📋 ANALISI DEL TUO CODICE ATTUALE

🎯 Quello che HAI GIÀ e Funziona Bene:

Dal tuo `Function_def_CompletaPre.m` e `Function_def_CompletaPost.m` hai già implementato:

✅ Analisi statistica solida:

```
V_AVG = mean(subIR);            % Media verticale
V_dev = std(subIR);             % Deviazione standard verticale
sigmaV = mean(V_dev');          % σV per ratioD
H_AVG = mean(tsubIR);           % Media orizzontale
H_dev = std(tsubIR);            % Deviazione standard orizzontale
sigmaH = mean(H_dev');          % σH per ratioD
```

✅ Parametri diagnostici corretti:

```
DH_AVG = H_AVG - Tot_AVG;       % Normalizzazione temperatura
T_int = polyfit(dV, DH_AVG, 1); % Interpolazione lineare
grad_pre = (max_DH - min_DH)/L; % Gradiente termico
```

✅ Visualizzazioni base funzionanti:

- Profili verticali e orizzontali
- Temperature normalizzate
- Interpolazioni lineari
- Combinazione grafici pre/post

❌ Cosa Manca per i Tuoi Obiettivi:

1. **Etichette automatiche complete** sui grafici
2. **Salvataggio automatico organizzato** in cartelle
3. **Grafici 3D** per forme ricorrenti (richiesta relatrice)

🔧 STEP 1: MIGLIORA LE TUE FUNZIONI ESISTENTI

Upgrading `Function_def_CompletaPre.m`

MANTIENI tutta la tua logica di calcolo, **AGGIUNGI** solo etichettatura e salvataggio:

```
function risultati = Function_def_CompletaPre_Enhanced(termografia,
idCampione, cartellaProgetto)
    % La tua analisi esistente rimane identica...
    % [tutto il tuo codice attuale fino alla generazione grafici]

    % === MIGLIORAMENTO 1: ETICHETTATURA AUTOMATICA ===

    % Invece di:
    % plot(dH, V_AVG, 'r-', 'LineWidth', 1.0, 'DisplayName', 'μ pre
intervento');

    % Usa:
    fig_vert_pre = figure('Name', sprintf('Profili Verticali - %s',
idCampione));
    plot(dH, V_AVG, 'r-', 'LineWidth', 2.0, 'DisplayName', 'μ pre-
intervento');
    hold on;
    plot(dH, V_AVG+2*V_dev, 'k--', 'LineWidth', 1.5, 'DisplayName', 'μ+2σ');
    plot(dH, V_AVG-2*V_dev, 'k--', 'LineWidth', 1.5, 'DisplayName', 'μ-2σ');

    % AGGIUNGI etichettatura completa:
    xlabel('Posizione Orizzontale X [cm]', 'FontSize', 12, 'FontWeight',
'bold');
    ylabel('Temperatura Media [°C]', 'FontSize', 12, 'FontWeight', 'bold');
    title(sprintf('Profili Verticali Temperatura - Pre-intervento\nCampione:
%s | Data: %s', ...
          idCampione, datestr(now, 'dd/mm/yyyy HH:MM')), 'FontSize', 14,
'FontWeight', 'bold');
    legend('Location', 'best', 'FontSize', 10);
    grid on; box on;

    % === MIGLIORAMENTO 2: SALVATAGGIO ORGANIZZATO ===

    % Crea cartelle se non esistono
    cartellaGrafici = fullfile(cartellaProgetto, 'Grafici_Elaborati');
    if ~exist(cartellaGrafici, 'dir'), mkdir(cartellaGrafici); end

    % Salva con nomenclatura sistematica
    timestamp = datestr(now, 'yyyy-mm-dd_HH-MM');
    nomeFile = sprintf('profili_verticali_pre_%s_%s.fig', idCampione,
timestamp);
    savefig(fig_vert_pre, fullfile(cartellaGrafici, nomeFile));

    % Ripeti per tutti i tuoi grafici esistenti...

    % === RESTITUISCI RISULTATI ===
    risultati.sigmaV = sigmaV;
```

```
    risultati.sigmaH = sigmaH;
    risultati.ratioD = sigmaV/sigmaH;
    risultati.gradiente = grad_pre;
    risultati.pendenza = m;
    risultati.intercetta = q;
    risultati.idCampione = idCampione;
end
```

Applica lo Stesso Pattern a `Function_def_CompletaPost.m`

Cambia solo:

- Colori: `'r-'` → `'g-'` (verde per post)
- Titoli: `'Pre-intervento'` → `'Post-intervento'`
- Nomi file: `'_pre_'` → `'_post_'`

