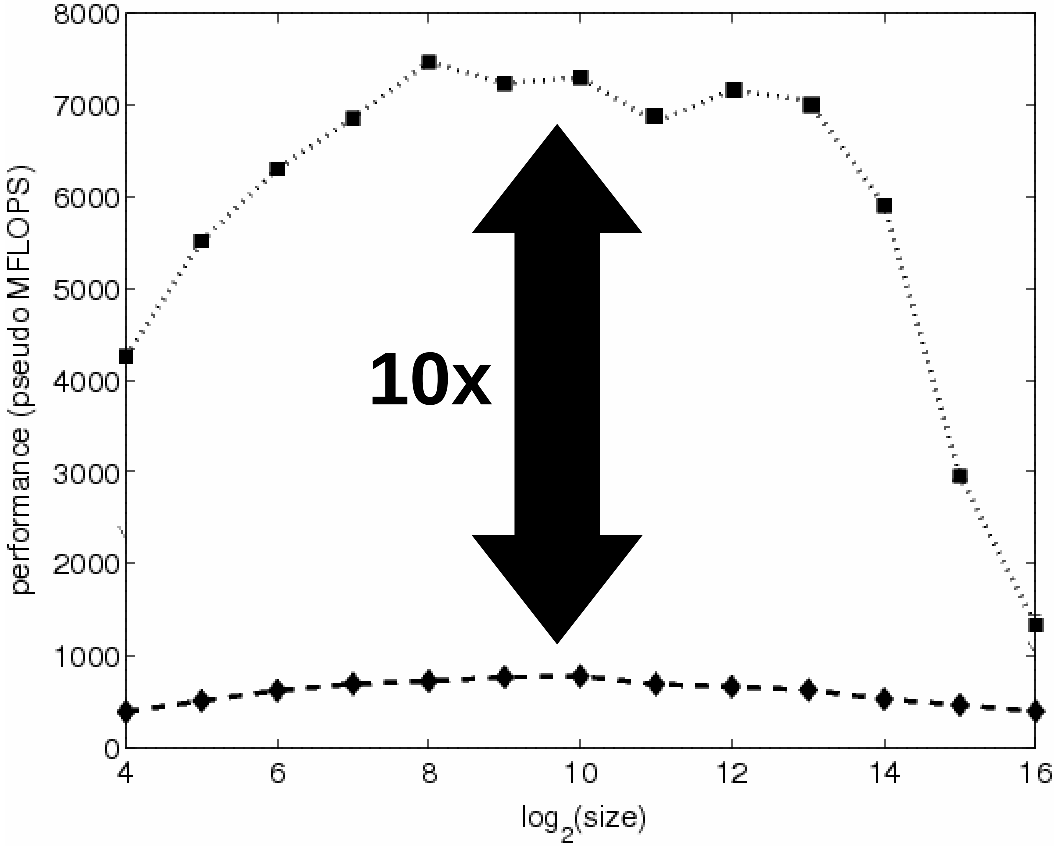
Algorithms and Computation in Signal Processing

special topic course 18-799B
spring 2005
1st Lecture Jan. 11, 2005

Instructor: Markus Pueschel
TA: Srinivas Chellappa

Motivation and Idea behind this Course

The Problem: Example DFT on Pentium 4



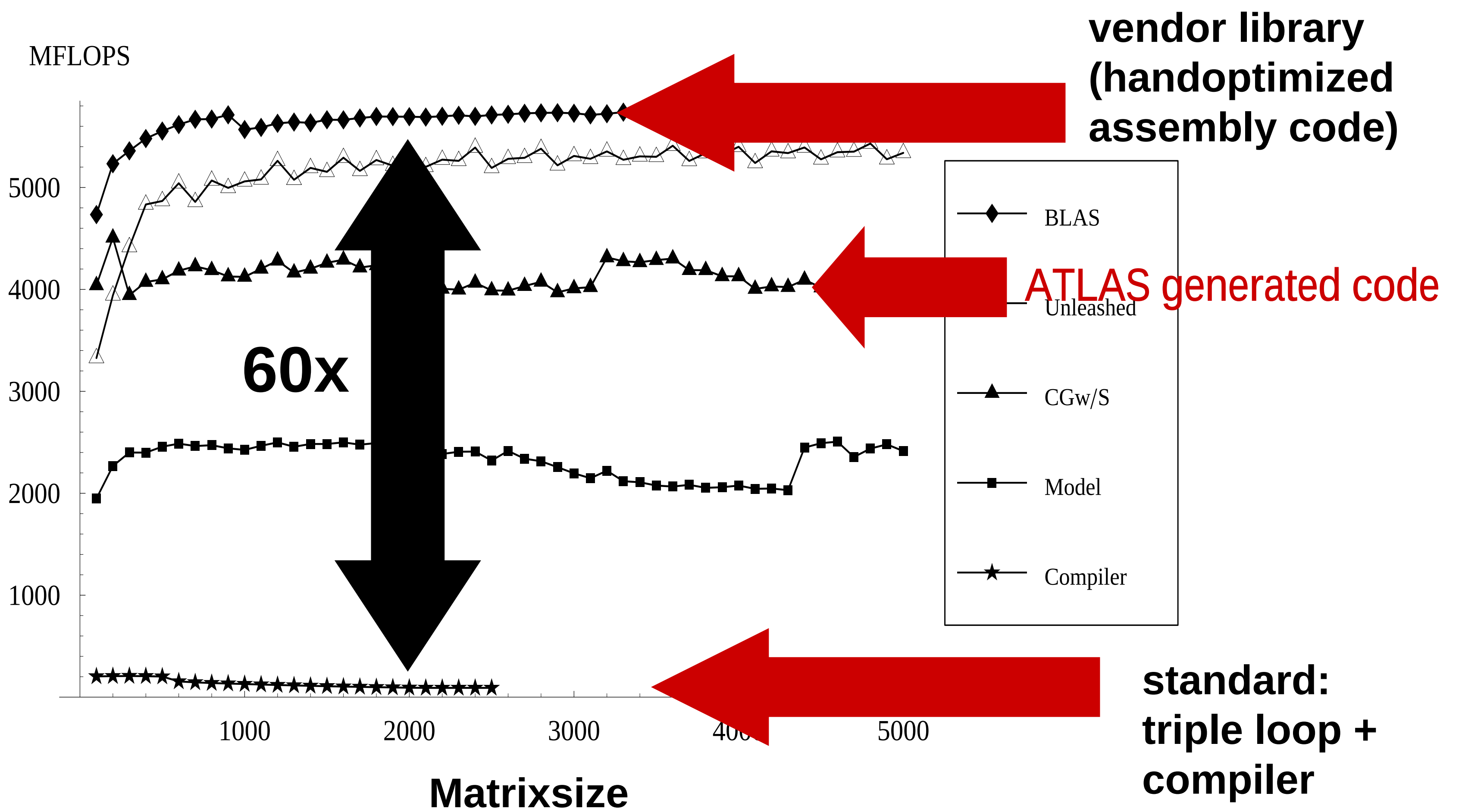
DFT size

Intel vendor library (hand-optimized assembly code)
but also FFTW, SPIRAL generated code

reasonable implementation (Numerical recipes. GNU scientific library)

