3D Object Reconstruction using DaVinci DSP

Raja Giryes

Advisors:

Alex Bronstein, Yair Moshe

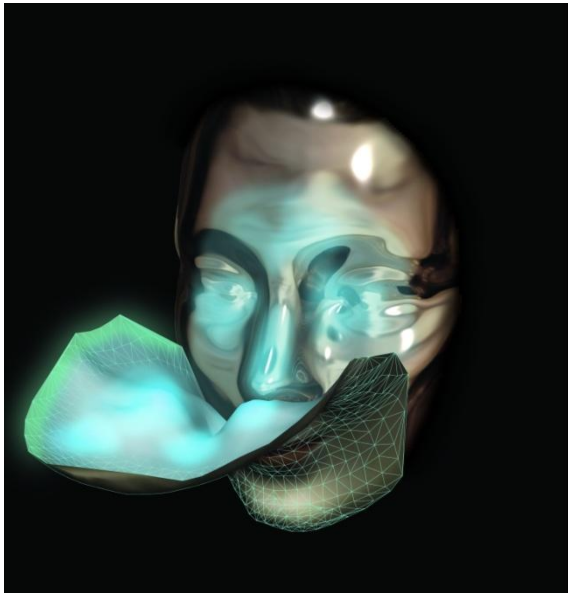
In association with GIP

Outline

- Background
- Project goal
- 3D reconstruction
- DaVinci platform
- Adapting the algorithm to the DaVinci
- Results
- Conclusion

Background – 3D Scanners

- A 3D scanner acquires an image in which every pixel has 3 coordinates
- Used for:
  - Biometric recognition (face)
  - Medical uses
  - Comparison of 3D surfaces
  - Architecture and civil engineering
  - Virtual reality
  - And more…