🎯 STEP 2: AGGIUNGI GRAFICI 3D PER FORME RICORRENTI

Nuova Funzione: `analisi3D_FormeRicorrenti.m`

```
function analisi3D_FormeRicorrenti(termografia_pre, termografia_post,
risultati_pre, risultati_post, idCampione, cartellaProgetto)
    % ANALISI3D_FORMERRICORRENTI - Implementa richiesta relatrice per
grafici 3D
    %
    % Tre variabili principali per identificare forme ricorrenti:
    % 1. Posizione X, 2. Posizione Y, 3. Variazione temperatura

    fprintf('Creazione grafici 3D per forme ricorrenti - Campione: %s\n',
idCampione);

    % Prepara dati
    IR_pre = rot90(termografia_pre);
    IR_post = rot90(termografia_post);

    % === GRAFICO 3D 1: SUPERFICI COMPARATIVE ===
    fig_3d_superfici = figure('Name', sprintf('Superfici 3D Comparative -
%s', idCampione));

    subplot(1,2,1);
    surf(IR_pre, 'EdgeColor', 'none');
    title(sprintf('Superficie Pre-intervento\n%s', idCampione),
'FontWeight', 'bold');
    xlabel('Posizione X [pixel]', 'FontWeight', 'bold');
```

```
    ylabel('Posizione Y [pixel]', 'FontWeight', 'bold');
    zlabel('Temperatura [°C]', 'FontWeight', 'bold');
    colorbar; colormap('hot');

    subplot(1,2,2);
    surf(IR_post, 'EdgeColor', 'none');
    title(sprintf('Superficie Post-intervento\n%s', idCampione),
'FontWeight', 'bold');
    xlabel('Posizione X [pixel]', 'FontWeight', 'bold');
    ylabel('Posizione Y [pixel]', 'FontWeight', 'bold');
    zlabel('Temperatura [°C]', 'FontWeight', 'bold');
    colorbar; colormap('cool');

    % === GRAFICO 3D 2: SCATTER VARIAZIONI (X, Y, ΔT) ===
    fig_3d_scatter = figure('Name', sprintf('Analisi Variazioni 3D - %s',
idCampione));

    % Calcola differenza temperatura
    temp_diff = IR_post - IR_pre;
    [X, Y] = meshgrid(1:size(temp_diff,2), 1:size(temp_diff,1));

    % Scatter 3D: Le tre variabili richieste dalla relatrice
    scatter3(X(:), Y(:), temp_diff(:), 30, temp_diff(:), 'filled');
    xlabel('Posizione X [pixel]', 'FontWeight', 'bold');
    ylabel('Posizione Y [pixel]', 'FontWeight', 'bold');
    zlabel('Variazione Temperatura [°C]', 'FontWeight', 'bold');
    title(sprintf('Analisi 3D Forme Ricorrenti\nX, Y, ΔTemperatura - %s',
idCampione), ...
          'FontSize', 14, 'FontWeight', 'bold');
    colorbar;
    grid on;

    % === GRAFICO 3D 3: IDENTIFICAZIONE PATTERN ===
    fig_3d_pattern = figure('Name', sprintf('Pattern Ricorrenti - %s',
idCampione));

    % Analisi variabilità locale per identificare pattern
    window_size = 5;
    [rows, cols] = size(temp_diff);
    variabilita = zeros(rows, cols);

    for i = window_size:rows-window_size+1
        for j = window_size:cols-window_size+1
            finestra = temp_diff(i-window_size+1:i+window_size-1, j-
window_size+1:j+window_size-1);
            variabilita(i,j) = std(finestra(:));
        end
    end

    surf(variabilita, 'EdgeColor', 'none');
```

```
    title(sprintf('Mappa Variabilità Locale\nIdentificazione Pattern - %s',
idCampione), ...
          'FontSize', 14, 'FontWeight', 'bold');
    xlabel('Posizione X [pixel]', 'FontWeight', 'bold');
    ylabel('Posizione Y [pixel]', 'FontWeight', 'bold');
    zlabel('Variabilità Locale', 'FontWeight', 'bold');
    colorbar;

    % === SALVATAGGIO ORGANIZZATO ===
    cartella3D = fullfile(cartellaProgetto, 'Visualizzazioni_3D');
    if ~exist(cartella3D, 'dir'), mkdir(cartella3D); end

    timestamp = datestr(now, 'yyyy-mm-dd_HH-MM');

    savefig(fig_3d_superfici, fullfile(cartella3D,
sprintf('superfici_3D_%s_%s.fig', idCampione, timestamp)));
    savefig(fig_3d_scatter, fullfile(cartella3D,
sprintf('scatter_3D_%s_%s.fig', idCampione, timestamp)));
    savefig(fig_3d_pattern, fullfile(cartella3D,
sprintf('pattern_3D_%s_%s.fig', idCampione, timestamp)));

    fprintf('✅ Grafici 3D salvati in: %s\n', cartella3D);

    % Analisi quantitativa forme ricorrenti
    zone_significative = abs(temp_diff) > std(temp_diff(:)) * 2;
    percentuale_cambio = sum(zone_significative(:)) /
numel(zone_significative) * 100;

    fprintf('📊 Forme ricorrenti identificate:\n');
    fprintf('   - Percentuale area cambio significativo: %.1f%%\n',
percentuale_cambio);
    fprintf('   - Variabilità media pattern: %.3f\n', mean(variabilita(:)));
end
```