Ok, but the DFT is kind of complicated, so let's take something simpler …

The Problem: Matrix-matrix Multiplication

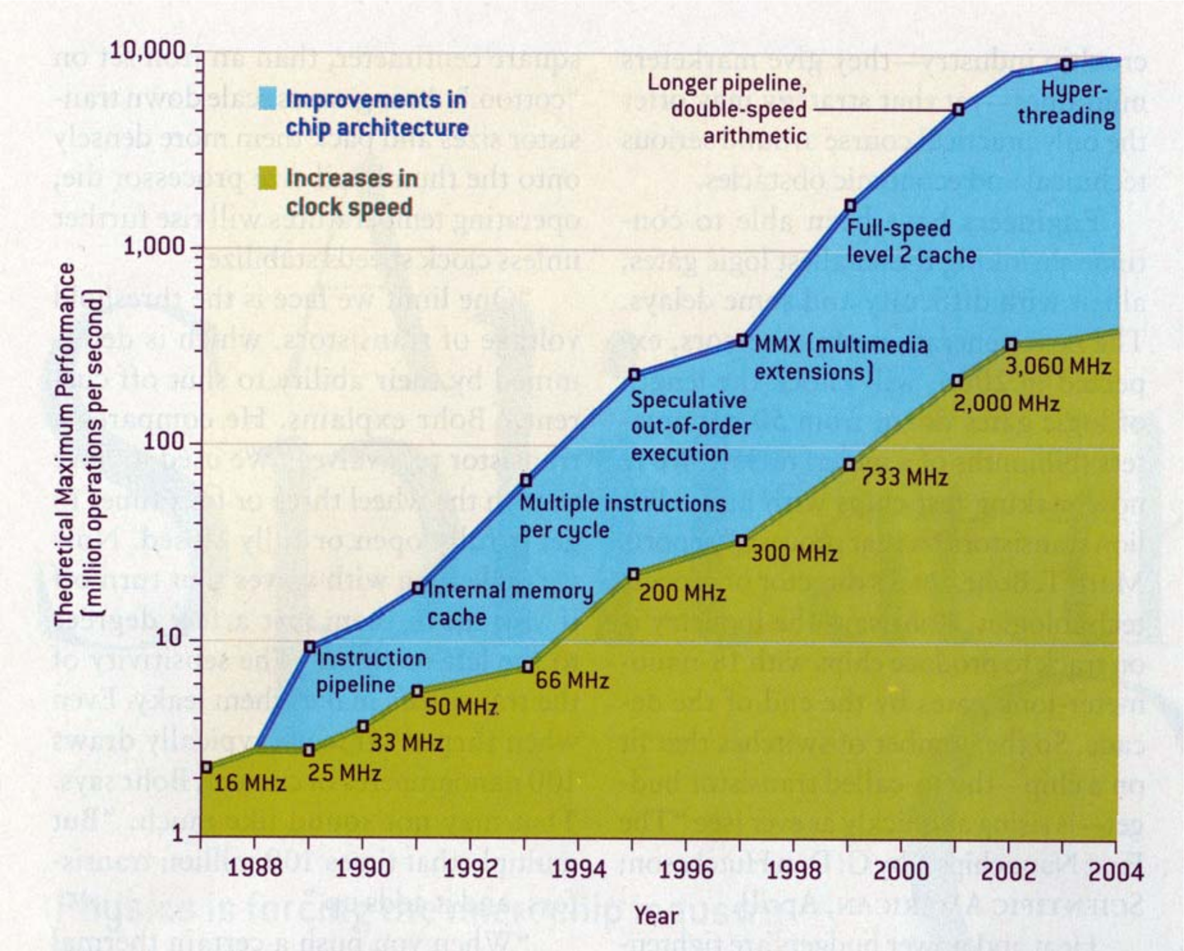


Why is that?

graph: Pingali, Yotov, Cornell U.

Moore's Law

- Moore's Law: exponential (x2 in ~18 months) increase number of transistors/chip



source: Scientific American, Nov 2004, p. 98

But everything has its price …

Moore's Law: Consequences

- Computers are very complex
  - multilevel memory hierarchy
  - special instruction sets beyond standard C programming model
  - undocumented hardware optimizations
- Consequences:
  - Runtime depends only roughly on the operations
  - Runtime behavior is hard to understand
  - Compiler development can hardly keep track
  - The best code (and algorithm) is platform-dependent
  - It is very difficult to write really fast code
- Computers evolve fast
  - Highly tuned code becomes obsolete almost as fast as it as written

What about the Future?

- It gets rather worse:
  End of Moore's Law and proliferation of multicore systems
  - Scientific American, Nov. 2004: "A Split at the Core,"
    subtitle: "[…] that is bad news for the software companies"
  - Dr. Dobb's Journal, 30(3), March 2005: "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software"

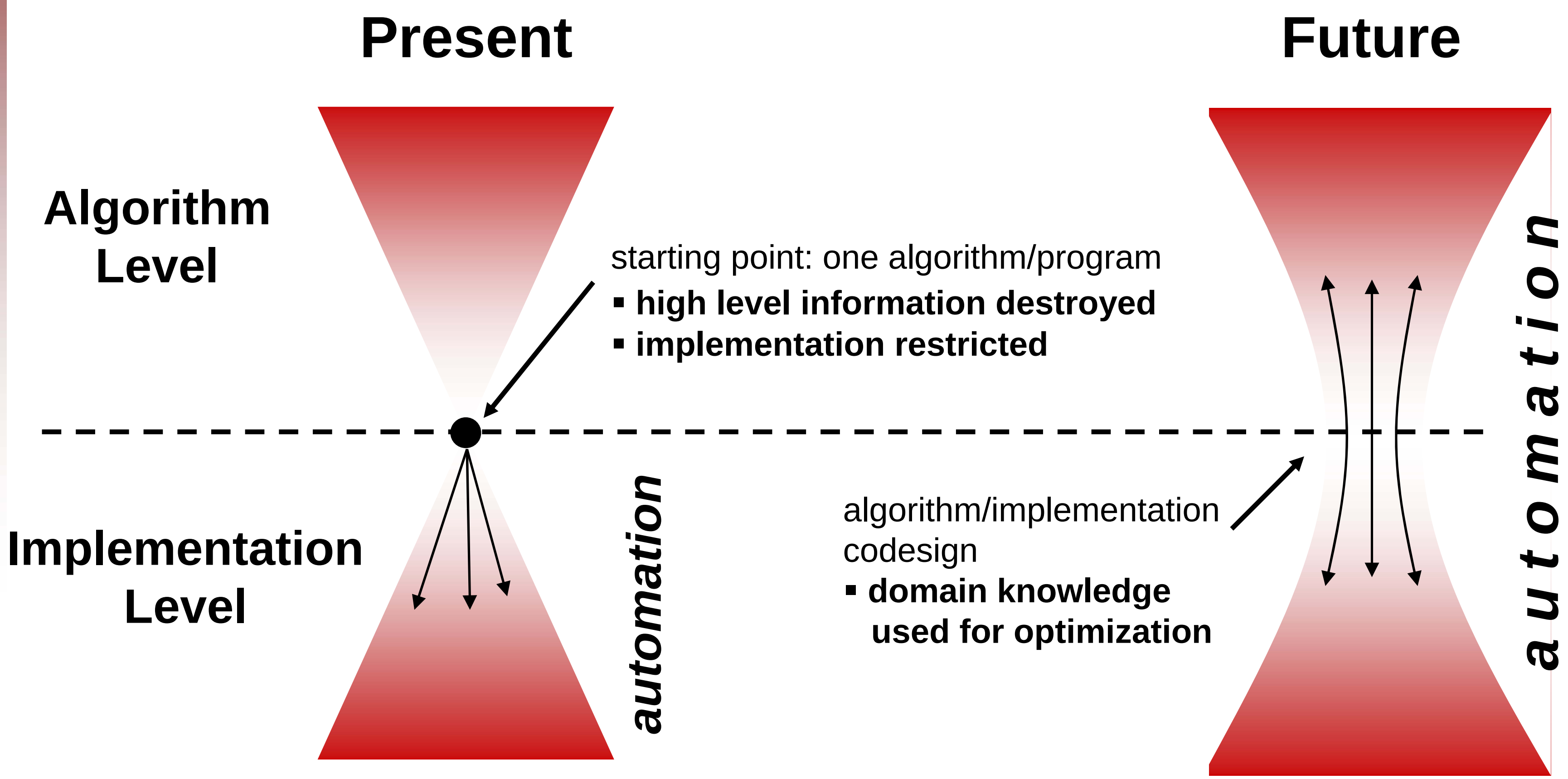
How to produce really fast code?
and with reasonable effort?