Background - 3D Scanning

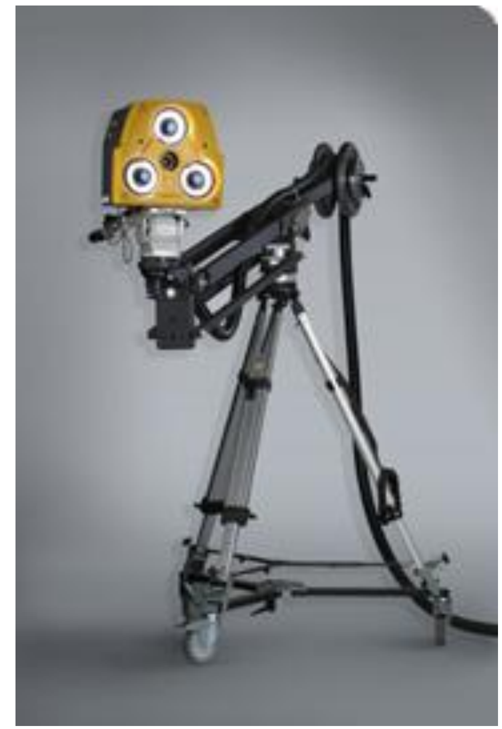
- Passive scanning
- Active scanning

Laser scanning

3D digital Escan

Time of light scanning

Leica ScanStation 2



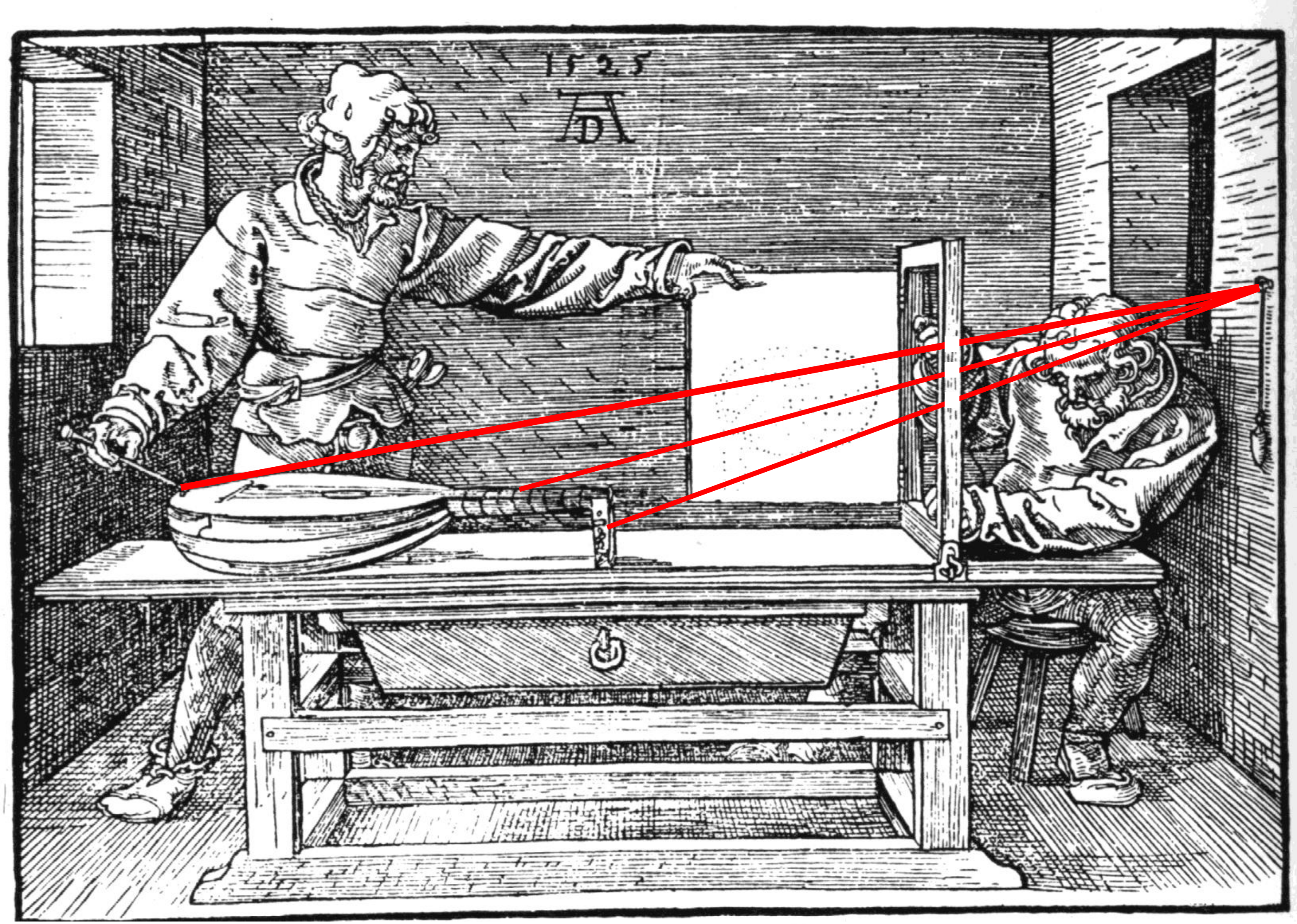
Structured light scanning

Cognitens optigo200

Project Goal

- Realtime implementation of 3D object reconstruction using the DaVinci platform
  - Structured light scanning
  - Using one standard camera and one standard projector
  - Low cost solution