🔧 STEP 3: MIGLIORA IL TUO `main.m`

Trasforma il tuo main in versione organizzata:

```
%% MAIN_ENHANCED.m - Versione migliorata del tuo main esistente

clear; clc; close all;

% === CONFIGURAZIONE PROGETTO ===
nomeProgetto = 'TesiMagistraleUmidita';
idCampione = 'T04_EdificioStorico';
```

```
% Crea struttura cartelle organizzata
cartellaProgetto = fullfile(pwd, sprintf('%s_%s', nomeProgetto, datestr(now,
'yyyy-mm-dd')));
if ~exist(cartellaProgetto, 'dir'), mkdir(cartellaProgetto); end

sottocartelle = {'Grafici_Elaborati', 'Visualizzazioni_3D',
'Dati_Statistici', 'Report'};
for i = 1:length(sottocartelle)
    cartellaSub = fullfile(cartellaProgetto, sottocartelle{i});
    if ~exist(cartellaSub, 'dir'), mkdir(cartellaSub); end
end

fprintf('📁 Progetto inizializzato: %s\n', cartellaProgetto);

% === CARICAMENTO DATI (il tuo codice esistente) ===
load('T04_1129.mat');
load('T04_2_1129.mat');

mat_pre = T04_1129;
mat_post = T04_2_1129;

fprintf('📊 Dati caricati: Pre %dx%d, Post %dx%d\n', size(mat_pre),
size(mat_post));

% === ANALISI PRE-INTERVENTO (tua funzione migliorata) ===
fprintf('\n🔍 Avvio analisi pre-intervento...\n');
risultati_pre = Function_def_CompletaPre_Enhanced(mat_pre, idCampione,
cartellaProgetto);

% === ANALISI POST-INTERVENTO (tua funzione migliorata) ===
fprintf('🔍 Avvio analisi post-intervento...\n');
risultati_post = Function_def_CompletaPost_Enhanced(mat_post, idCampione,
cartellaProgetto);

% === GRAFICI 3D FORME RICORRENTI (NUOVO - richiesta relatrice) ===
fprintf('🎯 Creazione grafici 3D per forme ricorrenti...\n');
analisi3D_FormeRicorrenti(mat_pre, mat_post, risultati_pre, risultati_post,
idCampione, cartellaProgetto);

% === UNIONE GRAFICI (il tuo codice esistente migliorato) ===
fprintf('🔗 Combinazione grafici pre/post...\n');
def_UnioneGrafici_Enhanced(risultati_pre, risultati_post, idCampione,
cartellaProgetto);

% === REPORT FINALE ===
fprintf('📄 Generazione report finale...\n');
generaReportFinale(risultati_pre, risultati_post, idCampione,
cartellaProgetto);
```

```
fprintf('\n✅ ANALISI COMPLETATA!\n');
fprintf('📁 Tutti i risultati salvati in: %s\n', cartellaProgetto);
```

🎯 STEP 4: COMPRENSIONE MATLAB AVANZATA PER LA TUA TESI

📚 Concetti MATLAB Chiave per il Tuo Lavoro:

1. Gestione Figure Professionali

```
% Invece di figure() generico, usa:
fig = figure('Name', 'Titolo Significativo', 'NumberTitle', 'off', ...
             'Position', [100 100 800 600]);  % [x y width height]

% Controllo assi avanzato:
ax = gca;  % Get Current Axes
set(ax, 'FontSize', 12, 'LineWidth', 1.5);
```

2. Salvataggio Multiplo Formati

```
% Per la tesi, salva in più formati:
savefig(fig, 'nome_grafico.fig');           % Formato MATLAB nativo
saveas(fig, 'nome_grafico.png', 'png');     % Per documenti
saveas(fig, 'nome_grafico.eps', 'epsc');    % Per pubblicazioni
```