Current Research: New Approaches to Software

- Linear Algebra:
  - LAPACK, ATLAS
  - BeBOP
- Tensor Computations (Quantum Chemistry): Sadayappan, Baumgartner et al. (Ohio State)
- Finite Element Methods: Fenics (U. Chicago)
- Signal Processing:
  - FFTW
  - SPIRAL
  - VSIPL (Kepner, Lebak et al., MIT Lincoln Labs)
- New Compiler Techniques (Domain aware/specific):
  - Model-based ATLAS
  - Broadway (Lin, Guyer, U. Texas Austin)
  - SIMD optimizations (Ueberhuber, Univ. Techn. Vienna)
  - Telescoping Languages (Kennedy et al., Rice)

See also upcoming Proceedings of the IEEE special issue on "Program Generation, Optimization, and Adaptation," http://www.ece.cmu.edu/~spiral/special-issue.html

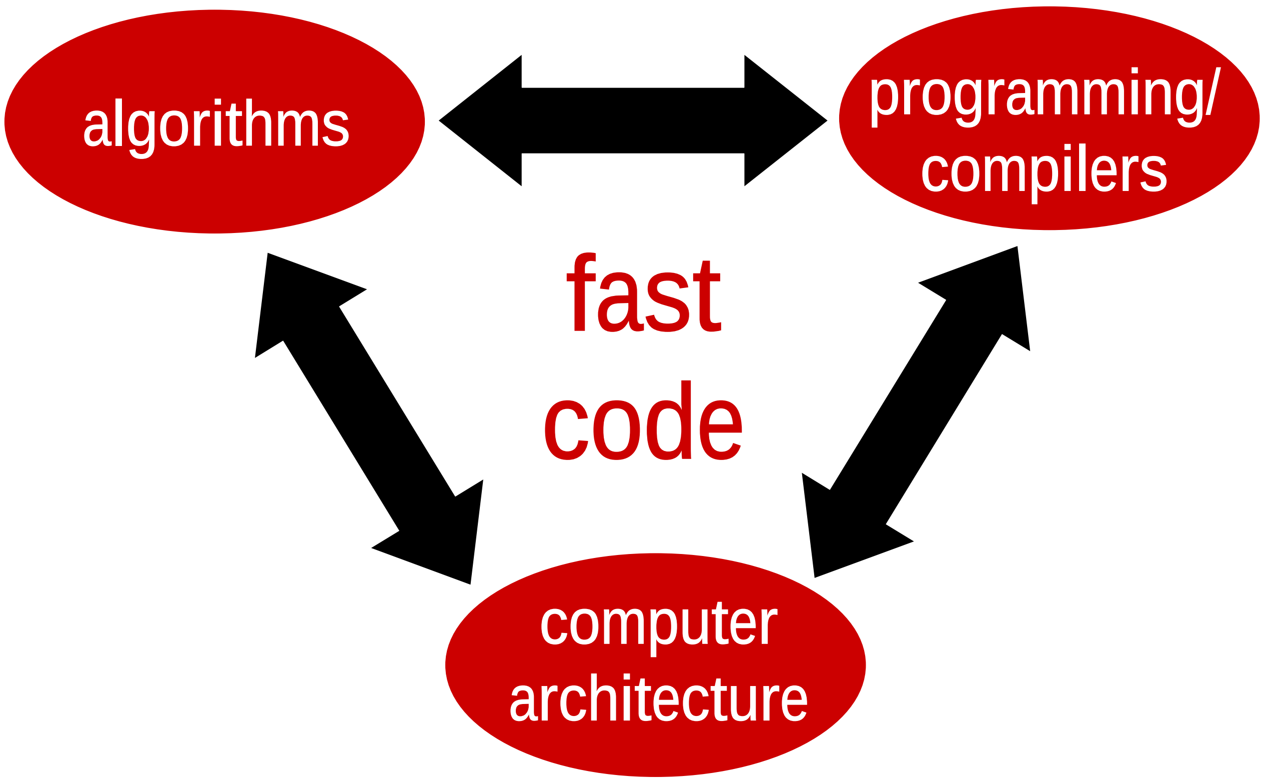
Possible Philosophy?



a new breed of domain-aware approaches/tools
push automation beyond what is currently possible
applies for software and hardware design alike

Idea of this Course

- Writing fast numerical code requires multidisciplinary knowledge of algorithms, programming/compilers, and computer architecture



- Study the interaction of algorithms, implementation, and architecture at hand of cutting edge adaptive numerical software
- Learn a guideline how to write fast code and apply it in a research project

Course Topics

- Foundations of algorithm analysis
  - cost and complexity, O-calculus, cost analysis through recurrences
- Computer architecture
  - architecture and microarchitecture, memory hierarchy/caches, execution units, special instruction sets (in particular, short vector instructions)
- Compilers
  - strengths, limitations, guidelines for use
- In detail: algorithms, complexity, and cutting edge adaptive software (extract design principles)
  - Discrete Fourier transform, other transforms, correlation, filters (FFTW, SPIRAL)
  - Matrix-matrix multiplication (ATLAS) and possibly other linear algebra functionality (LAPACK)
  - Sparse linear algebra (BeBOP)
  - other as time permits
  - work towards a guideline for writing fast numerical code
  - apply that guideline in your research project

About this Course

- Requirements
  - solid C programming skills
  - matrix algebra
  - senior or above

- Grading
  - 50% research project
  - 20% midterm
  - 20% homework
  - 10% class participation

- No textbook
- Office Hours: yet to be determined
- Website: www.ece.cmu.edu/~pueschel -> teaching -> 18-799B

Research Project

- Team up in pairs (preferably)
- Topic:
  Very fast, ideally adaptive, implementation of (or code generation for) a numerical problem
- End of January/early February:
  suggest to me a problem or I give you a problem
- Show "milestones" during semester
- Write 4 page standard conference paper (template will be provided)
- Give short presentation end of April

Midterm

- Mostly about algorithm analysis
- Some multiple-choice

Final Exam

- There is no final exam

Homework

- Exercises on algorithm analysis (Math)
- Implementation exercises
  - study the effect of program optimizations, use of compilers, use of special instructions, etc. (Writing C code + creating runtime/performance plots)
  - some templates will be provided
- Probably: More homework in the beginning, less in the end

Classes/Class Participation

- I'll start on time, duration ~1:30 (without break)
  - be on time, it's good style
- It is important to attend
  - many things I'll teach are not in books
  - I'll use part slides part blackboard
- Ask questions
- I will provide some anonymous feedback mechanism (maybe every 3-4 weeks)

Motivation from the Applications Side: Signal Processing

Definitions

- Definition: Signal Processing
  - [The discipline that is concerned with] the representation, transformation, and manipulation of signals and the information they contain (Oppenheim, Schafer 1999)
- Definition: Signal
  - (In signal processing) A function over an index domain

$$s: \; I \to \mathbb{K}, \qquad i \mapsto s(i)$$

Typical examples:

$\mathbb{K} = \mathbb{R}, \mathbb{C}, GF(2)$
(real, complex, bit-signals)
$I = \mathbb{R}, \mathbb{Z}, \{0, \dots, n-1\}$
(continuous, discrete, finite signals)

digital signal processing

Examples

- Multimedia
  - Speech (1-D), Image (2-D), Video (3-D)
  - Quality improvement, compression, transmission
- Biometrics
- Medical/Bioimaging
- Computer vision
- Communication

