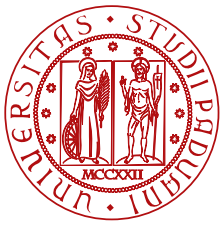

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN BIOINGEGNERIA

Automatic image processing and signal extraction of calcium dynamics in plant roots

Relatore
Prof. Morten Gram Pedersen

Laureando
Davide Rigato

ANNO ACCADEMICO 2023-2024

Data di laurea 27/11/2024

To my family

Summary

The aim of this master's thesis is to examine the Ca^{++}-signature in cell plants. The analysis is carried out by processing some videos describing the variation of Ca^{++} levels in a root. The code to analyze the videos is written in MATLAB. This master's thesis is composed of four chapters: the introduction chapter, the active contour chapter, the skeletonization chapter and the conclusive chapter.

Contents

List of Figures

Chapter 1

Introduction to Calcium Signaling in Plants

1.1 The role of calcium in plants

Ca^{++} is involved in every aspect of plant life. It regulates processes ranging from cell division and elongation to stomatal movement and response to biotic and abiotic stresses. The dual role of Ca^{++} as a structural component of cell walls and membranes and, as a dynamic signaling ion, underscores its importance. In signaling, Ca^{++} functions by altering its concentration in the cytosol in response to specific stimuli, leading to downstream effects on gene expression, metabolism, and cellular architecture. The **role** of calcium in plants to mitigate stress responses differs substantially from the animal counterpart. In a nutshell, Tuteja & Mahajan found 4 overall roles for calcium ions in plants' physiology [1]:

1. Calcium ions are essential plant nutrient required for **growth and development** of plant, especially the root and shoot tip. The tips are meristmatic (repetition of similar parts in the structure of an organism) and cell division occurs by mitosis. Ca^{++} helps in the formation of microtubules and those are essential for the anaphasic (relative to anaphase) movement of chromosomes.

2. Ca^{++} is an important divalent cation and is required for **structural roles** in the cell wall and membranes where it is found as Ca^{++} pectate. Ca^{++} accumulates as calcium pectate in the cell wall and binds the cells together.

3. It is also required as a counter-cation for inorganic and organic anions in the vacuole and as an **intracellular messenger** in the cytosol. Externally supplied Ca^{++} reduces the toxic effects of $NaCl$ and ameliorates stress.

4. Ca^{++} is required for **pollen tube growth and elongation**.

The calcium is taken up by roots from the soil solution and delivered to the shoot via the xylem. Ca^{++} may traverse the root either through the cytoplasm of the cells, linked together by plasmodesmata (the symplast) or through the spaces between the cells (the apoplast) [1].

1.1.1 Further notions on calcium usage & development in plants

An aspect to consider is the environment in which the plant developed in ancient times.
Metabolism in all cells requires the presence of orthophosphate (P_i) and phosphorylated organic compounds, particularly for cytosolic reactions associated with transduction of free energy [2]. The combination of Ca^{++} with P_i led to a low solubility enviroment in the cytosol, therefore the capacity of $[Ca^{++}]_{cyt}$ (calcium ions in the cytosol) to be at steady concentration would have been well below the millimolar concentrations, which prevailed in seawater. Thus, transport systems that export Ca^{++} from the cytosol are present in all cells to sustain steady state values of $[Ca^{++}]_{cyt}$ in the submicromolar range[2].

1.2 Calcium as a Universal Intracellular Messenger

Calcium ions (Ca^{++}) are one of the most versatile and ubiquitous signaling molecules found in both plant and animal cells. In plants, Ca^{++} plays a crucial role in mediating a wide range of physiological processes, from growth and development to stress responses. For example, many extracellular signals and environmental cues including light, abiotic and biotic stressors, elicit change in the cellular calcium levels, are termed as calcium signatures[1]. As a signaling molecule, Ca^{++} operates by acting as a secondary messenger that transduces external stimuli into specific cellular responses. Calcium is not the only messenger in plant cells, though. The signaling pathways include also *lipids, pH*, and *cyclic GMP (cGMP)*[2].
In plants, calcium ions are stored in **vacuoles**, endoplasmic reticulum (ER), mitochondria, chloroplast and cell wall[1][2]. The voltage across the membrane is around $150mV$ yealding an electrochemical potential difference of $50\frac{kJ}{mol}$.

The network of signalling events starts with a stimuli (stress perception) at the membrane level of the cell and ends with a cellular response[1] (Fig.1.4).
The ability of Ca^{++} to bind to various proteins and influence their activity makes it a key player in signal transduction networks.
The proper term to describe the main calcium ions messengers and channels is "Ca^{++} signalling toolkit" (Fig. 1.1) as described by Berridge & coworkers [3].
This toolkit is composed of:

1. **Influx**, maily Ca^{++} channels:

    - *Cyclic nucleotide-gated channels (CNGCs)*
    - *Glutammate receptors-like channels (GLRs)*
    - *Two-pore channels (TCPs)*
    - *Mechanosensitive channels (MCAs)*
    - *Reduced hyperosmolality-induced Ca^{++} increase channels (OSCAs)*
    - *Orai channels*

2. **Efflux**, ATPases and H^+/Ca^{++} antiporters:

    - *autoinhibited Ca^{++}-ATPases (ACAs)*
    - *ER-type Ca^{++}-ATPases (ECAs)*
    - *P1-ATPases (HMA1)*
    - *Mitochondrial calcium unimporter complex (MCUC)*
    - *Ca^{++} exchanges (CAX)*

3. **Decoding**, it is composed by more than 250 protein. The three main groups are:

    - *Calcium-dependent protein kinases (CDPKs)*
    - *Calcineurin B-like (CBL) protein kinases (CIPKs)*
    - *Calmodulin (CaM) and CaM-like proteins*

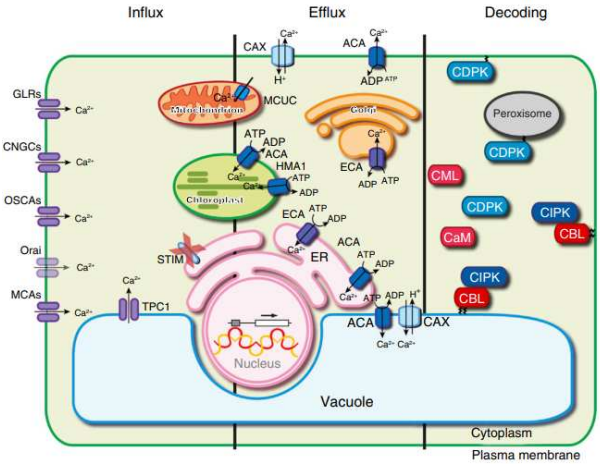

Figure 1.1: The plant Ca^{++} signalling toolkit

1.2.1 EF-hand

To properly bind with calcium, most of the Ca^{++} sensors use a helix-loop-helix (Fig.1.2) motif termed as the 'EF hand' (due to its shape) or the elongation factor. It binds a single Ca^{++} molecule with high affinity[1]. This structure was first discovered by Kretsinger & Nockolds in 1973[4].

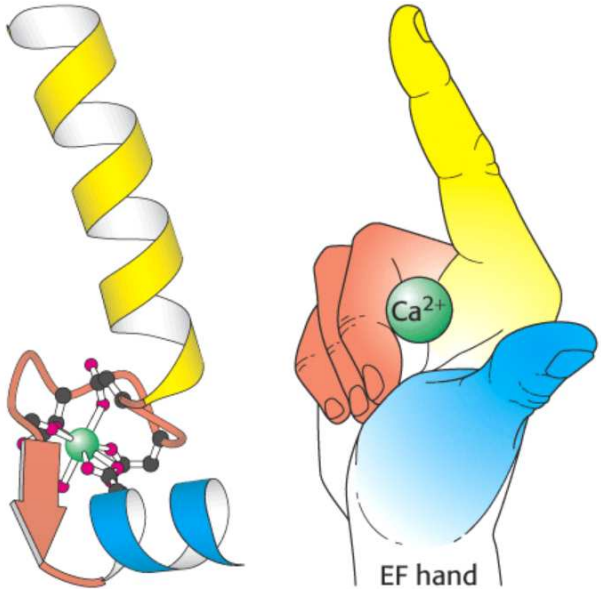

Figure 1.2: Ef-hand

The EF-hand motif is formed by 29 ammino acids. The structure is made of: one α-helix, composed by residue 1-10 , (the yellow helix in Fig.1.2) a loop, composed by residue 10-21 (orange component between the α-helix in Fig.1.2) and a final α-helix, constituted by residue 19-29 (the blue helix in Fig.1.2). It has been observed that all EF-hand have common proprieties:

- The EF-hand motifs mostly exist in pairs, which helps in the stabilization of the protein structure;
- The Ca^{++} ion is coordinated by an oxygen atom or by a bridging water molecule of the side chains;
- In most of the functional EF hand motifs, the first amino acid is Aspartate (Asp) and the twelfth is Glutamate (Glu), Glu contributes both its side chain oxygen atoms to the metal ion coordination;
- Most of the EF hand proteins are characterized by the relatively high percentage of acidic residues (amino acids that have side chains with a carboxyl group that can donate a proton, making them negatively charged at physiological pH around 7.4).

1.2.2 Ca^{++} binding proteins, additional features

In this subsection, the role of Ca^{++} binding proteins is further analyzed. Since there are more than 250 proteins present in nature, there must be some features in common to this pool. Those can be called Ca^{++} sensors. The processes linked to it are: **efflux** and **decoding**.
Properties essential for Ca^{++}-binding proteins (CaBPs) to function as intracellular-Ca^{++}receptor are:

- The Ca^{++} receptor/sensor must have Ca^{++}binding sites essentially unoccupied in the resting cell (free Ca^{++} is $100 - 200n$M) and occupied at levels;

- The protein must show selectivity and preference for Ca^{++} in the presence of other cations;

- After binding with Ca^{++}, a CaBP must undergo a conformational change that either alters its interaction with other molecules or changes its activity as an enzyme[5];

- The kinetics of interaction should be very fast;

1.2.3 Toxicity levels

High dosage of calcium ions can be poisonous for the organism. In plants, Ca^{++} is stored in intracellular compartments (such as plasmic reticulum or vacuole) and some calcium ions are found in the cytosol, the fluid portion of the cytoplasm. The concentration of $[Ca^{++}]_{cyt}$ at resting state is around $100 - 200n$M[6]. Other organell's concentration can be seen in Fig.1.3. When the singlling happens, calcium ions gradually enter in the cytosol from the intracellular compartments. This transition must be tightly regulated, because $[Ca^{++}]_{cyt}$ levels above the 10^{-7}M order are cytotoxic[3]. One of the key proteins related to sphingolipid-induced programmed cell death (PCD) are: *AtCPK3* [7].
Furthermore, the collective processes responsible for generating the increase in $[Ca^{++}]_{cyt}$ have been termed 'on mechanisms' [3]. To operate as an effective signalling system, there is a requirement to return $[Ca^{++}]_{cyt}$ to its pre-stimulus levels. This is achieved by the so-called "off mechanism" processes.[3].

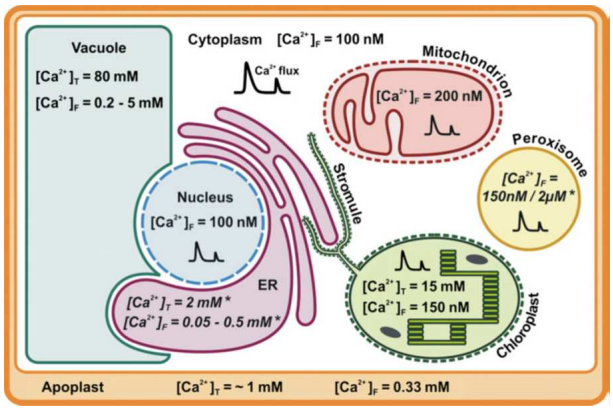

Figure 1.3: Various organelles concentrations

1.3 Historical Context and Discovery of Ca^{++} Signaling in Plants

The role of Ca^{++} as a signaling molecule in plants was first suggested in the late 20th century, following discoveries in animal systems where Ca^{++} was shown to be involved in muscle contraction and neurotransmission. Early studies in plants revealed that changes in cytosolic Ca^{++} levels were associated with environmental stimuli such as light, gravity, and pathogen attack. The identification of Ca^{++}-binding proteins, such as calmodulin, further solidified the concept of Ca^{++} as a critical component of plant signaling pathways.

1.3.1 Further distinction among Ca^{++} based plants

Two further distinctions have been made among Ca^{++} signaling based plants: calcifuges and calcicoles. Calcifuges are plants occurring on acidic soils with low Ca^{++}.[1] The term calciole is used to describe plants, which can grow on calcareous/Ca^{++} rich soils[1]. However, the requirements and ability to extract Ca^{++} from the soil differs from plant to plant. In particular, monocots require less Ca^{++} for optimal growth than do the dicots[1]. Ca^{++} enters plant cells through Ca^{++}-permeable ion channels in their plasma membranes. Ca^{++} competes with other cations both for these sites and for the uptake from the soil. The presence of high levels of Ca^{++} is known to better the effects of the uptake of toxic cations ($Al^{++}$and Na^{+}) from the soil while the presence of high levels of other cations (K^{+}, Mg^{++}) may reduce Ca^{++} uptake. Calcium uptake and requirements for optimal growth are thus strongly dependent on the presence of other cations. Calcium deficiency is rare in nature [1].

1.4 Evolution of Ca^{++} signalling

According to several textbooks, the generation of the primordial Ca^{++} signalling toolkit took place by an invagination of the cell membrane of a bacteria that trapped the genome of an archaebacterial able to handle Ca^{++} signatures[8]. By infolding of the cell membrane with ribosomes attached, the Endoplasmic Reticulum (ER) could have formed.
Scientists were able to date Ca^{++} signalling back since the unikont-bikont split[3], a major evolutionary division among eukaryotes, based on the number of flagella present in the ancestral forms. From this basis, the last eukaryote common ancestor (LECA) was potentially able to generate and decode Ca^{++} signatures.
Despite this proven facts, theories (yet to be proven) suggest that bacterial Ca^{++} regulation may have emerged in the ocean since the beginning of life. The alkaline conditions that favoured relatively low Ca^{++} concentrations could permeate into ancestral cells, influencing the ATP metabolism. [8].
It is important to notice, by the work by Marchadier et al., that the three main parts of the Ca^{++} signalling toolkit (influx, efflux and decoding) are the ones present in plants, algae and animals. As those author noticed in their work, animals differentially lost proteins specialized for Ca^{++} efflux during evolution, while in plants influx proteins have been predominantly lost.

1.4.1 Evolution of Ca^{++} signalling toolkit in plants: Influx

Contrary to their counterparts (algae and animals), plants have shown a tendency to reduce diversity of mechanisms responsible for Ca^{++} influx through the years. This reduction on variety of systems led to an enhancement of a smaller pool of channels like: *cyclic nucleotide gated channels (CNGCs), glutamate receptors (GLRs) and reduced hyperosmolality-induced* $[Ca^{++}]_{cyt}$ *increase (OSCAs) channels*.(Fig. 1.1)
According to Edel et al. [3], this reduction in the plant Ca^{++} signalling toolkit took place long before the colonization of the land took place.

1.4.2 Evolution of Ca^{++} signalling toolkit in plants: Efflux

In the context of the calcium signalling toolkit, Ca^{++} efflux refers to the three mechanisms[2]:

- the process that plant cells use to remove calcium ions (Ca^{++}) from the cytosol, thus restoring the low resting concentration of intracellular Ca^{++} after a signaling event;
- the process of loading $Ca{+}{+}$ into compartments inside the cell, such as the ER or vacuoli, to be used as calcium storage;

- the process of supplying Ca^{++} to various organelles to support specific biochemical functions;

The removal of Ca^{++} from the cytosol against its electrochemical gradient to either the apoplast or to intracellular organelles requires energized 'active' transport. Ca^{++}-ATPases (ATP pumps[2]) and H^+/Ca^{++} antiporters (or proton/calcium exchangers e.g.*CAX*) are the two key families of proteins catalyzing this movement[1]. It is important to notice that the ATPases family of proteins is subdivided in type 2A and type 2B[2].
Efficient Ca^{++} efflux is crucial because prolonged high levels of intracellular calcium can be toxic and can trigger unwanted cellular responses. As a matter of facts, abundance or activity of a Ca^{++} pump can indeed alter signal transduction. [9][10][11].
Namely, five major Ca^{++} efflux systems are encoded by the *Arabidopsis thaliana* genome: *Ca^{++}-ATPases (ACAs), ER-type Ca^{++}-ATPases (ECAs), P1-ATPases (HMA1), the mithocondrial calcium uniporter complex (MCUC) and Ca^{++} exchangers (CAX, concentration around 13μM)[2]).* (Fig. 1.1) Those enzymes perform several important functions. Tutja & Mahanja identified five roles where those proteins are employed[1]:

- They maintain **low** $[Ca^{++}]_{cyt}$ **concentration** at resting state;
- They **restore** $[Ca^{++}]_{cyt}$ **normal levels** after a signalling event;
- they **fill up again** calcium ions storage locations that allows other spikes in Ca^{++} signals;
- They **provide** Ca^{++} **in the ER** for the secretory system to function;
- They **remove some divalent cations**, such as Ni^{++}, Zn^{++}, Mg^{++} and Mn^{++} from the cytosol to prevent mineral toxicity;

Furthermore, by the work done by Hirischi (2001) [12] it can be found that Ca^{++}-ATPases, that have high calcium affinity but low capacity of transport, are responsible for maintaining $[Ca^{++}]_{cyt}$ homeostasis. On the other hand, H^+/Ca^{++} antiporters, that have low affinity but high capacity of calcium ions transport, are likely employed in the removal of calcium after a signalling event. Ca^{++}/H^+ antiporters, utilize the H^+ gradient generated by the tonoplast V-type H^+-pump and by a proton-pumping pyrophosphatase to sequester Ca^{++} in the Vacuole[1]. In general Ca^{++}-ATPases are less abundant in the cell membrane than the H^+-ATPases.

1.4.3 Evolution of Ca^{++} signalling toolkit in plants: Decoding

Decoding refers to how cells interpret and respond to the dynamic changes in intracellular Ca^{++} levels. The regulation of the deconding procedure can be seen as a feedback system[2]. This process involves a complex interplay of various molecular components, including calcium-binding

proteins, sensors, effectors, and signaling pathways. In other words, Ca^{++} signatures are the net result of the operation regarding the 'on' and 'off' systems. [3]
In this contexts we can see the true distinction between the animal, plants and algae kingdoms, because their Ca^{++}-decoding tools are significantly different. In plants those tools are represented by few distinct families. As we have stated before, the main groups are: *calcineurin B-like (CBL) interacting protein kinases (CIPKs) and Ca^{++}-dependent protein kinases (CDPKs)*(Fig. 1.1).
Key proprieties to consider when analyzing those stream of messanger are[2]:

- amplitude;
- duration;
- frequency;
- location of the Ca^{++} signal;

Interaction with other cellular components and signaling pathways that reflect the "physiological address" of the given cell should be taken into account [13].

1.4.4 How calcium influence gene expression

Evidence that different types of Ca^{++} signal can differentially influence gene expression is less direct for plant cells[2]. As we said before, different stimuli produces different response. A way to indirectly measure the intensity of the stimulus is the detect the magnitude of the Ca^{++} signature. This determines the degree of the response. These different Ca^{++} signals correlated well with changes in osmotic stress–induced patterns of gene expression and the acquisition of osmotic stress tolerance[2].

1.5 Ca^{++} Oscillations and Waves

One of the most intriguing aspects of Ca^{++} signaling in plants is the generation of Ca^{++} oscillations and waves, which encode information in the frequency, amplitude, and duration of the Ca^{++} spikes. This information is thought to reflect the sequential opening of hyperpolarization-activated Ca^{++} channels at the plasma membrane.
Cytosolic calcium waves are generated in the cytoplasm by successive recruitment of Ca^{++} channels to metabolize stress responses[1]. The oscillation of $[Ca^{++}]_{cyt}$ patterns are often referred to as "Ca^{++} signatures." Different stimuli, such as light (circadian oscillation [2]), mechanical stress, or pathogen attack, can induce unique Ca^{++} signatures that are interpreted by

the plant to trigger specific responses(Fig. 1.4). It may happen that secondary messanger, such as IP^3 or $cADPR$, might be generated during the Ca^{++} signature.
A generic signal transduction pathway has following steps[1]:

1. Perception of the signal by the membrane receptors;

2. Generation of second messengers;

3. A cascade of protein hosphorylation/dephosphorylation events that may target transcription factors controlling specific set of stress regulated genes;

4. Stress tolerance, plant adaptation and other phenotypic responses;

It is important to notice that the plant cell's response could be growth inhibition or cell death[1]. As an example of Ca^{++} signature, touch-induced Ca^{++} waves travel rapidly through plant tissues, facilitating swift responses to physical stimuli. Similarly, symbiotic interactions between plants and mycorrhizal fungi involve rhythmic Ca^{++} oscillations that are critical for the establishment of the relationship. The generation of Ca^{++} waves typically involves the concerted action of channels and exchangers located on the plasma membrane and organelle membranes. The rapid release of Ca^{++} from internal stores, followed by its re-uptake, creates a self-propagating wave that can travel long distances within the plant. These Ca^{++} waves are essential for coordinating responses at the whole-plant level, such as systemic acquired resistance (SAR) during pathogen infection.