3. Parametrizzazione Colori e Stili

```
% Definisci palette colori coerente:
colori_pre = [0.8 0.2 0.2];   % Rosso per pre-intervento
colori_post = [0.2 0.8 0.2];  % Verde per post-intervento
colori_sigma = [0.3 0.3 0.3]; % Grigio per bande σ

% Usa consistentemente:
plot(x, y, 'Color', colori_pre, 'LineWidth', 2);
```

4. Annotazioni e Frecce Esplicative

```
% Aggiungi annotazioni ai tuoi grafici:
annotation('arrow', [0.3 0.7], [0.2 0.8], 'Color', 'blue', 'LineWidth', 2);
annotation('textbox', [0.1 0.8 0.3 0.1], 'String', 'Zona Umidità Elevata',
```

```
...
            'FitBoxToText', 'on', 'BackgroundColor', 'white');
```

📊 Metriche Avanzate per la Tua Tesi:

```
function metriche_avanzate = calcolaMetricheAvanzate(risultati_pre,
risultati_post)
    % Calcola metriche specifiche per tesi magistrale

    % Efficacia intervento (0-100%)
    efficacia_ratioD = max(0, (risultati_pre.ratioD - risultati_post.ratioD)
/ risultati_pre.ratioD * 100);
    efficacia_gradiente = max(0, (abs(risultati_pre.gradiente) -
abs(risultati_post.gradiente)) / abs(risultati_pre.gradiente) * 100);

    % Score globale per tesi
    score_globale = (efficacia_ratioD + efficacia_gradiente) / 2;

    % Classificazione qualitativa
    if score_globale > 20
        classificazione = 'Intervento Molto Efficace';
    elseif score_globale > 10
        classificazione = 'Intervento Efficace';
    else
        classificazione = 'Intervento Poco Efficace';
    end

    metriche_avanzate.efficacia_ratioD = efficacia_ratioD;
    metriche_avanzate.efficacia_gradiente = efficacia_gradiente;
    metriche_avanzate.score_globale = score_globale;
    metriche_avanzate.classificazione = classificazione;
end
```

📝 STEP 5: INTEGRAZIONE CON LA TUA TESI

🎓 Come Presentare i Miglioramenti nella Tesi:

Nel Capitolo Metodologia:

```
"Il protocollo di analisi è stato ottimizzato attraverso:
1. Automazione completa dell'etichettatura grafica
2. Organizzazione sistematica degli output in cartelle tematiche
3. Implementazione di visualizzazioni tridimensionali per identificazione
```

```
pattern ricorrenti
4. Sviluppo di metriche quantitative di efficacia dell'intervento"
```

Nel Capitolo Risultati:

```
"L'analisi automatizzata ha generato un score globale di efficacia del XX%,
classificando l'intervento CNT come [classificazione]. Le visualizzazioni 3D
hanno identificato pattern ricorrenti in XX% dell'area analizzata."
```

Nelle Figure:

- Usa i grafici generati automaticamente con le etichette professionali
- Include le visualizzazioni 3D come innovazione metodologica
- Presenta il score di efficacia come metrica oggettiva

📊 Tabelle Risultati per Tesi:

```
function creaTabellaTesi(risultati_multipli, nomi_edifici)
    % Crea tabella riassuntiva per tesi

    T = table();
    T.Edificio = nomi_edifici';
    T.RatioD_Pre = [risultati_multipli.ratioD_pre]';
    T.RatioD_Post = [risultati_multipli.ratioD_post]';
    T.Miglioramento_Perc = [risultati_multipli.efficacia]';
    T.Classificazione = {risultati_multipli.classificazione}';

    % Salva per LaTeX
    writetable(T, 'tabella_risultati_tesi.csv');
    fprintf('Tabella risultati salvata per inclusione in tesi\n');
end
```

🛠️ TUTORIAL IMPLEMENTAZIONE PRATICA - MODIFICA STEP-BY-STEP

🎯 COME TRASFORMARE IL TUO CODICE ESISTENTE

Emma, questa guida ti mostra **esattamente** come modificare i tuoi file esistenti per raggiungere i tuoi obiettivi, mantenendo tutto il lavoro che hai già fatto.