Multimedia: Example Image Compression

Lena

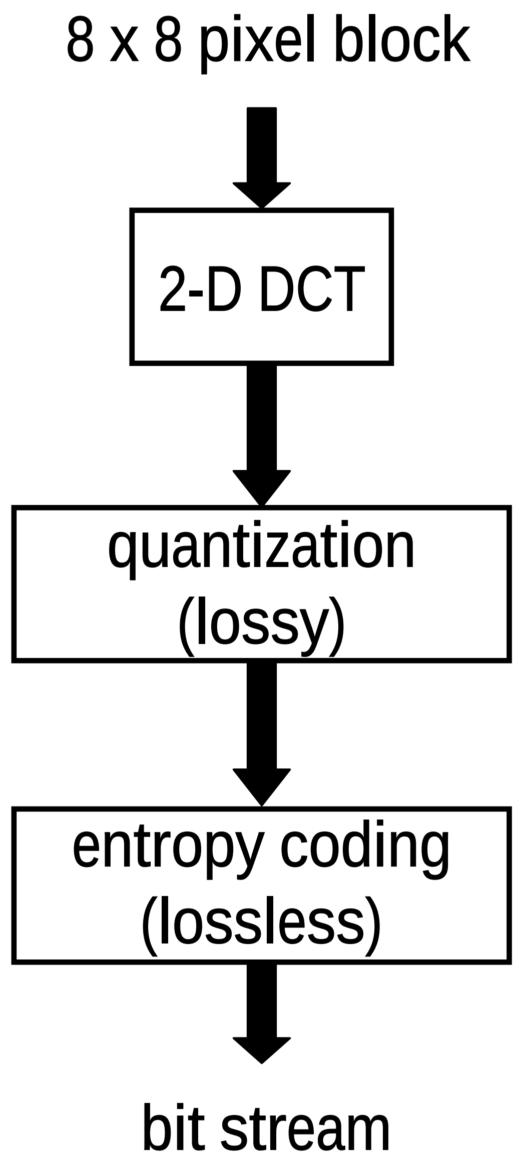
Pepper

Baboon

$512 \times 512 \times 3$ bytes = 768KB

With JPEG, ~32KB

JPEG: How does it Work?



JPEG versus JPEG2000

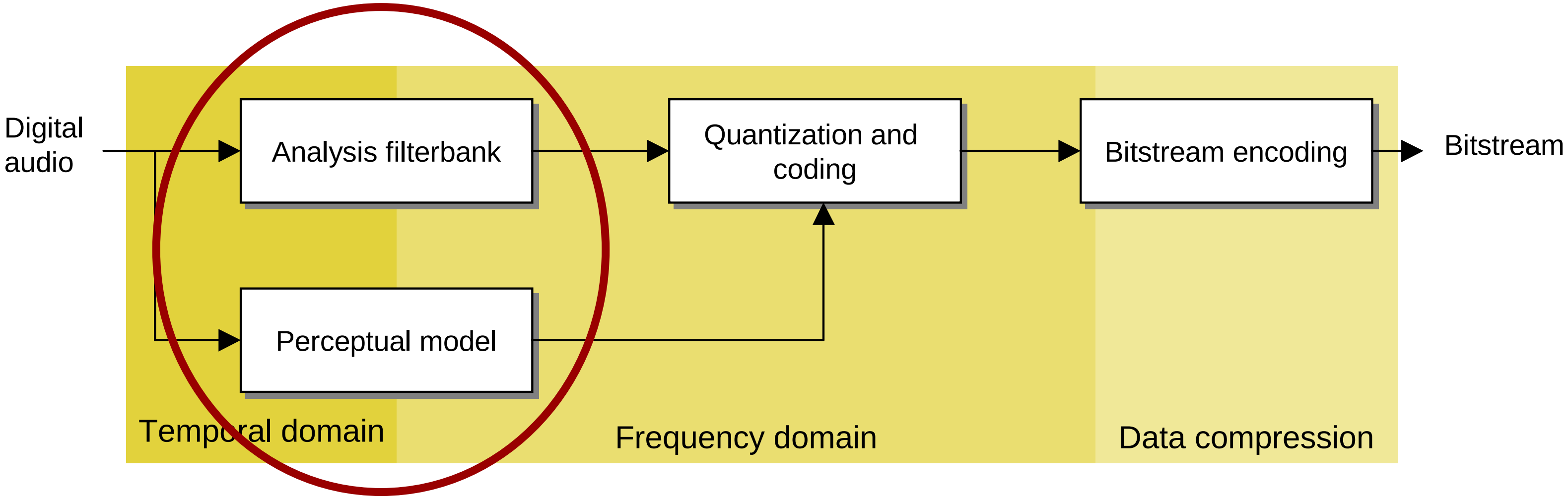
original: 3MB

JPEG: 19KB (DCT based)

JPEG2000: 19KB (wavelet based)

Multimedia Coding

- MPEG-I to MPEG-IV
- Includes standards for audio, image and Video
- Example: MPEG-II, layer III audio = MP3



