Gestione dei File

Credits Prof. Campi

Giacomo Boracchi,

Perché i file?

- Sono strutture dati **persistenti**
- Sono solitamente memorizzati sui dischi
  - Si usano dall'interno dei programmi
- Realizzano la *persistenza dei dati*
  - cioè del contenuto delle variabili
- Tramite i file, i dati possono sopravvivere al termine dell'esecuzione del programma
- N.B.: i file sono usati anche per memorizzare i programmi !!
  - Quando se ne chiede l'esecuzione, il sistema operativo copia il programma (eseguibile, conservato in un file) in memoria centrale e inizia a eseguirlo

File binari, file di testo

- I file sono strutture dati **sequenziali**
  - Sequenziale significa: si leggono (e scrivono) gli elementi del file in sequenza
- Un file ***binario*** è una sequenza di byte che non è "interpretata" in alcun modo
- Un file ***di testo*** è una sequenza di caratteri "interpretata":
  - Alcuni caratteri rappresentano separatori
    - Esempio: il carattere di "newline" è interpretato dalla stampante come "salto alla riga successiva"

File e sistema operativo

- I file sono gestiti dal S.O.
  - Sono resi visibili all'interno del linguaggio per essere manipolati attraverso opportune funzioni di libreria
- Per essere usato, un file deve essere prima ***aperto***, e dopo l'uso andrà ***chiuso***
  - *Aprire e chiudere il **"flusso di comunicazione"** tra il programma e il file*
- In C anche le periferiche sono viste come file (chiamati "file speciali")
  - **stdin** e **stdout** (terminali, stampanti, ecc)
  - **Si può "leggere" e "scrivere" con le stesse modalità (quelle dei file) da ogni device di I/O**

Tutte le periferiche sono "viste" come file!!

Possiamo "leggere" e "scrivere" con le stesse modalità (quelle dei file) in ogni operazione di I/O

QUINDI SAPPIAMO GIÀ TUTTO !!

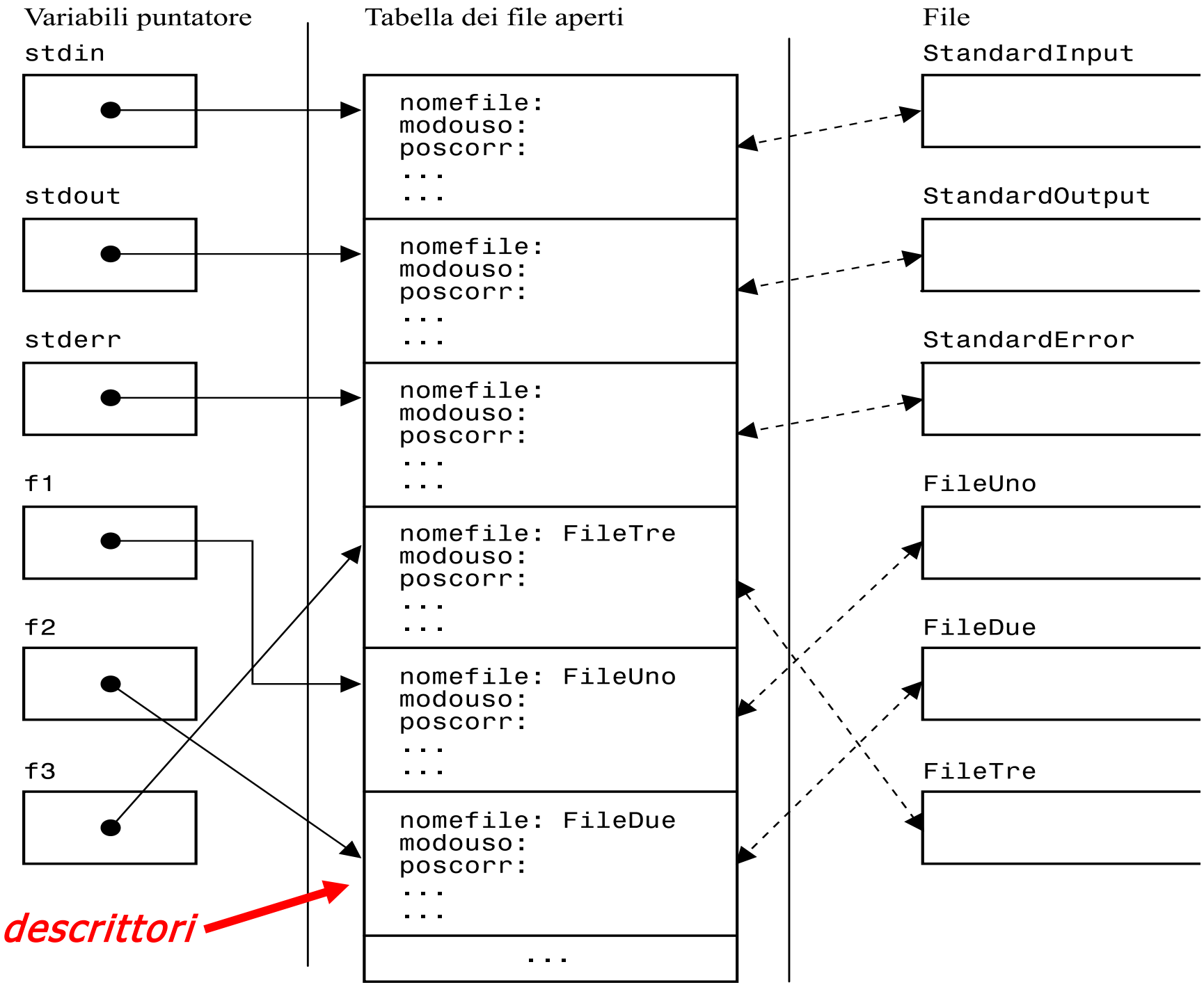
Rappresentazione interna dei file

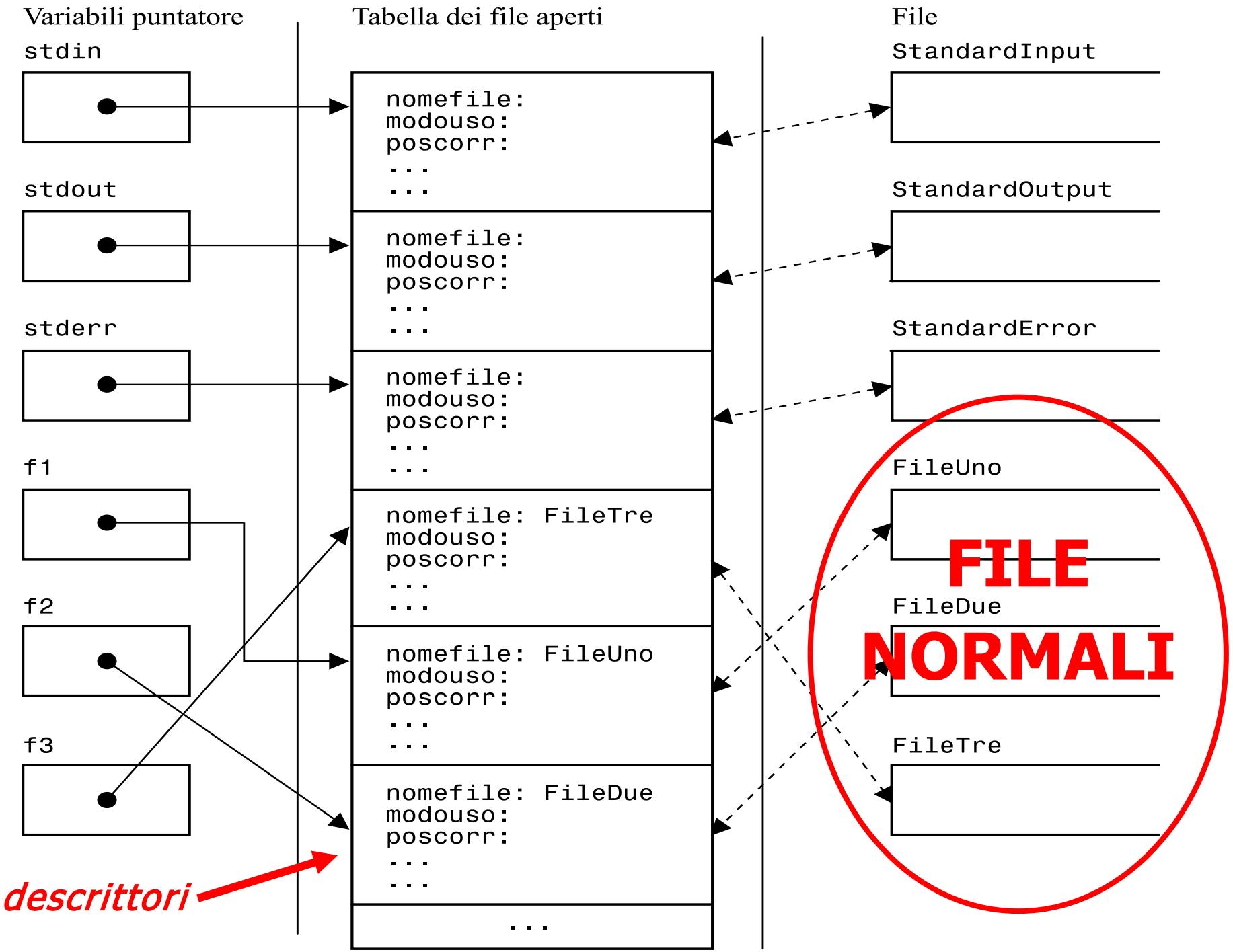
- Ogni file aperto da un prog. ha un **descrittore**
  - Risiede nella *tabella dei file aperti*, una delle strutture dati che il S.O. associa ai programmi in esecuzione
- Il descrittore memorizza:
  - la **modalità** d'uso (read, write)
  - la **posizione** corrente all'interno del file
  - l'indicatore di eventuale **errore**
  - l'indicatore di **EOF** (end-of-file)
- L'**apertura** del file restituisce un descrittore
  - Per la precisione, un **puntatore** a un descrittore