📝 MODIFICA 1: `Function_def_CompletaPre.m`

🔧 Cosa Cambiare nel Tuo File Esistente:

MANTIENI tutto il tuo codice fino a questa parte:

```
%% grafico temperatura profili verticali
dH=linspace(0,(xf-xi)/calibrationFactor,b_subIR);
fig_vert_pre = figure;
```

SOSTITUISCI questa sezione:

```
% IL TUO CODICE ATTUALE:
plot(dH, V_AVG, 'r-', 'LineWidth', 1.0, 'DisplayName', 'μ pre intervento');
hold on;
plot(dH, V_AVG+2*V_dev, 'k--', 'LineWidth', 0.7, 'DisplayName', 'μ+2σ');
plot(dH, V_AVG-2*V_dev, 'k--', 'LineWidth', 0.7, 'DisplayName', 'μ-2σ');
xlabel('Posizione X [cm]');
ylabel('Temperatura media [°C]');
title('Profili verticali: confronto pre/post intervento', 'FontWeight',
'bold');
legend('Location', 'northeast');
ytickformat('%.1f')
```

CON QUESTA VERSIONE MIGLIORATA:

```
% VERSIONE MIGLIORATA CON ETICHETTATURA COMPLETA:
% Configurazione figura professionale
set(fig_vert_pre, 'Name', sprintf('Profili Verticali Pre - %s',
termografia_id), ...
                  'NumberTitle', 'off', 'Position', [100 100 800 600]);

% Plot con stile migliorato
plot(dH, V_AVG, 'r-', 'LineWidth', 2.0, 'DisplayName', 'μ pre-intervento');
hold on;
plot(dH, V_AVG+2*V_dev, 'k--', 'LineWidth', 1.5, 'DisplayName', 'μ+2σ');
plot(dH, V_AVG-2*V_dev, 'k--', 'LineWidth', 1.5, 'DisplayName', 'μ-2σ');

% Etichettatura professionale completa
xlabel('Posizione Orizzontale X [cm]', 'FontSize', 12, 'FontWeight',
'bold');
ylabel('Temperatura Media [°C]', 'FontSize', 12, 'FontWeight', 'bold');
title(sprintf('Profili Verticali Temperatura - Pre-intervento\nCampione: %s
| Data: %s', ...
      termografia_id, datestr(now, 'dd/mm/yyyy HH:MM')), ...
```

```
      'FontSize', 14, 'FontWeight', 'bold');
legend('Location', 'best', 'FontSize', 10, 'Box', 'on');
grid on; box on;
set(gca, 'FontSize', 10, 'LineWidth', 1);
ytickformat('%.1f');

% AGGIUNTA: Salvataggio automatico organizzato
if exist('cartella_progetto', 'var')
    cartella_grafici = fullfile(cartella_progetto, 'Grafici_Elaborati');
    if ~exist(cartella_grafici, 'dir'), mkdir(cartella_grafici); end

    timestamp = datestr(now, 'yyyy-mm-dd_HH-MM');
    nome_file = sprintf('profili_verticali_pre_%s_%s.fig', termografia_id,
timestamp);
    savefig(fig_vert_pre, fullfile(cartella_grafici, nome_file));

    fprintf('✅ Salvato: %s\n', nome_file);
end
```

🔧 Aggiungi Queste Variabili All'Inizio della Funzione:

SUBITO DOPO `function FunctionCompletaPre(termografia)`**:**

```
function FunctionCompletaPre(termografia, termografia_id, cartella_progetto)
    % AGGIUNTA: Parametri per etichettatura e salvataggio
    if nargin < 2, termografia_id = 'Campione_Sconosciuto'; end
    if nargin < 3, cartella_progetto = pwd; end
```

🔧 Applica lo Stesso Pattern a TUTTI i Tuoi Grafici:

Per ogni `figure` **nel tuo codice, sostituisci:**

- ✅ Titoli generici → Titoli con ID campione e timestamp
- ✅ Labels semplici → Labels con font e stile professionale
- ✅ `savefig` semplice → Salvataggio organizzato in cartelle

📝 MODIFICA 2: `Function_def_CompletaPost.m`

🔧 Identico a Modifica 1, ma cambia:

```
% Colori: rosso → verde
plot(dH, V_AVG, 'g-', 'LineWidth', 2.0, 'DisplayName', 'μ post-intervento');

% Titoli: 'Pre-intervento' → 'Post-intervento'
```

```
title(sprintf('Profili Verticali Temperatura - Post-intervento\nCampione: %s
| Data: %s', ...

% Nomi file: '_pre_' → '_post_'
nome_file = sprintf('profili_verticali_post_%s_%s.fig', termografia_id,
timestamp);
```

