

# Gradient Descent for Performing Image-to-Image Change Detection

---

PATRICK COWHILL

ECE6560 26 APR 2022

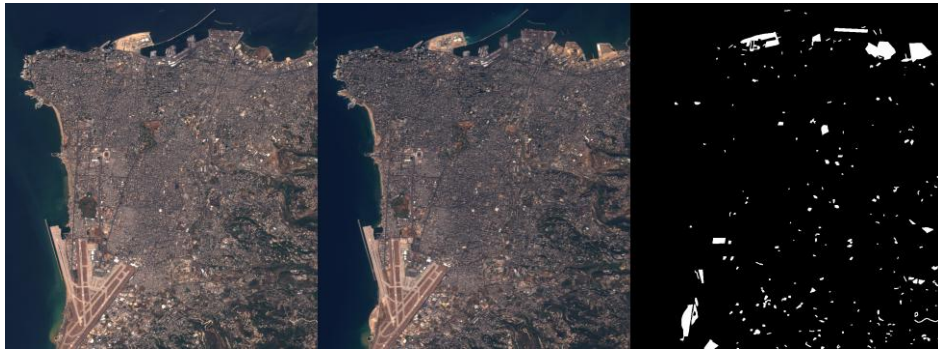
FINAL PROJECT PRESENTATION

# 1 The Problem

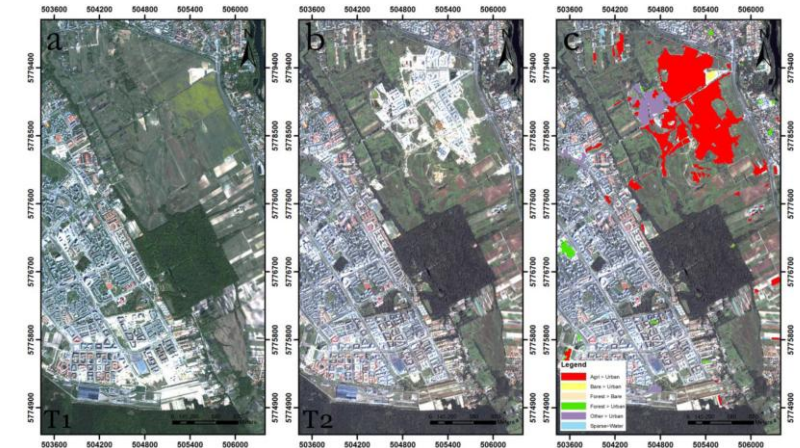
---

# Image-to-Image Change Detection

This is the process of **identifying differences** in the state or contents of a scene captured in images at different times. Automating change detection in computer vision can be an **extremely useful** technology.



Number of Frame	Original Frame	Mask Foreground Display of Learning Rate Variation			
		0.01	0.05	0.0033	0.0025
1					
117					
137					
700					



There are **different modern approaches** for designing an image processing solution to implement this functionality into a computer vision system. Many include more **standard computer science methods** of image processing such as calculating cross correlation, entropy and contour mapping, using histograms of pixel intensity and oriented gradients. Others leverage **machine learning** and neural networks, and many of these contain gradient descent or some similar energy-inspired metric for classifying changes within images.



# Solving with a Gradient Descent PDE

Using two aligned images of a scene taken at different times, the later of the two images will **slowly morph** to become identical to the earlier of the two images. During this process, the goal is to make portions of the second image that are already **similar** to the first image **morph quickly**, early in the processing while portions of the second image that are quite **different** from the first image **shift slowly**. If this goal is met, then in between the start of the processing and the end (when the two images are identical), there should be a state of the morphing image where all incidental differences between the images have been resolved and only **significant differences remain**. These differences identify changes that have occurred within the scene.

The combination of measurements and statistics generated from the two images make up the **weighting scheme** for determining how much to adjust each portion of the later image and in what direction. Pixel variations between the two images caused by **image noise** and differences in **brightness and contrast** are weighted in such a way that they dissipate quickly while portions of the images containing **patterns and edges** that are not present the same way in both images should be opposingly weighted to persist in the later image longer to allow for more miscellaneous anomalies to be removed.

# 2 The Energy Functional

---

To mathematically define a weighting scheme that exhibits the **variable-speed image adjustments** described in the previous section, the following energy functional was defined. It is characterized by three adjustable **parameter coefficients**  $a$ ,  $b$ , and  $c$ , and both the constant earlier image  $H$  and the time-evolving later image  $I$ :

$$E = \int_0^1 \int_0^1 \frac{a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2}{1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right]} dy dx$$

This functional is composed of three main components.