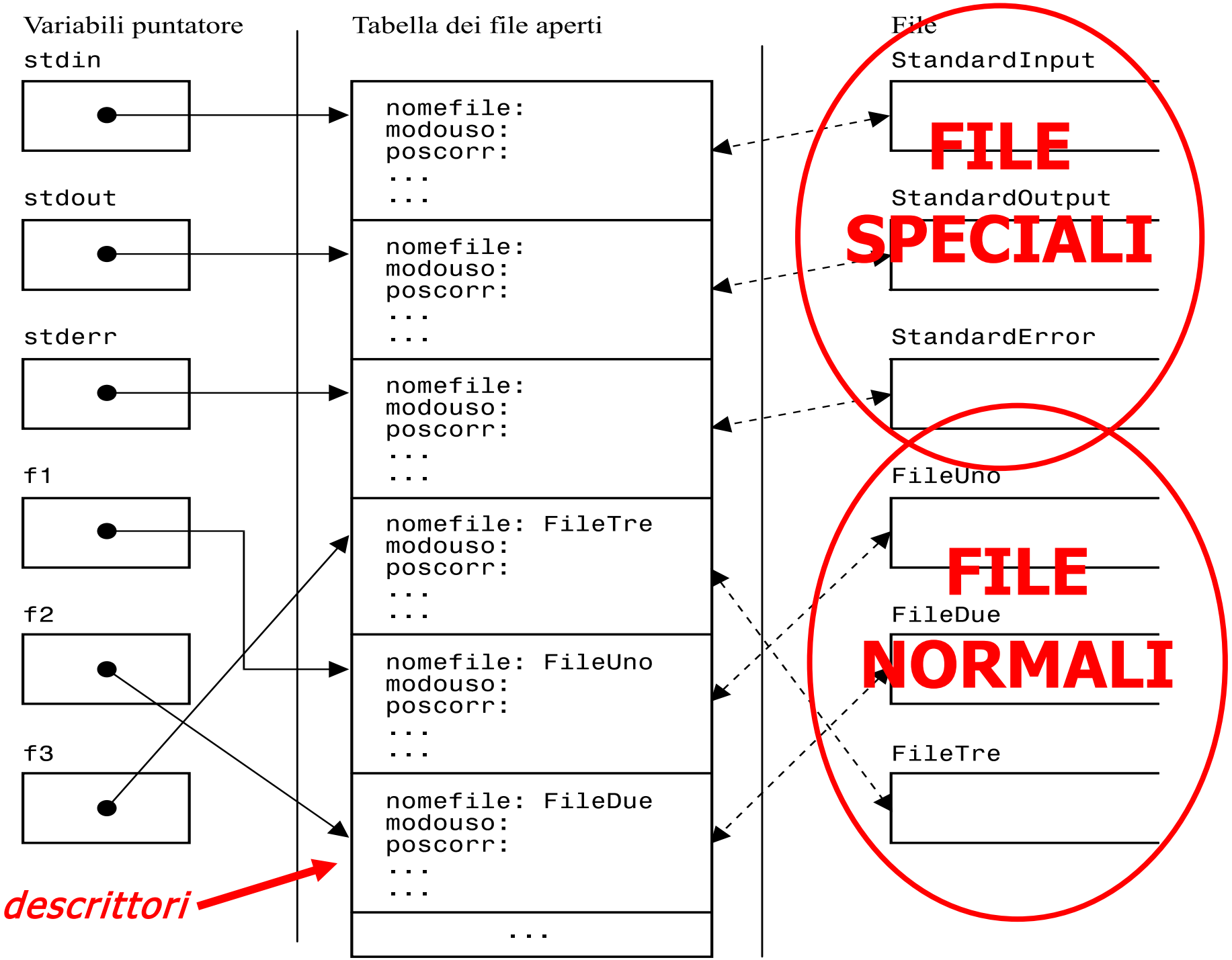
Rappresentazione interna dei file

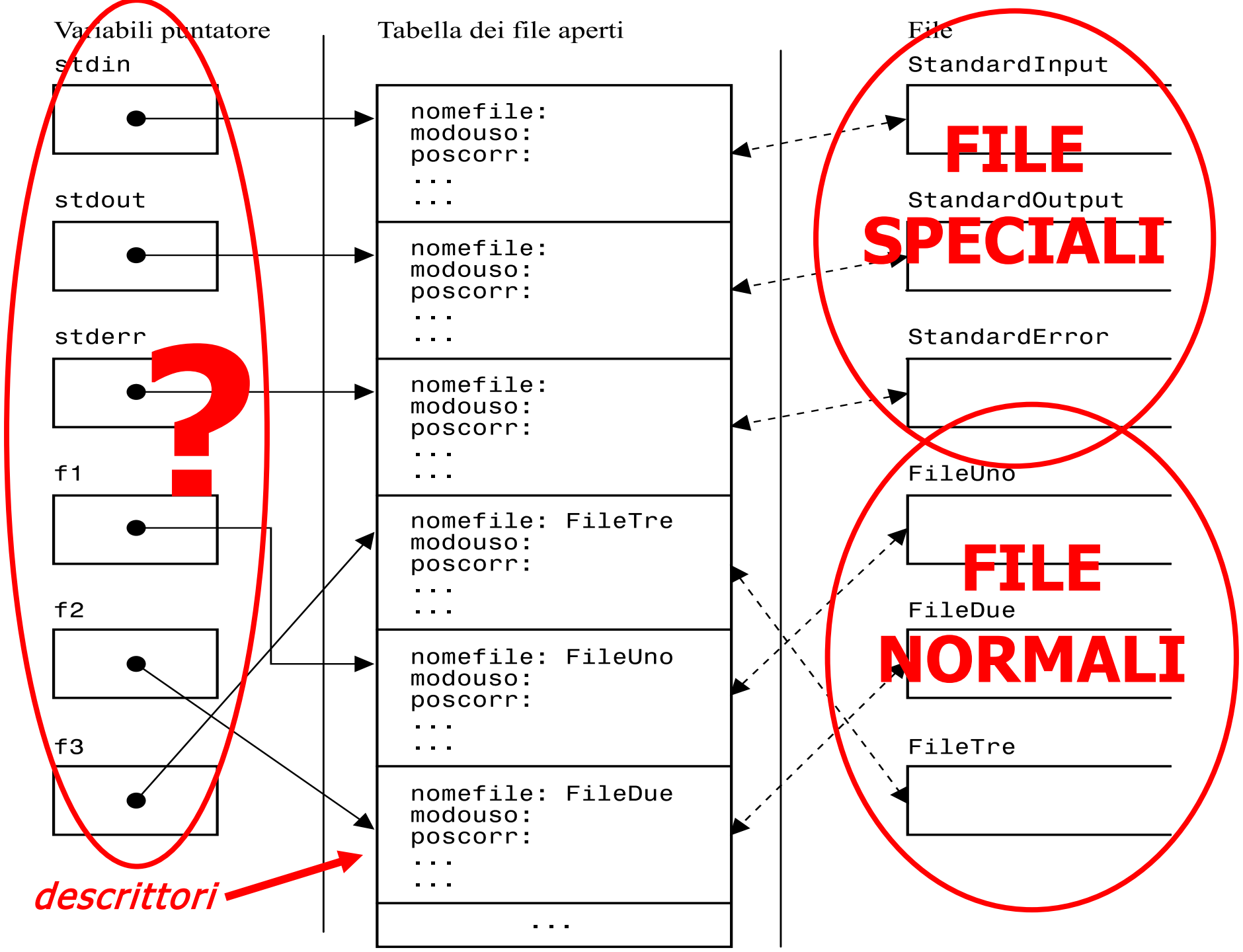
- Ogni file aperto da un prog. ha un **descrittore**
  - Risiede nella *tabella dei file aperti*, una delle strutture dati che il S.O. associa ai programmi in esecuzione
- Il descrittore memorizza:
  - la **modalità** d'uso (read, write)
  - la **posizione** corrente all'interno del file
  - l'indicatore di eventuale **errore**
  - l'indicatore di **EOF** (end-of-file)
- L'**apertura** del file restituisce un descrittore
  - Per la precisione, un **puntatore** a un descrittore

Visto che il S.O. salva la posizione, è possibile accedere in maniera sequenziale ai contenuti del file!


