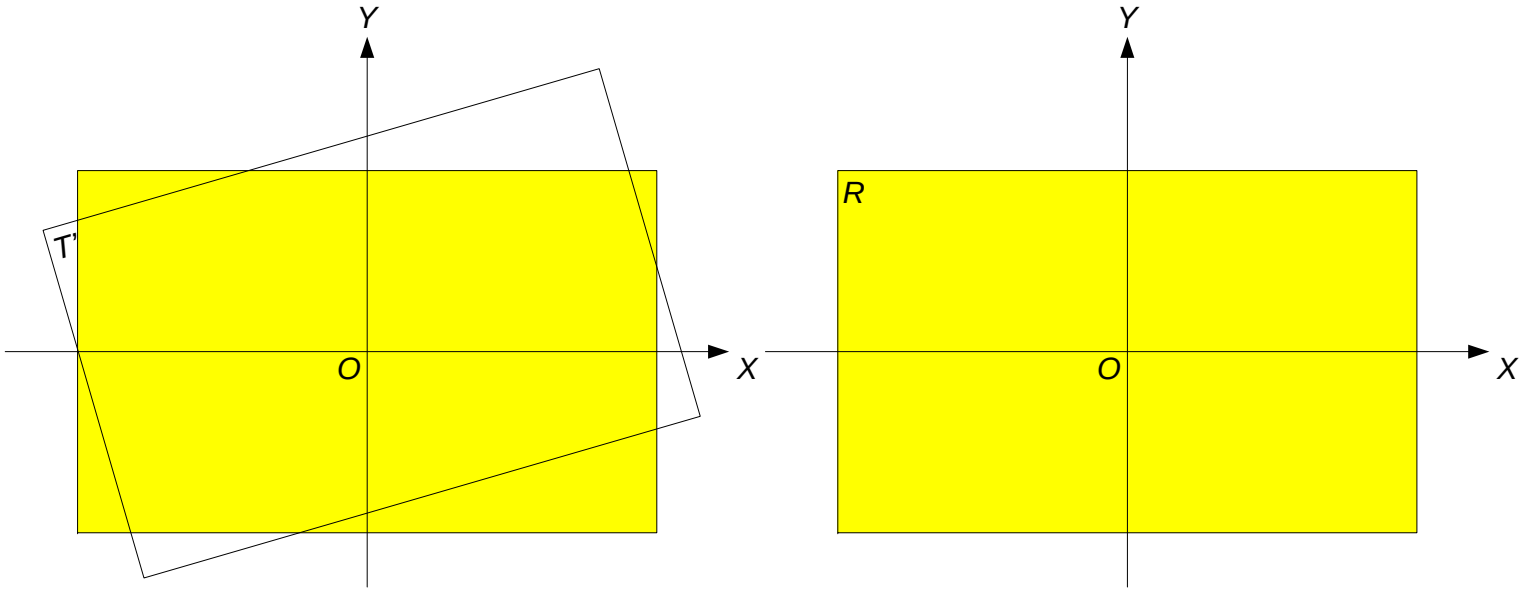
1. Initialize the memory region T' to be zero
2. Apply rigid-body transformation on T to get T'