Scaling Coefficient	Functional Structure	Purpose and Intent
$a$	$(I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2$	A simple way to capture <b>differential noise</b> between the two images and resolve it before they are classified as change.
$b$	$(I - H)^2$	The <b>driving term</b> to morph $I$ into an identical copy of $H$ .
$c$	$(I_x - H_x)^2 + (I_y - H_y)^2$	A dampening term to limit the effect of the other two components by reducing the energy consumption of portions of the image that contain different <b>patterns and edges</b> .

# 3 Deriving the PDE

---

To draw out a function that explicitly and precisely describes the adjustments made to later image  $I$  as it evolves over time, a particular form of the single-function, multivariate, second-order derivative form of the **Euler-Lagrange equation** is derived. For now, the integrand is **generalized** to some function; note that  $H$ ,  $a$ ,  $b$ , and  $c$  are constant, and are therefore not included as input arguments during this derivation.

$$E(I) = \iint \mathcal{L}(I, I_x, I_y, I_{xx}, I_{yy}, x, y) dy dx$$

$$E(I) = \iint \mathcal{L}(z_1, z_2, z_3, z_4, z_5, x, y) dy dx$$

The first five variable **inputs** of  $\mathcal{L}$  can be either referenced by the values used from the function passed to the energy functional  $(I, I_x, I_y, I_{xx}, I_{yy})$  or explicitly by index  $(z_1, z_2, z_3, z_4, z_5)$ .

Find how the energy functional changes as some **alteration** is applied to the later image:

$$\nabla E = \frac{\partial E}{\partial K} \Big|_I = \lim_{\varepsilon \rightarrow 0} \frac{E(I + \varepsilon K) - E(I)}{\varepsilon} \text{ for some "direction" } K \text{ where}$$

$$K(x, 0) = K(0, y) = K_x(x, 0) = K_x(0, y) = K_y(x, 0) = K_y(0, y) = 0$$



The derivative of  $E$  with respect to  $K$  is identical to finding the **infinitesimal change** in  $E$  when adjusting  $I$  :

$$\nabla E = \frac{d}{d\varepsilon} E(I + \varepsilon K) = 0$$

Using the **general form** of the energy functional to calculate  $\nabla E$  in terms of  $\mathcal{L}$  :

$$E(I) = \iint \mathcal{L}(I, I_x, I_y, I_{xx}, I_{yy}, x, y) dy dx$$

$$E(I + \varepsilon K) = \iint \mathcal{L}(I + \varepsilon K, I_x + \varepsilon K_x, I_y + \varepsilon K_y, I_{xx} + \varepsilon K_{xx}, I_{yy} + \varepsilon K_{yy}, x, y) dy dx$$

$$\frac{d}{d\varepsilon} E(I + \varepsilon K) = \iint \left( \frac{\partial \mathcal{L}}{\partial z_1} \frac{d(I + \varepsilon K)}{d\varepsilon} + \frac{\partial \mathcal{L}}{\partial z_2} \frac{d(I_x + \varepsilon K_x)}{d\varepsilon} + \frac{\partial \mathcal{L}}{\partial z_3} \frac{d(I_y + \varepsilon K_y)}{d\varepsilon} + \frac{\partial \mathcal{L}}{\partial z_4} \frac{d(I_{xx} + \varepsilon K_{xx})}{d\varepsilon} + \frac{\partial \mathcal{L}}{\partial z_5} \frac{d(I_{yy} + \varepsilon K_{yy})}{d\varepsilon} \right) dy dx$$

$$\frac{d}{d\varepsilon} E(I + \varepsilon K) = \iint \left( \frac{\partial \mathcal{L}}{\partial z_1} K + \frac{\partial \mathcal{L}}{\partial z_2} K_x + \frac{\partial \mathcal{L}}{\partial z_3} K_y + \frac{\partial \mathcal{L}}{\partial z_4} K_{xx} + \frac{\partial \mathcal{L}}{\partial z_5} K_{yy} \right) dy dx$$

Each of the terms in the integrand can be **reduced** using the boundary conditions.