📝 MODIFICA 3: Nuovo File `analisi3D_Emma.m`

🆕 Crea Questo Nuovo File per i Grafici 3D della Relatrice:

```
function analisi3D_Emma(termografia_pre, termografia_post, id_campione,
cartella_progetto)
    % ANALISI3D_EMMA - Grafici 3D per forme ricorrenti (richiesta relatrice)
    %
    % Input:
    %   termografia_pre  - Matrice termografia pre-intervento
    %   termografia_post - Matrice termografia post-intervento
    %   id_campione      - Nome identificativo campione
    %   cartella_progetto - Cartella principale progetto

    fprintf('🎯 Creazione grafici 3D per identificazione forme
ricorrenti...\n');

    % Preparazione dati (usa la tua stessa logica di rotazione)
    IR_pre = rot90(termografia_pre);
    IR_post = rot90(termografia_post);

    % Crea cartella dedicata
    cartella_3D = fullfile(cartella_progetto, 'Visualizzazioni_3D');
    if ~exist(cartella_3D, 'dir'), mkdir(cartella_3D); end

    %% === GRAFICO 3D 1: SUPERFICI COMPARATIVE ===
    fig_superfici = figure('Name', sprintf('Superfici 3D - %s',
id_campione), ...
                           'Position', [100 100 1200 600]);

    % Pre-intervento
    subplot(1,2,1);
    surf(IR_pre, 'EdgeColor', 'none');
    colormap('hot'); colorbar;
    title(sprintf('Superficie 3D Pre-intervento\n%s', id_campione), ...
          'FontSize', 12, 'FontWeight', 'bold');
    xlabel('Posizione X [pixel]', 'FontWeight', 'bold');
```

```
    ylabel('Posizione Y [pixel]', 'FontWeight', 'bold');
    zlabel('Temperatura [°C]', 'FontWeight', 'bold');
    view(45, 30);

    % Post-intervento
    subplot(1,2,2);
    surf(IR_post, 'EdgeColor', 'none');
    colormap('cool'); colorbar;
    title(sprintf('Superficie 3D Post-intervento\n%s', id_campione), ...
          'FontSize', 12, 'FontWeight', 'bold');
    xlabel('Posizione X [pixel]', 'FontWeight', 'bold');
    ylabel('Posizione Y [pixel]', 'FontWeight', 'bold');
    zlabel('Temperatura [°C]', 'FontWeight', 'bold');
    view(45, 30);

    % Salva
    savefig(fig_superfici, fullfile(cartella_3D,
sprintf('superfici_3D_%s.fig', id_campione)));

    %% === GRAFICO 3D 2: TRE VARIABILI (X, Y, ΔT) ===
    fig_scatter = figure('Name', sprintf('Scatter 3D - %s', id_campione),
...
                         'Position', [200 200 800 600]);

    % Calcola variazione temperatura
    delta_temp = IR_post - IR_pre;
    [X, Y] = meshgrid(1:size(delta_temp,2), 1:size(delta_temp,1));

    % Scatter 3D con le tre variabili richieste dalla relatrice
    scatter3(X(:), Y(:), delta_temp(:), 30, delta_temp(:), 'filled');

    % Etichettatura professionale
    xlabel('Posizione X [pixel]', 'FontSize', 12, 'FontWeight', 'bold');
    ylabel('Posizione Y [pixel]', 'FontSize', 12, 'FontWeight', 'bold');
    zlabel('Variazione Temperatura [°C]', 'FontSize', 12, 'FontWeight',
'bold');
    title(sprintf('Analisi 3D Forme Ricorrenti\nVariabili: X, Y,
ΔTemperatura - %s', id_campione), ...
          'FontSize', 14, 'FontWeight', 'bold');
    colorbar; grid on; view(45, 30);

    % Salva
    savefig(fig_scatter, fullfile(cartella_3D, sprintf('scatter_3D_%s.fig',
id_campione)));

    %% === GRAFICO 3D 3: IDENTIFICAZIONE PATTERN ===
    fig_pattern = figure('Name', sprintf('Pattern Ricorrenti - %s',
id_campione), ...
                         'Position', [300 300 800 600]);
```

```
    % Analisi variabilità locale (finestra 5x5)
    finestra = 5;
    [righe, colonne] = size(delta_temp);
    mappa_pattern = zeros(righe, colonne);

    for i = finestra:righe-finestra+1
        for j = finestra:colonne-finestra+1
            area_locale = delta_temp(i-finestra+1:i+finestra-1, j-
finestra+1:j+finestra-1);
            mappa_pattern(i,j) = std(area_locale(:)); % Variabilità locale
        end
    end

    % Visualizzazione 3D pattern
    surf(mappa_pattern, 'EdgeColor', 'none');
    colormap('jet'); colorbar;
    title(sprintf('Mappa Pattern Ricorrenti\nVariabilità Locale - %s',
id_campione), ...
          'FontSize', 14, 'FontWeight', 'bold');
    xlabel('Posizione X [pixel]', 'FontWeight', 'bold');
    ylabel('Posizione Y [pixel]', 'FontWeight', 'bold');
    zlabel('Indice Variabilità', 'FontWeight', 'bold');
    view(45, 30);

    % Salva
    savefig(fig_pattern, fullfile(cartella_3D, sprintf('pattern_3D_%s.fig',
id_campione)));

    %% === ANALISI QUANTITATIVA ===
    % Calcola metriche per la tesi
    zone_cambio_significativo = abs(delta_temp) > std(delta_temp(:)) * 1.5;
    percentuale_cambio = sum(zone_cambio_significativo(:)) /
numel(zone_cambio_significativo) * 100;
    variabilita_media = mean(mappa_pattern(:));

    % Report risultati
    fprintf('📊 RISULTATI ANALISI 3D:\n');
    fprintf('   • Percentuale area cambio significativo: %.1f%%\n',
percentuale_cambio);
    fprintf('   • Variabilità media pattern: %.3f\n', variabilita_media);
    fprintf('   • Range variazione temperatura: %.2f - %.2f °C\n',
min(delta_temp(:)), max(delta_temp(:)));
    fprintf('✅ Grafici 3D salvati in: %s\n', cartella_3D);

    % Salva risultati numerici
    risultati_3D.percentuale_cambio = percentuale_cambio;
    risultati_3D.variabilita_media = variabilita_media;
    risultati_3D.range_variazione = [min(delta_temp(:)),
max(delta_temp(:))];
```

```
    save(fullfile(cartella_3D, sprintf('risultati_3D_%s.mat', id_campione)),
'risultati_3D');
end
```