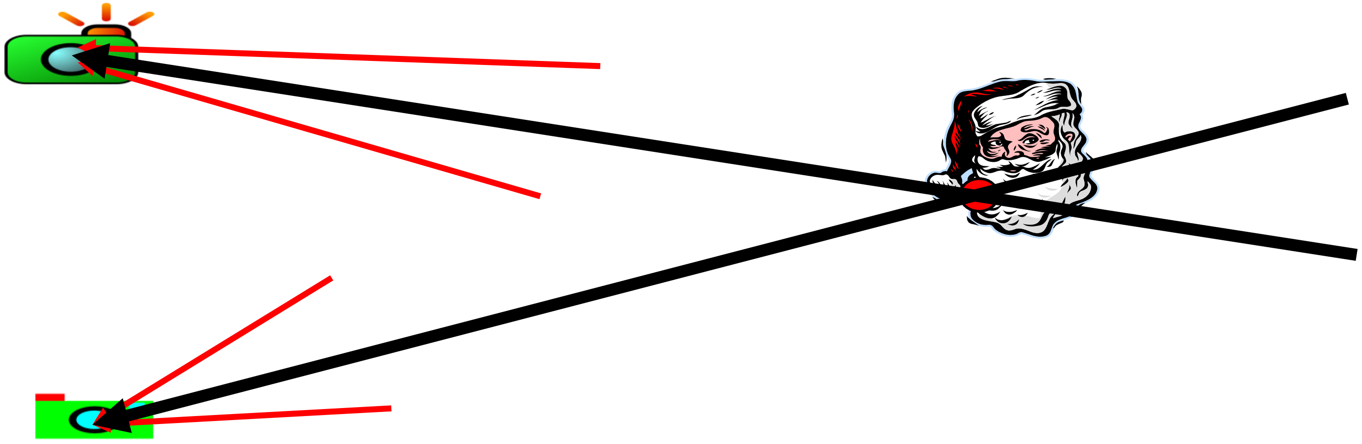
Perspective projection



The Draughtsman of the Lute, Albrecht Dürer

Image Reconstruction

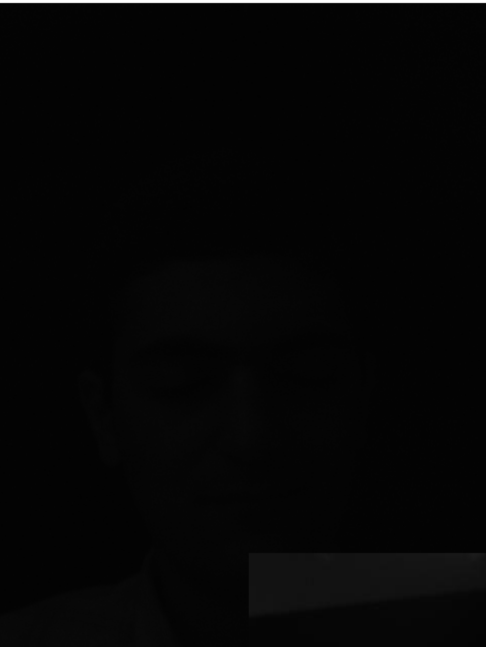
- Given 2 cameras
- If we identify a point in both of them we can know its coordinates (if the lines that goes into the camera are not parallel)



structured light method

- Using ‘active’ projector and a camera
- Projector cast light code on the object
- From the projected plane of the projector and the captured images in the camera we can make the reconstruction