$$\iint \frac{\partial \mathcal{L}}{\partial z_2} K_x dy dx = \int \frac{\partial \mathcal{L}}{\partial z_2} K \Big|_{x=0}^{x=1} dy - \iint \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_2} K dy dx = - \iint \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_2} K dy dx$$

$$\iint \frac{\partial \mathcal{L}}{\partial z_3} K_y dy dx = \int \frac{\partial \mathcal{L}}{\partial z_3} K \Big|_{x=0}^{x=1} dx - \iint \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_3} K dy dx = - \iint \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_3} K dy dx$$

$$\begin{aligned} \iint \frac{\partial \mathcal{L}}{\partial z_4} K_{xx} dy dx &= \int \frac{\partial \mathcal{L}}{\partial z_4} K_x \Big|_{x=0}^{x=1} dy - \iint \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_4} K_x dy dx = 0 - \left( \int \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_4} K \Big|_{x=0}^{x=1} dy - \iint \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial z_4} K dy dx \right) \\ \iint \frac{\partial \mathcal{L}}{\partial z_4} K_{xx} dy dx &= \iint \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial z_4} K dy dx \end{aligned}$$

$$\begin{aligned} \iint \frac{\partial \mathcal{L}}{\partial z_5} K_{yy} dy dx &= \int \frac{\partial \mathcal{L}}{\partial z_5} K_y \Big|_{y=0}^{y=1} dx - \iint \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_5} K_y dy dx = 0 - \left( \int \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_5} K \Big|_{x=0}^{x=1} dx - \iint \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial z_5} K dy dx \right) \\ \iint \frac{\partial \mathcal{L}}{\partial z_5} K_{yy} dy dx &= \iint \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial z_5} K dy dx \end{aligned}$$

Substituting the terms back into the equation for the gradient of the energy yields:

$$\begin{aligned}\nabla E = 0 &= \iint \left( \frac{\partial \mathcal{L}}{\partial z_1} K + \frac{\partial \mathcal{L}}{\partial z_2} K_x + \frac{\partial \mathcal{L}}{\partial z_3} K_y + \frac{\partial \mathcal{L}}{\partial z_4} K_{xx} + \frac{\partial \mathcal{L}}{\partial z_5} K_{yy} \right) dy dx \\ 0 &= \iint \left( \frac{\partial \mathcal{L}}{\partial z_1} K - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_2} K - \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_3} K + \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial z_4} K + \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial z_5} K \right) dy dx \\ 0 &= \iint \left( \frac{\partial \mathcal{L}}{\partial z_1} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_2} - \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_3} + \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial z_4} + \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial z_5} \right) dy dx\end{aligned}$$

Following from this is the Euler-Lagrange equation that will be leveraged to produce the PDE for this project.

$$\frac{\partial \mathcal{L}}{\partial z_1} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_2} - \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_3} + \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial z_4} + \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial z_5} = 0$$

$$\frac{\partial \mathcal{L}}{\partial z_1} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_2} - \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_3} + \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial z_4} + \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial z_5} = 0$$

Applying [gradient descent](#) to this equation to receive the rate of change of the later image follows with:

$$\frac{\partial I}{\partial t} = I_t = -\nabla E = -\frac{\partial \mathcal{L}}{\partial I} + \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial I_x} + \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial I_y} - \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial I_{xx}} - \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial I_{yy}}$$

Each of these terms can be calculated using the [previously engineered function](#):

$$\mathcal{L}(I, I_x, I_y, I_{xx}, I_{yy}, x, y) = \frac{a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2}{1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right]}$$

$$\mathcal{L}(I, I_x, I_y, I_{xx}, I_{yy}, x, y) = \frac{a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2}{1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right]}$$

$$\frac{\partial \mathcal{L}}{\partial I} = \frac{2b(I - H)}{1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right]}$$

$$\frac{\partial \mathcal{L}}{\partial I_x} = \frac{-2c(I_x - H_x) \left[ a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2 \right]}{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^2}$$

$$\frac{\partial \mathcal{L}}{\partial I_{xx}} = \frac{2a(I_{xx} - H_{xx})}{1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right]}$$

Each of these terms are also true if all  $\mathbf{x}$ s and  $\mathbf{y}$ s are switched

$$\frac{\partial \mathcal{L}}{\partial I_x} = \frac{-2c(I_x - H_x) \left[ a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2 \right]}{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^2}$$

$$\frac{d}{dx} \frac{\partial \mathcal{L}}{\partial I_x} = -2c \frac{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^2 \left[ (I_x - H_x) \left[ a \left[ 2(I_{xx} - H_{xx})(I_{xxx} - H_{xxx}) + 2(I_{yy} - H_{yy})(I_{yyx} - H_{yyx}) \right] + 2b(I - H)(I_x - H_x) \right] + \left[ a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2 \right] (I_{xx} - H_{xx}) \right] - \left[ (I_x - H_x) \left[ a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2 \right] 2 \left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] \right]}{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^4}$$