📝 MODIFICA 4: Aggiorna il Tuo `main.m`

🔧 Sostituisci il Tuo main.m con Questa Versione:

```
%% MAIN_ENHANCED.m - Versione potenziata del tuo main esistente
% Emma - Tesi Magistrale Umidità di Risalita

clear; clc; close all;

%% === CONFIGURAZIONE PROGETTO ===
nome_progetto = 'TesiMagistraleUmidita';
id_campione = 'T04_EdificioStorico';  % Cambia per ogni edificio

% Crea struttura cartelle organizzata
cartella_principale = fullfile(pwd, sprintf('%s_%s', nome_progetto,
datestr(now, 'yyyy-mm-dd')));
if ~exist(cartella_principale, 'dir'), mkdir(cartella_principale); end

% Sottocartelle per organizzazione
sottocartelle = {'Grafici_Elaborati', 'Visualizzazioni_3D',
'Dati_Statistici', 'Report'};
for i = 1:length(sottocartelle)
    cartella_sub = fullfile(cartella_principale, sottocartelle{i});
    if ~exist(cartella_sub, 'dir'), mkdir(cartella_sub); end
end

fprintf('🎯 TESI MAGISTRALE - ANALISI TERMOGRAFICA AUTOMATIZZATA\n');
fprintf('📁 Progetto: %s\n', cartella_principale);
fprintf('📊 Campione: %s\n', id_campione);

%% === CARICAMENTO DATI (il tuo codice esistente) ===
load('T04_1129.mat');
load('T04_2_1129.mat');

mat_pre = T04_1129;
mat_post = T04_2_1129;

fprintf('✅ Dati caricati: Pre %dx%d, Post %dx%d\n', size(mat_pre),
size(mat_post));

%% === ANALISI PRE-INTERVENTO (tua funzione potenziata) ===
fprintf('\n🔍 Analisi pre-intervento in corso...\n');
```

```
FunctionCompletaPre(mat_pre, id_campione, cartella_principale);

%% === ANALISI POST-INTERVENTO (tua funzione potenziata) ===
fprintf('🔍 Analisi post-intervento in corso...\n');
FunctionCompletaPost(mat_post, id_campione, cartella_principale);

%% === GRAFICI 3D FORME RICORRENTI (NUOVO - richiesta relatrice) ===
fprintf('🎯 Creazione grafici 3D per forme ricorrenti...\n');
analisi3D_Emma(mat_pre, mat_post, id_campione, cartella_principale);

%% === UNIONE GRAFICI (il tuo codice esistente) ===
fprintf('🔗 Unione grafici pre/post...\n');
cartella_grafici = fullfile(cartella_principale, 'Grafici_Elaborati');

% Trova i tuoi grafici salvati
file_pre = dir(fullfile(cartella_grafici, '*_pre_*.fig'));
file_post = dir(fullfile(cartella_grafici, '*_post_*.fig'));

% Applica la tua funzione esistente ai file trovati
for i = 1:min(length(file_pre), length(file_post))
    def_UnioneGrafici(fullfile(cartella_grafici, file_pre(i).name), ...
                      fullfile(cartella_grafici, file_post(i).name));
end

%% === REPORT FINALE ===
fprintf('📄 Generazione report finale...\n');
creaReportTesi(id_campione, cartella_principale);

%% === RIEPILOGO FINALE ===
fprintf('\n🎉 ANALISI COMPLETATA CON SUCCESSO!\n');
fprintf('📁 Tutti i risultati in: %s\n', cartella_principale);
fprintf('📊 File generati:\n');

for i = 1:length(sottocartelle)
    cartella_sub = fullfile(cartella_principale, sottocartelle{i});
    file_generati = dir(fullfile(cartella_sub, '*.*'));
    file_utili = file_generati(~[file_generati.isdir]);
    fprintf('   - %s: %d file\n', sottocartelle{i}, length(file_utili));
end

fprintf('\n✅ Obiettivi raggiunti:\n');
fprintf('   ✅ Etichettatura automatica completa\n');
fprintf('   ✅ Salvataggio organizzato in cartelle\n');
fprintf('   ✅ Grafici 3D forme ricorrenti\n');
fprintf('   ✅ Automazione processo completa\n');
```