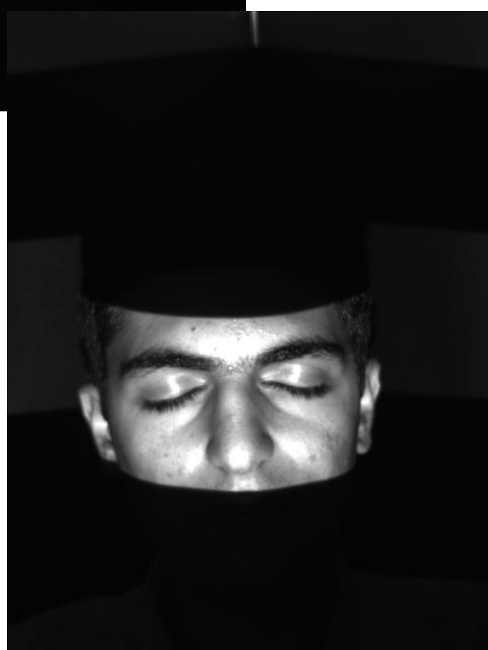
Structured Light Method



full darkness image

scanned pictures

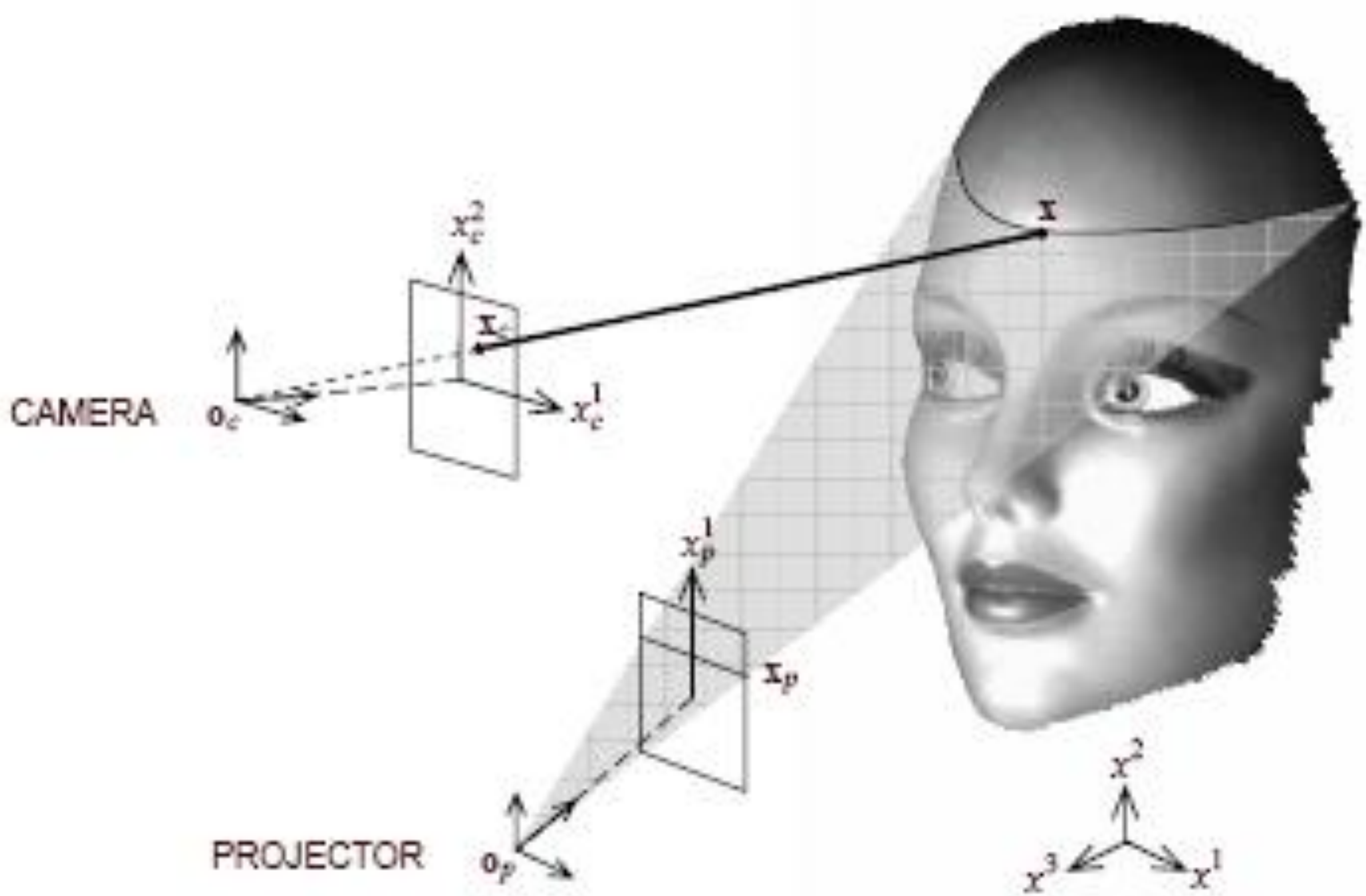
full illumination image



Bit 3

bit 6

bit 9


CAMERA
$\mathbf{o}_c$
x_c^2
x_c^1
$\mathbf{x}_c$
$\mathbf{x}$
x_p^1
$\mathbf{x}_p$
PROJECTOR
$\mathbf{o}_p$
x^2
x^3
x^1

Reconstruction

PPM world to camera

$$X_c = C_c X_w \qquad C_c = \alpha \begin{bmatrix} f_x & kf_y & x_c^0 \\ 0 & f_y & y_c^0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_c & t_c \end{bmatrix}.$$

PPM world to projector

$$X_p = C_p X_w \qquad C_p = \alpha \begin{bmatrix} f_p & 0 & x_p^0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_p & t_p \end{bmatrix}.$$

[1] A. M. Bronstein, M. M. Bronstein, E. Gordon, R. Kimmel, 2003.

Reconstruction 2

- We have $T : X_w \rightarrow \left(X_c, X_p \right)$
- But wants: $T^{-1} : \left(X_c, X_p \right) \rightarrow X_w$
- We can get that:

$$x_w = -R^{-1} s$$

- When:
- $$\left[R, s \right] = \begin{bmatrix} x_c c_3 - c_1 \\ y_c c_3 - c_2 \\ x_p p_2 - p_1 \end{bmatrix}$$