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

3. Calculate correlation coefficient between reference image R and transformed image T'

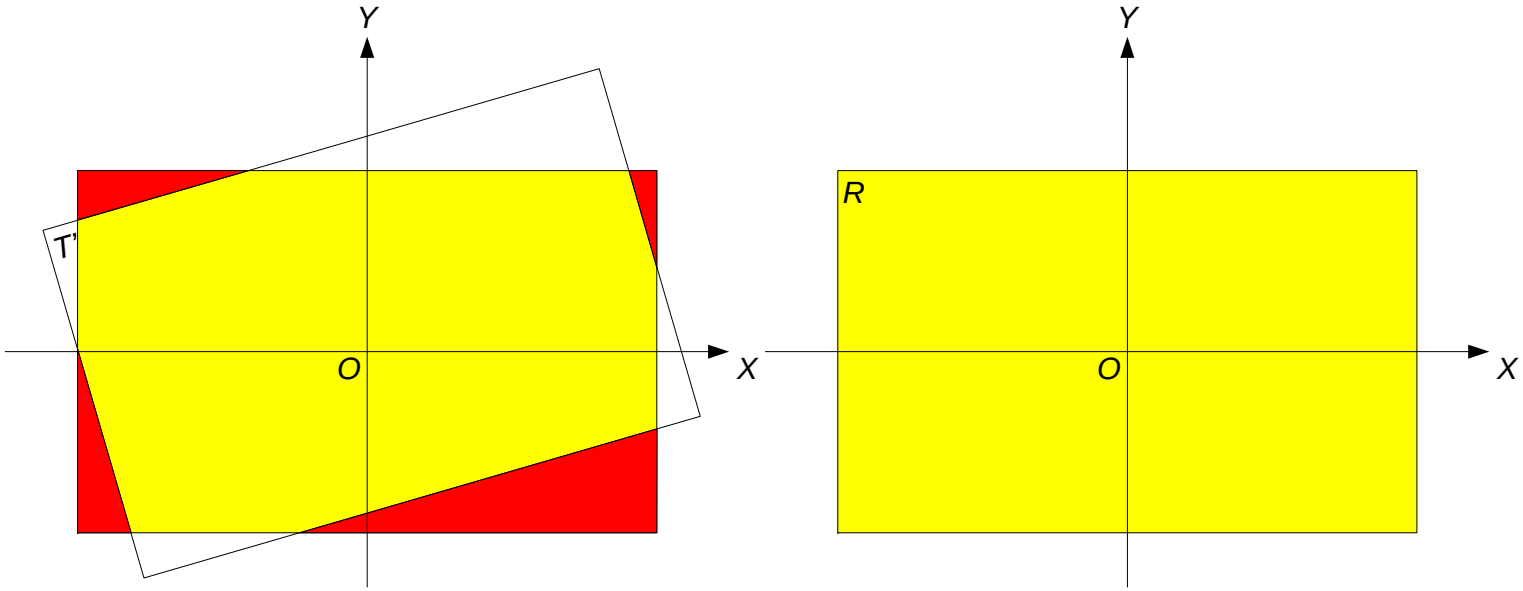
$$\frac{\sum_{x,y}(R(x,y)-\mu_R)(T'(x,y)-\mu_{T'})}{\sqrt{\sum_{x,y}(R(x,y)-\mu_R)^2 \sum_{x,y}(T'(x,y)-\mu_{T'})^2}}$$

Step 1: Initialize the memory region T' to be zero



Two reasons for initializing T':

Step 1: Initialize the memory region T' to be zero



Two reasons for initializing T':

- **Several regions (shown in red) may not covered by T'**

Step 1: Initialize the memory region T' to be zero

Two reasons for initializing T':

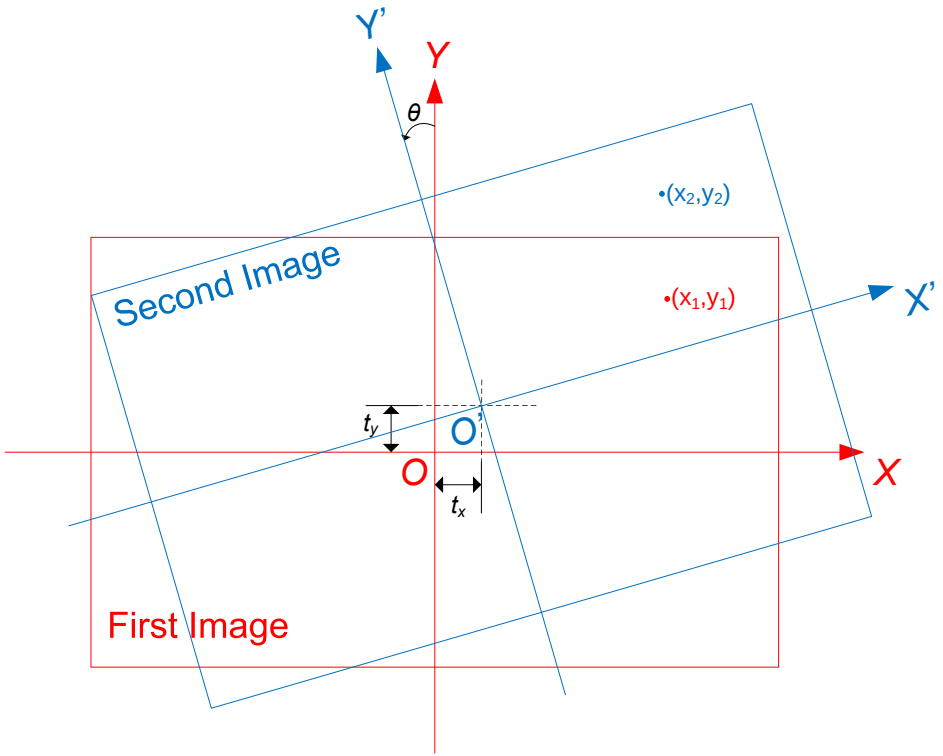
- Several regions (shown in red) may not covered by T'
- **Artifacts due to discretization in forward transformation (shown as dark spots in the transformed image)**