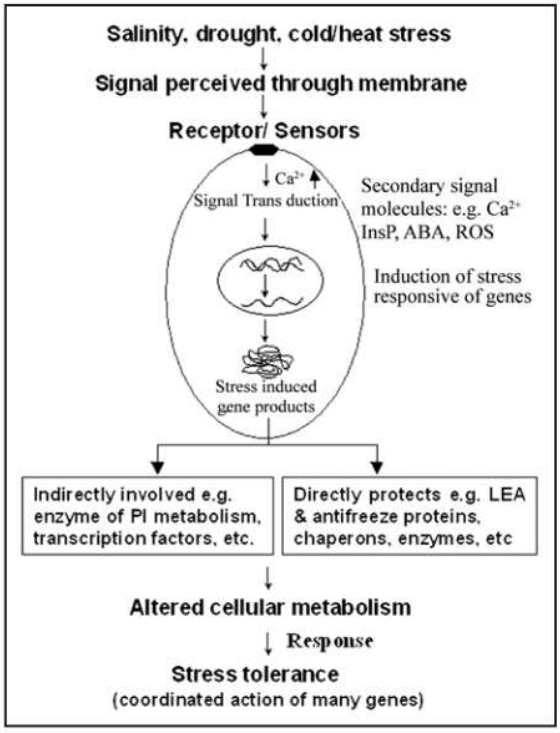

Figure 1.4: Generic pathway for plant response to stress

1.5.1 Examples of stimuli

In this subsection some examples of physiological stimuli of $[Ca^{++}]_{cyt}$ are show.

Stimulus	Example of response	Reference
Red light	Photomorphogenesis	Shacklock et al.(1992)[14]
Abscissic acid	Stomatal closure	McAinsh et al. (1990) [13]
Gibberellin	α-Amylase secretion	Bush and Jones (1988) [15]
Salinity/drought	Proline synthesis	Knight et al. (1997)[16]
Hypoosmotic stress	Osmoadaptation	Taylor et al. (1996)[17]
Touch	Growth retardation	Knight et al. (1991)[18]
Fungal elicitors	Phytoalexin synthesis	Knight et al. (1991)[18]
Cold	*KIN1* gene expression	Knight et al. (1996)[19]
Heat shock	Thermotolerance	Gong et al. (1998)[20]
Oxidative stress	Free radical scavenger induction	Price et al. (1994)[21]
NOD factors	Root hair curling	Ehrhardt et al. (1996)[22]

1.5.2 Grouping information together

So far we have analyzed sub cellular waves, therefore waves of calcium happening at a cellular level. In an organism, when a stressor comes into play, it generates ”waves” of signalling cells[1]. This means that many cells have the same subcellular response and generate many coordinated Ca^{++} signatures. This propagates through the plant tissue and it is the signal decoded by the organism.
Although signals are influenced by the nature of the stimulus and the plant itself, the most common shapes seen in this kind of environments are:

- **Spikes:** short, transient increases in Ca^{++} concentration that return quickly to baseline levels;
- **Waves:** Ca^{++} increases that propagate spatially across cells or tissues, like a ripple effect;
- **Oscillatory waveform:** repeated, rhythmic fluctuations of Ca^{++} levels over time, resembling a sine wave;
- **Plateaus:** a sustained elevation in Ca^{++} levels for a prolonged period before returning to baseline;

It is important to notice that prolonged exposure to this signalling events is lethal for the cells. The sphingolipid-induced cell death (PCD) implies apoptosis (cell death)[5]. Another remark is that there is always a presence of a concentration gradient of Ca^{++} in the cytosol that generates a dynamic equlibrium.

1.5.3 Repeated Ca^{++}signatures: calcium ”memory”

The term “memory” was put forward first by Knight et al. (1996)[19]. There is considerable evidence that $[Ca^{++}]_{cyt}$ signatures are modified by previous experience. A diminished $[Ca^{++}]_{cyt}$ elevation upon repetitive stimulation by the same environmental challenge or a developmental cue is a common observation. This is probably due to the fact that high levels of calcium for prolonged periods of time are toxic for the organism. There is also evidence that the $[Ca^{++}]_{cyt}$ signatures produced by one stressor can be modified by prior exposure to a contrasting one.

1.6 Spatial and Temporal Dynamics of Ca^{++} Signaling

The spatial and temporal dynamics of Ca^{++} signaling are critical for the specificity of the response. Ca^{++} signals can be highly localized, affecting only a specific part of the cell, such as the tip of a growing pollen tube, or they can be more widespread, influencing entire tissues.

The timing of Ca^{++} signals is also crucial, with some processes requiring sustained Ca^{++} elevations, while others depend on transient spikes.

For example, during stomatal closure in response to drought, ABA-induced Ca^{++} signals must be precisely timed to regulate the opening and closing of guard cells, balancing water loss with CO_2 uptake. Similarly, the duration and intensity of Ca^{++} signals in root cells can influence root growth direction, allowing the plant to optimize nutrient and water acquisition.

The complexity of Ca^{++} signaling is further enhanced by the ability of cells to integrate multiple Ca^{++} signals, leading to the activation of specific pathways depending on the context. This integration is mediated by Ca^{++}-binding proteins that interpret the Ca^{++} signals and initiate appropriate downstream responses.

1.7 Ca^{++} measuring techniques

Measuring free calcium ions in biological or chemical systems is crucial for understanding various physiological and biochemical processes. So far, reliable techniques to measure free Ca^{++} intracellularly have been developed. Those techniques consist mainly in [2]:

- Ca^{++}-selective **micorelectrodes** yelding consistent estimation of $[Ca^{++}]_{cyt}$;
- **Fluorescent dye** and the luminescent protein **aequorin**;
- **Atomic absorption spectrophotometry (AAS)**[23];
- Various **spectroscopy** methods [24];
- **Magnetic resonance imaging (MRI)**;
- **Super-resolution microscopy**[25];
- **patch clamp** technique;
- **Förster Resonance Energy Transfer (FRET)**;

In this context,a strong argument for a critical involvement of Ca^{++} requires the demonstration not only that a stimulus evokes a change in $[Ca^{++}]_{cyt}$ but also that two additional experimental criteria are met[2]:

1. a block of the $[Ca^{++}]_{cyt}$ transient also should block the downstream response;
2. appropriate Ca^{++}-sensitive response elements should be present;

1.7.1 Fluorescent dye and aequorin

Aequorin is a calcium-sensitive bioluminescent protein that was originally isolated from the jellyfish Aequorea victoria[26]. It emits blue light in the presence of calcium ions, which makes it useful for studying calcium dynamics in biological systems.
Aequorin is composed of a protein component (apoaequorin) and a prosthetic group called coelenterazine, a luminophore.
When calcium ions (Ca^{++}) bind to aequorin, the complex undergoes a conformational change that triggers the oxidation of coelenterazine, resulting in the emission of blue light (around 470 nm) 1.5.

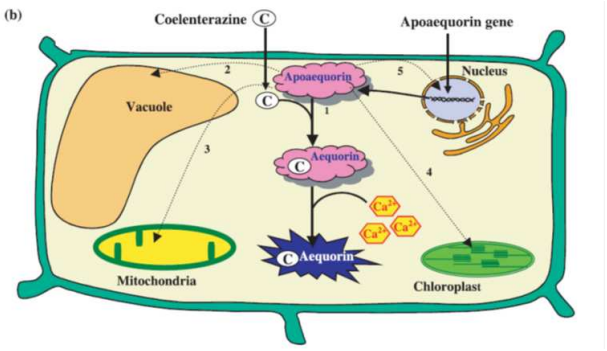

Figure 1.5: Aequorin pathway

1.7.2 MRI

Magnetic Resonance Imaging (MRI) is commonly used in medical and biological research to visualize internal structures in living organisms, including plants. In plant biology, MRI can be particularly useful for studying the dynamics of calcium, which plays a critical role in various cellular processes.

MRI typically relies on the magnetic properties of hydrogen nuclei (protons) in water molecules, but detecting calcium directly is more challenging because calcium ions (Ca^{++}) do not have the magnetic properties required for conventional MRI detection. Instead, MRI can be used indirectly to visualize calcium distribution and dynamics by using contrast agents. This technique requires the injection of magnetic calcium-responsive nanoparticles (MaCaReNas) and MRI, which helps to monitor the extracellular calcium signaling in tissues. It's a highly sensitive technique and can detect the very narrow range (0.1-1.0 mM) of change in calcium levels within a short span (within seconds) of time. MRI has been used for the imaging of whole organ system in plant, thus this method can be used for plant calcium ion detection in future[27].

1.7.3 AAS

Atomic Absorption Spectrophotometry (AAS) is commonly used in plant sciences to analyze essential elements in plant tissues. This technique is valuable in studying nutrient uptake, deficiency, and toxicity, as well as for monitoring environmental pollution and its effects on plant growth. Conventionally, this is how AAS works:

1. **Sample Collection and Preparation;**
2. **Atomization:** The sample solution is introduced into the AAS instrument, where it is atomized. This is done using one of two main methods:
   (a) *Flame Atomization:* The solution is sprayed into a flame;
   (b) *Graphite Furnace Atomization:* The sample is introduced into a small graphite tube, which is electrically heated to high temperatures, producing free atoms from the sample;
3. **Light Absorption:** As the light emitted form a source passes through the cloud of free atoms in the flame or graphite furnace, the atoms absorb light at specific wavelengths corresponding to the element being analyzed;
4. **Detection and Quantification;**

1.7.4 Spectroscopy

Spectroscopy is a scientific technique used to analyze the interaction between matter and electromagnetic radiation. It is based on the principle that different materials absorb, emit, or scatter electromagnetic radiation in unique ways depending on their structure, composition, and physical state.

1.7.5 Super-resolution microscopy

Ernest Abbe defined ”microscopic object” as object of size smaller or similar to that of the wavelength (λ) of the light used for visualization in microscopes [25]. From this notion of microscopic objects derives the formula for the resolution limit of a microscope, give by Abbe:

$$d = \frac{\lambda}{2NA} \tag{1.1}$$

where λ is the wavelength of the beam used to irradiate the specimen and NA is the ”numerical aperture” of the object used. This equations shows a linear proportion among the λ and the

resolution of the microscope (d).
Super-resolution microscopy is a collection of advanced imaging techniques that surpass the diffraction limit, a fundamental resolution limit imposed by the wave nature of light (eq.1.1), to achieve higher resolution than conventional light microscopy. The diffraction limit, approximately 200-300 nanometers, restricts the ability to resolve structures that are closer together than this distance. Super-resolution microscopy overcomes this barrier, enabling scientists to observe cellular structures and molecules with unprecedented detail, often down to the nanoscale. In this context, fluorophores dye (fluorescent molecules that absorb light at a specific wavelength and emit light at a longer wavelength) are used[25].

Historical background

the development of super-resolution microscopy methods dates back to 1994, relevant applications in plant cell imaging only started to emerge in 2010[25]. The Nobel Prize in Chemistry 2014 was awarded to three pioneers of microscopy, Eric Betzig, William Moerner, and Stefan Hell, who devoted substantial time and ingenuity to bending or breaking the diffraction limits of **far-field microscopy**, where the diffraction limit of the object observed is not surpassed, as defined by Ernst Abbe in 1873[25].

SIM

SIM (structured illimination microscopy) [28] manipulates the excitation light to scramble high-frequency components of the sample with a striped illumination pattern with a periodicity of approximately $\frac{\lambda_{exc}}{2}$ (λ_{exc} = lambda exitation) in the form of Moiré patterns[25]. A Moiré pattern is the geometrical design that results when a set of straight or curved lines is superposed onto another set, creating interference.
Multiple images (typically 9 to 25[25]) are acquired with the illumination pattern shifted or rotated in various ways. The light pattern is rotated by three or five angles (120° or 72°, respectively) and at each angular position is further phase shifted by three or five phase shifts (of $\frac{2\pi}{3}$ or $\frac{2\pi}{5}$ increments, respectively). In this way a single SIM frame may be calculated from the composite Moiré patterns and provides an optimal lateral resolution of 100 nm[25]. Commercial SIM have a frame rate of around 100 fps [25]. The resolution of a SIM goes well below Abbe's limit (eq.1.1), it can detects events below the 200 nm threshold[25].

PALM and STORM

Both PALM (Photo-activated localization microscopy) and STORM (Stochastic optical reconstruction microscopy) are **precision localization methods**. Both methods acquire multiple im-

ages where only a few fluorophores are stochastically found in the emission state at one time, and this requires special photophysical properties of the fluorophores used [25]. In the case of PALM, the isolation of individual emitters from a densely labeled diffraction-limited region can be achieved using a pulsed activation laser and continuous excitation illumination[25]. The key part of PALM analysis relies on the usage and proprieties of fluorophores. The class of proteins used, according to their characteristics are:

- photoswitchable proteins (can switch between light ”emitting” and ”non-emitting” state);
- reversibly photoswitchable proteins (a subset of photoswitchable proteins);
- photoactivatable fluorescent proteins (a class of fluorescent proteins that can be activated or deactivated in response to specific wavelengths of light);

STORM is is conceptually similar to PALM but mechanistically different. Like PALM, this analysis relies on the usage of photoswitchable proteins that can be turned ”on” and ”off” (blinking) when coupled with appropriate dye. While PALM works on proteins that remain in a non-fluorescent state until activated by exposure to specific wavelengths of light, STORM relies on the blinking proprieties of the photoswitchable proteins used.
STORM routinely achieves a spatial resolution of 20 to 30 nm[25].
Due to the huge number of cycles between the ”on” and ”off” state of fluorophores, PALM and STORM are considered slow methods.

STED

STED (Stimulated Emission Depletion) microscopy achieves lateral resolutions (the ability of a microscopy technique to distinguish between two points) on the order of 20–50 nanometers but a poor axial resolution (the ability of a microscopy technique to distinguish between two points along the z-axis) above 500 nm[25].
STED is based on scanning the sample with spatially arranged light comprising two different laser beams. One (the excitation line) corresponds to the excitation maximum of the fluorophore, while the other (the depletion line) is engineered into a doughnut shape through a phase mask to surround the excitation beam[25]. In this way, a small region of interest (ROI) can be achieved. Due to its high energy beam, STED is phototoxic for the living cells. The STED leaser is red-shifted and it has an intensity of the order of $\frac{GW}{\mu m^2}$.

1.7.6 Patch Clamp

Patch-clamp is a highly sensitive technique used to study the ionic currents that flow through individual ion channels in cells. This technique, which was developed by Erwin Neher and Bert

Sakmann in the 1970s (who were awarded the Nobel Prize in Physiology or Medicine in 1991 for their work), allows researchers to measure the activity of single or multiple ion channels on a cell's membrane. The first implementation of the patch clamp tecnique on NSCCs was done by Stoeckel & Takeda in 1989 [29].

1.7.7 Förster Resonance Energy Transfer (FRET)

This is the technique used to acquire the images in this project.
FRET stands for Förster Resonance Energy Transfer (also known as Fluorescence Resonance Energy Transfer). It is a distance-dependent interaction between the electronic excited states of two dye molecules, a donor (fluorophore) and an acceptor (chlorophore)[30]. When the donor molecule is excited by a photon, it can transfer its energy non-radiatively to a nearby acceptor molecule if they are within a certain proximity (1-10 nm range [30]). The efficiency of transfer rate k_t is highly dependent on the distance between the donor and acceptor, typically following a $\frac{1}{r^6}$ relationship (eq.1.2), where r is the distance between the two molecules. The formula to describe transfer rate is as follows:

$$k_t = \frac{1}{\tau_0}(\frac{R_0}{r})^6. \tag{1.2}$$

The transfer rate depends also on 3 other parameters:

1. the overlap of the donor emission and acceptor absorption spectra (parameter: overlap integral J);

2. the relative-orientation of the donor absorption and acceptor transitions moments (parameter: κ^2, range 0–4)

3. the refractive index (parameter: n^{-4},range 1/3–1/5);

$$R_0^6 = c_0 \kappa^2 J n^{-4} (k_f \tau_0) \tag{1.3}$$

$$\tau_0^{-1} = k_f + k_{nr} + k_{isc} + k_{pb} \tag{1.4}$$

where k_f and k_{nr} are the radiative and non radiative deactivation rate costant, k_{isc} represent the intersystem crossing and k_{pb} is the photobleaching effect. In addition, $c_0 = 8.8 * 10^{-28}$ nm, $J = 10^{17} \int q_{d,\lambda} \epsilon_{a,\lambda} \lambda^4 d\lambda \frac{nm^6}{mol}$ and $q_{d,\lambda}$ is the normalized donor emission spectrum [30].

FRET, like other fluorescence microscopic techniques, generates contrast on the specimen based on three proprieties of light emission[30]:

- **Sensitivity:** the ability to detect low levels of emitted light from fluorescent molecules;

- **Selectivity:** the ability to distinguish specific signals from desired fluorescent molecules or processes against other unwanted or background emissions;
- **Modulation:** the ability to alter the properties of emitted light (such as intensity, wavelength, phase, polarization, or timing) to enhance detection and improve selectivity;

Like super-resolution microscopy, FRET far exceedes the ”diffraction limit” given by eq. 1.1 [30].

Yellow Cameleon 3.6 biosensor and a confocal microscopy

In our case, a Cyan Fluorescent Protein (CFP) donor has been paired with a Yellow Fluorescent Protein (YFP) acceptor, in combination with a chain composed of CaM and M13 calmodulin-binding molecule (Fig. 1.6). The CaM molecule works as the binding site for calcium. The light needed to excite the donor has $\lambda_{CFP} = 440$ nm, the light emitted in the final stage when calcium binds to CaM is of $\lambda_{YFP} = 527$ nm.
It is important to use as little excitation energy as possible to reduce photobleaching, the irreversible loss of fluorescence from a fluorophore due to prolonged exposure to light.
When Ca^{++} molecules bind to the CaM molecule, there is a structural change in the conformation of the CFP-CaM-M13-YFP chain. This brings the two folorescent proteins (CFP and YFP) into closer proximity, doing so they are able to undergo the FRET phenomena[31].

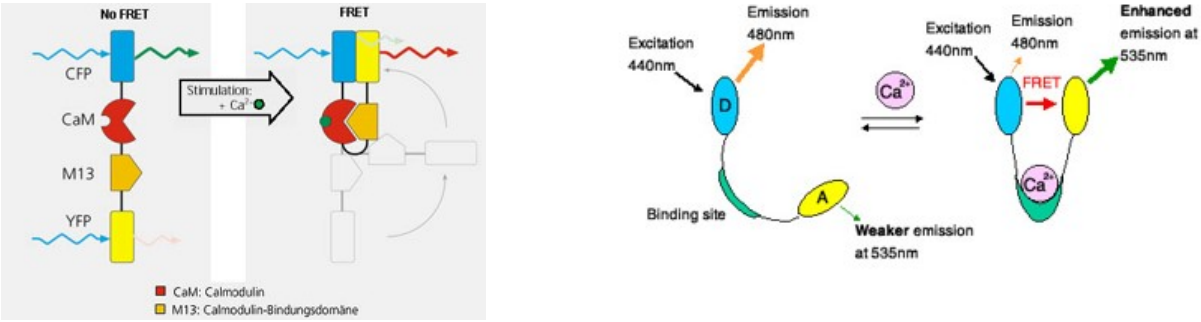

Figure 1.6: FRET emission for Calcium ions

In the methodology described above, a cofocal microscopy is generally used. Most cofocal microscopy have some characteristics useful for the correct implementation of FRET[31]:

1. An emission line from the argon ion laser suitable for CFP/FRET excitation;
2. Filters able to separate the CFP and FRET fluorescence (they have different λ);
3. two channels capable of collecting the CFP and FRET signal simultaneously;

4. detectors that allow quantitative analysis of the resultant fluorescence signals;

The use of YC3.6 allowed the technical staff to perform time-resolved calcium imaging in intact Arabidopsis plants. The setup of the experiment can be seen in Fig. 1.7.