Variabili puntatore
Tabella dei file aperti
File
stdin
stdout
stderr
f1
f2
f3
nomefile:
modouso:
poscorr:
...
...
nomefile: FileTre
nomefile: FileUno
nomefile: FileDue
StandardInput
StandardOutput
StandardError
FileUno
FileDue
FileTre
descrittori


Variabili puntatore
Tabella dei file aperti
File
stdin
stdout
stderr
f1
f2
f3
nomefile:
modouso:
poscorr:
...
...
nomefile: FileTre
nomefile: FileUno
nomefile: FileDue
StandardInput
StandardOutput
StandardError
FileUno
FileDue
FileTre
FILE
NORMALI
descrittori


Variabili puntatore
Tabella dei file aperti
File
stdin
stdout
stderr
f1
f2
f3
nomefile:
modouso:
poscorr:
...
nomefile: FileTre
nomefile: FileUno
nomefile: FileDue
StandardInput
StandardOutput
StandardError
FILE SPECIALI
FileUno
FileDue
FileTre
FILE NORMALI
descrittori


Variabili puntatore
Tabella dei file aperti
File
stdin
stdout
stderr
f1
f2
f3
nomefile:
modouso:
poscorr:
...
nomefile: FileTre
nomefile: FileUno
nomefile: FileDue
StandardInput
StandardOutput
StandardError
FileUno
FileDue
FileTre
FILE SPECIALI
FILE NORMALI
descrittori

Dichiarare e aprire un file

- Puntatore al descrittore**: FILE * fp**
- Apertura del file:
  - FILE * **fopen** (char * *nomefile*, char * *modalità*)

    nomefile e *modalità* sono stringhe

    nomefile dà il percorso (path), oppure il nome è interpretato nella cartella in cui si lancia l'eseguibile

    apre il file (oppure lo crea, se è inesistente)
  - modalità di apertura

    "r" lettura modalità testo, posizionamento inizio file (**read**)

    "w" scrittura modalità testo, posizionamento inizio file (**write**)

    "a" scrittura in modalità testo, posizionamento fine file (**append**)

    "rb", **"wb"** e **"ab"** (idem, ma considerando il file come **binario**)
- Se si verifica un errore, fopen() restituisce **NULL**

Cancellare, ridenominare, chiudere

int remove (char * nomefile)

- cancella file nomefile
- restituisce 0 se buon fine, != 0 altrimenti

int rename (char *oldname, char *newname)

- cambia nome al file
- restituisce 0 se buon fine, !=0 altrimenti

int fclose (FILE * fp)

- fp diventa NULL, descrittore di tipo FILE rilasciato
- restituisce 0 se buon fine, altrimenti EOF

Gestione degli errori

int ferror (FILE * fp)

- restituisce ≠0 se errore (variabile errno set)

int **feof** (FILE * fp)

- restituisce 0 (falso) se NON si è alla fine

void clearerr (FILE * fp)

- riporta al valore normale gli indicatori di errore e eof

Lettura e scrittura

- Si opera sui file in quattro modi possibili
- Tre modi per i file di testo:
  - Precisando la **formattazione** dell' I/O
  - Un **carattere** alla volta
  - Per **linee** di testo
    - Fino ad ogni prossimo '\n'
- Un modo per i file binari:
  - Per **blocchi** di byte
    - approccio "à-la-sizeof"

Lettura / scrittura formattata

- scanf e printf fanno riferimento a stdin e stdout
  - Non serve specificare su quale file agiscono!!
- **f**printf e **f**scanf fanno riferimento a file generici e si usano esattamente come scanf e printf

int fprintf (FILE * **fp**, char * *str_di_controllo*, *elementi*)

int fscanf (FILE * **fp**, char * *str_di_controllo*, *indirizzo_elementi*)

- Restituiscono il numero di elementi effettivamente letti/scritti, o zero se errore

Leggere, mostrare a video e salvare il contenuto di una struct (un campo intero e uno char)

```
#include <stdio.h>
int main ()  {
    FILE * fp1, * fp2;
    int numero;
    char c;
    fp1 = fopen (“nomeFile1“,"r");/*file lettura, modalità testo */
    fp2 = fopen (“nomeFile2“,“w");/*file scrittura,modalità testo*/
    if (fp1 != NULL && fp2 != NULL ) {
          fscanf(fp1,“%d%c”,&numero,&c);
          printf(“%d%c”,numero,c);
          fprintf(fp2,“%d%c”,numero,c);
    } else
          printf ("Il file non può essere aperto.\n");
     if (fp1 != NULL) {
          fclose (fp1);
     }
     if (fp2 != NULL) {
          fclose (fp2);
     }
    return 0;
}
```

```c
#include<stdio.h>

int main()
{
FILE *p, *q;
char nome[100];
int peso;

p = fopen("pesiInfoA.txt","r");// ogni riga del file ha il seguente formato: nome peso
q = fopen("pesiAlti.txt","w");

if(p != NULL && q != NULL)
{
    // processa tutto il file puntato da p, fino alla fine
    while(!feof(p))
    {
        // legge da file
        fscanf(p, "%s %d", nome, &peso);
        // stampa a schermo
        printf("\n%s pesa: %d", nome, peso);
        // scrittura selettiva su altro file
        if(peso > 60)
            fprintf(q, "\n%s,", nome);
    }
    fclose(p);
    fclose(q);
}
else
    printf("Errore apertura file");

return 0;
}
```

Lettura carattere per carattere

- *int getchar (void)*
  - *legge un carattere **da standard input**, restituendolo come intero*
- *int putchar (int c)*
  - *scrive un carattere **su standard output***
- int **f**getc (FILE * fp)
- int **f**putc (int c, FILE * fp)
  - leggono/scrivono un carattere dal/sul **f**ile descritto da *fp, restituendolo come inter

Se fp è stdin/stdout è identico scrivere getc() e putc(c)

Leggere e mostrare a video un file

```
#include <stdio.h>
#include <stddef.h>
int main ()  {
    FILE * fp;
    int c;
    fp = fopen ("filechar", "r");  /* file lettura, modalità testo */
    if (fp != NULL) {
        c = fgetc (fp);
        while (c != EOF)  {      /* oppure while (! feof (fp)) */
                putchar (c);
                c = fgetc (fp);   /* oppure c=fgetc (fp); */
        }
        fclose (fp);
    } else
        printf ("Il file non può essere aperto.\n");
    return 0;
}
```

```
while ((c=fgetc(fp)) != EOF)
    putchar(c);
```