Step 1: Initialize the memory region T' to be zero

Two reasons for initializing T':

- Several regions (shown in red) may not covered by T'
- Artifacts due to discretization in forward transformation (shown as dark spots in the transformed image)
- d **pixels can be initialized to zero every clock cycle**
  - $\frac{m}{d}$ **clock cycles are needed**

Step 2: Apply rigid-body transformation on T to get T'



$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

- Start calculating $(x_1, y_1) \rightarrow (x_2, y_2)$ every clock cycle
  - *m* clock cycles required
- Calculate the mean of image T', $\mu_{T'}$, in the Step 2

Step 3: Calculate correlation coefficient between R and T'

$$\frac{\sum_{x,y}(R(x,y)-\mu_R)(T'(x,y)-\mu_{T'})}{\sqrt{\sum_{x,y}(R(x,y)-\mu_R)^2\sum_{x,y}(T'(x,y)-\mu_{T'})^2}}$$

- Start the processing of d pixels every clock cycle
  - $\frac{m}{d}$ clock cycles are required
- Compute $\sum_{x,y}(R(x,y)-\mu_R)(T'(x,y)-\mu_{T'})$, $\sum_{x,y}(R(x,y)-\mu_R)^2$, and $\sum_{x,y}(T'(x,y)-\mu_{T'})^2$ at the same time
  - μ_R is precomputed by μP and then forwarded to FPGA
  - $\mu_{T'}$ is computed in Step 2

Hardware Computation Time using Exhaustive Search Algorithm

Given

- Search space: $(\Delta\Theta, \Delta X, \Delta Y)$
- Both reference image and test image are in 8-bit gray-scale and consist of m pixels
- $n+2$ local memory modules
  - The access ports are d-byte wide