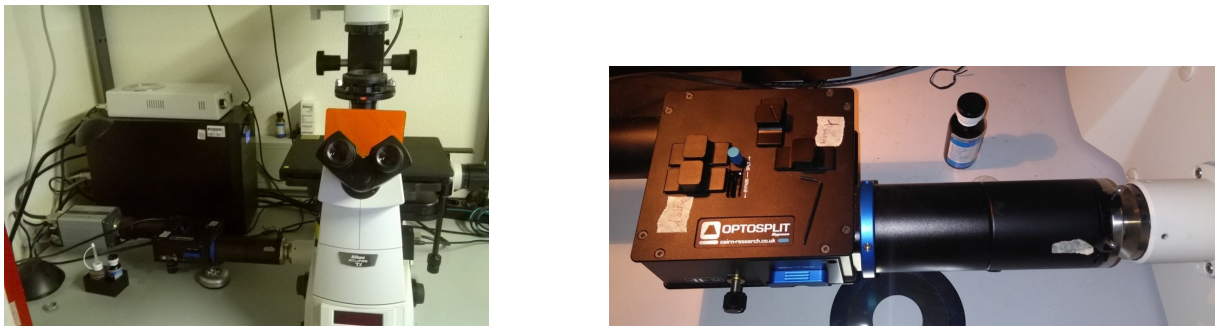

Figure 1.7: Measuring setup

1.8 The Dataset

The dataset that we used is composed of 9 videos of 9 different roots. The videos contains the Ca^{++} response of the root to different environmental stressors. In addition, to better isolate the calcium signature, 9 video_ratio were given to us. The total number of videos we were able to analyze is 18.

1.8.1 Challenges

The main difference among the videos is that the root may or may not contract when it undergoes environmental stress, this is particularly evident in "video848". We tried to solve this challenge by adding a flag that collects an user input to tell if the root moves or not. The procedure on how to do so is explained in the last chapter.
The following pictures (Fig.1.8) portraits the root at rest:

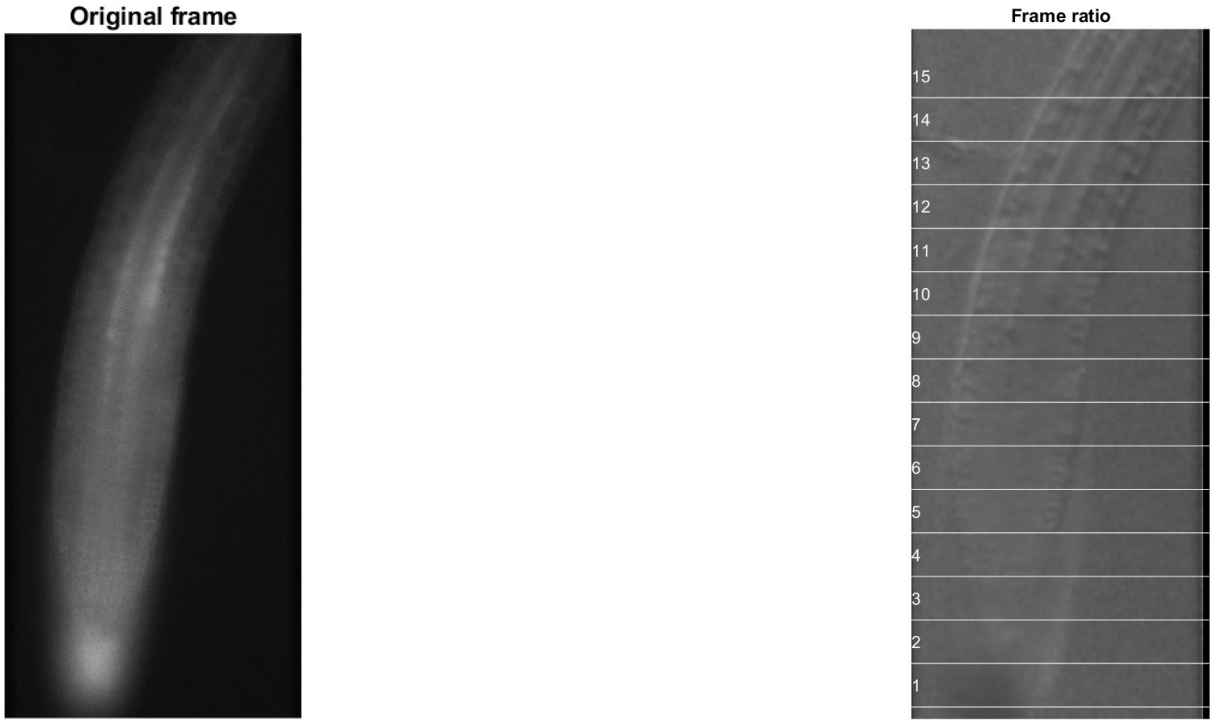

Figure 1.8: Original frame non-ratio and ratio

The following pictures (Fig.1.9) portraits the contracted root:

Figure 1.9: contracted root, frame non-ratio and ratio

We decided to display 5 images in one figure. We used the subplot function in MATLAB.

The five figures are the following:

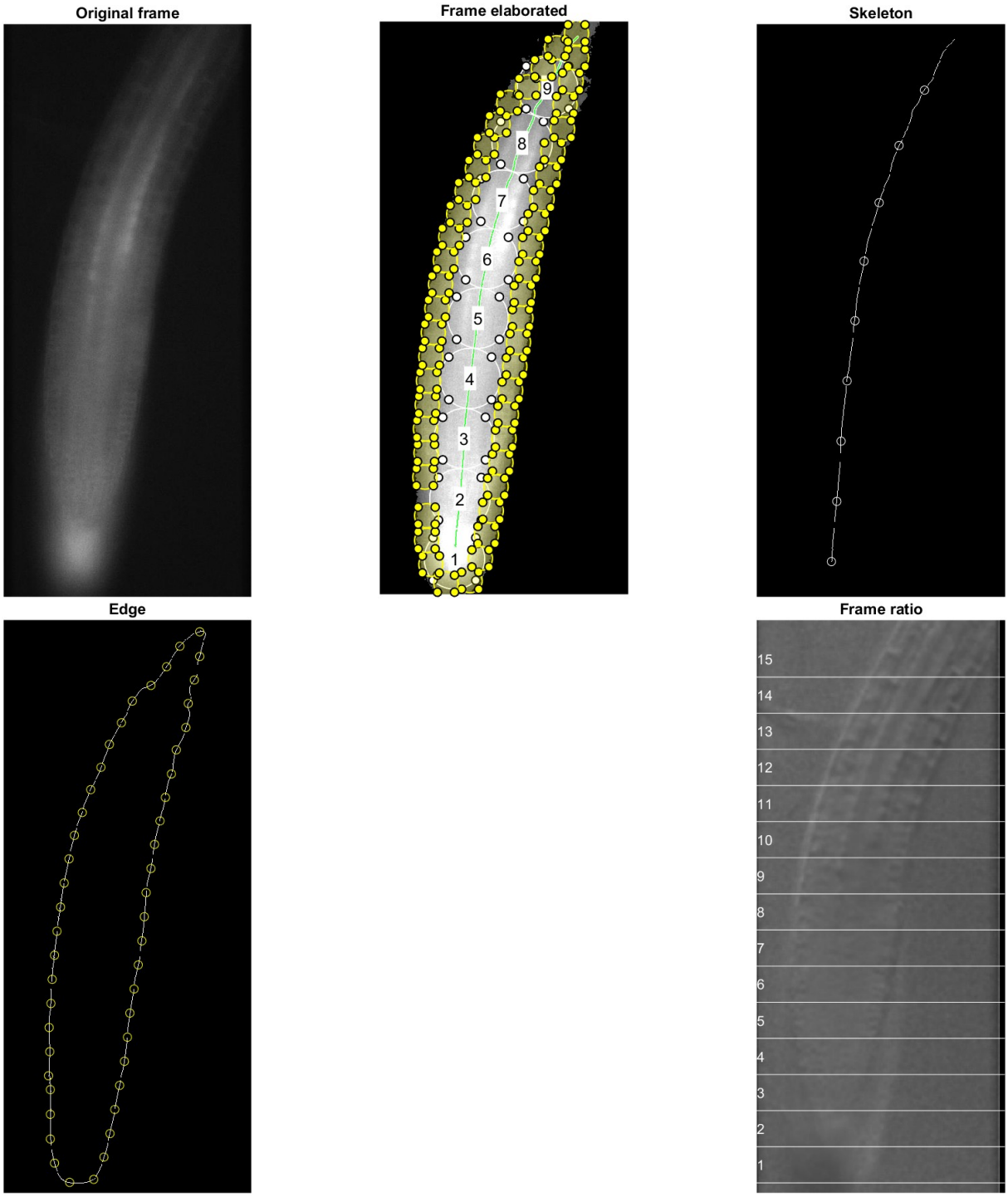

Figure 1.10: The five images

Respectively the five images contains:

1. **Original frame:** the original frame from the video untouched;

2. **Frame elaborated:** the elaborated frame with all the regions of interest (ROIs) attached to it. The main distinction is among the ROIs on the skeleton and the ROIs on the contour. The contour is found using the active contour model.

3. **Skeleton:** a plot of the skeleton of the root, obtained with the function ”bwmorp()”;

4. **Frame ratio:** The frame from the video_ratio. This frame is divided in section and those section correspond to the subdivision of the ROIs on the contour.

Chapter 2

Active Contour

2.1 Theory behind active contours

The first theory behind active contours was developed by Marr and Nishihara in their paper: ”Visual information processing: Artificial Intelligence and the sensorium of sight”[32].
Their work provides a foundational framework for understanding how visual perception is structured both in humans and artificial intelligence. They propose a multi-stage model of vision that breaks down the complex process of interpreting visual data into a series of hierarchical representations.
As stated before, Marr & Nishahara divided the visual process in phases and in each one the optical input is processed and abstracted more and more. This hierarchical approach moves from raw sensory data to a structured understanding of the environment. In particular, those authors identified three main stages[32]:

1. **Primal Sketch:** The first stage. It captures basic features like edges, textures, and contours from the raw image. This stage identifies the fundamental structure of the visual scene, providing a ”sketch” of the visual input;

2. **2.5D Sketch:** The second stage. The visual system organizes information into a viewer-centered representation of surfaces and their orientations. It captures depth and spatial relationships relative to the observer but remains dependent on the viewer’s perspective;

3. **3D Model Representation:** The final stage involves a full 3D model of objects that is independent of the viewer’s perspective. This representation allows for recognition and interaction with objects regardless of changes in viewpoint, scale, or orientation;

2.2 Introduction

Active contours, commonly referred to as ”snakes”, are a prominent technique in computer vision and image processing used for detecting object boundaries in images. A snake is an energy-minimizing spline guided by external constraint forces and influenced by image forces that pull it towards the feature such as lines, boundaries and edges of an image[33]. The application of active contours to visual problems regards: detection of edges, lines, subjective contours (not necessarily drawn), motion tracking and stereo matching.
As stated before, the total energy of a snake is typically defined as a combination of internal energy (related to the shape and smoothness of the curve) and external energy (derived from the image data that attracts the curve towards features of interest like edges or lines). This energy minimization problem is solved iteratively, with the curve adjusting its position to better fit the target features as it moves through the image space.
Active contours can be broadly categorized into two types: **parametric active contours** [33] (the original snakes, the ones that has been used in this master thesis) and **geometric active contours** [34](such as level set methods). Parametric contours explicitly represent the curve and rely on gradient descent methods to minimize the energy. This approach had tow main drawbacks:

- It fails to detect non-convex objects;
- It is sensitive to initialization;

This issues were partially solved by Xu and Prince in their *Gradient Vector Flow* approach[35]. There were other problems associated with the classical approach[34]:

- The user needed to handle the changes in topology of the snake explicitly;
- the lack of a parametrization independent energy definition;
- numerical instabilities;
- resampling problems arising in solving the energy minimization problem;

All this difficulties were solved in the geometric contours approach.
Geometric contours, introduced by Caselles et al in 1993[36], implicitly represent the curve as a level set (**level set methods**, presented by Osher and Sethian) of a higher-dimensional function, providing greater flexibility in handling topological changes such as merging or splitting of contours. In the level set methodology, the main intuition is that the curve can be seen as the zero level set of a function in higher dimension. Then, the snake motion can be expressed as a velocity along the normal direction of the curve[34]. The perks of this approach are:

- Changes in tolpology of the active contour are handled implicitly during the curve evolution;

- For numerical approximation, a fixed discrete grid in the spatial domain and finite-difference approximations for spatial and temporal derivatives can be used;

- level set methods can be extended to any dimension;

Geometric active contour models are based on designing a speed term so that the evolving front gradually attains zero speed as it gets closer to the object boundaries and eventually comes to a stop[34]. The main component of the geometric active contours is the definition of the speed function. Further developments on geometric active contours are:

- Geodesic active contours (Caselles et al [37]);

- Area based active contours (Siddiqi et al. [38])

- Active contour model without edges (Chan and Vase [39])

The versatility and adaptability of active contours make them suitable for a wide range of applications, including medical imaging (e.g., segmenting organs or tumors), video tracking (e.g., following moving objects), and shape modeling (e.g., reconstructing 3D shapes from 2D images). Despite their success, active contours also face challenges, such as sensitivity to initialization, susceptibility to noise, and computational complexity. We decided to use the active contour models in our code to find the edge of the root, we then placed some ROIs on that contour so to pick up the Ca^{++} signature from the contour (Fig. 1.10 picture 2 and 4).

2.3 Historical background

Introduced by Kass, Witkin, and Terzopoulos in 1988 in the paper ”Snakes: Active Contour Models”[33]. The theoretical fundations to which this paper is based can be found in the Marr & Nishahara work, called: ”Visual information processing: Artificial Intelligence and the sensorium of sight”[32]. The objective of Kass, Witkin, and Terzopoulos was to find a new algorithm that would **actively** adjust to the contour of an object in an image with little dependence from a starting point [33]. In this way the user can actively interact with the segmentation of the contour. Previous similar works have carried out by Sperling in his stereo models but they did not reach the same efficiency as the active contours[33].

2.4 Mathematical Formulation of Snake Behaviour

The basic snake model is called *controlled continuity* spline. This model is designed to provide more flexible control over the contour's smoothness and adaptability to image features. It is the sum of the "image forces" and the "external constraint forces"[33]. The **image forces** push the snake towards salient image features like lines, edges and subjective contours (the ones not drawn but perceived by our brain as such). The **external constraint forces** are responsible for putting the snake near the desired local minimum, which function can come from an user interface, automatic attentional mechanism (like masks) or high level interpretations. Furthermore, the external constraint forces are responsible to give continuity to the active contour.

Given the position of a snake, parametrized as $\mathbf{v}(s) = (x(s),y(s)), s \in [0,1]$, the energy function of the active contour can be written as:

$$E^*_{snake} = \int_0^1 E_{snake}(\mathbf{v}(s))ds = \int_0^1 E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s))ds \tag{2.1}$$

where:

- E_{int} represent the internal energy of the spline;
- E_{image} gives the image forces;
- E_{con} define the external constraint forces;

Another formulation of the problem was given by Aubert and Kornprobst's notation [34]. First, let C be a set of curves in R^2 given by

$$C = \{c : [a,b] \rightarrow \Omega, c\, piecewise\, C^1, c(a) = c(b)\} \tag{2.2}$$

The energy functional to be minimized is given by $J(c)$:

$$J(x) = \alpha \int_a^b |c'(q)|^2 dq + \beta \int_a^b |c''(q)|^2 dq + \lambda \int_z^b g^2(|\nabla I(c(q))|)dq \tag{2.3}$$

where c' and c'' denote the first and the second derivative of c. The $g(\nabla I)$ function is an edge detector function with constraints:

$$g(0) = 1, \lim_{s\rightarrow\infty} g(s) = 0. \tag{2.4}$$

The usual choice for $g(\nabla I)$ is:

$$g(\nabla I) = \frac{1}{1 + |\nabla I|^2} \tag{2.5}$$

In equation 2.3, the first term ($\alpha \int_a^b |c'(q)|^2 dq + \beta \int_a^b |c''(q)|^2 dq$) represent the **internal energy** of the active contour. The second term ($\lambda \int_z^b g^2(|\nabla I(c(q))|) dq$) represent the **external energy** of the snake. Those two concepts will be developed in subsection 2.4.1 and subsection 2.4.2.
The Euler-Lagrange equation associated with $J(c)$ are a forth order system:

$$-\alpha c'' + \beta c'''' + \lambda \nabla F|_c(c) = 0 \tag{2.6}$$

$$c(a) = c(b) \tag{2.7}$$

Where $F(c_1,c_2) = g^2(|\nabla I(c_1,c_2)|)$.
The traditional approach of active contours had several drawbacks (as previously stated). The solution of those issues were solved partially by Xu and Prince [35] in their *gradient vector flow approach* (discussed in section 2.4.4).

2.4.1 Internal Energy

The internal energy is responsible for the snake elongation and flexibility. In this phase, it is possible to think of it as rubber band.
In particular, the internal energy is composed of two set of forces:

- **Tension:** that makes the active contour move like series of spring, penalizing the spline elongation (See $|\frac{\partial v(s)}{\partial s}|^2$ eq. 2.8);
- **Rigidity forces:** those make the spline robust against bending (See $|\frac{\partial^2 v(s)}{\partial s^2}|^2$ eq. 2.8);

The internal spline energy can be written as:

$$E_{int} = \frac{\alpha(s)|\frac{\partial v(s)}{\partial s}|^2 + \beta(s)|\frac{\partial^2 v(s)}{\partial s^2}|^2}{2} \tag{2.8}$$

The spline energy is composed of a first-order term controlled by $\alpha(s)$ and second-order term controlled by $\beta(s)$. The first-order term makes the snake act like a membrane and the second-order term makes it act like a thin plate[33]. To better understand how the curve works, one can try to play with the parameters. For example, setting $\beta(s) = 0$ at a point allows the active contour to develop a corner.
The procedure to adjust the curve to the contour can be seen as a *regularization* problem [33]. Those kind of "ill-posed" problems (contrary to the well-posed problems qualified by

Hadamard), defined as not solvable in a practical sense, have been previously studied by Tikonov in his work called ridge regularization (or Tikonov regularization).
In the context of snakes, Kass et al. developed their minimization problem in case of $\alpha(s)$ and $\beta(s)$ constant as:

$$\alpha\frac{\partial^2 x}{\partial s^2} + \frac{\partial^4 x}{\partial s^4} + \frac{\partial E_{ext}}{\partial x} = 0 \tag{2.9}$$

$$\alpha\frac{\partial^2 y}{\partial s^2} + \frac{\partial^4 y}{\partial s^4} + \frac{\partial E_{ext}}{\partial y} = 0 \tag{2.10}$$

When $\alpha(s)$ and $\beta(s)$ constant, the minimization problem is solved by going back to eq. 2.8 and rewriting the eq. 2.1 in descrete case as:

$$E^*_{snake} = \sum_{i=1}^{n} E_{int}(i) + E_{ext}(i) \tag{2.11}$$

The minimization of eq. 2.11 is done with the aids of numerical methods techniques[33].
The procedure is an $O(n)$ iterative technique using sparse matrix methods [33].
The for the MATLAB implementation of this minimization problem of the internal energy, see [40].

2.4.2 Image Energy

The ”image energy”, also called ”external energy”, is a set of fictitious forces that attract the snake toward regions of interest, such as edges, lines, or other prominent features.
The image energy function can be expressed as:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \tag{2.12}$$

In this subsection, three main class of functionals are presented.

Line Functional

Line functionals are intended to attract the snake to regions of specific pixel intensity, typically corresponding to lines, ridges, or elongated structures. This makes line functionals particularly useful in scenarios where the object of interest is characterized by lines rather than sharp boundaries.
If we set, $I(x,y)$ equals to the intensity of the pixel in position (x,y) and we define the functional as

$$E_{line} = I(x,y) \tag{2.13}$$

then, depending on the sign of w_{line}, the active contour will be attracted either to to light lines or dark ones. In this way, the snake will try to align with the nearby contour.
With this functional, the use of a symbolic attentional mechanism, like a mask, is strongly suggested[33].

Scale Space

This is not a functional, it is a variation of the Line functional. Scale space is a framework used in computer vision that represents an image at multiple levels of resolution, allowing features to be analyzed at various scales. In other words, it is a multi-resolution representation of an image created by progressively smoothing the image using a Gaussian filter with increasing standard deviations (scales). At lower scales (small filter sizes), fine details are preserved, whereas at higher scales (large filter sizes), only coarse structures remain. This allows for a robust analysis of image features at different levels of abstraction.
In this context, an important result is given by the Marr-Hildreth Theory of Edge Detection [41]. This theory is a foundational approach in computer vision that seeks to identify edges within an image by detecting zero-crossings in the second derivative of the image intensity function. It is grounded in the understanding of human visual perception and aims to mimic the way biological vision systems detect edges.
Kass et al. developed a new functional for edge detection described as:

$$E_{line} = -(G_\sigma * \nabla^2 I)^2 \tag{2.14}$$

where G_σ is a Gaussian of standard deviation σ.
In this way, if a part of a snake finds a low-energy image feature, the spline term will pull neighbour parts of the snake towards a possible continuation of the feature [33].
Minima of this function lie on zero-crossings of $G_\sigma * \nabla^2 I$ which define edges in the Marr-Hildret theory[33]. The snake will then be attracted to zero-crossing, but still constrained by its own smoothness.