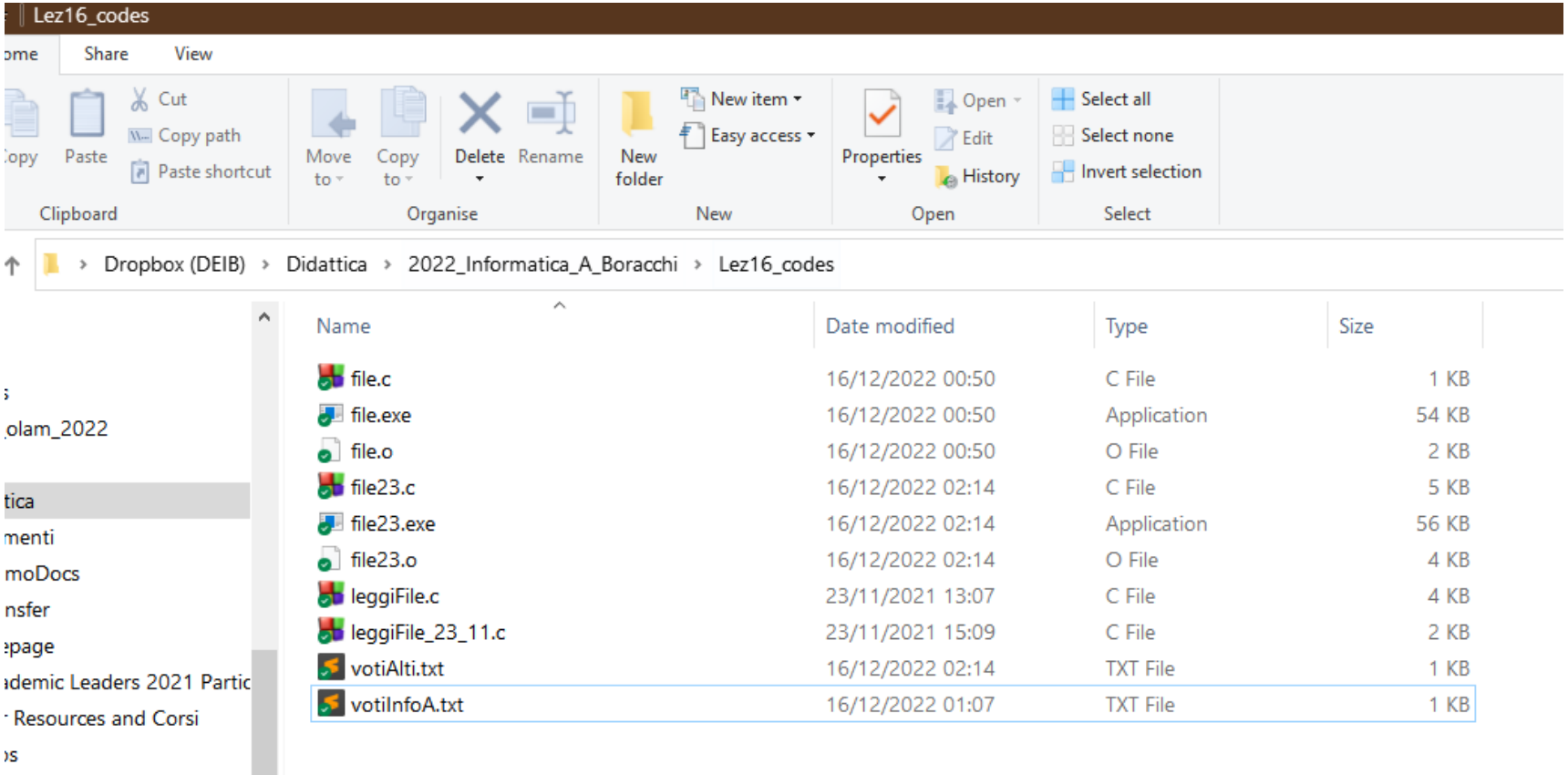
Lettura / scrittura per linee di testo

- Su **stdin** e **stdout**:
  - char * gets (char * s)
    - s è l'array in cui copiare la stringa letta da stdin
      - s risulta terminata da un '\0', aggiunto in automatico
    - Non si può limitare la dimensione dei dati in input
      - Non controlla che la stringa s sia sufficientemente grande
    - In caso di errore, restituisce **NULL**
  - int puts (char * s)
    - scrive la stringa s, escluso il '\0'
      - al posto del '\0' che si trova nella stringa scrive un '\n'
    - Restituisce n>=0 se OK, EOF in caso di errore

Nota Bene

Il file di testo che si apre deve trovarsi

- Nella stessa cartella dell'eseguibile, oppure
- In una posizione precisata dal percorso nel disco



Esercizio Studenti

Il file votiInfoA.txt ci sono registrati i voti di tutti gli studenti che hanno passato il primo appello di Info A secondo il seguente formato:

```
carlo 18
paolo 22
giovanni 23
```

…

Si scriva un programma che indirizza le seguente richieste

- Apre il file votiInfoA.txt e copia nel file votiAlti.txt tutti gli studenti che hanno preso un voto maggiore a 18
- Senza caricare in memoria i nomi ed i voti di tutti gli studenti, si calcoli il voto medio ed il nome di uno studente che ha preso il voto più vicino al voto medio (nel caso fossero più di uno è indifferente quale viene restituito)
- Carica tutti i dati in una listaStudenti opportunamente definita e calcola il voto mediano
- Definisce un secondo tipo: listaVoti i cui nodi sono ordinati e contengono ciascuno un voto e una lista di tutti gli studenti che hanno preso quel voto
- Si scriva la funzione stampa (su file) del contentuto della lista

Soluzione in C fornita nei codici

Tutorato

https://www.ingindinf.polimi.it/it/studenti/servizi/tutorato

Avete un tutor che risponderà a domande e vi aiuterà a risolvere problemi che riscontrate in preparazione all'esame

Tutorato online, via webex (stanza tutor) secondo un calendario definito a breve

Presentatevi a tutorato con domande, esercizi su cui avete avuto problemi e richieste di chiarimenti.

Vi verrà probabilmente inviato un form in cui inserire le richieste

NEXT STEPS…

Option 1: Artificial Neural Networks And Deep Learning

Magistrale, Primo Semestre

Option 2: Mathematical Models and Methods for Image Processing

Magistrale, Secondo Semestre

Laboratory Course

Please have a look at:

https://boracchi.faculty.polimi.it/teaching/AN2DL.htm

And possibly video recordings in the course calendar

What is this course about?

What is this course about?

It is about **algorithms** *for processing* **images** *and solving image-related problems.*

Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K. *"Image denoising by sparse 3-D transform-domain collaborative filtering"* TIP 2007

What is this course about?

It is about ***algorithms*** *for processing* ***images*** *and solving image-related problems.*

..like denoising

Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K. *"Image denoising by sparse 3-D transform-domain collaborative filtering"* TIP 2007

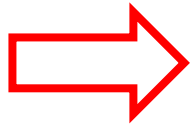
Who cares about images?

Who cares about images?

Everybody!

We will see algorithms solving problems customarily addressed in our phones,

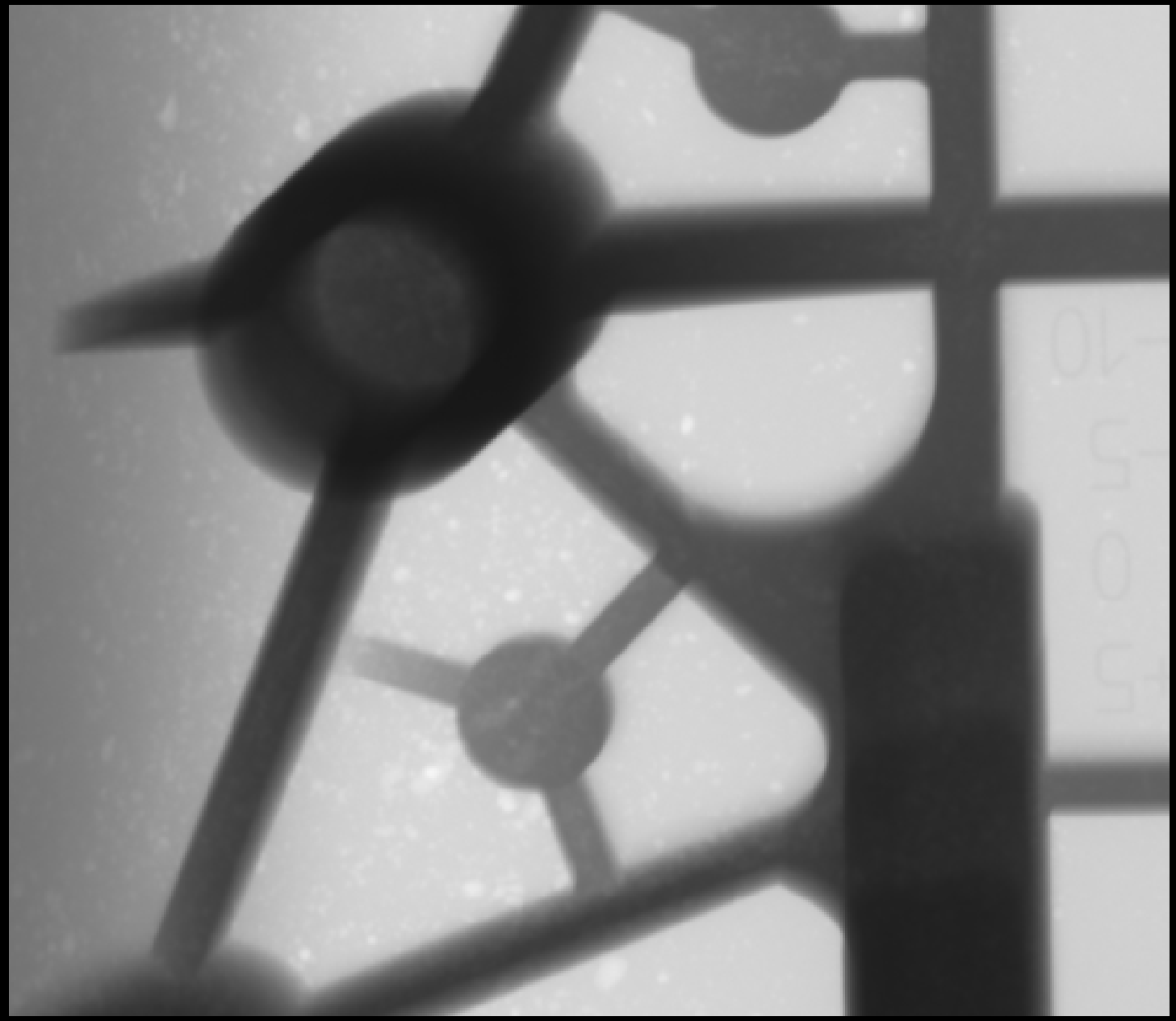
$$z = y + \eta\,, \qquad \eta \sim \mathcal{N}(0, \sigma^2)$$