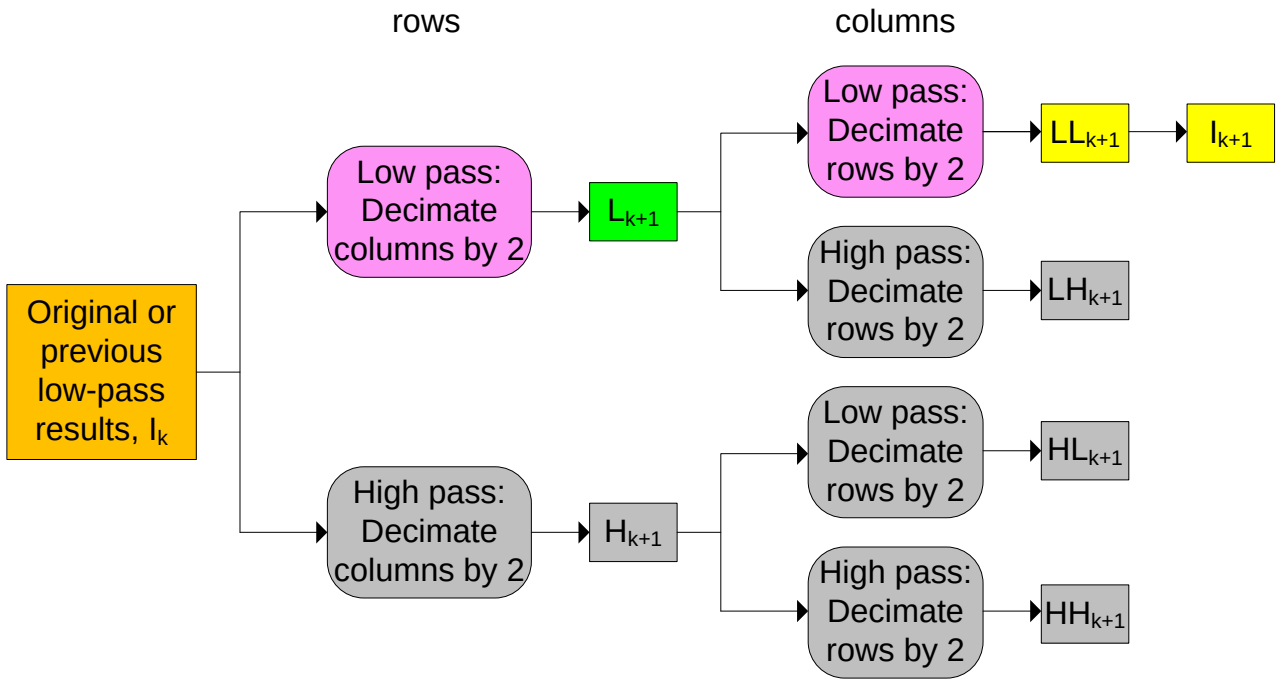
Computation Time

- Each tuple (θ, t_x, t_y) takes approximately $\frac{d+2}{d}m$ clock cycles
- The search of overall space takes approximately

$$\frac{\Delta\Theta \cdot \Delta X \cdot \Delta Y \cdot (d+2)}{n \cdot d} m \quad \text{clock cycles}$$