Edge Functional

This formulation leverages the gradient of the image to guide the snake to align with boundaries. Edge functionals are particularly effective in scenarios where objects are defined by clear, sharp boundaries.
If we set $\nabla I(x,y)$ equals to the gradient of the intensity of the pixel and we define the functional as

$$E_{edge} = -|\nabla I(x,y)|^2 \tag{2.15}$$

then the snake will be attracted to contours with large image gradient. Again, like for Line functionals, the sign of w_{edge} determines whether the active contour is attracted to a dark edge or a light one.

Termination Functional

The purpose here is to find the of line segments, edges or corners. To introduce the functional, first we have to define some parameters:

- Let $C(x,y) = G_\sigma(x,y) * I(x,y)$ be a slightly smoothed version of the image;
- Let $\theta = \arctan(\frac{C_y}{C_x})$ be the gradient angle;
- Let $\mathbf{n} = (\cos\theta, \sin\theta)$ be a unit vector along the the gradient direction;
- Let $\mathbf{n}_\perp = (-\sin\theta, \cos\theta)$ be a unit vector perpendicular to the gradient direction;

The curvature of the the level contour in $C(x,y)$ can be written as:

$$E_{term} = \frac{\partial\theta}{\partial\mathbf{n}_\perp} = \frac{\frac{\partial^2 C}{\partial\mathbf{n}_\perp^2}}{\frac{\partial C}{\partial\mathbf{n}}} = \frac{C_{yy}C_x^2 - 2C_{xy}C_xC_y}{(C_x^2 + C_y^2)^{\frac{3}{2}}} \tag{2.16}$$

The combination of E_{edge} and E_{term} generates an active contour that moves towards edges and terminations.
In this way the span of possibilities of active contours increases, because the same snake that finds subjective contours (the ones that are not really there) can very effectively find more traditional edges in natural imagery [33]. It is important to point out that snake can be subject to *hysteresis* when showing moving stimuli. Hysteresis is a phenomenon where the output or state of a system depends not only on its current input but also on its history of past inputs. Basically, the active contour has memory.
Kass et al. defined this hysteresis as "uncharacteristic of a purely bottom-up process and global optimization" [33].

2.4.3 External Constraint Forces

External constraint forces are defined in physics as forces applied to a system from outside sources that restrict or control the movement of an object or system. These forces are not generated by the system itself but are imposed externally to maintain or alter its state of motion or equilibrium. In the context of image processing, this are fictitious forces that constraint the development of the active contour. Typically, those constraint are product of the user that tries to define a ROI (Region of interest). This is usually done with a mask.

2.4.4 Gradient Vector Flow

Instead of defining an external energy and minimizing the energy functional $J(c)$ by solving the system of eq. 2.6, Xu and Prince work directly on eq. 2.6 and define an external force field that attracts the curve to boundaries[34].
Their approach was based on providing a new static external force field $F_{est} = \vec{v}(c)$, which was called the *gradient vector flow*. Then, the Euler-Lagrange equation (eq.2.6) became:

$$\frac{\partial c}{\partial t}(t,q) = -\alpha c'' + \beta c'''' + \vec{v}(c) \tag{2.17}$$

The vectors in an edge map generally have large magnitudes only in the immediate vicinity of the edges. And in homogeneous regions, where ∇g (eq.2.5) is nearly constant, the vectors are nearly zero.
In this way, the equation to minimize is:

$$\epsilon(c) = \int \mu(|\nabla u|^2 + |\nabla v|^2) + |\nabla g|^2 |[u,v]^T - \nabla g|^2 dq \tag{2.18}$$

when ∇g is small, the energy is dominated by sum of the squares of the partial derivatives of the vector field, leading to slowly varying field. On the other hand, when ∇g is large, the second term dominates the integrand, and is minimized by setting $\vec{v} = \nabla g$. This produces the desired effect of keeping $\vec{v}$ nearly equal to the gradient of the edge map when it is large, but forcing the field to be slowly-varying in homogeneous regions. The parameter μ is a regularization parameter which should be set higher in noisy images[34].
In comparison with Kass et al. model, the Xu and Prince variation is less sensitive to initialization. In addition, their model can detect nonconvex objects.

2.5 Geometric Active Contours

This variant of active contours is based on level set methods, introduced by Osher and Sethian in 1988 [42]. This approach arises from a problem with the classical formulation: the numerical instabilities and the need for sampling rate changes. The active contour models introduced in this part will make use of flows governed by equations (speed term) of the form:

$$\frac{\partial c}{\partial t} = FN \tag{2.19}$$

$$c(0,q) = c_0(q) \tag{2.20}$$

With this formulation the curve $c(t,q)$ moves along its normal (N) with speed F. In this way, the curve can be seen as the zero level set of a function in higher dimension. Given a generic function in the form of:

$$u(t,c(t,q)) = 0,\ \forall q, \forall t \geq 0 \tag{2.21}$$

The normal derivative is chosen to vanish on the boundary and u is initialized to be the signed distance function to the initial curve 0. The final model given by Otsher and Sethian [42] is :

$$\frac{\partial u}{\partial t}(t,x) = F|\nabla u(t,x)|\ for(t,x) \in]0,\infty[\times\Omega \tag{2.22}$$

$$u(0,x) = \hat{d}(x,c_0)\ (\hat{d} : signed\ distance) \tag{2.23}$$

$$\frac{\partial u}{\partial N} = 0\ for(t,x) \in]0,\infty[x \times \Omega \tag{2.24}$$

To apply those formulae, the curve motion must be expressed as velocity along the normal direction.
The perks of this formulation are:

1. The evolving function $u(t, x)$ remains a function during evolution as long as F is smooth.
2. for numerical approximations, a fixed discrete grid in the spatial domain and finite-difference approximations for spatial and temporal derivatives can be used.
3. geometric elements of the front such as the normal vector and the curvature can be expressed with respect to u.
4. level set methods can be extended to any dimension.

2.5.1 Models based on Boundary Function

The main issue with geometric active contours is the definition of a velocity function (F) in eq. 2.22. As a reminder, this speed function should gradually get close to zero as the curve approaches the boundary. The critical point here is to design a stopping function such that the curve velocity will stop along the edge. In the following sub subsection we will discuss two main stopping functions.

Image Gradient Based Stopping Function

This stopping function was proposed by Caselles et al. in their paper “A geometric model for active contours” [36].

They defined a new speed function F. The process of finding the new F starts from eq. 2.22. The snake was given by solving:

$$\frac{\partial u}{\partial t} = g(|\nabla I|)(\kappa + \alpha)|\nabla u| \tag{2.25}$$

In this equation $g(|\nabla I|)$ represents the stopping term and $(\kappa + \alpha)$ represents the curvature + the constant t. In particular, α is constant and κ is the curvature. κ is defined as:

$$\kappa = div(\frac{\nabla u}{|\nabla u|}) \tag{2.26}$$

This term, κ, imposes smoothness constraint on the curvature. On the other hand, the constraint term α (also called the "balloon force") makes the detection of the of nonconvex object easier and it increases the speed of convergence. $g(|\nabla I|)$ is the stopping term. Further analyzing the stopping term, Caselle et al. have given the following formulation:

$$g(|\nabla I|) = \frac{1}{1 + |G_\omega * \nabla I|} \tag{2.27}$$

It is important to point out the $*$ is the convolution operator and G_ω is a gaussian kernel (a small matrix used to apply effects, usually used to reduce noise) with standard deviation equals to σ. This formulation made the model immune to topological changes. However, the stopping term was not robust and sometimes the snake leaked from the boundaries. The pulling back feature was not strong. This meant that if the front propagated and crossed the goal boundary, then it could not come back[34].

Geodesic Active Contours

We have used this model to isolate the contour of the root in our code.
Later on, Caselle et al. introduced a new model for active contours: Geodesic Active Contours[37]. It was based on minimizing an intrinsic weighted Euclidean length. Starting from Aubert & Kornprobst formulation, eq. 2.3, we introduce a new formulation with $\beta = 0$:

$$J_1(x) = \alpha \int_a^b |c'(q)|^2 dq + \lambda \int_z^b g^2(|\nabla I(c(q))|) dq \tag{2.28}$$

This is done because the term preceded by β is redundant. In this way, the curvature of the curve is decreased.

Caselle et al. introduced a new functional which is independent from parametrization (intrinsic):

$$J_2(c) = \int_b^a g(|\nabla I(c(q))|)|c'(q)|dq \tag{2.29}$$

$J_2(c)$ can be seen as a new length by weighting the Euclidean length [34]. $g(|\nabla I(c(q))|)$ contains information regarding object boundary.
The correspondence between J_1 and J_2 has been proven by Aubert and Blanc-Feraud [34]. Thus, geometric active contours unify the curve evolution framework with classical energy minimization techniques. Following this lead, the equation 2.25 becomes:

$$\frac{\partial u}{\partial t} = g(|\nabla I|)(\kappa + \alpha)|\nabla u| + \langle \nabla g, \nabla u \rangle \tag{2.30}$$

In this equation, the term $\langle \nabla g, \nabla u \rangle$ attracts the curve further to the boundary. In this way, they solved the ”weak” stopping term problem.

Weighted area gradient flow

This is a work done by Siddiqi et al. [38]. It introduces area based active contour models. This model minimized a weighted area functional given by:

$$A(c) = -\frac{1}{2}\int_0^L g(|\nabla I|)\langle c(s),N\rangle ds \tag{2.31}$$

where:

- L is the length of the curve;
- $c(s)$ is the arc length parametrization of the curve;
- $g(|\nabla I|)$ is the weighting function;

The minimization of the area proceeds as follows:

$$\frac{\partial c}{\partial t} = (g(|\nabla I|) + \frac{1}{2}\langle c, \nabla g(|\nabla I|)\rangle)N \tag{2.32}$$

The level set representation of area minimizing flow is:

$$\frac{\partial u}{\partial t} = \frac{1}{2} div\left[\binom{x}{y} g(|\nabla I|)\right]|\nabla u| \tag{2.33}$$

In this equation, x, y represent the coordinates of a given image. This formulation combined with weighted length minimizing flow of Caselles et al. (eq. 2.30) give rise to:

$$\frac{\partial u}{\partial t} = g(\kappa + \alpha)|\nabla u| + \langle \nabla g, \nabla ur \rangle + \frac{\gamma}{2} div(\left[\binom{x}{y} g \right] |\nabla u|) \tag{2.34}$$

In this equation γ is constant and $div\left[\binom{x}{y} g \right] |\nabla u|$ provides an additional attractive forces when the front is in the proximity of the edge.

2.5.2 Region Based Methods

Region-based active contours, also known as region-based snakes or Chan-Vese models, are methods used in image segmentation to identify objects or regions of interest based on intensity values rather than edges. Unlike edge-based active contour models, which rely on detecting gradients or edges in an image, region-based methods look at the overall region properties, making them more robust to noise and weak or blurred edges. In the previously formulated models, structures such as interior of objects are not segmented. This was due to the fact that the level set formulation on the final contour was always based on a close edge.
Region-based active contours focus on segmenting the image into different regions by considering the intensity distribution inside and outside the contour. The most notable region-based active contour model is the Chan-Vese model, which is widely used for segmenting images with weak or blurry edges.

Active Contours without Edges

This model was formulated by Chan & Vase[39] in 1999. They called it the ”Active Contour Model without Edges”. The intuition here was to consider the information inside the region and not only on the boundary.
This methodologies is well suited for images where the boundaries are not well defined by the gradient.
The energy term in this model is defined as:

$$F(\phi,c_1,c_2) = \mu \int_\Omega \delta(\phi)|\nabla \phi| + v \int_\Omega H(\phi)dxdy + \tag{2.35}$$

$$+\lambda_1 \int_\Omega |u_0 - c_1|^2 H(\phi)dxdy + \lambda_2 \int_\Omega |u_0 - c_2|^2 (1 - H(\phi))dxdy \tag{2.36}$$

This long equation is composed of mainly four terms:

- $\int_\Omega \delta(\phi)|\nabla\phi|$, which is the length of c;
- $\int_\Omega H(\phi)dxdy$, which is the area inside c;
- $\lambda_1 \int_\Omega |u_0 - c_1|^2 H(\phi)dxdy$, which is the first fitting term of $F_1(c)$. This term guides the snake towards the boundary.;
- $\lambda_2 \int_\Omega |u_0 - c_2|^2 (1 - H(\phi))dxdy$, which is the second fitting term for $F_2(c)$. This term guides the snake towards the boundary.

In addition: U_0 is the original image, c_1 and c_2 are two constants and $H(\phi)$ is a step function of ϕ.
The perk of this model is that it searches for the best approximation of u_0 as a set of region with just two different intensities (c_1 and c_2). In this way, one region will be detected as the object itself and the other one will be treated as the background of the image. The active contour c will place itself in the boundary between those two regions. The Euler-Lagrange equation for the active contour model without edges is:

$$\frac{\partial\phi}{\partial t} = \delta\left(\mu\ div\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - |u_0 - c_1|^2 - |u_0 - c_2|^2\right) \tag{2.37}$$

Furthermore, the c_1 and c_2 terms can be expressed as

$$c_1 = \frac{\int_\Omega u_0 H(\phi)dx}{\int_\Omega H(\phi)dx}$$
$$c_2 = \frac{\int_\Omega u_0 (1 - H(\phi))dx}{\int_\Omega (1 - H(\phi))dx}$$

2.6 Active Shape Models

This is not a methodology used for the code associated with this master thesis. Although, for the sake of presenting all possible implementation of the snakes, this procedure must be shown. Since it is directly linked to active contours, this methodology is also called ”smart snakes” [43]. The purpose of this methodology is to build and use flexible statistical models of image whose shape can vary. Those models allow for considerable variability but are still specific to an original structure they represent. The main idea is to represent the original set of images ,or objects, (training set) as a set of points aligned in some ways. The points can represent the boundary, internal features or even external ones, such as the center of a concave section of boundary. The selection of points is a **manual procedure** done by the user, from this set of points a ”Point Distribution Model” (PDM) is derived. Once this PDM is defined, with a new

image a rough guess for the best shape, orientation, scale and position is set. Then, by comparing the hypothesized model instance with image data, the shape of the boundary is deformed so to a match the new image data. This deformation is placed under some global shape constraints.

2.6.1 Point Distribution Model (PDM)

The Point Distribution Model consists in a series of labelled ”landmarks” placed by the user in the training set of images. The amount of images chosen to act as reference images is not important. Although, it is important that the images captures the vast majority of cases available.

Labelling the Training Set

As we said, we represent the shape of an image by a set of points. This procedure must be done manually for each shape in the training set. The labelling of the points is important because each labeled point represent a particular part of the object or its boundary. If the labelling is incorrect, with a particular point placed at a different site on each training shape, the method will fail to capture the shape variability reliably.
It is important to notice that the manual placement of landmark is done only in the training phase. Landmarks can be classified in three categories according to their purpose:

1. The points mark parts of the object with particular application-dependent significance;
2. The points can mark application-independent things;
3. Other points which can be interpolated from points of type 1 and type 2;

Landmarks of type 1 are preferable to those of type 2, since they are in general easier to identify precisely. However, points of type 2 and 3 are are almost always necessary to define the boundary of a flexible shape. One of the main differences among smart snake and classical snakes is that the PDM can be defined for spatially separated objects. This could not be done for classical active contour models.
The use of only type 3 landmarks (usually not done) to describe the curve is quite effective and computationally more efficient.

Alignment the Training Set

As previously stated, this model works by examining the statistics of the coordinates of the labelled points over the training set. In order to compare corresponding points from different shapes, the PDM must be aligned with respect to a set of axes. This is done by scaling, rotating and translating the the training shapes. The further step is to minimize the weighted sum of

squared distance among between equivalent points of different shapes.
Let x_i be a vector describing the n points (landmarks) of the i^{th} shape:

$$x_i = (x_{i0},y_{i0},x_{i1},y_{i1},...,x_{ik},y_{ik},...,x_{i(n-1)},y_{i(n-1)})^T \tag{2.38}$$

Let $M(s,\theta)[x]$ be a rotation by θ and a scaling by s.
Once we find a new shape x_j to match to the training set image x_i we find a peculiar θ_j,s_j and translation (t_{xj},t_{yj}) so to map one image (x_j) into another (x_i). The formula describing this mapping is $M(s_j,\theta_j)[x_j] + t_j$. In order to find the optimal match, the weighted sum of squares must be minimized:

$$E_j = (x_i - M(s_j,\theta_j)[x_j] + t_j)^T W (x_i - M(s_j,\theta_j) \tag{2.39}$$

where

$$M(s,\theta) \begin{bmatrix} x_{jk} \\ y_{jk} \end{bmatrix} = \begin{pmatrix} s\cos\theta x_{jk} & s\sin\theta y_{jk} \\ s\sin\theta x_{jk} & s\cos\theta y_{jk} \end{pmatrix}, \tag{2.40}$$

$$t_j = (t_{xj},t_{yj},...,t_{xj},t_{yj})^T \tag{2.41}$$

and W is a diagonal matrix of weights for each point. For further details on the calculations regarding point alignment refer to [44]. The diagonal matrix W is chosen so to give more significance to those points which tend to be most ”stable” over the set (the ones that moves the least).
In order to align the PDM to the shape the following algorithm is used:

- Rotate, scale and translate each shape to align with the first shape in the set;
- **Repeat:**
  - Calculate the mean shape from the aligned shape;
  - Normalize the orientation, shape and origin of the current mean to proper defaults;
  - Realign every shape with the current mean;
- **Until** the process converges;

The process of normalizing the mean is required to make sure that the algorithm converges. Constraints on the pose and scale of the mean allow the equation to have unique solution[44].

Computing the Statistics of a Set of Aligned Shapes

In the point distribution model some of the vertices show little variability over the training set (those points are almost like fixed), on the other hand others form more diffuse ”clouds”. The

PDM tries to model the variation of the coordinates within those clouds. It is important to notice that the landmarks are not free to move, their position is partially correlated with each other.
In this way each image of the training set can be represented by a single point in a $2n$ dimensional space (with n = number of landmarks). Thus a set of N example shapes gives a cloud of N points in this $2n$ dimensional space. This space is called the "Allowable Shape Domain" and it gives an indication of the shape and size of this region of interest.
Another important remark is that the shape of the previously mentioned "cloud" is assumed to be ellipsoidal, in this way the center and its axes can be computed.
The center of the ellipsoid can be calculated as:

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.42}$$

while the principal axes of a $2n$-D (with D number of images and n number of landmarks) ellipsoid can be computed using Principal Component Analysis (PCA). The procedure works as follow, first for each shape in the training set we calculate its deviation from the mean, dx_i, where

$$dx_i = x_i - \overline{x}. \tag{2.43}$$

Then the $2n \times 2n$ covariance matrix is calculated, using:

$$S = \frac{1}{N}\sum_{i=1}^{N} dx_i dx_i^T \tag{2.44}$$

The next step is to calculate the eigenvalue and eigenvectors of the covariance matrix S. Then, the principal axes are given by the unit eigenvectors p_k of S such that:

$$Sp_k = \lambda_k p_k \tag{2.45}$$

As always, in the PCA the variance explained by each eigenvector is equal to the corresponding eigenvalue. Most of the variation can usually be explained by the first (if the eigenvalue are in order) few number of modes t. This means that the $2n$ dimensional ellipsoid is approximated by a t dimensional ellipsoid. In order to choose the best candidate eigenvector and relative eigenvalues, we sum up all the eigenvalues up to the point were a certain amount of variance is explained:

$$\lambda_T = \sum_{k=1}^{2n} \lambda_k < x \tag{2.46}$$

where x is the amount of variance that we want to explain through the sum of the eigenvalues. Any point in the Allowable Shape Domain can be reached by taking the mean and adding a linear combination of the eigenvectors. The k^{th} eigenvector related to the point l moves along a vector parallel to (dx_{kl},dy_{kl}) due to image variation. A shape in the training set can be approximated using the mean shape and a weighted sum of these deviations obtained from the first t modes calculate with the PCA:

$$x = \overline{x} + Pb \tag{2.47}$$

where $P = (p_1, p_2,..., p_t)$ is the matrix of the first t eigenvalues, and $b = (b_1, b_2,...,b_t)$ is a vector of weights. The above equations allow us to generate new examples of the shapes by varying the parameters (b_k) within suitable limits, in this way the new shape will be similar to those in the training set.
Suitable limits for the values of b are usually

$$-3\sqrt{\lambda_k} \leq b_k \geq 3\sqrt{\lambda_k} \tag{2.48}$$

since most of the population lies within three standard deviation of the mean.