transforms: DFT, MDCT, DCT

Example: Biometrics

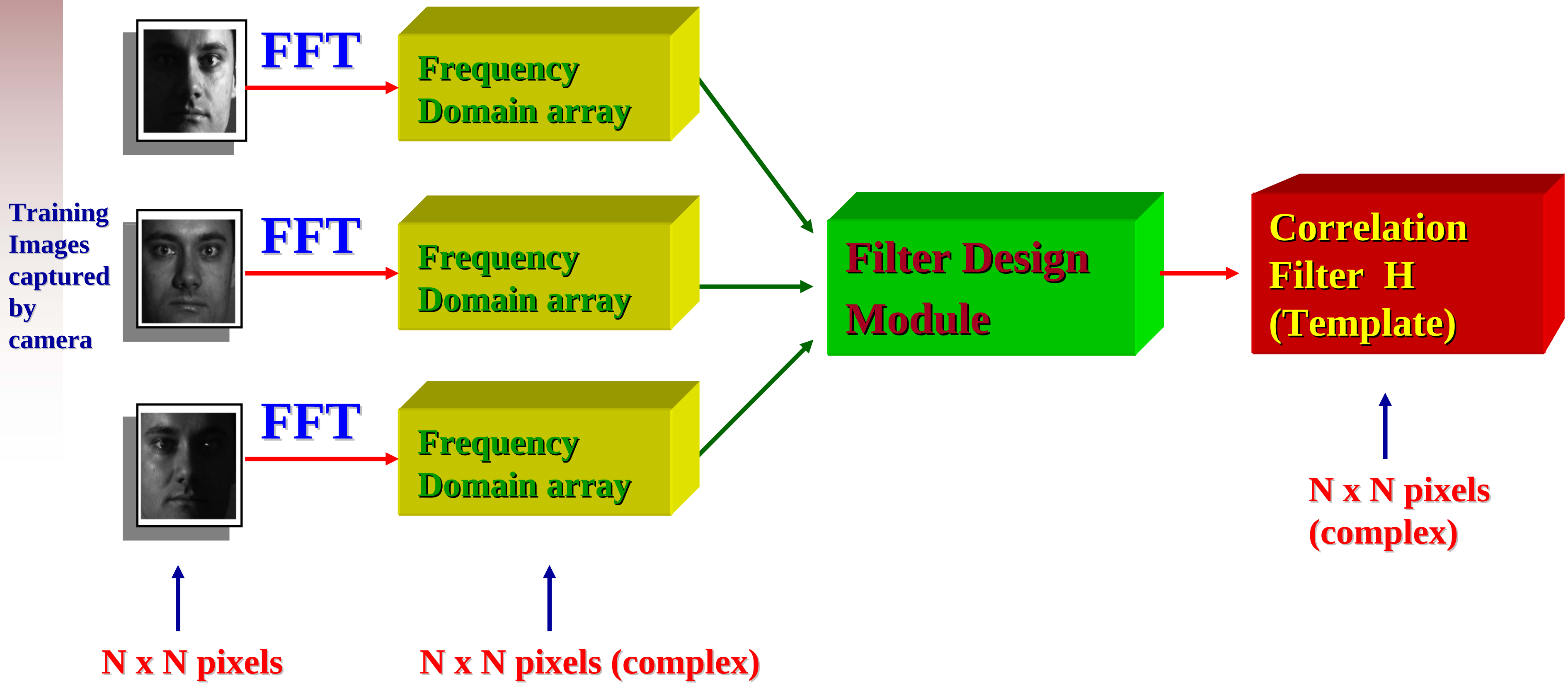
Facial Expression

Fingerprints

Illumination

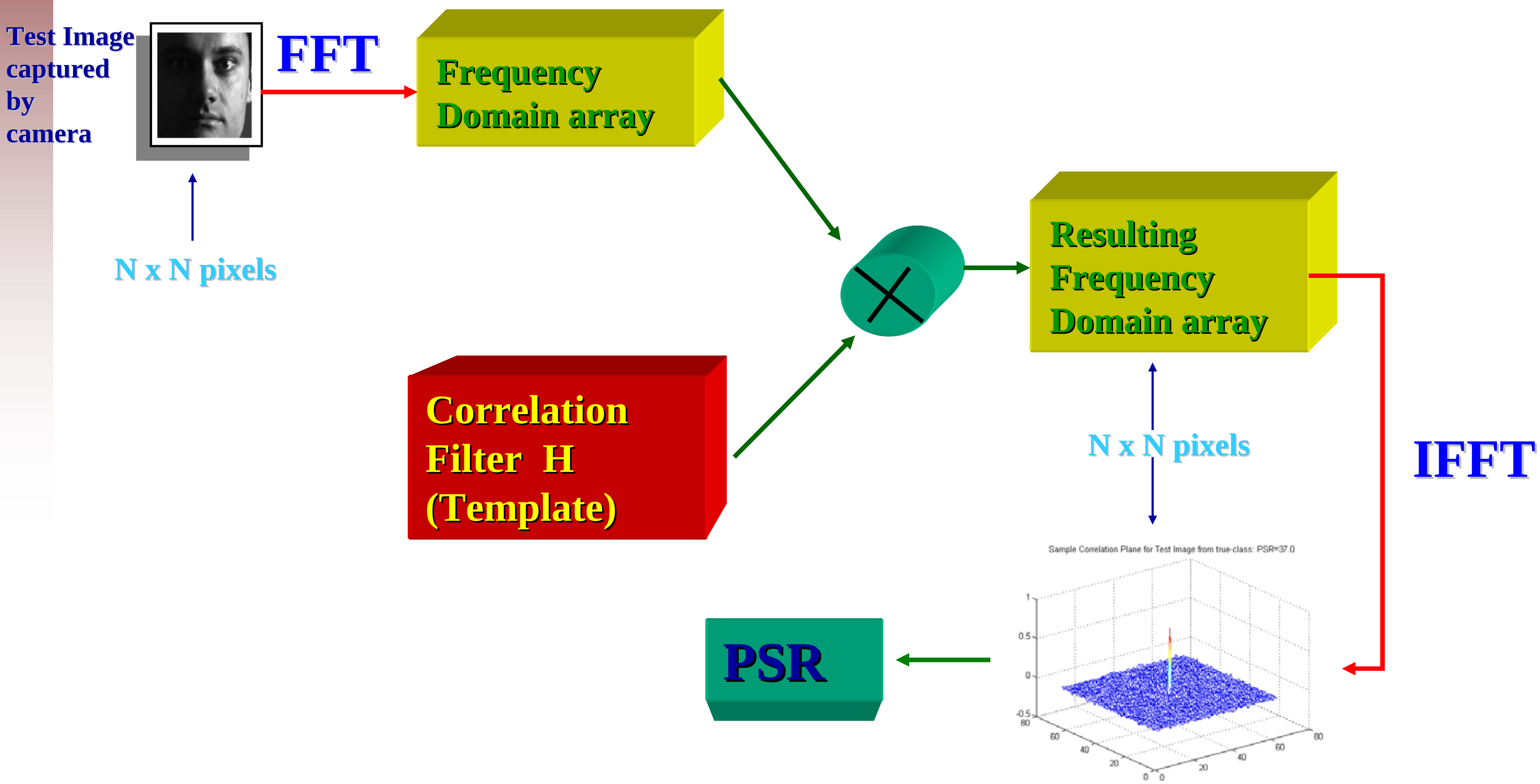
Source: Bhagavatula/Savvides

How does it Work?: Registration



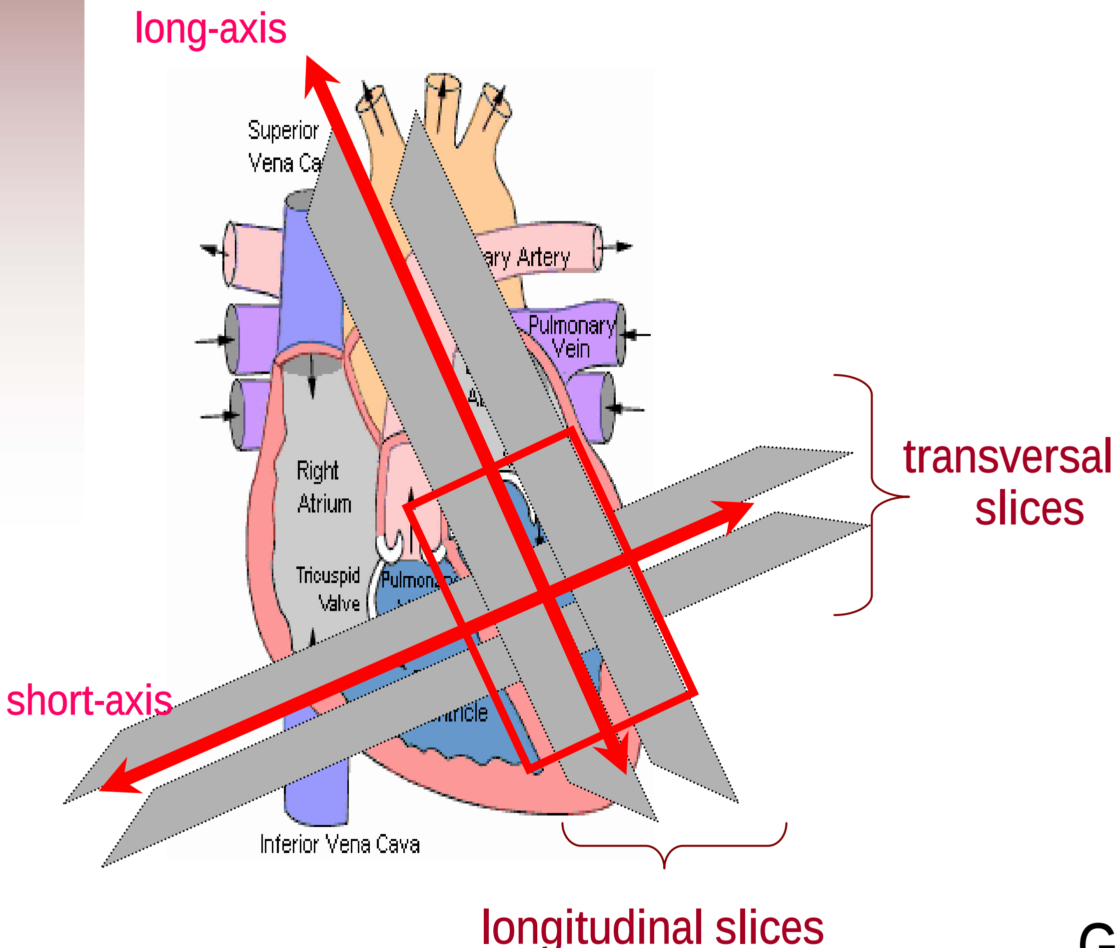
Source: Bhagavatula/Savvides

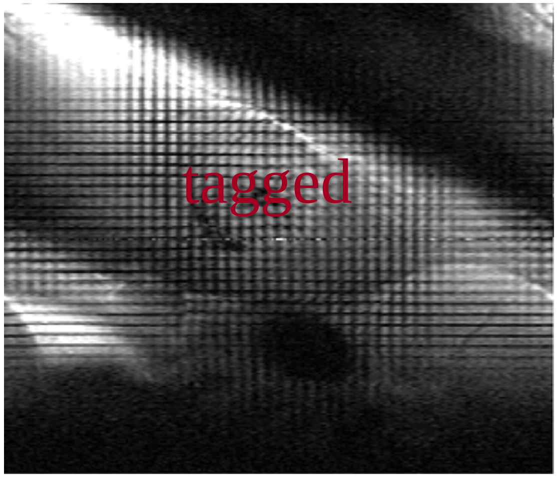