Outline

Reduce Computation Demand through DWT Decomposition

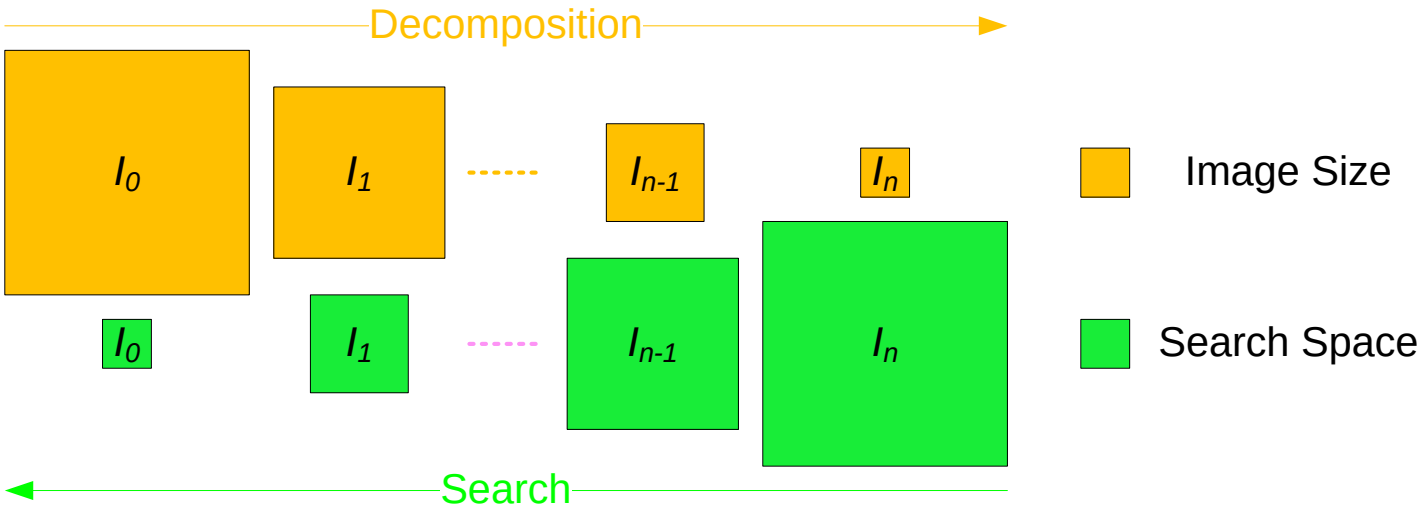


	I_0	I_1	$\cdots$	I_n
Size	$H \times W$	$\frac{H}{2} \times \frac{W}{2}$	$\cdots$	$\frac{H}{2^n} \times \frac{W}{2^n}$
Search Resolution	$\delta_\theta, \delta_x, \delta_y$	$2\delta_\theta, 2\delta_x, 2\delta_y$	$\cdots$	$2^n \cdot \delta_\theta, 2^n \cdot \delta_x, 2^n \cdot \delta_y$

[Moigne 2002] J. L. Moigne, W. J. Campbell, and R. F. Cromp, "An automated parallel image registration technique based on the correlation of wavelet features," *IEEE Trans. Geosci. Remote Sensing*, vol. 40, no. 8, pp. 1849–1864, Aug. 2002.
[El-Ghazawi 1997] T. A. El-Ghazawi, P. Chalermwat, and J. L. Moigne, "Wavelet-based image registration on parallel computers," in *Proc. ACM/IEEE 1997 Supercomputing Conference (SC'97)*, Nov. 1997, pp. 20–28.

Search Strategy Summary

Decomposition Level	Search Space	Search Resolution	Result
n	$\Delta\Theta$	$2^n * \delta_\theta$	θ_n
$n-1$	$[\theta_n - 2^n * \delta_\theta; \theta_n + 2^n * \delta_\theta]$	$2^{n-1} * \delta_\theta$	θ_{n-1}
$n-2$	$[\theta_{n-1} - 2^{n-1} * \delta_\theta; \theta_{n-1} + 2^{n-1} * \delta_\theta]$	$2^{n-2} * \delta_\theta$	θ_{n-2}
$\cdots$	$\cdots$	$\cdots$	$\cdots$
2	$[\theta_3 - 2^3 * \delta_\theta; \theta_3 + 2^3 * \delta_\theta]$	$2^2 * \delta_\theta$	θ_2
1	$[\theta_2 - 2^2 * \delta_\theta; \theta_2 + 2^2 * \delta_\theta]$	$2 * \delta_\theta$	θ_1
0	$[\theta_1 - 2 * \delta_\theta; \theta_1 + 2 * \delta_\theta]$	δ_θ	θ_0



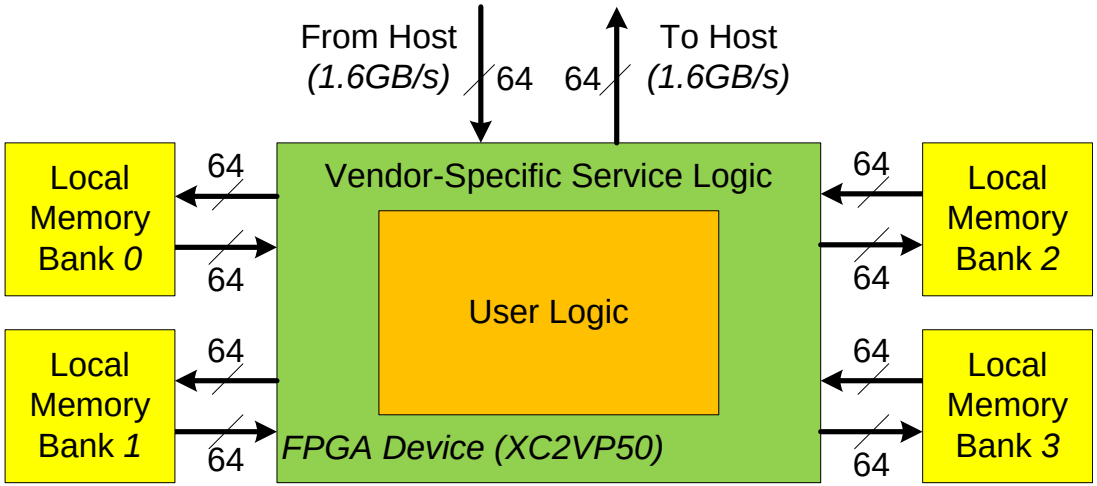
Decrease the search space and increase the search resolution in the search process