2.6.2 Active Shape Models

As we said, the purpose here is to match the PDM of the training set to a new image. Summing up what we have seen so far, we have a model described by an instance:

$$X = M(s,\theta)[x] + X_c \tag{2.49}$$

where $X_c = (X_{c1}, Y_{c1}, X_{c2}, Y_{c2}, ..., X_{cn}, Y_{cn})^T$ is the translation vector, $M(s,\theta)[]$ is a rotation by θ and scaling by s, and (X_{ci}, Y_{ci}) is the position of the centre of the model image frame.
The iterative methodology described in this section is concerned with finding the appropriate X given a rough starting approximation. The starting value of X does not need to be very close to the final solution.
The main idea is to place the current estimate of X into the image and examine a region of the image around each model point to determine the displacement which moves it to a better location. This is done so to align to the Point Distribution Model, it ensures the shape of the model example to remains similar to those of the training set. The procedure is repeated until convergence.

Calculating a Suggested Movement for Each Model Point

Given a new image, there is the need to find a set of adjustment which will move each point towards a better position (in alignment with the PDM). When the model points represent the boundaries of object this involves moving them towards the image edge. This can be done in various ways. One approach consists in using a gradient like in traditional snakes.
However those adjustments are obtained, we denote them as a vector dX where

$$dX = (dX_0,dY_0,..., dX_{n-1},dY_{n-1})^T \tag{2.50}$$

Computing Changes in the Pose and Shape Parameters

Once we get the displacement of the landmarks from the original PDM, we can move the vector of X to be close to the suggested new locations $(X+dX)$ as can be arranged while still satisfying the shape constraints of the model. Obviously the each iteration the parameters (X_c,Y_c), θ and s must be updated. This is achieved by finding the translation (dX,dY), rotation $d\theta$ and scaling factor $(1 + ds)$ which best map the current set of points X, onto the set of points given by $(X + dX)$. To see the method to update those parameters refer to [43] (Appendix A). At this point, we have all the adjustment to make ((dX,dY), $d\theta$ and $(1 + ds)$) in order to deform the shape of the model so to match the training set with the new image. This means moving the points from X by dX.
As said before, the initial position of the points in the image frame is given by eq. 2.49

$$X = M(s,\theta)[x] + X_c$$

The set of residual adjustment dx in the local model coordinates frame is given by

$$M(s(1 + ds)),\ (\theta + d\theta)[x + dx] + (X_c + dX_c) = (X + dX) \tag{2.51}$$

Thus

$$M(s(1 + ds)),\ (\theta + d\theta)[x + dx] + (X_c + dX_c) = (M(s,\theta)[x] + dX) - (X_c + dX_c).$$

Furthermore, since

$$M^{-1}(s,\theta)[] = M(s^{-1}, -\theta)[]$$

we can write

$$dx = M((s(1 + ds))^{-1},\ -(\theta + d\theta))[y] - x, \tag{2.52}$$

where $y = M(s,\theta)[x] + dX - dX_c$. This equation calculates the suggested movements to make in order to match the PDM with the new image. Obviously there will not be an exact overlap. This is the point where we apply shape constraints.
In order to apply the shape constraints we transform dx into a model parameter space, giving db, the changes in model parameters required to adjust the model points as closely to dx as allowed. From equation 2.47 we can write:

$$x = \overline{x} + Pb$$

The idea is to find db such that

$$x + dx \approx \overline{x} + P(b + db) \tag{2.53}$$

Although, the available modes of variation are t ($< 2n$) and dx can move points in $2n$ different degrees of freedom, we can only achieve an approximation to the deformation required.
Subtracting eq. 2.47 from 2.53 gives

$$dx \approx p(db)$$

in this way

$$db = P^T dx$$

P is an orthogonal matrix with unit length, therefore $P^T = P^{-1}$.
It can be shown that the last equation is equivalent to using a least-squares approximation to calculate the shape parameter adjustment db.

Updating the Pose and Shape Parameters

In the previous section we discussed how to calculate changes to the pose variables and adjustments, dX_c,dY_c, $d\theta$ and ds, to the shape parameters db so to match the new image with the PDM of the training set.
The parameters are then updated according to their weights:

$$X_c \to X_c + w_t dX_c \tag{2.54}$$

$$Y_c \to Y_c + w_t dY_c \tag{2.55}$$

$$\theta \to \theta + w_\theta d\theta \tag{2.56}$$

$$s \to s(1 + w_s ds) \tag{2.57}$$

$$b \to b + W_b db \tag{2.58}$$

In those equations w_t, w_s and w_θ are scalar weights and W_b is a diagonal matrix of weights, one for each mode. This can be the identity, or each weight can be proportional to the standard deviation of the corresponding shape parameter over the training set[44]. The latter allows more rapid movements in modes, this translates in larger shape variations.
To ensure that the model only deforms into shapes consistent with the training set by placing limits on the value of b_k. We have done that in eq. 2.48. In this way the vector b should lie within the hyperellipsoid about the origin.
There are other ways to put constraints on the movement of the model by using the Mahalanobis distance [44].

2.6.3 Summing up

In this section we described first the Point Distribution Models (PDMs) that are statistical models of shape which can be constructed from training sets of currently labelled image.
Active Shape Models (ASMs) exploits the linear formulation of PDMs in an iterative search procedure capable of rapidly locating the modeled structures in noisy cluttered images, even if they are partially occluded.
The important point to stress is precisely the same software can be applied to a broad range of image interpretation problems.

Chapter 3

Skeletonization

In the technical literature, the concepts of skeleton and medial axis are used interchangeably [45]. The skeletonization is a core function of the algorithm that we implemented. We extracted the contour of the root with the ”snake” and then we searched for the medial axes of the figure. In this way, we would have a reference of a structure fully describing the proprieties and the changes in root.
Below you can see a comparison of the original frame of the root and its medial axes Fig.3.1.

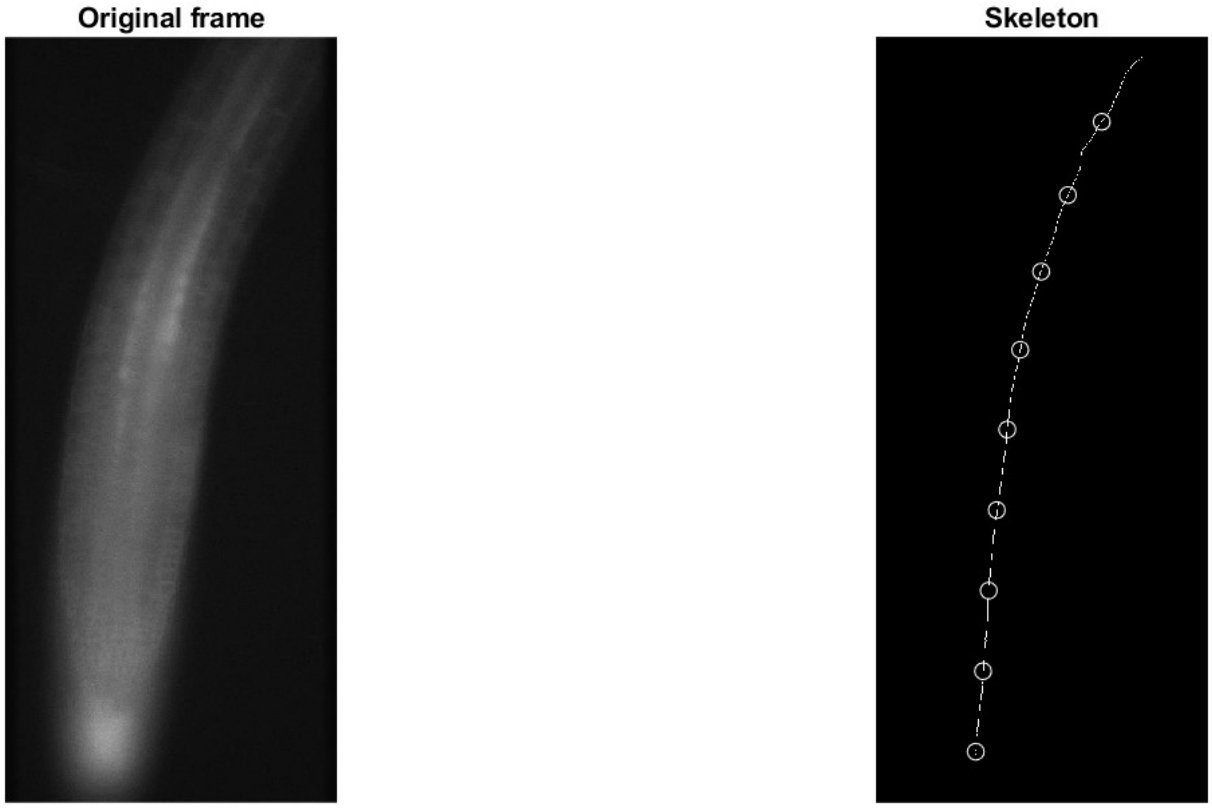

Figure 3.1: Original frame & medial axes

The MATLAB code associated with the thinning process is composed of two steps:

1. In the first subiteration, delete pixel p if and only if the conditions G_1, G_2, and G_3 are all satisfied.

2. In the second subiteration, delete pixel p if and only if the conditions $G1$, $G2$, and G_3' are all satisfied.

The conditions G_1, G_2, G_3 and G_3' are listed below, where $\overline{x}$ is the logical not of the value of the pixel (of the black and white image):
Given an 8-neighbourhood like in Fig. 3.2:

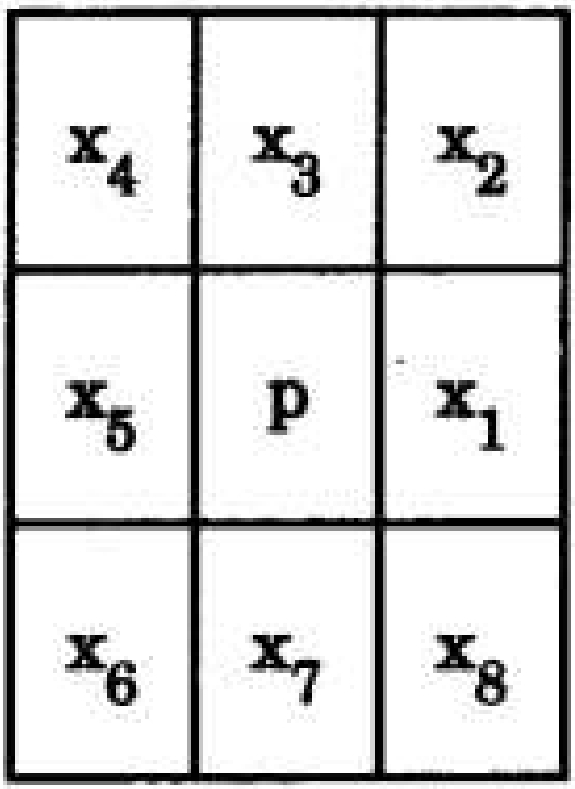

Figure 3.2: 8-neighborhood

Condition G_1:

$$X_H(p) = 1$$

where

$$X_H(p) = \sum_{i=1}^{4} b_i$$

and $b_i = 1$, if $x_{2i-1} = 0$ and $(x_{2i} = 1$ or $x_{2i+1} = 1)$ otherwise $b_i = 0$.

Condition G_2:

$$2 \leq \min\{n_1(p), n_2(p)\} \leq 3$$

where

$$n_1(p) = \sum_{k=1}^{4} x_{2k-1} \wedge x_{2k}$$

$$n_2(p) = \sum_{k=1}^{4} x_{2k} \wedge x_{2k+1}$$

Condition G_3

$$(x_2 \wedge x_3 \wedge \overline{x}_8) \vee x_1 = 0$$

Condition G_3'

$$(x_6 \wedge x_7 \wedge \overline{x}_4) \vee x_5 = 0$$

The two subiterations together make up one iteration of the thinning algorithm [46].

3.1 Thinning algorithms

In this section we analyse how thinning algorithms, the algorithms used to define the skeleton, are implemented.
This procedure arose with the need to find a way to reduce the amount of information processed to the minimum possible for the recognition of patterns (e.g. letters of the alphabet). The reduction of an image to its essential features can delete some contour noise (distortions) while keeping relevant topological and geometrical characteristics. This means highlight features such:

- edge points;
- junction points;
- connection among components;

The whole point of thinning algorithms is to compress data.

3.1.1 Overview of Thinning procedure

With the ”skeleton” term we usually refer to a representation of a shape by a collection of thin (usually one pixel wide) arcs and curves.

With the term ”medial axes” we denote the locus of centers of maximal blocks that are also equivalent to the local maxima of a chessboard distance (the distance between two squares on the chessboard e.g. Manhattan distance or Chebyshev distance).
As we said before, those two nomenclatures are equivalent.
Although, with the ”medial axis transform” people usually refer to methods involving determination of centers of maximal blocks. With ”thinning”, we usually refer to the procedure of reduction of generally elongated patterns to line-like representation. In this section, we will consider some general aspects of iterative thinning algorithms or, more precisely, the algorithms that delete successive layers of pixels on the boundary of the pattern until only a skeleton remains. The deletion or retention of a pixel p is strictly correlated to its 4,8-neighbourhood. The representation and nomenclature of the pixel in the neighbourhood can be seen in Fig 3.2.
According to the way in which this algorithms examines pixel, this algorithms can be classified in **sequential** or **parallel**.
In sequential algorithms, the pixels are examined for deletion in a fixed sequence each iteration , and the deletion of p in the n^{th} iteration depends on the operations that have been performed so far. On the other hand, in parallel algorithms the deletion of pixels in the n^{th} iteration would depend only on the results that remains after the $(n-1)^{th}$. In this way all pixels can be examined independently in a parallel manner in each iteration.
There is the need to introduce some important notations that will be used thorough the chapter. The pixels x_1,x_2,...,x_8 in Fig 3.2 are the 8-neighbours of p. Those pixels are collectively denoted as $N(p)$. They are said to be 8-adjacent to p.
The pixels x_1,x_3,x_5,x_7 in Fig. 3.2 are the 4-neighbours of p and are 4-adjacent to p.
The number of black pixels in $N(p)$ is denoted by $b(p)$. A sequence of points y_1,y_2,y_3,...,y_n is called an 8-path (or a 4-path) if y_{i+1} is an 8-(or 4-)neighbour of y_i with $i = 1,2,...,n-1$. A subset Q of a picture P is 8-(or 4-)connected if for every pair of points x,y in Q there exists an 8-(4-)path from x to y consisting in points in Q. In this case, Q is said to be an 8-(4-)component of P.
The order of connectivity of P is the number of components if its complement $\overline{P}$, and if this order is 1, we say that P is simply connected; otherwise P is multiply connected.
In this algorithms the pixels considered for deletion are contour pixels.
For the purpose of thinning, it is desirable for skeleton to have unit width. For this reason, in the algorithm, the 8-connectivity is chosen for P and the 4-connectivity is chosen for $\overline{P}$ [47]. This choice is done so to preserve the connectivity of P by deleting the only pixels that are 4-adjacent to $\overline{P}$. Therefore, contour pixels are usually defined as those having at least one white pixel in the 4-neighbour. This pixels include edge points and border points.
Noncontour points that black pixels are said to be interior points and pixel that have $b(p) = 1$

will be called end points. The statement that $b(p) = 1$ will be the end point condition.
The difference among thinning algorithms relies on the definition given by developers to ensure connectivity. Following this set of definition, there is also the need to define the **crossing number**. The crossing number specifically refers to the number of transitions from 0 to 1 in the neighborhood of a pixel as one moves around the pixel in a clockwise or counterclockwise manner. The crossing number helps in determining whether a pixel is a part of the skeleton or whether it can be deleted without altering the overall structure of the object.
There are two main definitions of crossing number: the **Rutovitz** definition and the **Hilditch** definition.
In Rutovitz's definition, the crossing number (denoted as $X_R(p)$) for a pixel $N(p)$ in a binary image is defined based on its 8-neighborhood. The crossing number is calculated as half of the number of 0-to-1 transitions around the pixel as you move counterclockwise through its eight neighboring pixels. In other words, the crossing number can be defined as :

$$X_R(p) = \sum_{i-1}^{8} |x_{i+1} - x_i| \tag{3.1}$$

where $x_9 = X_1$.and it is equal to twice the number of black 4-components in $N(P)$.
Skeletons obtained using Rutovitz crossing number can contain 8-deletable pixels, and this skeleton are said to be imperfectly 8-connected (this means that can be more than one pixel wide) [47].
The Hilditch crossing number is defined similarly to other crossing number definitions. It counts the number of transitions from 0 to 1 as you traverse the neighborhood in a clockwise direction. In other words, this crossing number $X_H(p)$ is the number of times one crosses over from a white point to a black point when the points in $N(p)$ are traversed in order, cutting the corner from between 8-adjacent black 4-neighbours. Therefore:

$$X_H(p) = \sum_{i=1}^{4} b_i \tag{3.2}$$

where $b_i = 1$, if $x_{2i-1} = 0$ and ($x_{2i} = 1$ or $x_{2i+1} = 1$) otherwise $b_i = 0$.
In particular, in the implementation of our algorithms, we have used the **Hilditch** variant.
By this definition, we can consider the Hilditch crossing number as the number of black 8-components in $N(p)$. In case of a whole black 4-neighbours, the crossing number is $b(p) = 0$.
If $X_H(p) = 1$, the deletion of p would not change the 8-connectedness of the pattern. Furthermore, an important distinction between $X_R(p)$ and $X_H(p)$ is that the condition $X_H(p) = 1$ implicitly implies that the pixel p is a contour point. On the other hand, $X_R(p) = 2$ does not imply that the pixel is a contour pixel[47]. In order to match X_H case of contour point to the

one in X_R another condition should be added, for example $b(p) \leq 6$.
The Hilditch, the crossing number implemented in our algorithm, is constructed using another definition. This definition is more computationally efficient. The 8-connectivity number is:

$$N_c^8(p) = \sum_{i=1}^{4} (\overline{x}_{2i-1} - \overline{x}_{2i-1}\overline{x}_{2i}\overline{x}_{2i+1}) \tag{3.3}$$

where $\overline{x}$ is the logical negation of x. The efficient formulation for the 4-connectivity number is :

$$N_c^4 = \sum_{i-1}^{4} (x_{2i-1} - x_{2i-1}x_{2i}x_{2i+1}) \tag{3.4}$$

and it represent the number of 4-connected components containing the black 4-neighbour of $N(p)$.
Another approach to preserve the topological structure of the image during thinning is that the genus of P and $\overline{P}$, which is the number of "holes" or disconnections associated with the pixel and its surrounding neighborhood, should remain invariant. In other words, the number of connected components of P minus the number of holes in P should remain invariant during the process of thinning.
The genus G of a region can be entirely determined by the configuration of $N(p)$. If deletion of p does not change G, p is said to be simple.
Pixels with connectivity number $N_c^8(p)$ greater than one also belong to the category of multiple pixels. The peculiarity of this points is that they are considered to occur when a pattern "folds" onto itself, this includes:

- endpoints of branches;
- strokes that are two pixels wide;
- pixels that should be assigned to the skeleton;

This pixels are kept in the thinning process.

3.1.2 Sequential thinning algorithms

In sequential thinning algorithms, contour pixels are examined for deletion one by one and in a predetermined order. The order in which pixel can be "visited" can be accomplished by either **raster scan** or **contour following**.
Raster scan is a technique used for systematically traversing or displaying pixel information in a two-dimensional grid, such as an image. The term "raster" refers to the rectangular grid of pixels that make up a digital image. In a raster scan, the pixels are processed in a specific order,

typically from left to right and top to bottom.
Contour following , also known as boundary following or border tracing, is a technique to extract the boundary or contour of objects in a binary image. The purpose of this technique is to trace the outer edge of connected components (objects) within the image, helping to identify their shape, size, and other important features for further analysis.
Contour following algorithms can visit every border pixel of a simply connected object, and of a multiply connected picture, if all the borders of the picture and holes are followed. These algorithms have an advantage over raster scans. This advantage is nested in the way they examine pixels: they are required to check only the contour pixels instead of all the pixels in P and $\overline{P}$ in every iteration.
The contours in contour following algorithms is defined with the aid of the Freeman chain code. The Freeman chain code is a method for representing the boundary or contour of a shape in a binary image using a sequence of directional moves. This method encodes the contour of an object as a sequence of directions corresponding to movements between adjacent boundary pixels. It's useful for compactly representing the shape of an object. Keeping in mind Fig. 3.2, the possible move that one can make to pass form a pixel p_i (center) to a pixel p_{i+1} are:

- Move right, encoded by a 0;
- Move up-right, encoded by a 1;
- Move up, encoded by a 2;
- Move up-left, encoded by a 3;
- Move left, encoded by a 4;
- Move down-left, encoded by a 5;
- Move down, encoded by a 6;
- Move down-right, encoded by a 7;

The steps to generate the Freeman chain code are:

1. Find a starting point;
2. Trace the step-by-step contour using the encoding listed above;
3. Continue until the boundary is closed;

The result is a sequence of directional codes (0-7) that describe the boundary.
So far we have seen that when a contour pixel p is examined, the deletion of retention of p is strictly connected to the configuration of N(p). A safety measure, employed by sequential algorithms, to prevent elimination of an entire branch in one iteration is that a sequential algorithm usually marks (or flags) the pixels to be deleted, and all the marked pixels are then removed at the end of an iteration. This ensures that only one layer of pixels would be removed in each cycle.
Generally, in thinning algorithms, we will assume that the pixel p considered for deletion satisfies all the following proprieties:

1. p is a black pixel;
2. p is not an isolated or end point, e.g. $b(p) \geq 2$
3. p is a contour pixel, e.g. p has at least one white 4-neighbour.