$$\hat{y} \approx y$$

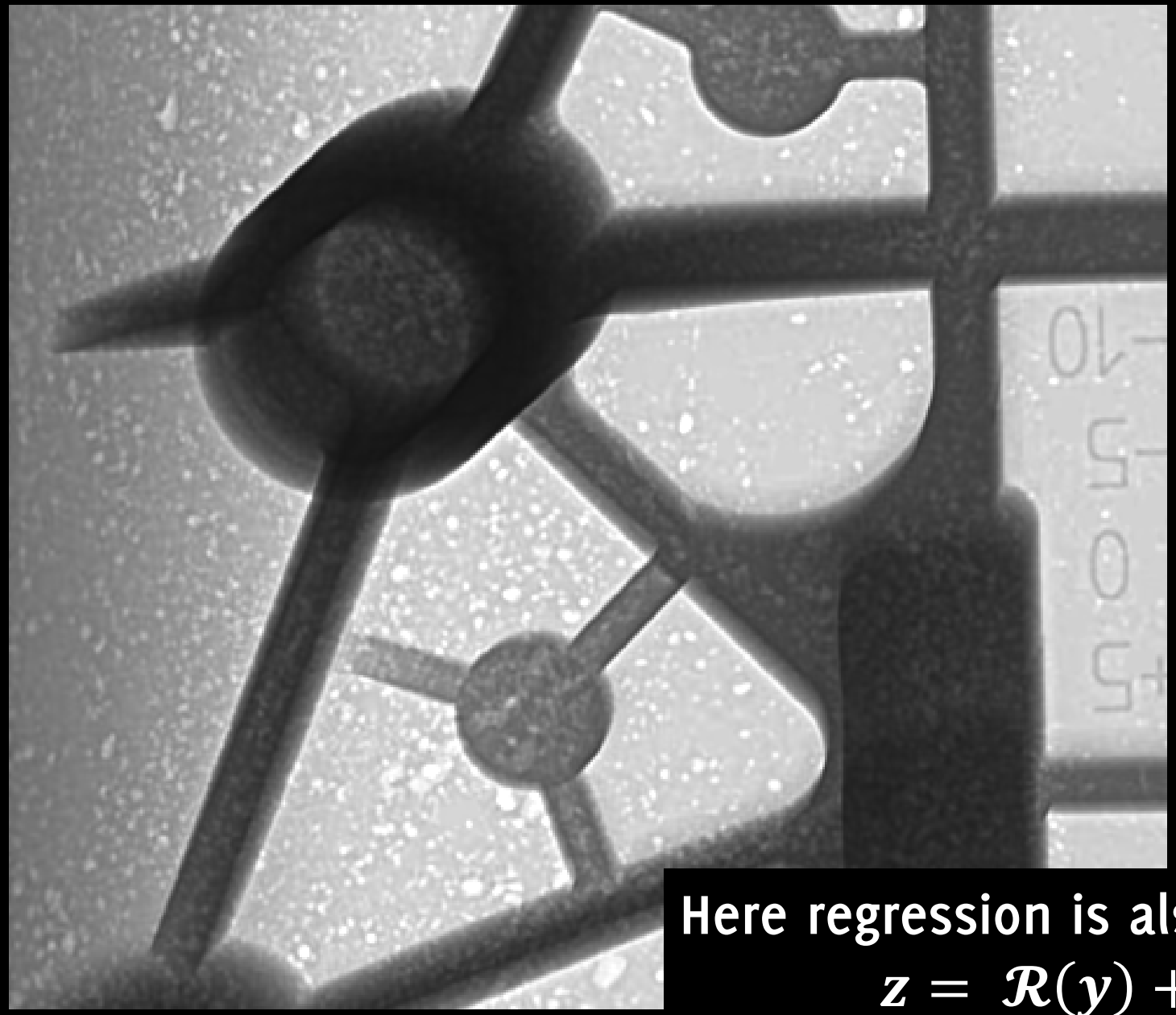
Denoising is a regression problem: given the noisy z, estimate $\hat{y}$ close to the unknown y

Who cares about images? Quality Inspection



G. Boracchi

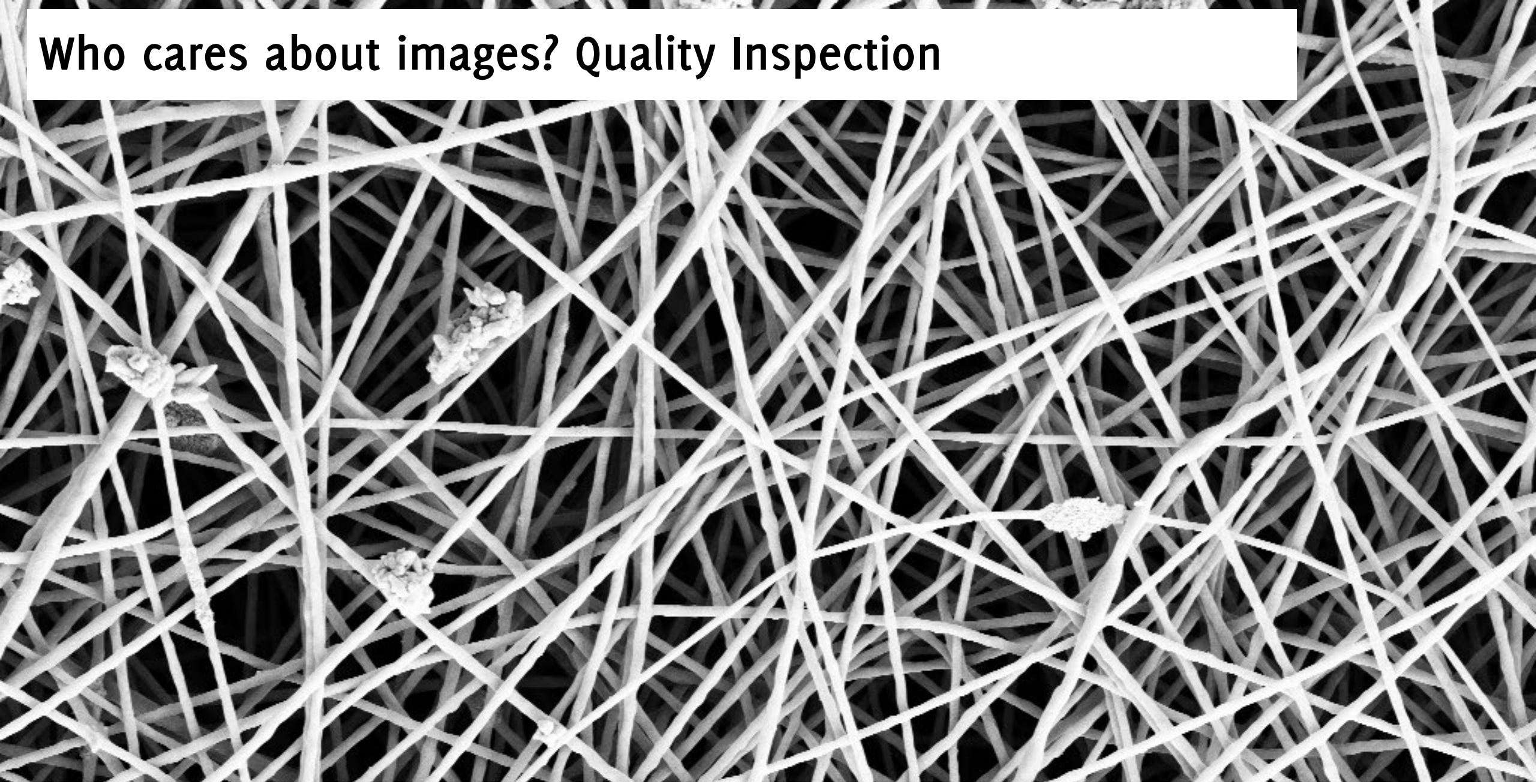
Who cares about images? Quality Inspection