How does it Work?: Identification



Source: Bhagavatula/Savvides

Example: Cardiac MRI

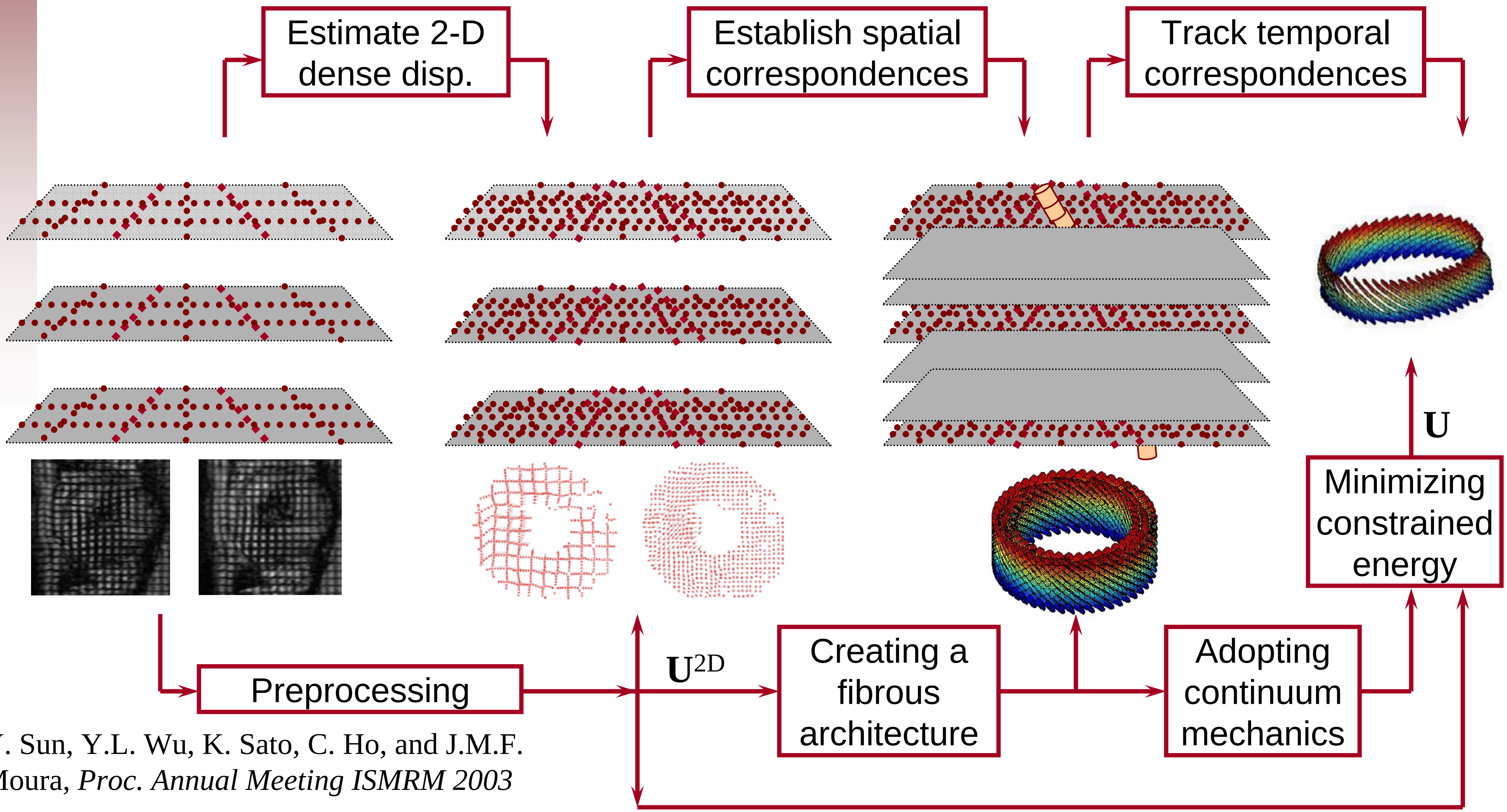




Goal: 3D-movie from 2D data

Source: Hsien/Moura

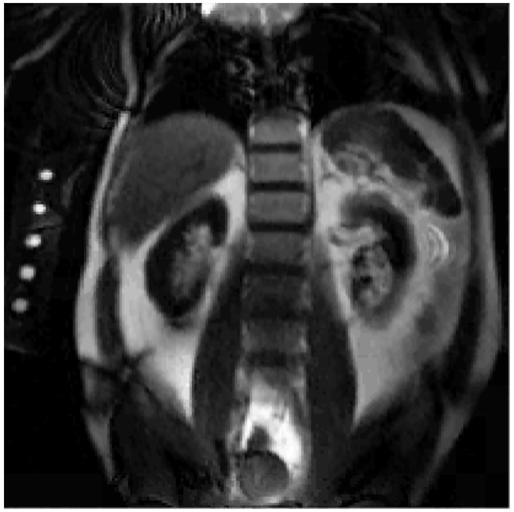
3-D Motion Estimation Procedure



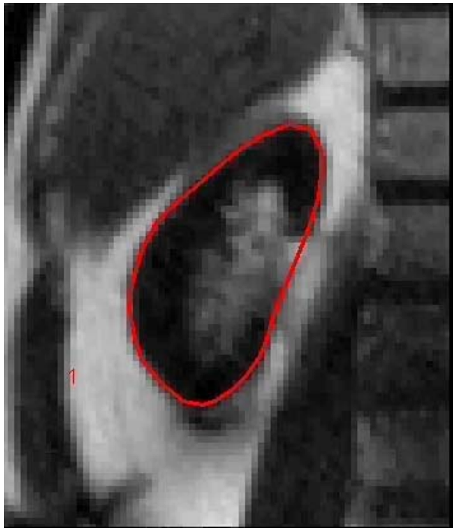
Y. Sun, Y.L. Wu, K. Sato, C. Ho, and J.M.F. Moura, *Proc. Annual Meeting ISMRM 2003*