Examples of parallel thinning algorithms

An important milestone paper in the development of sequential thinning algorithms is the "Linear skeletons from square cupboards"[48]. In this paper, Hilditch (the author) uses the $X_H(p)$ crossing number. The pattern is scanned from left to right and top to bottom. In order to delete a pixel, the additional following conditions were added:

1. At least one black neighbour of p must be unmarked;
2. $X_H(p) = 1$ at the beginning of the iteration;
3. if x_3 is marked, setting $x_3 = 0$ does not change $X_H(p)$;
4. Same as condition 3, with x_5 replacing x_3;

The first condition here was designed to prevent excessive erosion of small "circular" subsets. The second condition was intended to maintain connectivity. Condition 3 and 4 are made to preserve two-pixel wide lines. It was found in later researches [47] that a sequential thinning method based on a raster scan and 3×3 operation can be implemented using pipeline structure to reduce the memory and processing time required.
The crossing number $X_H(p)$ together with a rough estimate $K(p)$ of the convexity at p are used as deletion criteria for some algorithms [47]. $K(p)$ is the maximum number of 4-connected white points in $N(p)$ and it is intended as a discrete equivalent for the notion of curvature. Under this conditions, it is possible to define a deletion criteria for p such as $K(p) \leq k_T$, where K_T is a prefixed threshold. This inequality must be paired with condition 2,3 and 4 from the previous

Figure 3.3: 3 by 3 Breakpoint Configuration

list.

It is important to notice that an high threshold K_T is results in a skeleton less affected by contour protrusion.

In later papers [47] the thinning criteria are extended to a $k \times k$ windows (with $k \geq 3$). In this configuration the "core" of the window, a $(k-2) \times (k-2)$ pixels scheme, can be deleted together if the boundary pixels in the window have Hilditch crossing number of 1 and if they contain the more than $(k-2)$ 4-connected white pixels and more than $(k-2)$ black pixels. For every black pixels, its $k \times k$ windows are examined in order of decreasing k until $k < 3$ or the core is deleted[47]. This requires fewer iteration to reach the skeleton. Unfortunately, this can produce a noisy skeleton. Furthermore, in thinning procedure the use of a larger k can be damaging for the computational speed.

In other algorithms [47], different criteria have been used. In those algorithms contour or edge points are used. For example, p is a west edge point if there is at most one black pixel in the first column of $N(p)$ and at least 3 black pixels in the rest of $N(p)$. In this casuistry, another condition must be added to prevent an interior pixel to be wrongfully marked as an edge point. Edge points are marked in one raster scan after which the marked points are compared against six $k \times k$ windows and deleted if they are not break points. In this way, pixels removal do not create breaks in the pattern. The procedure is repeated until no deletion of pixels occurs or a maximum of 3 iteration is reached. After this iterations, a single "clean-up" scan removes all black pixels that are not end points, break points or interior points. The final step of this algorithm is a smoothing process that outputs the final skeleton. Variant of this kind of methods implements a smoothing process after each iteration. The classical thinning algorithms use a 3×3 windows like in Fig. 3.3 as breakpoint configuration. Since this windows only represent connectivity preservation, their sequential application to a thinning methodology may result in excessive erosion or more than necessary shortening of the branches[47].

The last algorithm implementing the Hilditch crossing number is the "SPTA algorithm". In this one, two raster scans are used per cycle (1:left to right, 2:top to bottom). Contrary to other algorithms, p is not marked by deletion but by retention in each scan.

The condition to satisfy in order to be retained are:

1. $N(p)$ satisfies the configuration in Fig. 3.3 or its rotation;

2. $N(p)$ contains exactly two 4-adjacent black points;

It is important to notice that the second condition prevents excessive erosion.
The boolean expression (more computationally efficient) of the last two conditions is:

$$x_1(x_2 + x_3 + x_7 + x_8)(x_3 + \overline{x}_4)(x_7 + \overline{x}_6) = 0 \tag{3.5}$$

3.1.3 Parallel thinning algorithms

This is the class of algorithms implemented in MATLAB.
Parallel thinning algorithms are techniques used to reduce the dimensionality of objects in binary images (usually represented as a grid of pixels) by removing pixels in a way that preserves the object's topological properties, such as connectivity, until the object is reduced to a skeleton. The key concepts of thinning are 3:

- **Topological Preservation:** The algorithm must retain the fundamental properties of the shape, such as connectivity and structure.

- **Parallel Execution:** Pixels are processed in parallel, allowing for faster computation, especially on large images or with more computational power.

- **Iterative Process:** Thinning is typically done iteratively, removing pixels layer by layer while ensuring the skeleton is formed.

In the process of parallel thinning, the examination of pixels occurs based on the results of only the previous iteration. This is suitable for processors that implements a strong pipeline calculation (GPU).
The usual practice is to use a 3×3 neighbourhoods but to divide each iteration into subiteration (or subcycle) in which only a subset of contour pixels are considered for removal. When a subcycle is completed the image is updated. The number of subcycle associate with this procedure is varies up to a maximum of 4, in each subcycle the contour points (north,east,south west pixels) are removed. The partition of the image is done in a checkerboard manner, as in sequential thinning. The subdivision of parallel thinning algorithms happens according to the number of subcycles used.

Examples of parallel thinning algorithms

A fundamental one-iteration algorithm in this field is described in the "Pattern Recognition" paper [47] by Rutoviz. In this algorithm, a pixel p is deleted if and only if all the following conditions are true:

1. $b(p) \geq 2$;
2. $X_R(p) = 2$;
3. $x_1x_3x_5 = 0$ or $X_R \neq 2$;
4. $x_7x_1x_3 = 0$ or $X_R \neq 2$;

Some variation of this algorithm add the condition of $b(p) \leq 6$ to ensure that p has a white 4-neighbour, in this way the deletion of p would not create a hole.
Furthermore, due to the asymmetric nature of conditions 3 and 4 [47], the skeleton would not lie centrally. Therefore, $180°$ rotations of these rules were introduced later on in another algorithm. The result is a complete set of rules for the removal of p:

1. $X_R(p) = 0$,2 *or* 4;
2. $b(p) \neq 1$;
3. $x_1x_3x_5 = 0$;
4. $x_1x_3x_7 = 0$;
5. If $X_R(p) = 4$, then in addition the following two conditions must hold:
   (a) $x_1x_7 = 1$, $x_2 + x_6 \neq 0$, and $x_3 + x_4 + x_5 + x_8 = 0$;
   (b) $x_1x_3 = 1$, $x_4 + x_8 \neq 0$, and $x_2 + x_5 + x_6 + x_7 = 0$;

There would be other another 3 conditions to add to the list, by they are rotations of $180°$ of conditions 3-5 according to Fig.3.2.
This renewed class of algorithms use two subcycles to satisfy all the above conditions. The first subiteration deletes pixels that satisfy conditions 1-5, the second deletes according to condition 1-2 and 6-8. One of the perks of this algorithm is that it deletes isolated pixels.
The complexity of rules 1-5 led to the observation that if two consecutive 4-neighbours of p are 1's, then the value of the corner pixel q in between has no effect on weather p should be deleted [47]; q can be considered to be 1.
The crossing number $X_R(p)$ can also be used as a criterion for filling operations before thinning. An important work in this field is the Zang and Suen parallel algorithm for skeletonization[49].

This algorithm is composed of two subiteration.
In the first subiteration p is deleted if the following conditions are satisfied:

1. $2 \geq b(p) \leq 6$;
2. $X_R(p) = 2$;
3. $x_1 x_3 x_7 = 0$;
4. $x_1 x_7 x_5 = 0$;

In the second iteration, condition 3 and 4 are substituted by their $180°$ rotation. In this way, the fist subiteration deletes 4-simple pixels on the south and east border as well as north-west corner pixels. On the other hand, the second iteration deletes pixels with opposite orientation (pixels are deleted from diagonal neighbors). One of the perks of this algorithm is that it is immune to contour noise.
One observation proposed by the community is that the first condition ($2 \geq b(p) \leq 6$) should be replaced by $3 \geq b(p) \leq 6$. In this way, two pixels wide diagonal lines were preserved otherwise eroded too much during the process. One side effect of this new condition is that more extraneous pixels would be retained.
The more recent version of this algorithm does not delete isolated pixels ($X_R(p) = 0$). It requires the condition $2 \geq b(p) \leq 6$, and it does not preserve 4-connectivity. Therefore, a pixel p at a vertex of a right angle formed by two 4-neighbours would be deleted, leading to a condition $x_2 + x_6 \neq 0$. Some variants of this method implement look-up tables to assess the rules and the configurations[47].
So far we have seen an algorithms that do not preserve topology. On the other hand, a 4-iteration algorithm that does that is the one of Bel-Lan and Montoto described in "A thinning transform for binary images" [47]. In this method, skeletal pixels found in each iteration are assigned the iteration number for the subsequent calculation of object width. The values of skeletal pixels from previous iteration are left unchanged, whereas those of interior pixels are incremented by one in each iteration. In the n^{th} iteration, the pixel p is tested for for deletion as a north border element if $x_3 = 0$, p has value n, and $x_7 \neq 0$. In this way such a border element p is considered skeletal if there exists $x_i \in N(p)|x_i > 0$ and $N(x_i) \cap N(p) = p$,x_i, otherwise p is deleted. Unfortunately, this algorithm is found to be sensitive to boundary noise and to the order of subiteration[47].
Stefanelli and Rosenfeld in their work "Some parallel thinning algorithms for digital pictures" proposed other two 4-subiteration algorithms that preserved topology. In each subcycle, the final "skeletal" pixels are stored. In order to avoid deletion of contour points that are also final points, all the contour pixels are fist deleted, after which the final points are added. The condition for final point deletion can be seen in Fig.3.4.

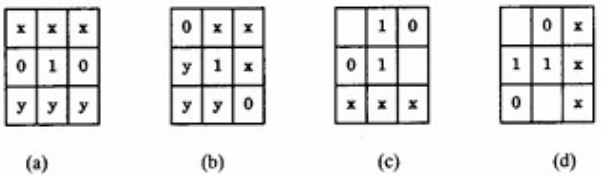

Figure 3.4: Final points condition

The parallel thinning algorithm used in MATLAB

As we said, to process the image we used the ”image processing” toolbox in MATLAB. In this library, there is a function called ”bwmorph()” that we implemented as

```
frame_skel= bwmorph(frame_contour,'thin',Inf);
```

Where ’thin’ specifies the fact that we want to use a thinning algorithm, ’Inf’ specifies the number of iteration that we want to use and frame_contour is the output of the function:

```
frame_contour = activecontour(frame, frame_thresh_emp_filt,'edge');
```

In the last function, the input argument is the original frame (frame), the frame with a threshold applied to it and filtered with a median filter(frame_thresh_emp_filt) and the type of active contour to use (without edges). The first function presented, ”bwmorph(frame_contour,’thin’,Inf)”, is described in the paper ”Parallel thinning with two subiteration algorithms” by Guo and Hall[47]. This paper is a direct evolution of the procedure presented in the ”Binary picture thinning by an iterative parallel two subiteration algorithm”[47]. In both paper the Hilditch crossing number $X_H(p)$ is used to examine pixels for deletion. Both algorithms uses a two subiteration method. Starting from ”Binary picture thinning by an iterative parallel two subiteration algorithm”, this algorithm implements approximately a 4×4 window shown in Fig. 3.5.

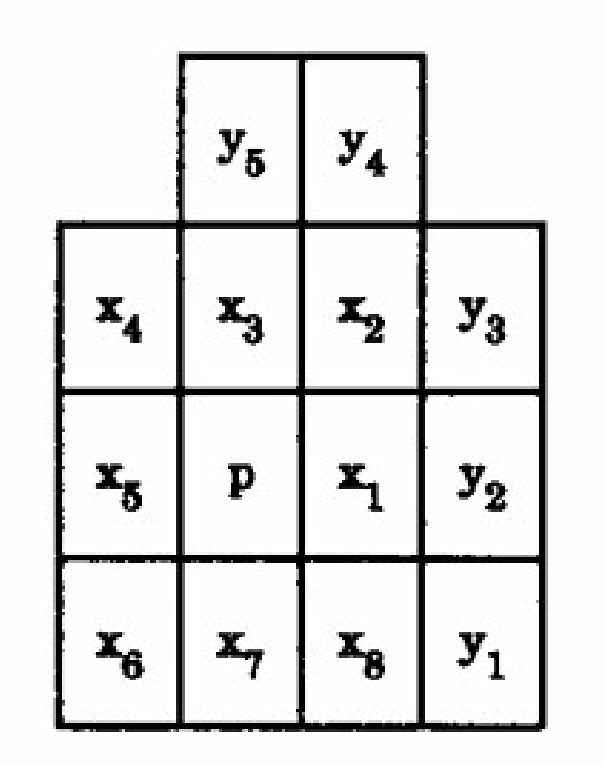

Figure 3.5: Window first MATLAB algorithm

The pixels removed in the first subcycle are deletable north contour pixels, and

- deletable north contour pixels;
- west contour pixels that are deletable after the pixels in the previous point have been removed;

In this subcycle, p is deletable if and only if:

1. $X_H < 2$
2. $(b(p) > 2) \vee [b(p) = 2 \wedge \sum_{k=1}^{8} x_k x_{k+1} = 0]$ and
3. $(x_3 = 0) \vee [(x_5 = 0) \wedge S4 \wedge S5]$, where
4. S4: $x_2 \vee \overline{x}_8(y_1 \vee y_2 \vee y_3) \vee \overline{y}_2 y_3 = 1$ and
5. S5: $\overline{y}_4 \vee y_5 \vee \overline{x}_2 y_4 = 1$

For the second subcycle the conditions are 1,2 and the $180°$ rotations of 3-5. Now that we have the basics of the ”Binary picture thinning by an iterative parallel two subiteration algorithm” we can analyze further the ”Parallel thinning with two subiteration algorithms”. In this paper two algorithms are presented.

In the first algorithm, the two subiteration explained in the first paper are implemented as well. For the first one, the image is divided into two distinct subfields in a checkerboard pattern, and in each subiteration the pixel p is deleted if and only if p is a contour point, $b(p) > 1$, and $X_H(p) = 1$. This procedure results in a noisy spike and zigzagging vertical or horizontal lines. The second algorithm, the one implemented in MATLAB, is a modification of the algorithm presented in "A fast parallel algorithm for thinning digital pattern" by Zhang and Suen [49] briefly described in the previous subsection. Instead of using $X_R(p)$, this method uses $X_H(p)$ and the result is an 8-connected skeleton.
Under this scheme, p is deleted if and only if:

1. $X_H(p) = 1$;
2. $2 \leq \min n_1(p), n_2(p) \leq 3$, where $n_1(p) = \sum_{k=1}^{4} x_{2k-1} \vee x_{2k}$ and $n_2(p) = \sum_{k=1}^{4} x_{2k} \vee x_{2k+1}$ represent the number of 4-adjacent pair of pixels in $N(p)$ containing one or two black pixels, and
3. $(x_1 \vee x_3 \vee \overline{x}_8) \wedge x_1 = 0$ in the first subiteration and its $180°$ rotation in the second.

A further adjustment has been implemented in this scheme where Labelling has been added. In particular, one subcycle thinning is accomplished by recording the pattern pixel to incorporate connectivity information from a 5×5 into the coded pixels of $N(p)$. Two initial scans are used to recode the pixels into core (*c*), interior(*i*), rim (*r*) and skeleton points(*s*). This is the labelling procedure.
The basic rule is to replace *r* pixels by *b* (background) pixels if horizontal or vertical *irb* sequence is present, with exceptions made to preserve connectivity and end points. In this way, this "bwmorp", the algorithm that we used in MATLAB, is similar to the "ideal" method for thinning proposed by Eckhardt and Maderlechner in "Thinning algorithms for document processing systems" [50].

Chapter 4

Conclusion

So far we have seen an introduction in the first chapter, where the various role and components of Calcium signature were presented. In the second chapter, a key function of our code, the ”active contour without edges”, was given. In the third chapter another key function of our code was presented, the thinning algorithm for skeletonization.
Finally in this last chapter, the whole algorithm is explained.

4.1 The Code

Other than beginning with the usual ”close all, clear all, clc”, the code that we wrote starts by reading two videos, one of the root and another of the ”ratio” of the root. Then the we added an user input to tell if the root in the video moves or not. Depending on the user input, the algorithm will take two different paths in posing the ROIs to collect the calcium signature. Another line is added to save the results of the skeletonization in another video. Finally, we preallocate some variables.

```
%Loading the two videos: video and video_ratio
video = VideoReader('video848.avi');
video_ratio = VideoReader('video_ratio848.avi');

%Collect user input
flag = input('If the root moves, enter 1: ');

%Creating the elaborated video
writerObject = VideoWriter('draftfinal.avi');
writerObject.FrameRate = 30;
open(writerObject)
```

```
%Preallocate the variabiles
length = zeros(1,video.NumFrames);
area = zeros(1,video.NumFrames);
area_ROIs_edge = NaN(12,video.NumFrames,50);
```

It important to notice the preallocation of the variable ”area_ROIs_edge”. This variable will be important further on.
Then a big for loop is introduced to read frame by frame the video and analyze the content of it. This loop contains the core of the algorithm.

```
for i = 1:video.NumFrames %i = number of frames
    ...
end
```

Diving right into the for loop, the first thing to do is read a frame of the ”video” and one of the ”video_ratio”. Then those images are converted in gray scale and converted sequentially into double for the sake of doing operations.

```
%Reading the number of frames-----------------------------------------
frame = readFrame(video);
frame_ratio = readFrame(video_ratio);

%Converting rgb to gray scale-----------------------------------------
if ndims(frame)==3
frame = rgb2gray(frame);
end

if ndims(frame_ratio)==3
frame_ratio = rgb2gray(frame_ratio);
end

frame = im2double(frame);
frame_ratio = im2double(frame_ratio);
```

Since the video_ratio was not of the same size of the video, we had to pad the array using the ”padarray” function from the image processing toolobox.

```
frame_ratio = padarray(frame_ratio, ...
[0, (size(frame,2)-size(frame_ratio,2))], "post");
```

The function adds 0s at up to the Δ of the number of columns from frame and frame_ratio. This padding is added at the end of the image (right part).
In the next step we decided to apply a empirical threshold of the image. This threshold is calculated as the mean of the matrix constituting the image converted into gray scale and double. Then a median filter is applied to the thresholded image and the final "frame_elab" is the result of the thresholding applied as a mask to the original frame. The image is then adjusted according to its gray scale.
The same thing is done, with a different strategy, to the frame_ratio.

```
%Thresholding-------------------------------------------------------
% Empirica
th_emp = mean2(frame); % 0.3*std2(frame);
frame_thresh_emp = frame>th_emp;


frame_thresh_emp_filt = medfilt2(frame_thresh_emp);
frame_elab = imadjust(frame-imcomplement(frame_thresh_emp_filt));
frame_ratio_elab = frame_ratio.*frame_thresh_emp_filt;
```

Then we calculated the contour of the image to which apply the thinning algorithm with the function "geodesic active contour", explained in Chapter 2.

```
% Snakes (Nice, Heavy computationally)--------------------------------
frame_contour = activecontour(frame, frame_thresh_emp_filt,'edge');
```

This function is applied to the original frame, using "frame_thresh_emp_filt" as a starting point on where to draw the snake. The last parameter, "edge" specifies the use of geodesic active contour model, instead of the Chan-Vase one.
The next operation is required for later on, when we will find the Region of Interest (ROIs) on the edge of the root.

```
%Here I find the edge-----------------------------------------------
frame_edge = edge(frame_contour, 'prewitt');
```

This function uses Sobel approximation to the derivative [51].
The next line of code implements the thinning algorithm described in chapter 3.

```
%Here I find the skeleton-------------------------------------------
frame_skel= bwmorph(frame_contour,'thin',Inf);
```

The parameters of this function are the black and white image to which apply the thinning process, the operation ("thin") and the amount of iteation to do before stopping. In our case, with

the parameter ”Inf” we ask the algorithm to stop when there are no sigificant changes in the skeleton from one iteration to another.
The next chunck of code explains how to find the points along the medial axes (skeleton). There are some controls done on the skeleton in order to match the x-values and the y-values.

```
%Here I create the ROIs-------------------------------------------------
%starting point
x_init = sum(frame_skel,2);
%Body
[row_init,~] = find(x_init==1); % y-axes
[~,col_init] = find(frame_skel(row_init,:)==1); %x-axis

%If the line drawn bt row_int and col_init (a series of points) don't
%match the one of frame_skel, I flip the row_init. In this way it
%matches
if ~(frame_skel(row_init(1),col_init(1)))
    row_init = flip(row_init);
end

%Same as above, but for the columns in case they don't match
% if ~(frame_skel(row_init(1),col_init(1)))
%     col_init = flip(col_init);
% end
```