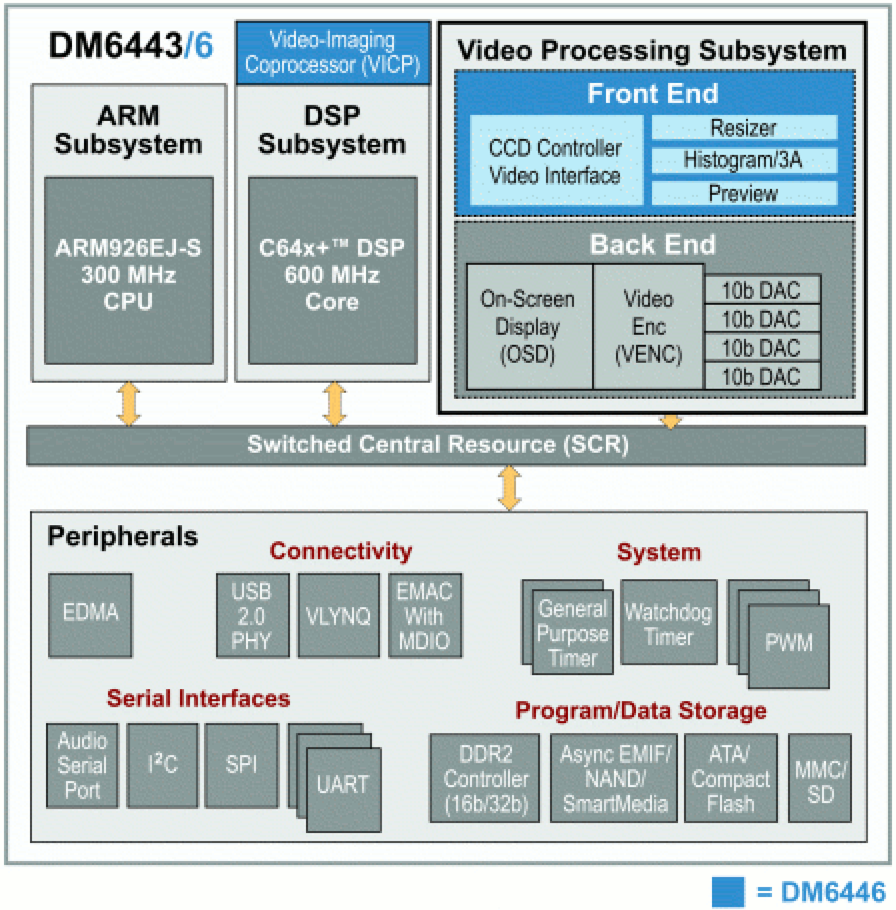
c_i is the row i in the C matrix and p_i is row i in the P matrix

Previous implementation

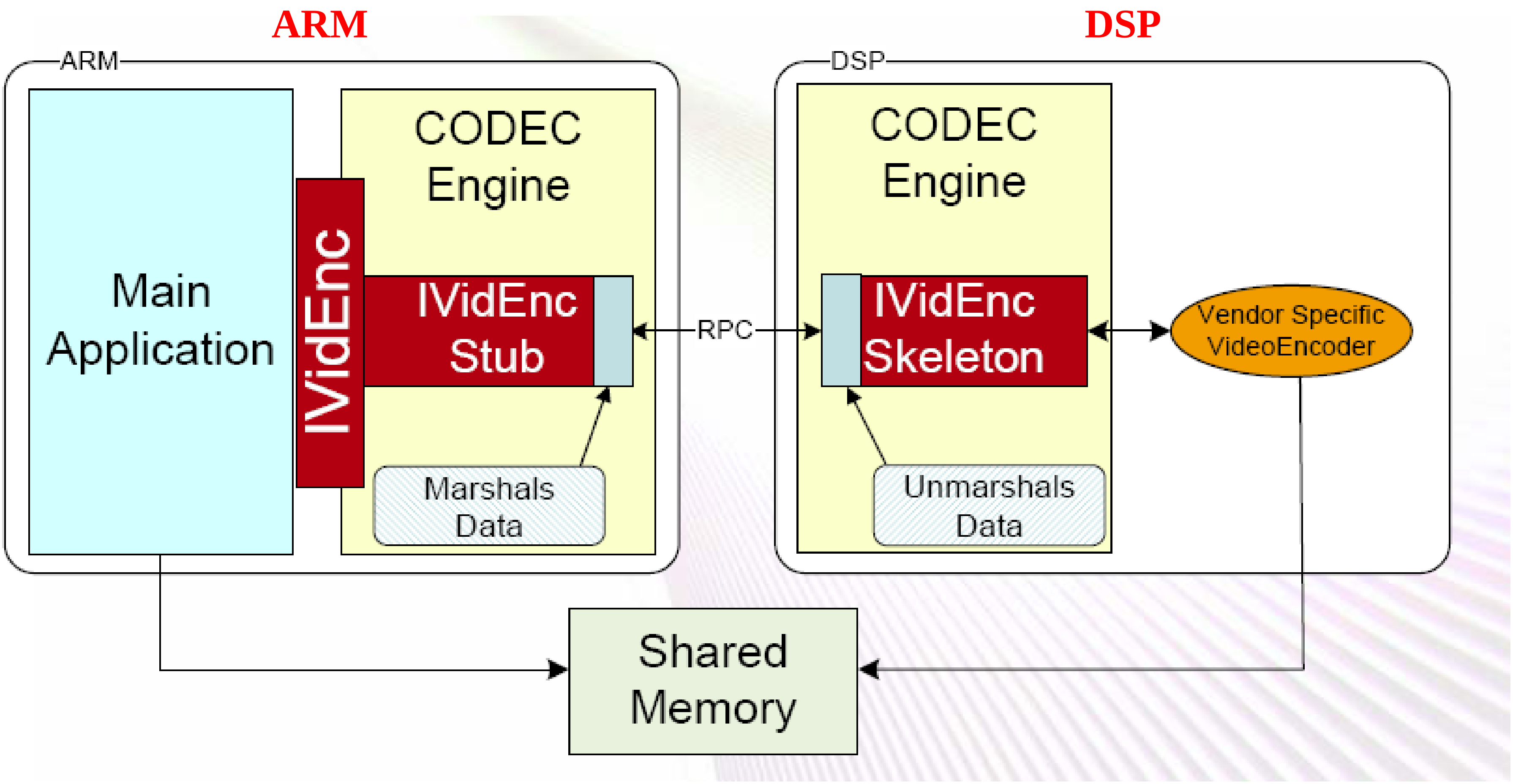
- FireWire black and white CCD camera
- computer-controlled DLP projector
- 10 binary stripe images and additional full dark and full illumination images
- Code implemented in C language on PC
- 3 3D images per second
- Reconstruction using highly optimized Pentium IV code takes 280ms