📝 MODIFICA 5: Funzione Report per Tesi

🆕 Aggiungi Questo File `creaReportTesi.m`:

```
function creaReportTesi(id_campione, cartella_progetto)
    % CREAREPORTTESI - Genera report finale per tesi magistrale

    cartella_report = fullfile(cartella_progetto, 'Report');
    nome_file = sprintf('Report_Tesi_%s_%s.txt', id_campione, datestr(now,
'yyyy-mm-dd_HH-MM'));
    percorso_completo = fullfile(cartella_report, nome_file);

    fid = fopen(percorso_completo, 'w');

    fprintf(fid,
'============================================================\n');
    fprintf(fid, '    REPORT ANALISI TERMOGRAFICA - TESI MAGISTRALE\n');
    fprintf(fid, '         Umidità di Risalita negli Edifici Storici\n');
    fprintf(fid,
'============================================================\n\n');

    fprintf(fid, 'CAMPIONE ANALIZZATO: %s\n', id_campione);
    fprintf(fid, 'DATA ELABORAZIONE: %s\n', datestr(now, 'dd/mm/yyyy
HH:MM'));
    fprintf(fid, 'METODOLOGIA: Protocollo automatizzato con grafici
3D\n\n');

    fprintf(fid, 'OBIETTIVI RAGGIUNTI:\n');
    fprintf(fid, '✅ Etichettatura automatica completa dei grafici\n');
    fprintf(fid, '✅ Salvataggio organizzato con nomenclatura
sistematica\n');
    fprintf(fid, '✅ Visualizzazioni 3D per identificazione forme
ricorrenti\n');
    fprintf(fid, '✅ Automazione integrale del processo di analisi\n\n');

    fprintf(fid, 'INNOVAZIONI METODOLOGICHE:\n');
    fprintf(fid, '• Analisi tridimensionale pattern spaziali\n');
    fprintf(fid, '• Identificazione automatica zone cambio
significativo\n');
    fprintf(fid, '• Quantificazione variabilità locale per forme
ricorrenti\n');
    fprintf(fid, '• Strutturazione output per riproducibilità
scientifica\n\n');

    fprintf(fid, 'FILE GENERATI:\n');
    sottocartelle = {'Grafici_Elaborati', 'Visualizzazioni_3D',
'Dati_Statistici'};
    for i = 1:length(sottocartelle)
```

```
        cartella_sub = fullfile(cartella_progetto, sottocartelle{i});
        file_generati = dir(fullfile(cartella_sub, '*.*'));
        file_utili = file_generati(~[file_generati.isdir]);
        fprintf(fid, '• %s: %d file\n', sottocartelle{i},
length(file_utili));
    end

    fprintf(fid, '\nNOTE PER TESI:\n');
    fprintf(fid, '• I grafici generati sono pronti per inclusione
documento\n');
    fprintf(fid, '• Le visualizzazioni 3D rappresentano innovazione
metodologica\n');
    fprintf(fid, '• La struttura automatizzata garantisce
riproducibilità\n');
    fprintf(fid, '• I risultati quantitativi supportano validazione
scientifica\n\n');

    fprintf(fid, 'SVILUPPI FUTURI:\n');
    fprintf(fid, '• Estensione database edifici storici\n');
    fprintf(fid, '• Implementazione machine learning per pattern
recognition\n');
    fprintf(fid, '• Integrazione sensori IoT per monitoraggio
continuo\n\n');

    fclose(fid);

    fprintf('📄 Report tesi generato: %s\n', nome_file);
end
```