Outline

Testbed: Cray XD1 Reconfigurable Computer

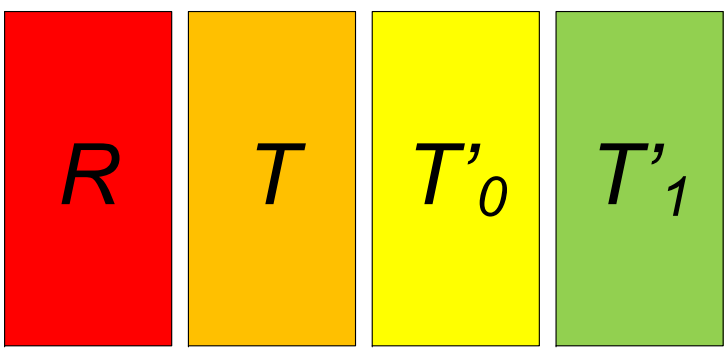
Specifications:

- FPGA device: Xilinx Virtex-II P50
- 4 local SRAM banks, 4 MBytes each
- Maximum user logic clock rate: 200 MHz

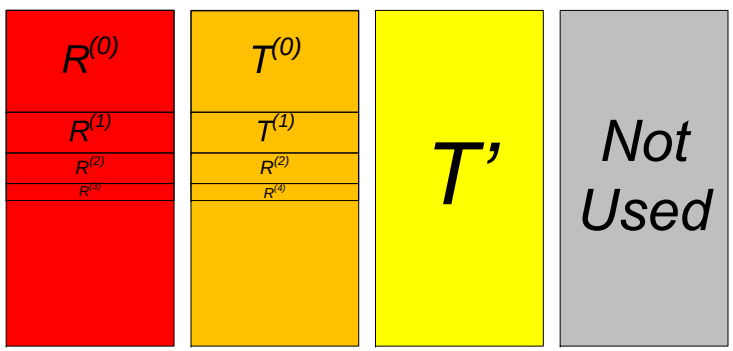


Memory usage of two different approaches:

Exhaustive Search



DWT-based Search



Experimental Results

Algorithm Name	Computing Time (s)			Resource Utilization	
	μP	Cray XD1	Speedup	Slices	MULT. Blocks
Exhaustive Search	157.347	16.193	9.72	10,766 (45%)	42 (18%)
DWT-based Search	1.298	0.829	1.57	20,205 (85%)	108 (55%)

- The reference and test image are both 1024$\times$1024 pixels and in 8-bit gray-scale
- The search spaces of $\Delta\Theta$,ΔX and ΔY are all from -8 to +8
- Three levels of DWT decomposition, which is based on LL coefficients, are performed in the DWT-based search
- The referenced μP is AMD Opteron 2.4 GHz
- The hardware computation time is end-to-end time
  - Data transportation time + data processing time
- *The hardware implementation was developed using Verilog HDL and VHDL; Xilinx ISE 8.1 was used for synthesis and p&r*

Outline

Conclusions

- Two related image registration algorithms based on rigid-body transformation are presented
  - For exhaustive search algorithm, the performance is linearly proportional to the available number of local memory banks
  - For DWT-based search algorithm, the efficiency is improved by applying DWT decomposition on both reference and test images to reduce the search scope
- Compared to software implementation on AMD Opteron 2.4 GHz, experimental results on Cray XD1 show
  - Exhaustive search algorithm: 10$\times$ speedup
  - DWT-based search algorithm: 2$\times$ speedup
    - Can be improved further by increasing the parallelism in hardware implementation