DaVinci platform

- 2 Cores:
  - ARM9
  - C64x+ DSP
- Memory
  - On-Chip L1/SRAM -112KB DSP, 40 KB ARM
  - On-Chip L2/SRAM – 64KB DSP
- TI's solution for Video processing.
- contains large set of compatible software



DaVinci xDM



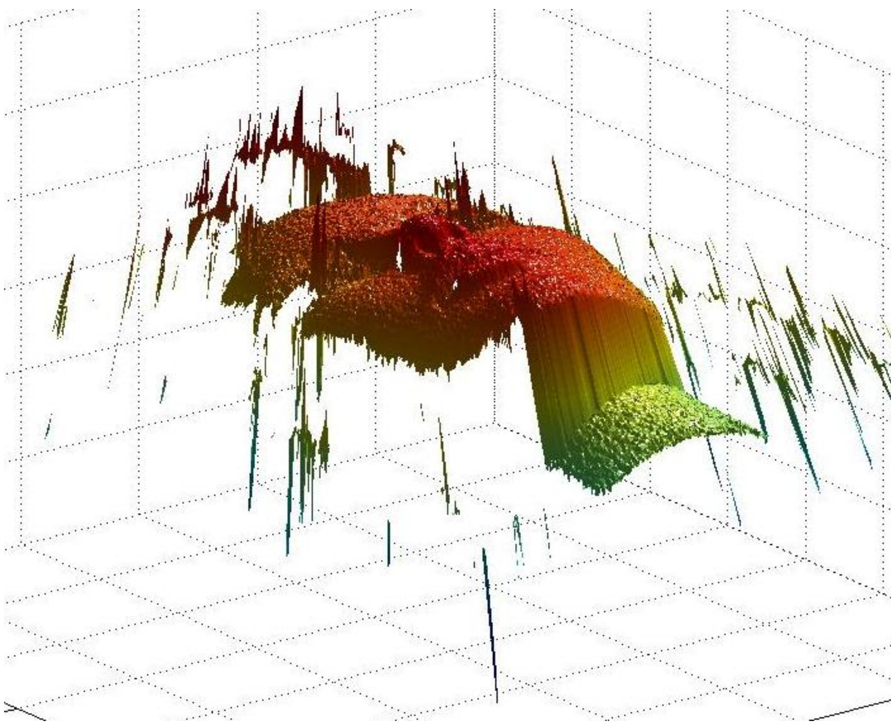
ARM
DSP
ARM
Main Application
IVidEnc
CODEC Engine
IVidEnc Stub
Marshals Data
RPC
DSP
CODEC Engine
IVidEnc Skeleton
Unmarshals Data
Vendor Specific VideoEncoder
Shared Memory