Source: Hsien/Moura

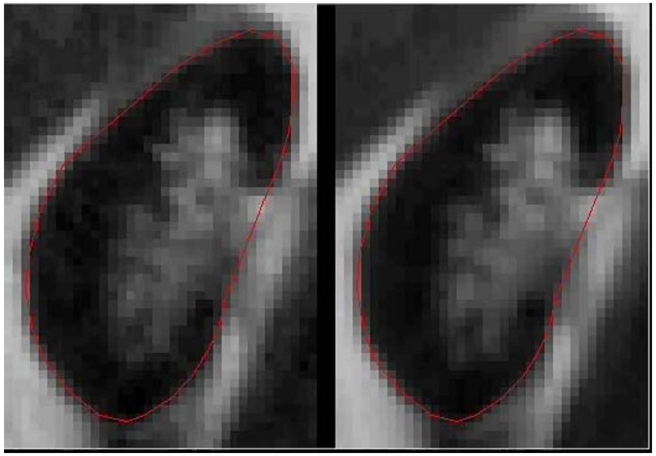
Example: MRI



MRI



Kidney tracking

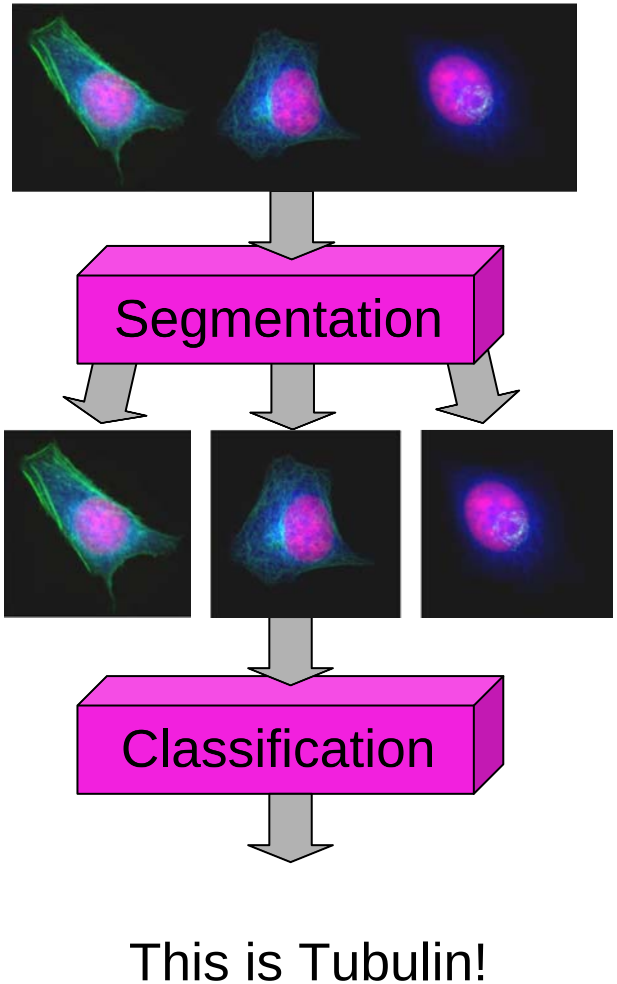


Compensation for motion

Source: Sun/Moura

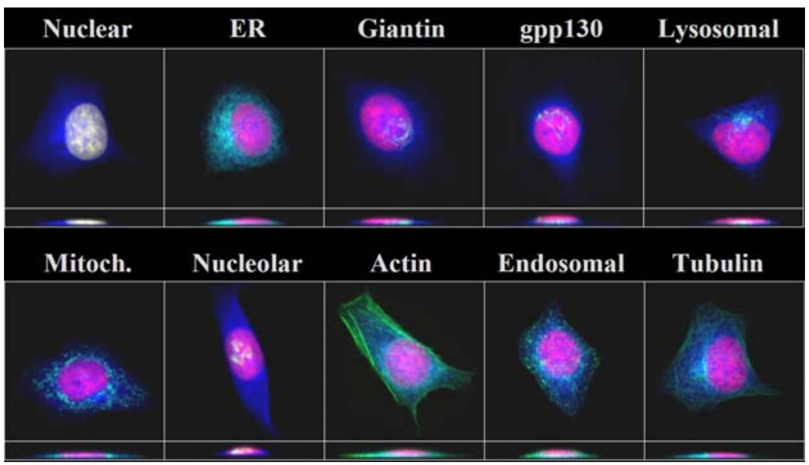
Example: Bioimaging

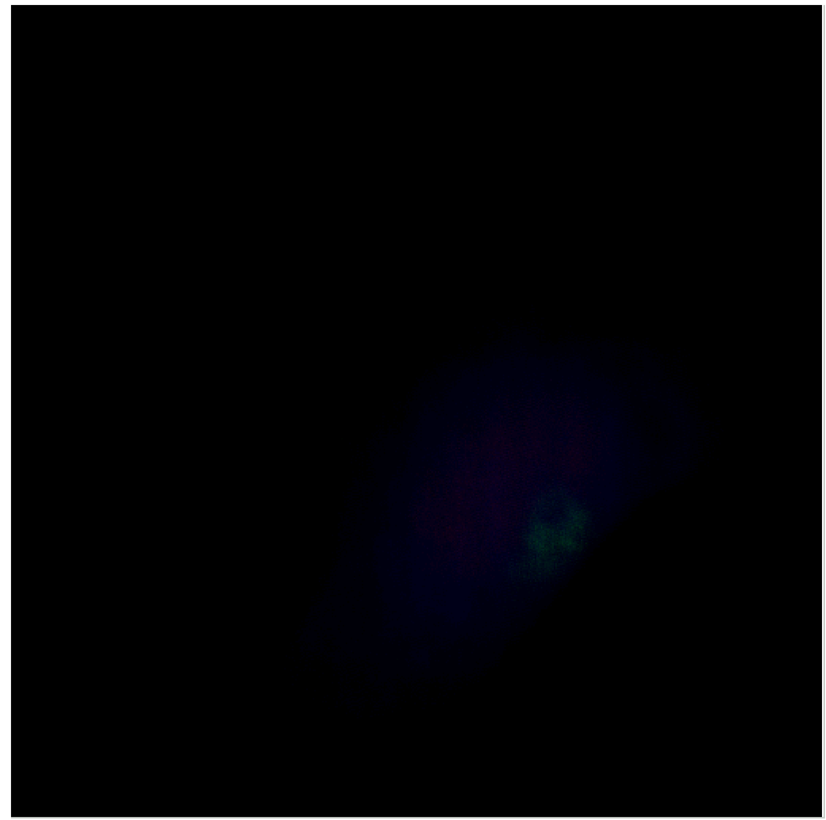
- Goal: automatic, fast, reliable identification of proteins from their distribution in the cell
- Signal processing
  - Segmentation
  - Classifikation (Wavelets, Frames)



Source: Kovacevic/Murphy

Images





Source: Kovacevic/Murphy

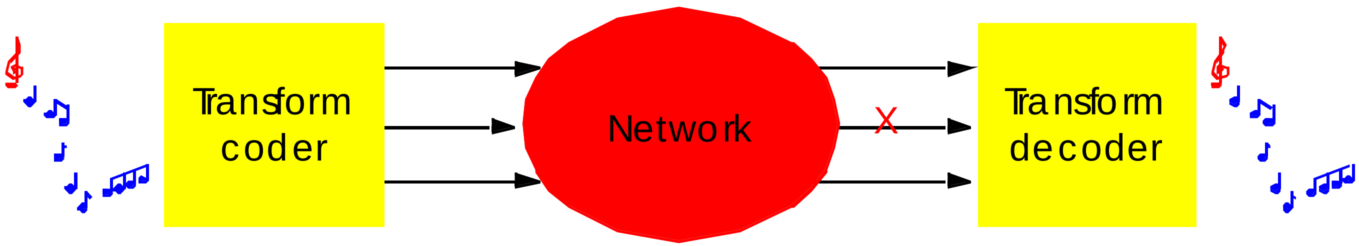
Example: Computer Vision

Suberbowl 2001 (Kanade et al.)