Then we searched for the centers of the ROIs by having a distance of 100 pixel wise one form another. The distance used is the euclidean distance.

```
%Isolating the centers-------------------------------------------------

z = 1; %Counter number of centers
centers(1,1,i) = col_init(1); %x-coordinate
centers(1,2,i) = row_init(1); %y-coordinate

%Here I start counting from 1
for j = 1:size(row_init)
    temp_x = col_init(j); %Temp value x imagine
    temp_y = row_init(j); %Temp value y image
    %Norm
```

```
        dist = sqrt((centers(z,1,i)-temp_x).^2+(centers(z,2,i)-temp_y).^2);

        %If the distance between the center and the temp_point(x,y) is
        %bigger than 100 (empirical measure) I choose the candidate point
        %as a new center
        if dist> 100
            z = z+1;
            centers(z,1,i)= temp_x;
            centers(z,2,i)= temp_y;
        end
        %This if control that the number of centers is constant throughout
        %the whoule video. Since we have an arry, MATLAB implements it as
        %constant in the number of colums that it has. I check if the
        %last row of the current frame have been filled or not by using the
        %length of the previously created (1:i-1) array to match the size.
        %If I don't do this, it can happen that the number of rows of the
        %matrix containg the number of centers changes (i.e from 6 to 7)
        %halfway through the video. (It can be imoproved)
        if ((i>1) && (centers(size(centers,1),1,i)~=0))
            break;
        end
    end
```

In the next section we searched for the ROIs on the edge of the root. A particular useful function that we used is the ”bwtraceboundary()” function that, given a starting point, gives the sequence of the points on the boundary. We decided to search for the next pixel in the ”south” direction of the $N(p)$ neighbourhood (Fig. 3.2). We then decided to search for the centers of the ROIs with a distance one from another of 40 pixels. Again, the distance used here is the euclidean distance.

```
     %Here I create the ROIs on the border----------------------------------
    %With "find" I look for the starting point of the edge
    [row_edge, col_edge] = find(frame_edge==1,1);
    %With bwtraceboundary I select the row and the column of the edge
    %SEQUENTIALLY
    boundary_edge = bwtraceboundary(frame_edge, [row_edge col_edge],'S');

    %Counter of the number of centers on the edge
```

```
u = 1;
%Initialization of the number of centers
centers_edge(1,1,i) = boundary_edge(1,1);
centers_edge(1,2,i) = boundary_edge(1,2);

for e = 1:size(boundary_edge,1) %e = n rows boundary edge
    temp_x_edge = boundary_edge(e,2);
    temp_y_edge = boundary_edge(e,1);
    dist_edges = sqrt((centers_edge(u,1,i)-temp_x_edge).^2+ ...
        (centers_edge(u,2,i)-temp_y_edge).^2);
    if dist_edges > 40
        u = u+1;
        centers_edge(u,1,i) = temp_x_edge;
        centers_edge(u,2,i) = temp_y_edge;
    end
end
```

The centers of the ROIs on the edge are stored in the centers_edge variable.
The next chunk of code describes the subdivision of the ROIs on the edge of the root. The delta_y variable is the smallest interval in which the bigger set of centers_edge is subdivided. The centers_edge interval is subdivided in 15 parts.
The y_delta variables contains all the ROIs selected with delta_y. The value of the corresponding centers_edge is then stored in a cell array, since the length of each column is different.

```
%Here I try to divide the centers_edge in sections---------------------
%Delta y
delta_y = ceil((max(centers_edge(:,2,1))-min(centers_edge(:,2,1)))/15);

%Position (x,y) of the centers belonging to one of the 5 classes. Here
%I put k = 1:16 because some centers where left out when dividing the
%edge ROIs in segments after the 15th  iteration of the loop
for k  = 1:16
    %Here I structured the y_delta like this because when it shrinks,
    %the section will be proportional to shrinkage. This is because I
    %subdivide the intervals (shrinked and non shrinked) in 5
    %proportional sections
    y_delta = find((centers_edge(:,2,i)>(k-1)*delta_y) & ...
        (centers_edge(:,2,i) < k*delta_y));
```

```
        for kk = 1:size(y_delta)
            %Here I had to add memory to the variable coordinates_group_x
            %and y otherwise the plot of the ROIs were distorted

            %x-coordinates
            coordinates_group_x{kk,k,i} = centers_edge(y_delta(kk),1,i);
            %y-coordinates
            coordinates_group_y{kk,k,i} = centers_edge(y_delta(kk),2,i);
        end
    end
```

Then there are the plots. We decided to creat an unique figure with 5 subplots.
Here we have the subplot for the original frame.

```
figure(1)

%Subplot 1
subplot(151)
imshow(frame)
title('Original frame')
```

Here we have the subplot for the frame_elaborated. This section is composed of a big if clauses and a loop for each clause. The first if is directly connected to the user input to see if the root moves or not. If the root moves (input = 1), then the ROIs are moved with the skeleton. If the root does not move, the ROIs are fixed in the skeleton. This is the best way that we found to pick up the Ca^{++} signature from the root. The radius of the ROIs on the skeleton is 50 pixels. The next step is to draw the ROIs on the edge of the root. We had a few problems with those but we managed to plot them right. The radius of the ROIs on the edge is 20 pixels.
To calculate the signal intensity, in both ROIs_edge and ROI_skeleton, we decided to:

- isolate the pixels according to the ROI on the frame_ratio frame;
- create a temp image that isolate the pixels from frame_ratio_elab according to the mask. The frame_ratio_elab is the frame_ratio image with the threshold obtained from the original frame applied to it.

```
%Subplot 2
subplot(152)
imshow(frame_elab)
```

```
hold on
%Vision boundaries
visboundaries(frame_skel,'Color','g','LineWidth',1)
%Here I put the user input in the if clause
if flag == 1
    % Now I create multiple masks
    %Here I create the circular ROI
    for u = 1:size(centers,1) % size(centers,1):-1:1
        h = drawcircle('Center', [centers(u,1,i) centers(u,2,i)], ...
            'Radius', 50, ...
            'Color', 'w', 'LineWidth', 1,'Label', num2str(u));
        % Here I create a mask such that it has only ones insde the ROI
        mask = createMask(h);
        % Here I isolate the pixels with the mask from the frame_ratio
        temp_image = frame_ratio_elab.*mask;
        % Here I apply the ROI to the frame_threshold
        temp_ROI = frame_thresh_emp_filt.*mask;

        %area_ROIs is calculated by summing the pixels of the
        % frame_ratio with the threshold and the ROI applied to it and
        % the sum of the pixels of the frame_threshold with the ROI
        % applied to it

        % Here I calculate the Calcium content inside the ROI
        area_ROIs(u,i) = sum(temp_image(:))/(sum(temp_ROI(:)));
    end
    %plot(centers(:,1,i),centers(:,2,i), '*k')
else
    for u = 1:size(centers,1) %size(centers,1):-1:1
        %Here I create the constant circular ROIs
        h = drawcircle('Center', [centers(u,1,1) centers(u,2,1)], ...
            'Radius', 50, ...
            'Color', 'w', 'LineWidth', 1, 'Label', num2str(u));
        % Here I create a mask such that it has only ones insde the ROI
        mask = createMask(h);
        % Here I isolate the pixels with the mask from the frame_ratio
```

```
        temp_image = frame_ratio_elab.*mask;
        % Here I apply the ROI to the frame_threshold
        temp_ROI = frame_thresh_emp_filt.*mask;

        %area_ROIs is calculated by summing the pixels of the
        % frame_ratio with the threshold and the ROI applied to it
        % and the sum of the pixels of the frame_threshold with the ROI
        % applied to it

        % Here I calculate the Calcium content inside the ROI
        area_ROIs(u,i) = sum(temp_image(:))/(sum(temp_ROI(:)));
    end
end

%Here I plot the centers_edge to see the possibile position of the ROI
% ii = number of columns of coordinates_group_x. 1 out of 6 groups
%      (subdivision of the total)
% u = number of rows (y) coordinates_group_x
for ii = size(coordinates_group_x,2):-1:1 %1:size(coordinates_group_x,2)

        for uu = 1:size(coordinates_group_x,1)
            if isempty(coordinates_group_x{uu,ii,i})
                break;
            end
            %Here I create the circular ROI
            h_edge = drawcircle('Center', [cell2mat( ...
                coordinates_group_x(uu,ii,i)) cell2mat( ...
                coordinates_group_y(uu,ii,i))],'Radius', 20, ...
                            'Color', 'y', 'LineWidth', 1);
            % Here I create a mask such that it has only ones
            % insde the ROI
            mask_edge = createMask(h_edge);
            % Here I isolate the pixels with the mask from the
            % frame_ratio
            temp_image_edge = frame_ratio_elab.*mask_edge;
            % Here I apply the ROI to the frame_threshold
```

```
            temp_ROI_edge = frame_thresh_emp_filt.*mask_edge;

            area_ROIs_edge(uu,i,ii) = sum( ...
                temp_image_edge(:))/(sum(temp_ROI_edge(:)));
        end
    end
    %Here I delete the extra columns
    area_ROIs_edge(:,:,(ii+1):end) = [];

    hold off
    title('Frame elaborated')
```

Then we have the third subplot. In this subplot we plot the skeleton along with the centers of the ROIs on the skeleton. In here there is the same type of if clause as in the previous subplot.

```
%Subplot 3
subplot(153)
imshow (frame_skel)
hold on
if flag == 1
    plot(centers(:,1,i),centers(:,2,i), 'ow')
else
     plot(centers(:,1,1),centers(:,2,1), 'ow')
end
hold off
title('Skeleton')
```

In the fourth subplot, we decided to plot the centers of the ROIs on the edge.

```
%Subplot 4
subplot(154)
imshow(frame_edge)
hold on
%Here I plot the centers edge
plot(centers_edge(:,1,i),centers_edge(:,2,i),'yo')
title('Edge')
hold off
```

The last subplot is the one representing the frame_ratio. We decided to add few separating lines according to the subdivision done on the ROIs on the edge.

```
%Subplot 5
subplot(155)
imshow(frame_ratio)
title('Frame ratio')
hold on
for g = 1:15
line([0 size(frame_ratio,2)], ...
    [(max(boundary_edge(:,1))-(g-1)*delta_y) ...
    (max(boundary_edge(:,1))-(g-1)*delta_y)], 'Color','white')
text(0, (max(boundary_edge(:,1))-g*delta_y)+delta_y/2, num2str(g), ...
    'Color', 'white');
end
hold off
```

Then some operations are added to the final frame in order to output a video of the frame and the skeleton

```
%Operazioni sul frame finale prima di creare il video------------------
final_frame = im2double(frame_elab+frame_skel);
final_frame = insertText(final_frame,[10 50;10 100], ...
    {['length skel=',num2str(length(i))],[' area=', num2str( ...
    area(i))]}, TextColor="white", FontSize = 20);
%This is because insertText writes labels in color
final_frame = rgb2gray(final_frame);

%Final video-----------------------------------------------------------
writeVideo(writerObject,final_frame);
```

This is the end of the for loop. We analyzed frame by frame the Ca^{++} signal on the skeleton and on the edge of the root. We picked up the signal, which is directly proportional to the intensity of the pixel, and we stored in the variables area_ROIs and area_ROIs_edge.
The following part of the code is concerned with the plot of the results, therefore the normal plot, the heatmap and the surface plot of the Ca^{++} signature that we picked up.
First we plotted the signal from the ROIs along the skeleton using a normal plot function:

```
    %Here I plot the values of the ROIs----------------------------------------
figure(2)
%This for loop plots the values inside each ROI
for q = 1:size(area_ROIs,1)
    %This if clause checks if there are Infs or NaNs inside the values
    %collected by each ROI. If there is a NaN or an Inf the results of the
    %ROI(q) are not plotted. In this way I take out the ROIs that moved and
    %did not collected the signal of the root continously.
    %(It can be improved)
    if (sum(isinf(area_ROIs(q,:)))==0 && sum(isnan(area_ROIs(q,:)))==0)
        plot(1:size(area_ROIs,2), area_ROIs(q,:),'-', 'DisplayName', ...
            strcat('ROI number=',num2str(q)))
        hold on
    end
end
title('Calcium singnal in the root')
xlabel('Frame')
ylabel('Signal in pixel intensity')
% I don't know how to show a lengend that varies according to the number of
% plots in MATLAB
legend('show')
```

The results (Fig.4.1) is the following for each ROI:

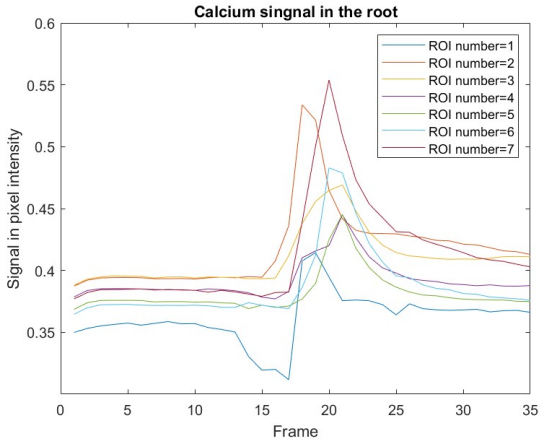

Figure 4.1: Skeleton ROIs plot

The spiky profile that we can see in this plot suggest that the Ca^{++} signature is produced by salt in the cytosol. Other responses can be attributed to NES (Nuclear Export Signal) or 4-MET.

In NES the signalling event is due to an amount of ATP in the cytosol. In 4-MET the signalling event is due to an amount of ATP in the mitochondria. The two peofiles can be seen below:

NES:

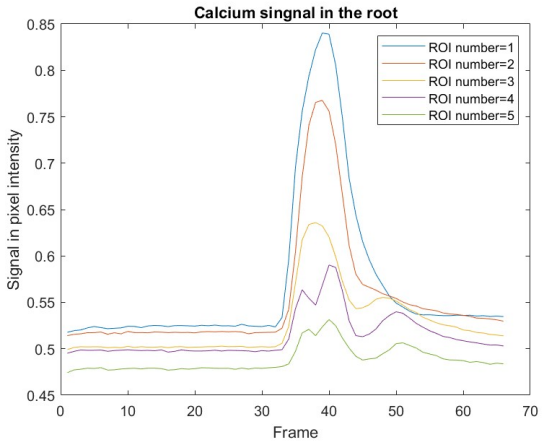

Figure 4.2: NES signature

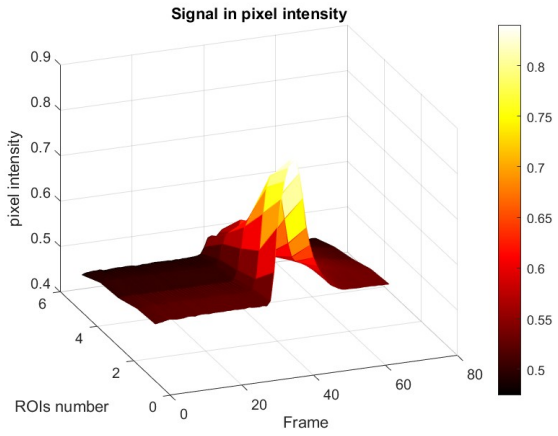

Figure 4.3: NES surface signature

As it can be seen from Fig.4.2 and Fig.4.3, the profile of the peak in signal is more smooth if the event that generates it is the ATP from the cytosol.

4-MET:

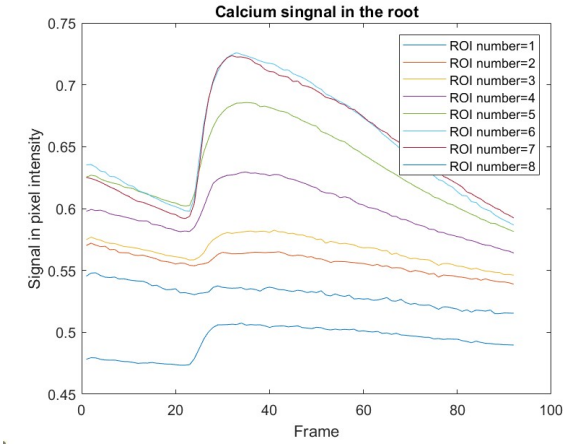

Figure 4.4: 4-MET signature

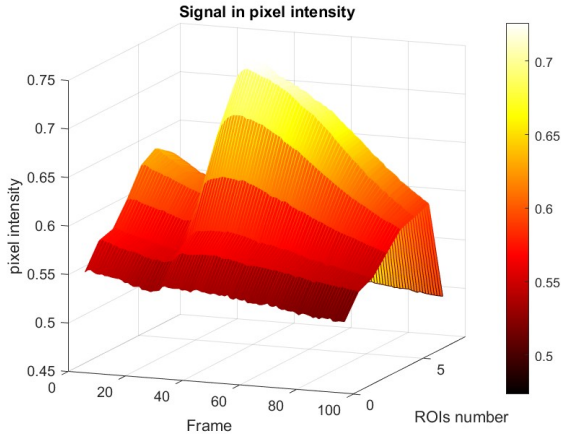

Figure 4.5: 4-MET surface signature

The difference in profile here is remarkable in both Fig.4.4 and Fig.4.5. The profile is more smooth and elongated.

The final part of the code is concerned with plotting the results of the ROIs on the edge. Due to the heavy operations done on the edge ROIs, this is the most fragile part of the code. We decided to plot the profiles using a normal plot and an heatmap.

```
% Plots of edge ROI-------------------------------------------------------
% I average the by the group ROI
%I preallocate the variable reduced_dim_area_ROIs_edge
 reduced_dim_area_ROIs_edge = NaN(size(area_ROIs_edge,3), ...
     size(area_ROIs_edge,2));
% I reduce the dimenionality by averaging
for t = 1:size(area_ROIs_edge,3)
```

```
    reduced_dim_area_ROIs_edge(t,:) = mean(area_ROIs_edge(:,:,t), ...
        'omitnan');
end

%I plot the results using HEATMAP
figure(7)
hh = heatmap(reduced_dim_area_ROIs_edge);
colormap('hot')
colorbar;
hh.Title = 'Signal in pixel intensity on the edge';
hh.XLabel = 'Frame';
hh.YLabel = 'group ROIs number averaged (height on the root)';
```

For what it concerns the signature due to the salt in the cytosol, the results are the following:

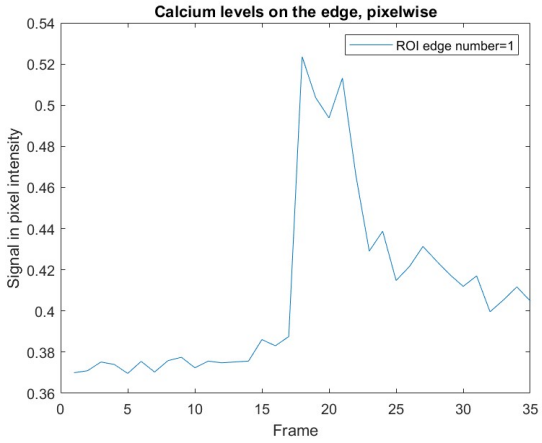

Figure 4.6: Plot ROIs edge

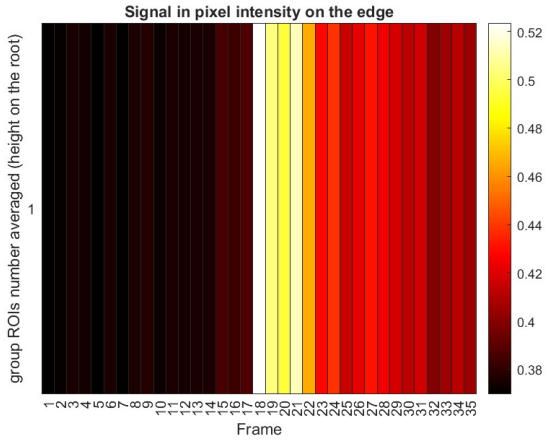

Figure 4.7: Heatmap ROIs edge

In addition, in the last few line we display some parameters such the length of the skeleton and the area of the root:

```
    % Final display of the maximum variation of the length of the skeleton and
% the area inside the contour-----------------------------------------------
disp(['variazione lunghezza scheletero nel tempo:', ...
    num2str((max(length)-min(length))),' pixel', ...
    '   |    variazione area della radice nel tempo:', ...
    num2str(max(area)-min(area)),' pixel']);
```

4.2 The Results

As we have mentioned before, the videos represent three types of experiments conducted:

- The NES experiments (video1211, video1213 and video1214);
- The 4MET experiments (video1263, video1264 and video1267);
- Salt in the cytosol experiments (video847, video848 and video852);

NES (Nuclear Export Signal) calcium signaling refers to mechanisms involving the transport of calcium-related signaling molecules and proteins between the nucleus and the cytoplasm in the cell. The Ca^{++} signature is generated by the ATP in the cytosol.
The 4MET calcium signaling refers to an experiment carried out stimulating the stress response generated by the mitochondrial ATP.
The last set of experiments consists on analyzing the stress response of the plant when salt is put in the cytosol.