Here regression is also crucial

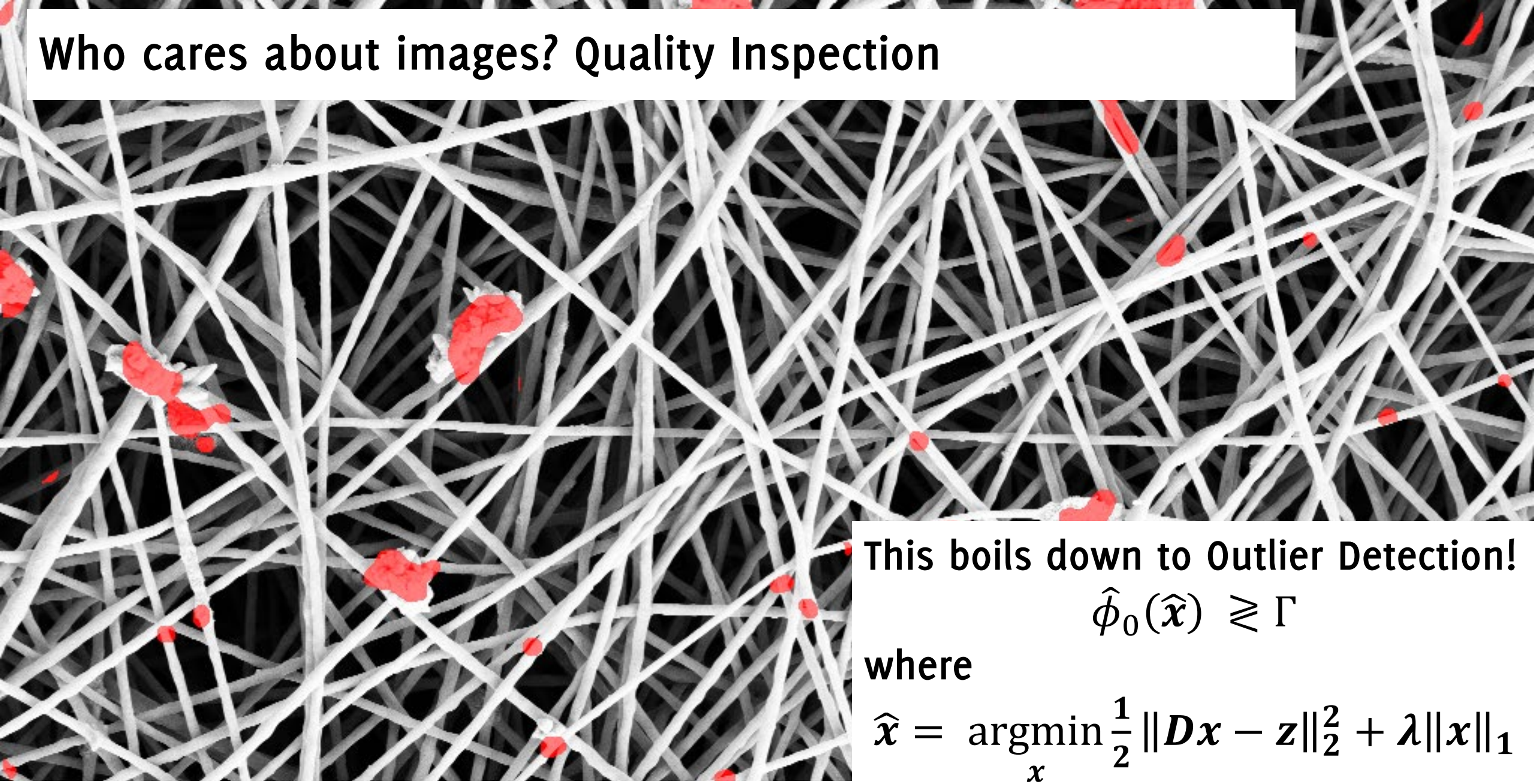
$$z = \mathcal{R}(y) + \eta$$

Who cares about images? Quality Inspection



Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

Who cares about images? Quality Inspection



This boils down to Outlier Detection!

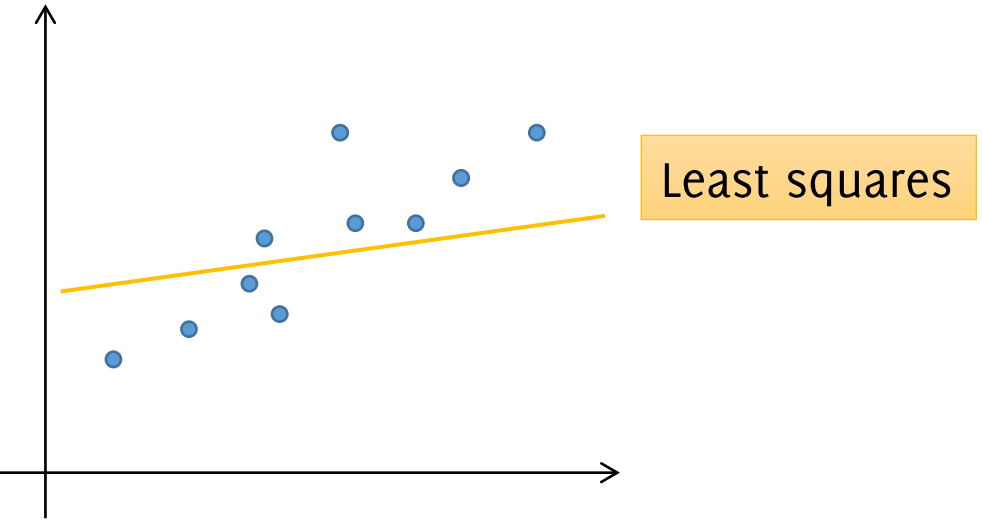
$$\hat{\phi}_0(\hat{\boldsymbol{x}}) \gtrless \Gamma$$

where

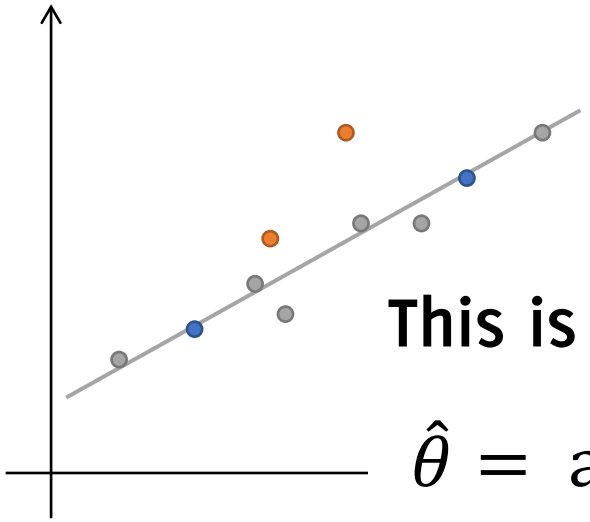
$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{D}\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \boldsymbol{\lambda}\|\boldsymbol{x}\|_1$$

Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

Who cares about images? *visual recognition systems*



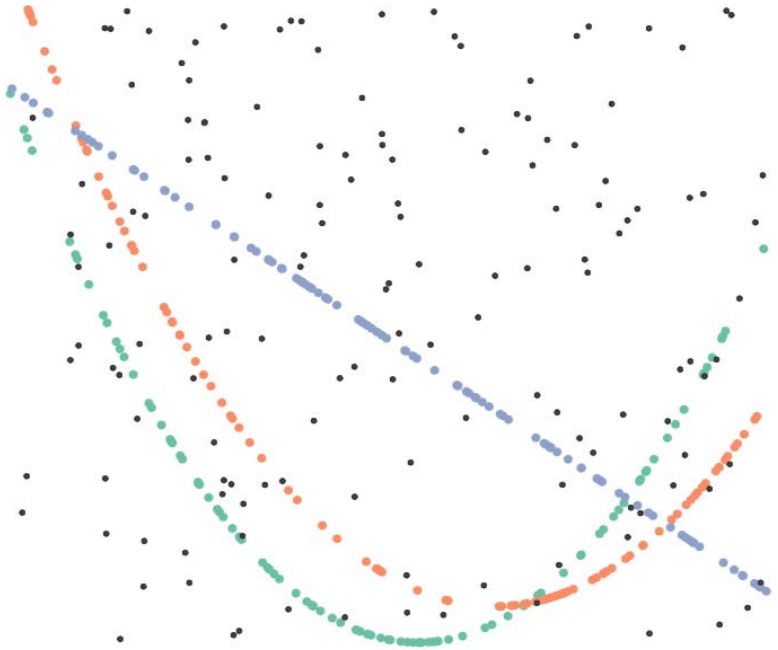
Least squares

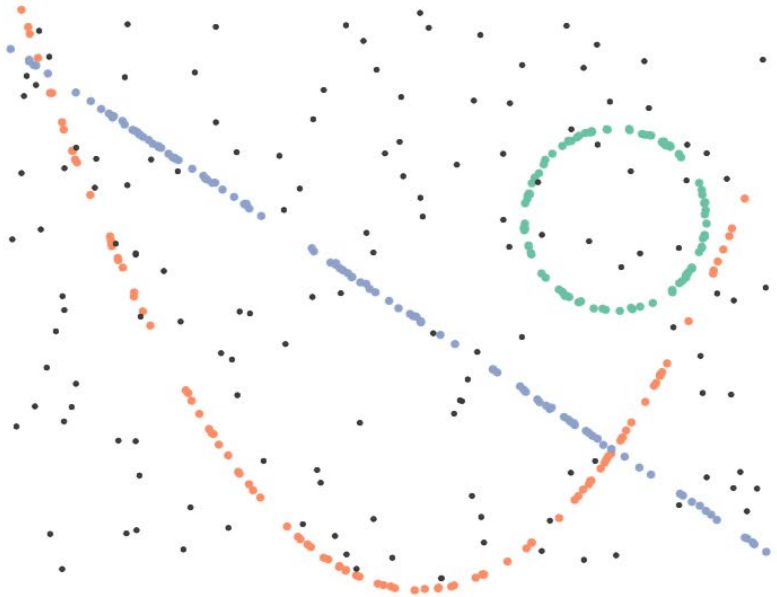


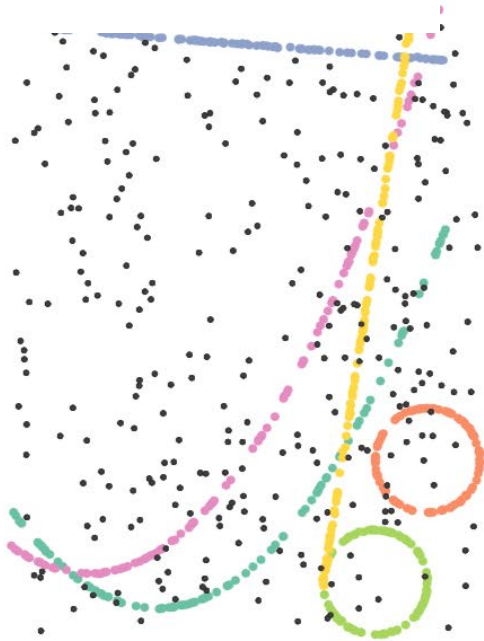
Robust Fit
(RANSAC)

This is a (robust) fitting problem

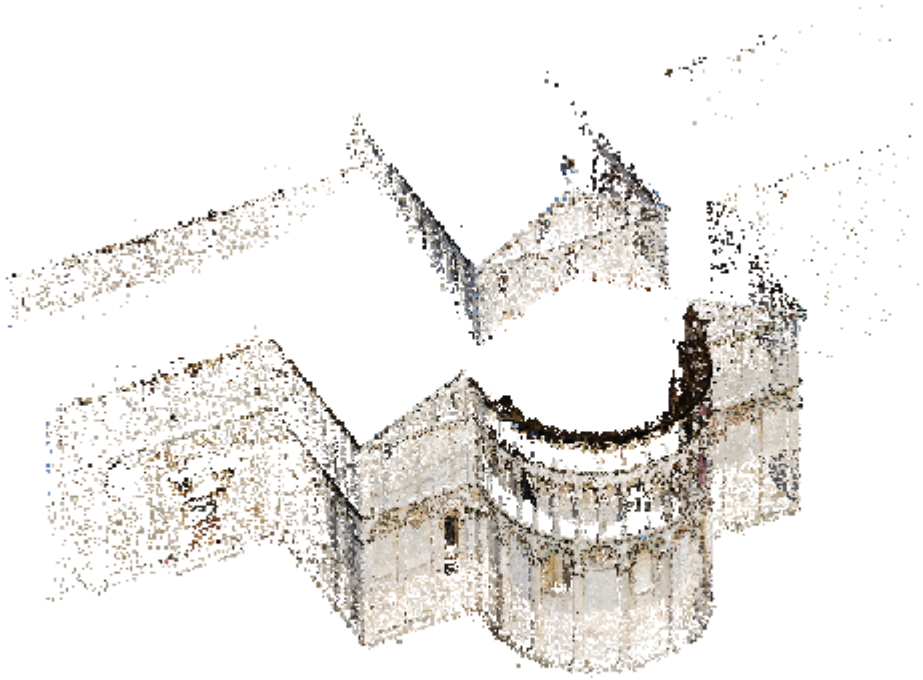
$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{\boldsymbol{x_i}} \rho(\operatorname{dist}(\boldsymbol{x_i}, \mathcal{M}_\theta))$$



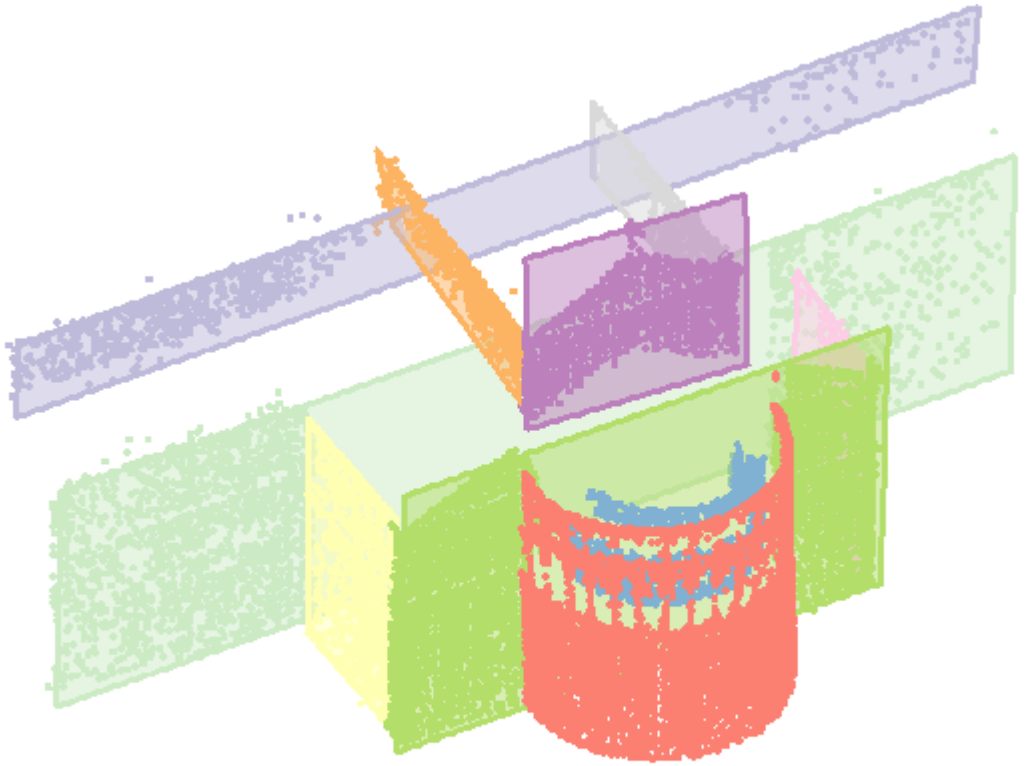




Who cares about images? *visual recognition systems*



(a) Input point cloud



(b) Recovered structures

This is a (robust) fitting problem

Magri, Leveni, Boracchi “MultiLink: Multi-class Structure Recovery via Agglomerative Clustering and Model Selection”, CVPR 2021

Who cares about images? *visual recognition systems*

Who cares about images? *visual recognition systems*

This is a (robust) fitting problem

Is this interesting for a (perspective) Mathematical Engineer?

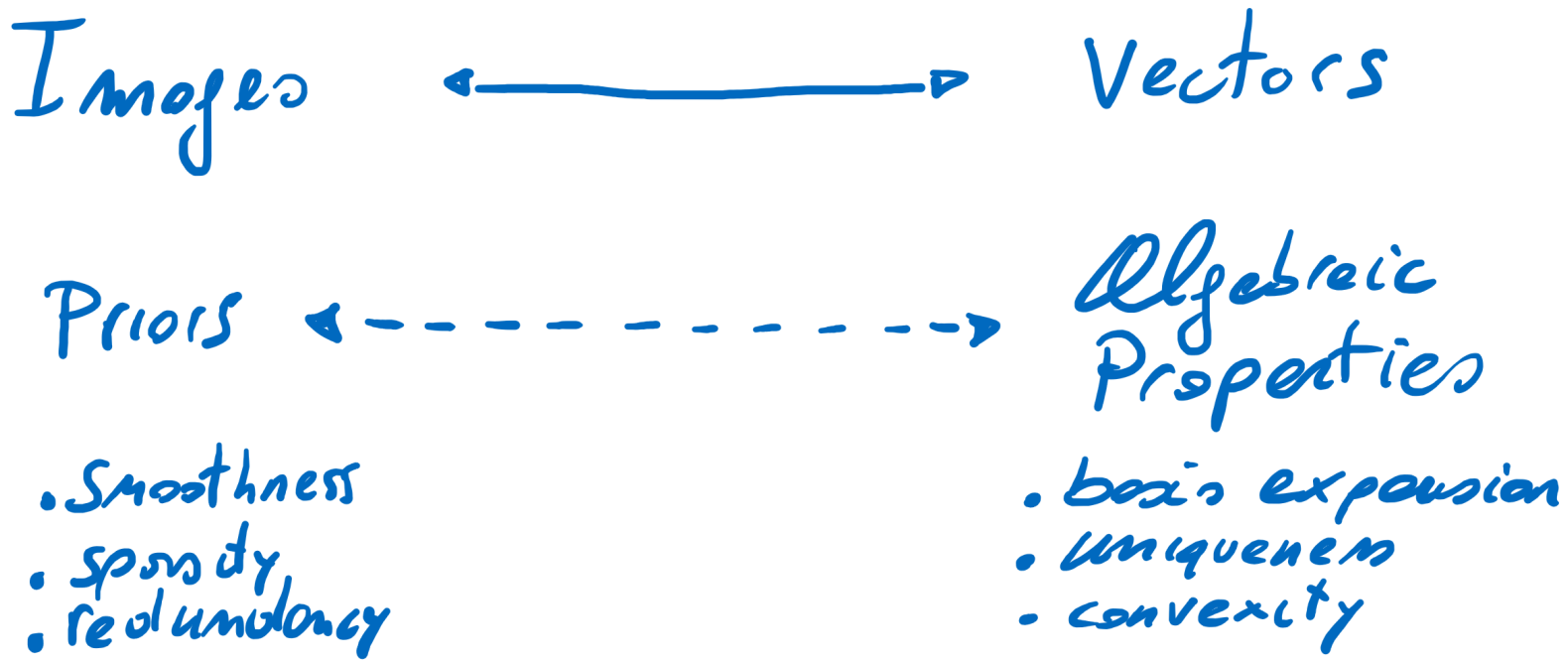
Is this interesting? Sure!