Adapting to the DaVinci

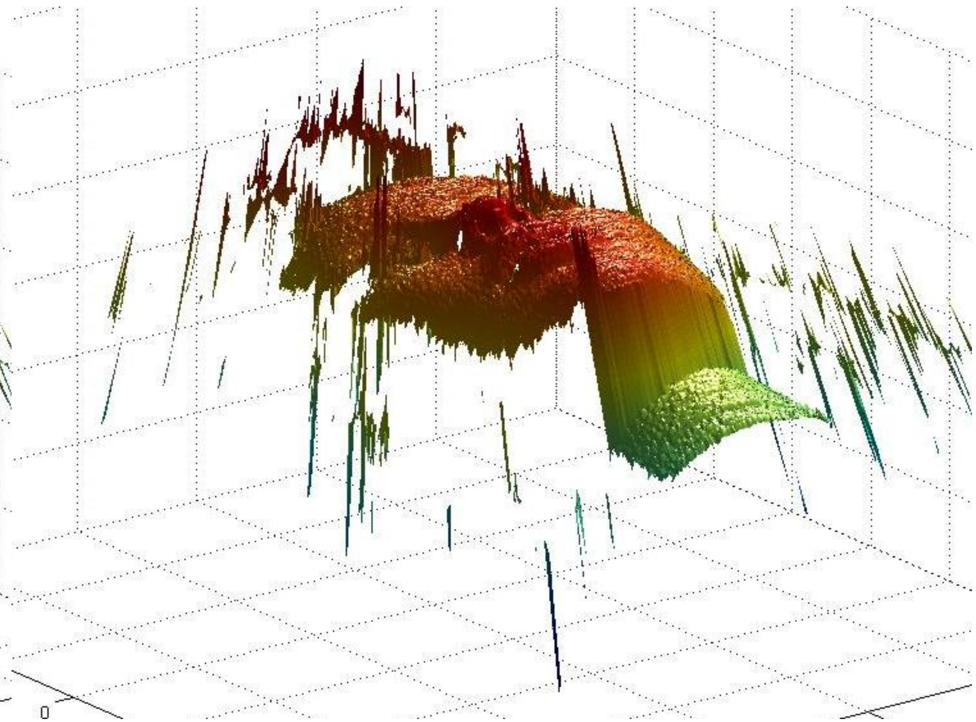
- Floating point to fixed point
  - Changing output format to homogenous coordinates
  - Fine-tuned scaling of all the values
  - Changing filter sizes for efficient division
  - Changing functions to look-up-tables
  - ….
- 1 core to 2 cores
  - Using TI's xDM standard and VISA interface for inter-core communication
  - ARM connects to peripherals and feeds the DSP
  - DSP makes the reconstruction and output the results back to the ARM