4.2.1 NES

The profiles that we found on the NES experiments can be seen in the figures down below:

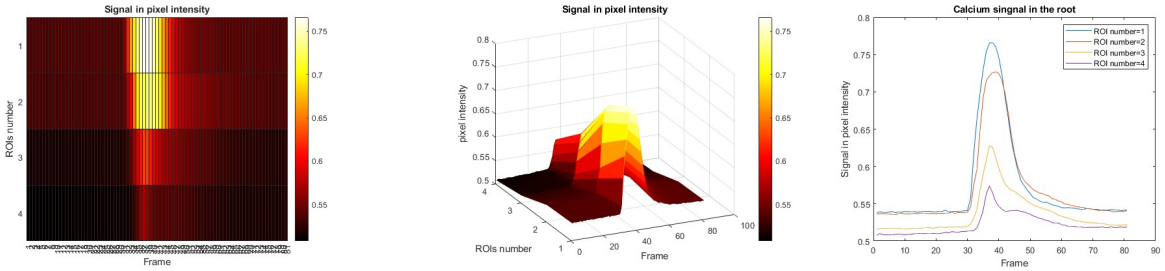

Figure 4.8: experiment 1211

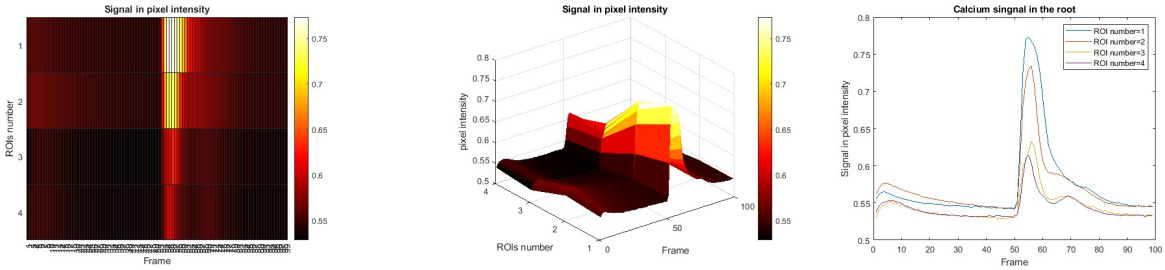

Figure 4.9: experiment 1213

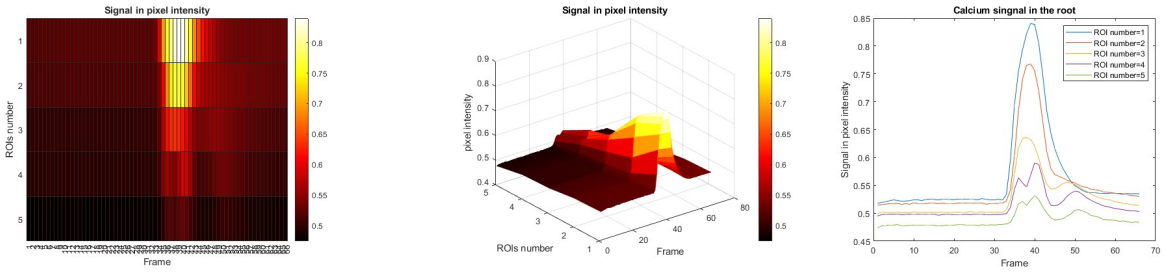

Figure 4.10: experiment 1214

Fig.4.8 refers to experiment 1211, Fig 4.9 refers to experiment 1213 and Fig.4.10 refers to experiment 1214.

As it can be seen from the plots, the profiles are not spiky. All the three shapes match. There is a gradient that goes from the tip of the root, the part of the graph with highest intensity, to the top of it, the part with less signal. The peak of the signal is registered around frame 30-50. We can refer to this figures to identify the characteristic shape of stress response provoked by ATP in the cytosol.

4.2.2 4MET

The profiles that we found on the 4MET experiments can be seen in the figures down below:

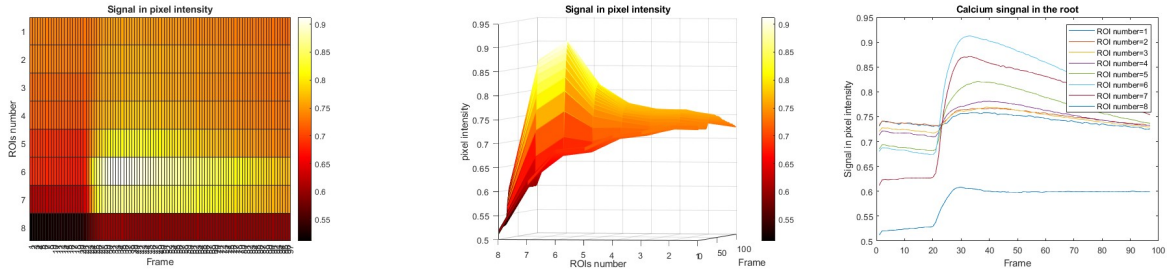

Figure 4.11: experiment 1263

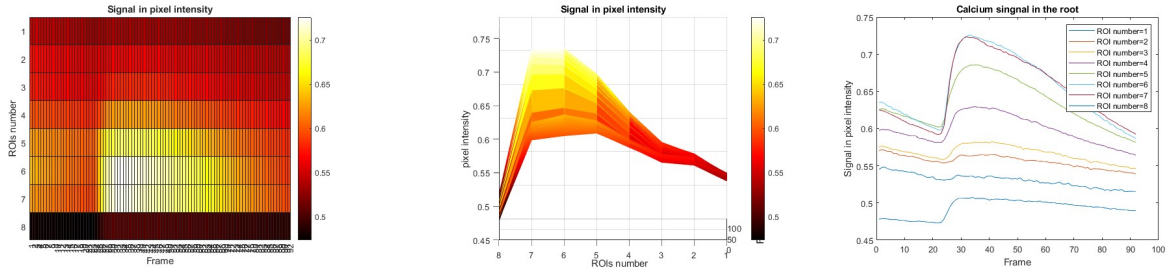

Figure 4.12: experiment 1264

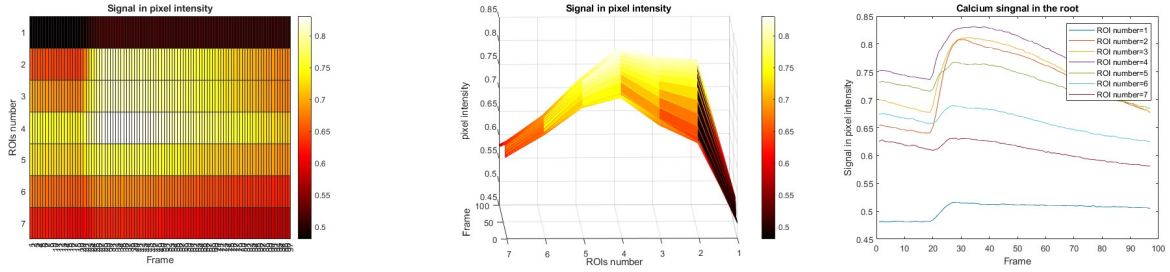

Figure 4.13: experiment 1267

Fig.4.11 refers to experiment 1263, Fig 4.12 refers to experiment 1264 and Fig.4.13 refers to experiment 1267.

The profiles on the graphs match. The only concern is experiment 1267 (Fig.4.13) where in the last ROI the signal is very noisy. Here an initial peak is registered around frame 20-30, after that there is a slow decreasing slope up until the end.

We can refer to this figures to identify the characteristic shape of stress response provoked by ATP in the cytosol.

4.2.3 Salt in the Cytsol

The profiles that we found on the salt in the cytosol experiments can be seen in the figures down below:

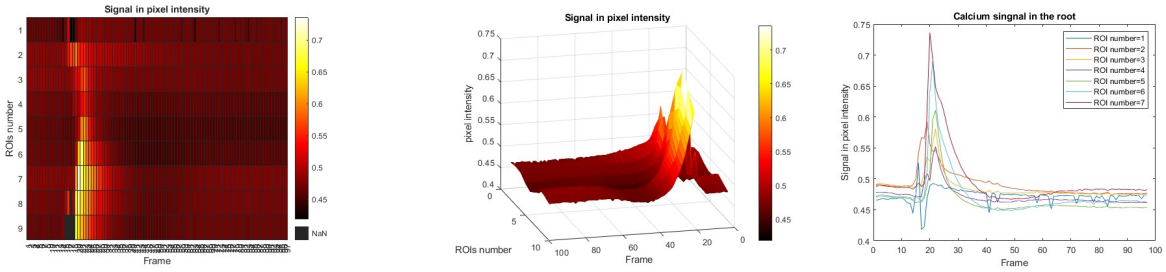

Figure 4.14: experiment 847

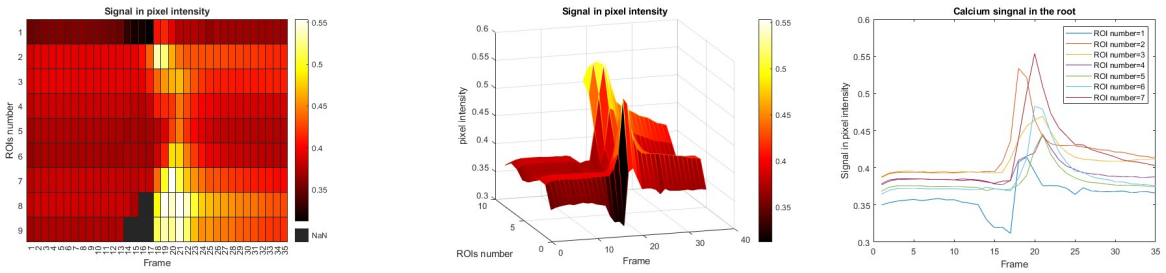

Figure 4.15: experiment 848

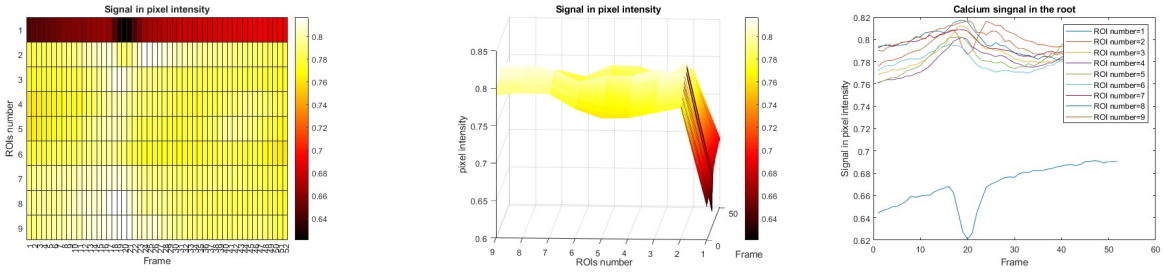

Figure 4.16: experiment 852

Fig.4.14 refers to experiment 847, Fig 4.15 refers to experiment 848 and Fig.4.16 refers to experiment 852.

Here the profiles match, excepts for experiment 852.

The shape of the signal is very spiky, this is typical in this type of experiments. All the peaks from the different ROIs are more or less aligned. The analysis of experiment 852 is not optimal, the profiles that come out from it have to high intensity. This is may due to a non-optimal data (video). Overall I would consider experiment 847 (Fig.4.14) and experiment 848 (Fig.4.15) as reference signals for future analysis.

4.2.4 ROIs on the edge

For what it concerns the ROIs on the edge the results were less significant. We were able to place the region of interest along the contour, but the results were not optimal. Maybe a finer work should be done in dividing the ROIs in group. The best profiles that we were able to isolate are the ones of video_848, video_852 and video_1213:

Figure 4.17: experiment 848 edge

Figure 4.18: experiment 852 edge

Figure 4.19: experiment 1213 edge

As you can see in Fig.4.17, the profiles are not optimal, as there are two spikes. In Fig.4.18, the profile is not smooth and it is really noisy. In Fig.4.19 some data points are missing, the final result is an incomplete profile of the Ca^{++}-signature.
The output figures for the other experiments were partial or completely absent.

4.2.5 Conclusions

The three experiments have fairly different profiles. There ar normal peaks in ”NES” experiments, there are long lasting peaks in the ”4MET” experiments and there are spiky peaks in the ”salt in the cytosol” experiments. Overall, one could use the software that we developed to analyze the type of stress response that undergoes a plant in an experiment. The optimal settings to make this program work properly are the ones regarding experiment 848, where the root is position with the top on the right and its profile can be seen well (Fig.1.10).

Bibliography

[1] N. Tuteja and S. Mahajan, "Calcium signalling netwoek in plants," *Plant Signaling & Behaviour*, 2007.

[2] D. Sanders, C. Brownlee, and J. F. Harper, "Communicating with calcium," *The Plant Cell*, 1999.

[3] K. Edel, E. Marchadier, C. Brownlee, J. Kudia, and A. Heterington, "The evolution of calcium-based signalling in plants," *Current Biology*, 2017.

[4] R. Kretsinger and C. Nockolds, "Carp muscle calcium-binding protein. ii. structure determination and general description," *J. Biol Chem*, 1973.

[5] B. Ranty, D. Aldon, V. Cotelle, J.-P. Galaud, P. Thuleau, and C. Mazars*, "Calcium sensors as key hubs in plant responses to biotic and abiotic stresses," *Frontiers in Plant Science*, 2016.

[6] D. S. Bush, "Calcium regulation in plant cells and its role in signaling," *Annu. Rev. Plant Physiol*, 1995.

[7] C. Lachaud, E. Prigent, P. Thuleau, S. Grat, D. Da Silva, and C. Briere, "14-3-3-regulated Ca^{++}-dependent protein kinase cpk3 is required for sphingolipid-induced cell death in arabidopsis.," *Cell Death Differ*, 2013.

[8] H. Plattener and A. Verkharktsky, "Cell calcium," *Elsver*, 2014.

[9] P. Camacho and J. Lechleiter, "Increased frequency of calcium waves in xenopus laevis oocytes that express a calcium-atpase," *Science 260*, 1993.

[10] A. Cubitt, I. Reddy, S. Lee, J. McNally, and R. Firtel, "Coexpression of a constitutively active plasma membrane calcium pump with green fluorescence protein identifies roles for intracellular calcium in controlling cell sorting during morphogenesis in dictyostelium," *Dev. Biol. 196*, 1998.

[11] J. Lechleiter, L. John, and P Camacho, "Ca^{++} Wave dispersion and spiral wave entrainment in xenopus laevis oocytes overexpressing Ca^{++} atpases.," *Biophys. Chem*, 1998.

[12] K Hirschi, "Vacuolar H^+/Ca^{++} transport: Who's directing the traffic?" *Trends Plant Sci*, 2001.

[13] M. McAinsh, C. Brownlee, and A. Hetherington, "Abscisic acid–induced elevation of guard cell cytosolic ca21 precedes stomatal closure," *Nature 343*, 1990.

[14] P. Shacklock, N. Read, and A. Trewavas, "Cytosolic free calcium mediates red light induced photomorphogenesis," *Nature 358*, 1992.

[15] D. Bush and R. Jones, "Cytoplasmic calcium and a-amylase secretion from barley aleurone protoplasts," *Cell Biol 46*, 1988.

[16] H. Knight, A. Trewavas, and M. Knight, ". calcium signaling in arabidopsis thaliana responding to drought and salinity," *Plant J. 12*, 1997.

[17] A. Taylor, N. Manison, C. Fernandez, J. Wood, and C Brownlee, "Spatial organization of calcium signaling involved in cell volume control in the fucus rhizoid," *Plant Cell 8*, 1996.

[18] M. Knight, A. Campbell, S. Smith, and A. Trewavas, "Transgenic plant aequorin reports the effects of cold shock and elicitors on cytoplasmic calcium," *Nature 352*, 1991.

[19] H Knight, A. Trewavas, and M. Knight, "Cold calcium signaling in arabidopsis involves two cellular pools and a change in calcium signature after acclimation," *Plant Cell*, 1996.

[20] M. Gong, A. Van de Luit, M. Knight, and A. Trewavas, "Heat-shock-induced changes in intracellular ca21 level in tobacco seedlings in relation to thermotolerance," *Plant Physiol. 116*, 1998.

[21] A. Price, A. Taylor, S. Ripley, A. Griffiths, A. Trewavas, and M. Knight, "Oxidative signals in tobacco increase cytosolic calcium," *Plant Cell 6*, 1994.

[22] D. Ehrhardt, R. Wais, and S. Long, "Calcium spiking in plant root hairs responding to rhizobium nodulation signals," *Cell 85*, 1996.

[23] S. S. Virk and R. E Cleland, "Calcium and the mechanical properties of soybean hypocotyl cell walls: Possible role of calcium and protons in cell-wall loosening," *Planta*, 1988.

[24] S. Rongpipi, W. J. Barnes, O. Siemianowski, *et al.*, "Measuring calcium content in plant using nexafs spectroscopy," *Frontiers in Plant Science*, 2023.

[25] G. Komis, O. Samajová, M. Ovecka, and J. Samaj, "Super-resolution microscopy in plant cell imaging," *Trends in Plant Science*, 2015.

[26] Aequorin Wikipedia, https://en.wikipedia.org/wiki/Aequorin.

[27] S. Dixit, A. Shukla, and D. K. Singh, "Methods for detection and measurement of calcium in plants," *Calcium Transport Elements in Plants*, 2021.

[28] I. Cone, https://www.youtube.com/watch?v=DbWtjTFSV-0, Moiré patterns and SIM.

[29] H. Stoeckel and K. Takeda, "Calcium-activated, voltage-dependent, non-selective cation currents in endosperm plasma membrane from higher plants," *Proceedings of Royal Society, London Series B 237*, 1989.

[30] E. A. Jares-Erijman and T. M. Jovin, "Fret imaging," *Nature Biotechnology Vo. 21*, 2003.

[31] S. j. Swanson and S. Gilroy, "Imaging changes in cytoplasmic calcium using the yellow cameleon 3.6 biosenor and cofocal microscopy," *Methods in Molecular Biology*, 2013.

[32] D. Marr and H. K. Nishihara, "Visual information processing: Artificial intelligence and the sensorium of sight," *TECHNOLOGY REVIEW*, 1978.

[33] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, 1988.

[34] S. K. Balcı and B. Acar, "Active contours: A brief review," *EE 570 Image Processing*,

[35] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing, vol. 7*, 1998.

[36] V. Caselles, F. Catte, T. Coll, and F. Dibos, "A geometric model for active contours," *Numer. Math., vol. 66*, 1993.

[37] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *IEEE Int'l Conf. Computer Vision*, 1995.

[38] K. Siddiqi, Y. B. Lauriere, A. Tannenbaum, and S. W. Zucker, "Area and length minimizing flows for shape segmentation," *IEEE Trans. Image Processing*, 1998.

[39] T. Chan and L. Vese, "Active contour model without edges," *Proc. Int'l Conf. Scale-Space Theories in Computer Vision*, 1999.

[40] R. T. Whitaker, "A level-set approach to 3d reconstruction from range data," *International Journal of Computer Vision, Volume 29, Issue 3, pp. 203-231*, 1998.

[41] D. Marr and E. Hildreth, "A theory of edge detection," *Proceedings of the Royal Society of London. pp-187-217*, 1980.

[42] Osher and Sethian, "Fronts propagation with curvature dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics, vol 29*, 1988.

[43] https://en.wikipedia.org/wiki/Active_shape_model.

[44] T. Cootes, C. Taylor, D. Cooper, and J.Graham, "Active shape models - their training and application," *Computer vision and image understanding*, 1994.

[45] https://en.wikipedia.org/wiki/Topological_skeleton.

[46] https://se.mathworks.com/help/images/ref/bwmorph.html#bui7chk-1.

[47] L. Lan, S.-W. Lee, and C. Y. Suen, “Thinning methodologies-a comprehensive survey,” *IEEE Transition Pattern Analysis and Machine Intelligence, Vol 13*, 1992.

[48] C. Hilitch, “Linear skeletons from square cupboards,” *Machine Intell*, 1969.

[49] T. Y. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns,” *Communications of the ACM, Volume 27, Issue 3,Pages 236 - 239*, 1984.

[50] Eckhardt and Maderlechner, “Thinning algorithms for document processing systems,” *Proc. IAPR Workshop Computer Vision: Spec. Hardware Ind. Application*, 1988.

[51] https://it.mathworks.com/help/images/ref/edge.html#d126e107373.