All the algorithms build upon:

- a **clear problem formulation**
- a **simple mathematical model (…often linear combinations!)**
- Sound mathematical solutions (linear algebra, least squares, convex optimization)

…and the result is not just a number... it's an image!

Is this interesting? Sure!



Ok, to recap

Mathematical Models and Methods for Image Processing (5 CFU)

The primary **goal** *of this laboratory course is to* **let the students design, implement and practice algorithms** *based* **on**

simple mathematical models *from* **linear algebra and convex optimization**,

and **solve** *challenging inverse* **problems in image processing** *(denoising, deblurring, inpainting, anomaly detection)*

Mathematical Models and Methods for Image Processing (5 CFU)

The course **topics include**:

- **Image models based on orthonormal bases** (Fourier, wavelets), **data-driven basis** (PCA, Gram-Schmidt) and **local polynomial approximation**.

- **Sparsity and redundancy**.
  - Away from Orthonormal Basis, redundant set of generators
  - Sparse coding with ℓ^0 (OMP) or ℓ^1 norm (convex optimization ISTA, IRLS, LASSO)
  - Dictionaries yielding sparse representations and dictionary learning (KSVD)
- **Applications of sparse models** to image denoising, inpainting, anomaly detection and classification.

- **Robust fitting** methods (RANSAC, LMEDS, HOUGH) and their sequential counterparts for object detection in images.

Course Organization

Lectures: 20 hours

Laboratory: 30 hours

There will be short theory recap and then you will be invited to develop and practice presented algorithms. Some demo code to fill in will be provided.

Simple assignment provided during lectures, oral exam.

Frequently Asked Questions

Q: Any specific background?

A: linear algebra, statistics and calculus

Q: Any programming skill required?

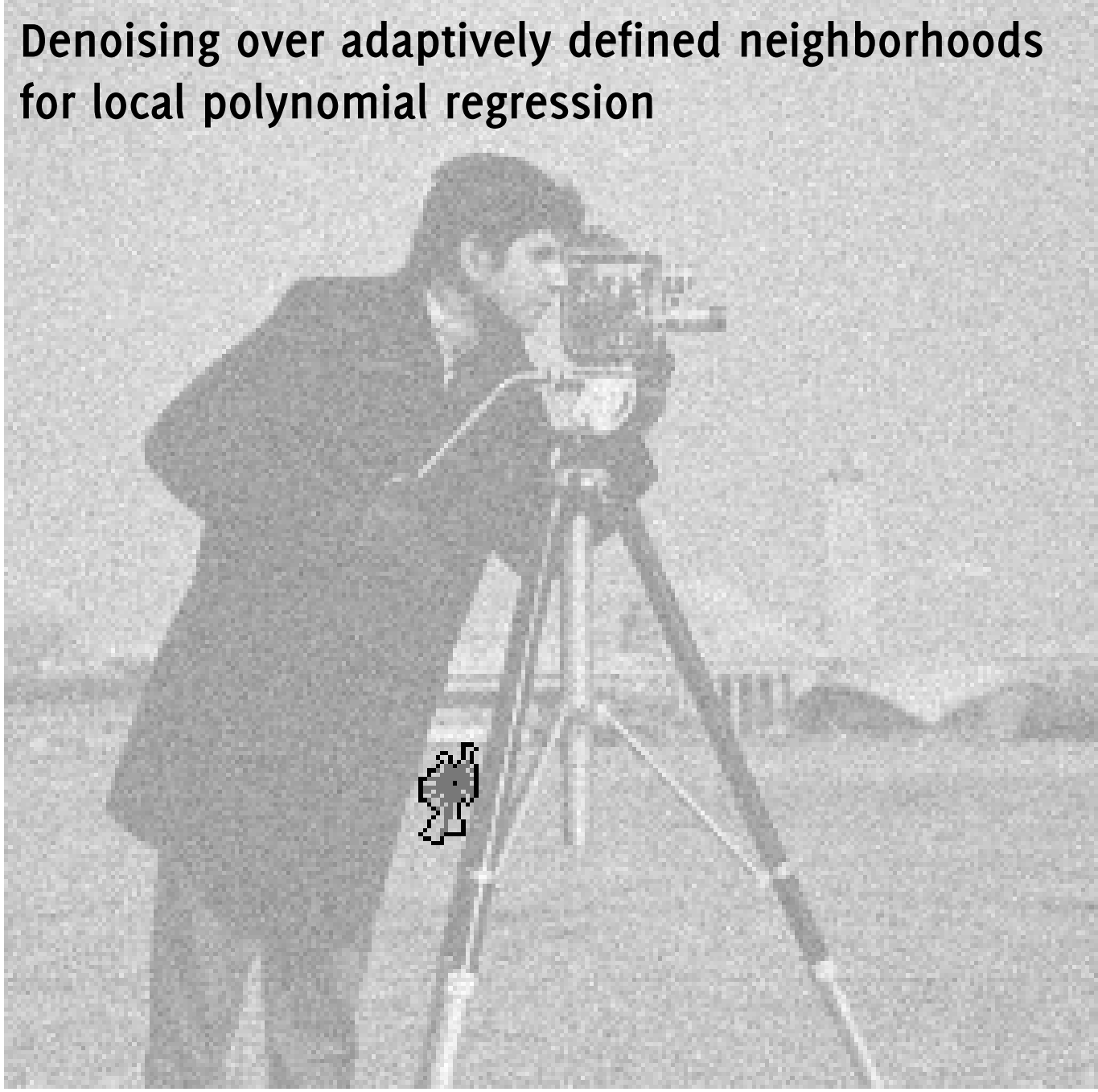
A: Proficiency in Matlab or Python

Q: Plenty of neural networks then?

A: No way. No neural networks allowed here ☺*
Only expert-driven algorithms designed upon a clear mathematical modeling that admits closed-form solutions / sound optimization schemes.

*** Interested in neural networks? Refer to «Artificial Neural Network and Deep Learning» in the first semester**

Questions?


Denoising over adaptively defined neighborhoods
for local polynomial regression

Please have a look at:

https://boracchi.faculty.polimi.it/teaching/MMMIP.htm

And possibly video recordings in the course calendar

Option 3: Join the team for a Thesis

Magistrale, quando sarete in dirittura di arrivo… ☺

The Team

We are 3 faculties, 7 PhD students, 2 Research Assistants

Giacomo Boracchi

Luca Magri (Researcher)

Federica Arrigoni (Researcher)

Filippo Leveni

Antonino Rizzo

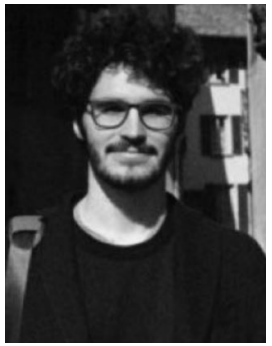
Michele Craighero

Andrea Schillaci

Diego Stucchi

Loris Giulivi

Andrea Porfiri Dal Cin



Giuseppe Bertolini

Edoardo Peretti

Research Collaborations

Major research collaborations:

Major research projects:

Thesis Information

- We typically illustrate thesis opportunities in February and September
- Thesis concern either Deep Learning or Computer Vision.

- Thesis are primarily research thesis, or thesis on industrial projects
- Sometimes we open internship with companies we are collaborating with

- We typically send proposals for Honours Program in Research (for those of you interested in research perspectives)

- We are always interested in brilliant candidates and perspective PhD students

That’s all Folks!

.. ci vediamo all’esame ☺
(option 0)