Reconstruction Results

Fixed point version



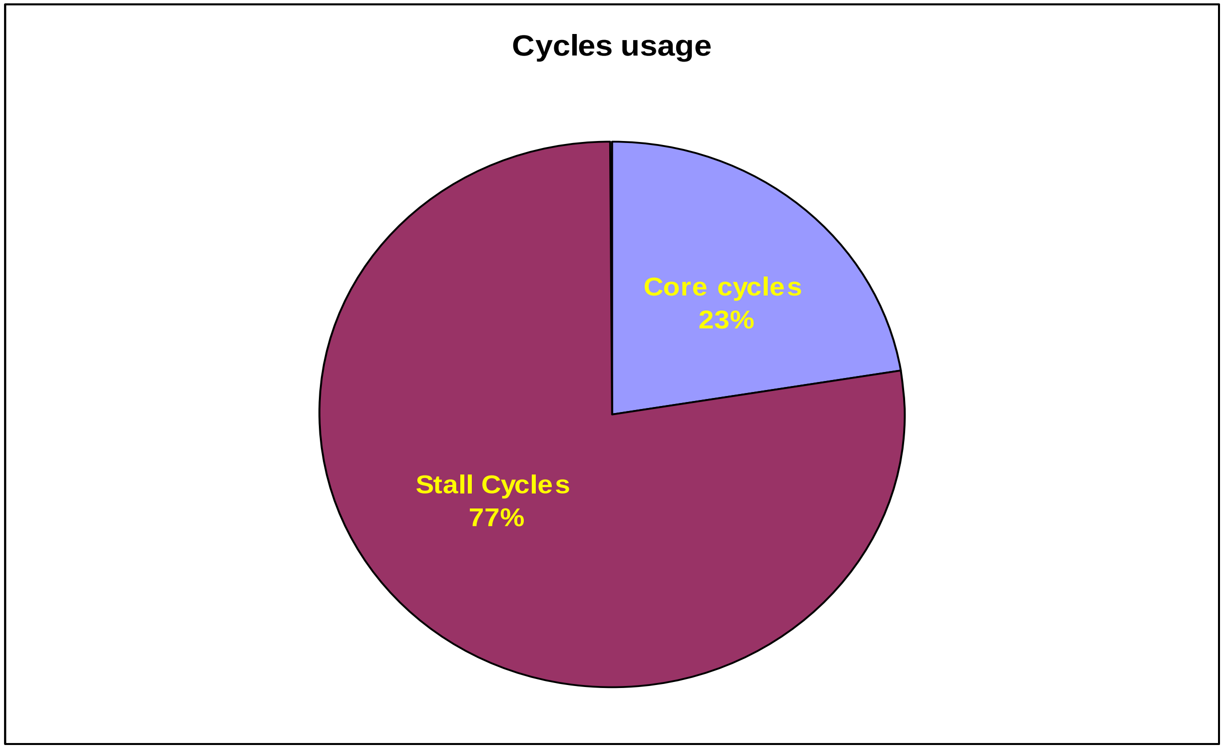
floating point version



RMS = 0.0271

Non-optimized Time Performance

- Reconstruction takes 2.5 sec



What’s next

- Adding the camera and the projector and controlling them using the ARM
  - In progress
- Using the DMA controller for more efficient use of memory
- More code optimization
- Adding the calibration part

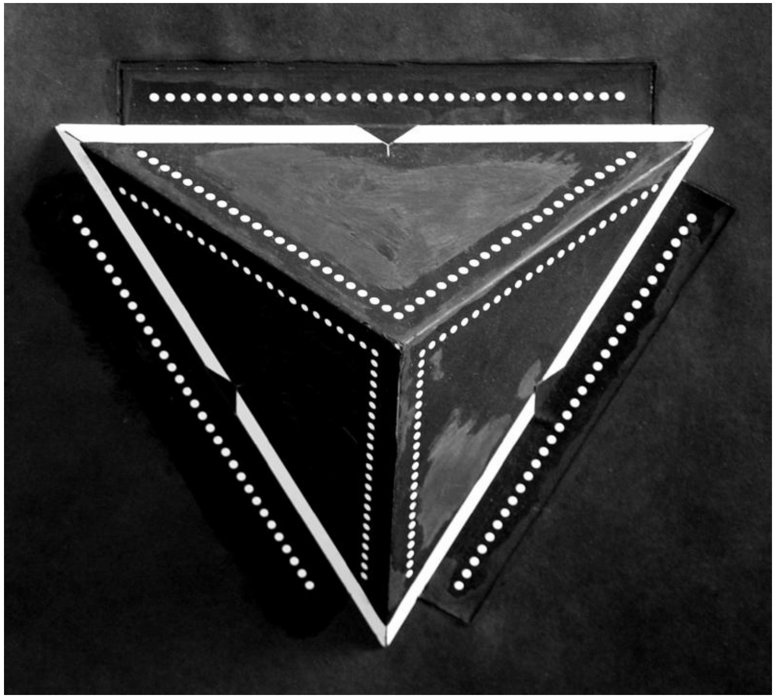
Thank you

Active Stereo techniques

- Gray level multiplexed
- Color multiplexed
- Space multiplexed
- Time multiplexed (we use this)

Calibration

- Known world coordinates
- Building C_c and C_p
- The calibration object:



Calibration 2

- Solving $$\begin{aligned}(x_c)_k &= \mathrm{C}_c(x_w)_k \\ (x_p)_k &= \mathrm{C}_p(x_w)_k,\end{aligned}$$
- No accurate solution due to finite precision
- Instead we will solve:

$$\mathrm{C}_c = \operatorname{argmin} \sum_{k=1}^{N} \|\mathrm{C}_c(x_w)_k - (x_c)_k\|_2^2 \quad \text{s.t.} \quad \mathrm{C}_c \in \mathrm{PPM}$$

$$\mathrm{C}_p = \operatorname{argmin} \sum_{k=1}^{N} \|\mathrm{C}_p(x_w)_k - (x_p)_k\|_2^2 \quad \text{s.t.} \quad \mathrm{C}_p \in \mathrm{PPM}.$$

- But actually we are interested in T^{-1} error:

$$\mathrm{T} = \operatorname{argmin} \sum_{k=1}^{N} \left\|\mathrm{T}^{-1}(x_c, x_p)_k - (x_w)_k\right\|_2^2 \quad s.t. \quad \mathrm{C}_c, \mathrm{C}_p \in textPPM.$$

Technical details

	C64x+	ARM9
Peak MMACS	4752	
Freq	594	297
Memory on chip (KB)	16 (ROM) 40 (L1/SRAM)	112 (L1/SRAM) 64 (L2/SRAM)