Plot: Kanade

Example: Communication

- Goal: Robustness to losses in transmission



Source: Kovacevic

Photo-to-Grandma Problem

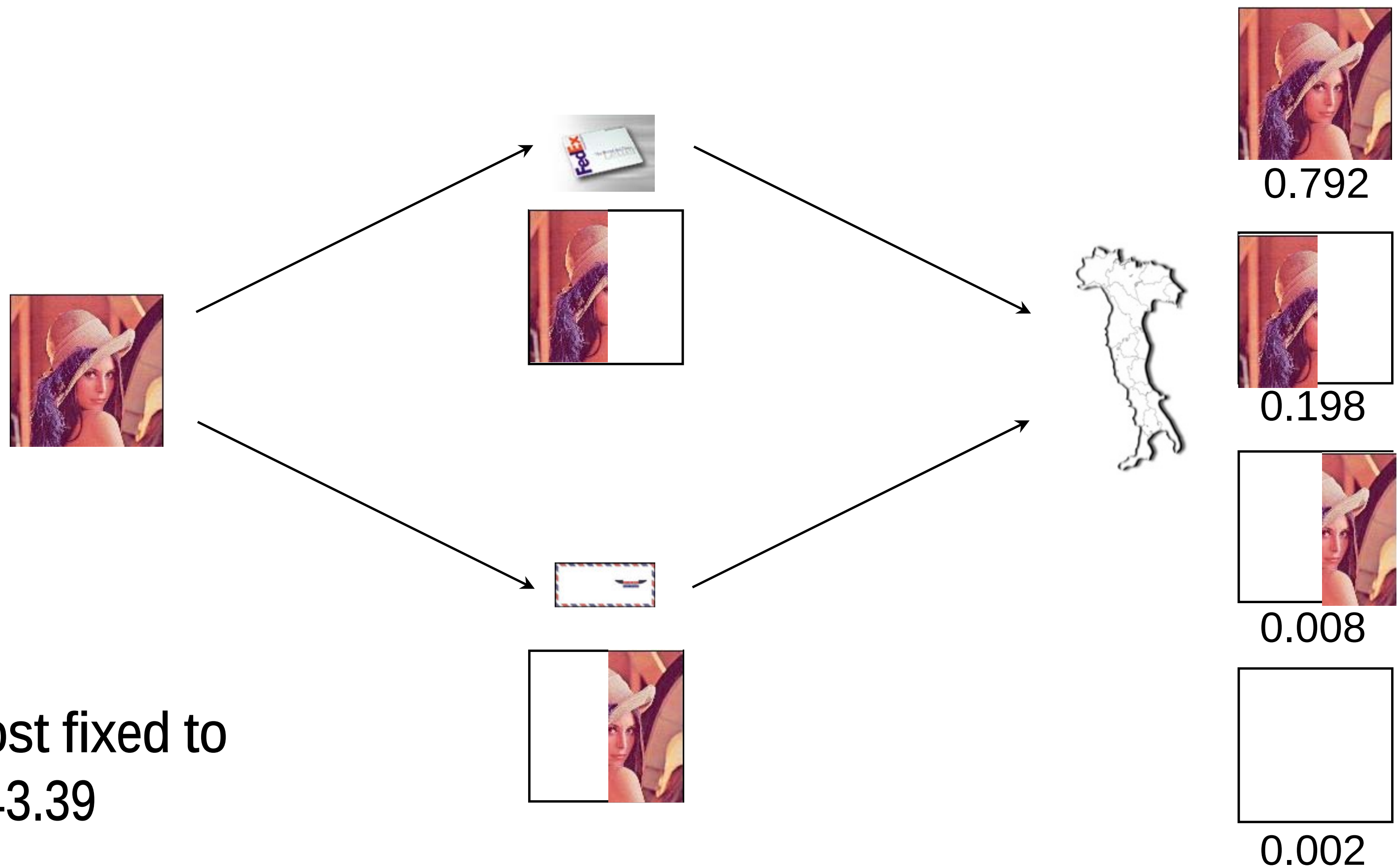
- Goal: send a digital photo to Italy
- Available: FedEx or regular “post channel”
  - FedEx 99% reliable, cost $39.99
  - Postal 80% reliable, cost $3.40
- 1 floppy per envelope only
- Photo needs 2 floppies (CDs haven’t been invented yet)

new girlfriend

Grandma lives in Italy

Source: Kovacevic

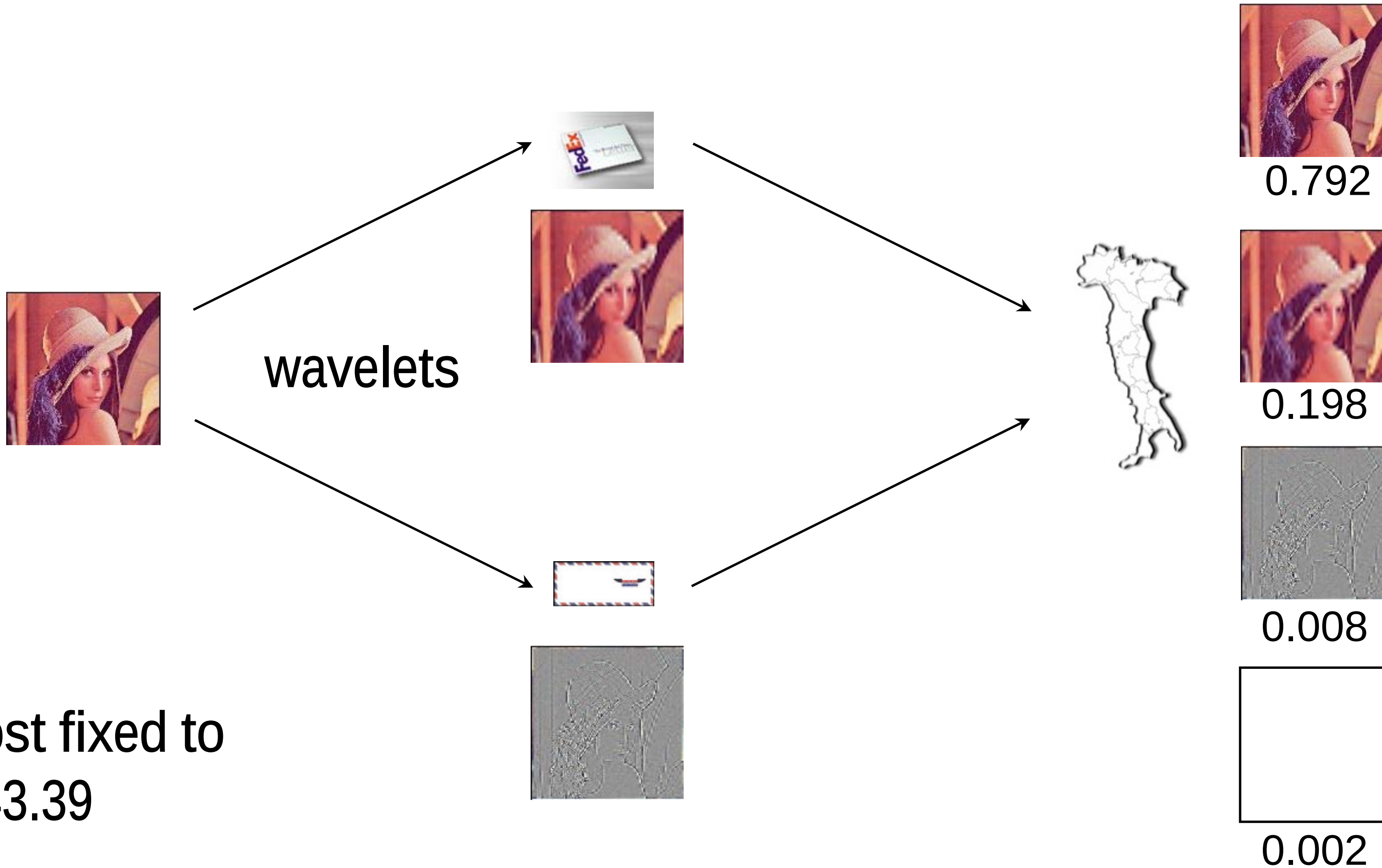
Heterogenous Channel (Dumb Solution)



cost fixed to
$43.39

Source: Kovacevic

Heterogenous Channel (Smart Solution)



cost fixed to
$43.39

Source: Kovacevic

Summary:
Computational Kernels in Signal Processing

Filter:
FIR, IIR, correlation,
filter banks

Signal transforms:
DFT, DCT, wavelets, frames

Linear algebra:
vector sum,
matrix-vector product,
…
singular value decomposition,
matrix inversion,
…

Coding:
Huffman, arithmetic,
Viterbi, LDPC

Most DSP computation is linear algebra

Numerical Computation Beyond DSP

- More than 90% of all numerical computation are linear algebra computations/algorithms
- Sciences: Chemistry, Physics, Biology; Economics; Engineering; etc.

Implementation

- Practically infinite speed requirements
  - Very large data sets
  - Realtime
- Multitude of platforms
  - Hardware: ASIC, FPGA
  - Software
    - Single vs. multiprocessor computers
    - Workstation versus embedded processor
    - Floating point vs. fixed point arithmetic
  - Combined hardware/software platforms
- Problems: Implementation difficult, expensive (time/money), becomes quickly obsolete

In this course: Single processor workstations