$$\frac{d}{dx} \frac{\partial \mathcal{L}}{\partial I_x} = -2c \frac{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^2 \left[ 2(I_x - H_x) \left[ a \left[ (I_{xx} - H_{xx})(I_{xxx} - H_{xxx}) + (I_{yy} - H_{yy})(I_{yyx} - H_{yyx}) \right] + b(I - H)(I_x - H_x) \right] + \left[ a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2 \right] (I_{xx} - H_{xx}) \right] - \left[ \begin{aligned} & (I_x - H_x) \left[ a \left[ (I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2 \right] + b(I - H)^2 \right] \\ & 2 \left[ \left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] \right] \end{aligned} \right]}{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^4}$$

Each of these terms are also true if all  $\mathbf{x}$ s and  $\mathbf{y}$ s are switched



$$\frac{\partial \mathcal{L}}{\partial I_{xx}} = \frac{2a(I_{xx} - H_{xx})}{1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right]}$$

$$\frac{d}{dx} \frac{\partial \mathcal{L}}{\partial I_{xx}} = 2a \frac{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] (I_{xxx} - H_{xxx}) - (I_{xx} - H_{xx}) 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right]}{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^2}$$

$$\frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial I_{xx}} = 2a \frac{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^2 \left[ \left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] (I_{xxxx} - H_{xxxx}) + (I_{xxx} - H_{xxx}) 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] - \left[ (I_{xx} - H_{xx}) 2c \left[ (I_x - H_x)(I_{xxx} - H_{xxx}) + (I_{xx} - H_{xx})(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yxx} - H_{yxx}) + (I_{yx} - H_{yx})(I_{yx} - H_{yx}) \right] + 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] (I_{xxx} - H_{xxx}) \right] - \left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] (I_{xxx} - H_{xxx}) - (I_{xx} - H_{xx}) 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] \right] 2 \left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] \right]}{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^4}$$

$$\begin{aligned} \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial I_{xx}} = 2a \frac{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^2 \left[ \begin{aligned} &\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] (I_{xxxx} - H_{xxxx}) + (I_{xxx} - H_{xxx}) 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] \\ &- \left[ (I_{xx} - H_{xx}) 2c \left[ (I_x - H_x)(I_{xxx} - H_{xxx}) + (I_{xx} - H_{xx})(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yxx} - H_{yxx}) + (I_{yx} - H_{yx})(I_{yx} - H_{yx}) \right] \right. \\ &\quad \left. + 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] (I_{xxx} - H_{xxx}) \right] \\ &- \left[ \left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] (I_{xxx} - H_{xxx}) - (I_{xx} - H_{xx}) 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right] \right] \end{aligned} \right]}{2 \left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right] 2c \left[ (I_x - H_x)(I_{xx} - H_{xx}) + (I_y - H_y)(I_{yx} - H_{yx}) \right]} \right]}{\left[ 1 + c \left[ (I_x - H_x)^2 + (I_y - H_y)^2 \right] \right]^4} \end{aligned}$$

Each of these terms are also true if all  $\mathbf{x}$ s and  $\mathbf{y}$ s are switched

# 4 Discretization

---

Now, the **partial derivatives** of  $I$  describing  $I_t$  are:

$$I_x, I_y, I_{xx}, I_{yy}, I_{xy}, I_{yx}, I_{xxx}, I_{yyy}, I_{xyy}, I_{yxx}, I_{xxy}, I_{yyx}, I_{xxxx}, I_{yyyy}, I_{xxyy}, I_{yyxx}$$

and

$$H_x, H_y, H_{xx}, H_{yy}, H_{xy}, H_{yx}, H_{xxx}, H_{yyy}, H_{xyy}, H_{yxx}, H_{xxy}, H_{yyx}, H_{xxxx}, H_{yyyy}, H_{xxyy}, H_{yyxx}$$

Since the partial derivatives of  $x$  and  $y$  **commute**, this can be reduced to:

$$I_x, I_y, I_{xx}, I_{yy}, I_{xy}, I_{xxx}, I_{yyy}, I_{xyy}, I_{xxy}, I_{xxxx}, I_{yyyy}, I_{xxyy}$$

and

$$H_x, H_y, H_{xx}, H_{yy}, H_{xy}, H_{xxx}, H_{yyy}, H_{xyy}, H_{xxy}, H_{xxxx}, H_{yyyy}, H_{xxyy}$$

To start with, all discretizations of derivatives will use **central difference**. Prior to experimentation, there is no convincing reason to **bias** either direction and no reason to increase or decrease the **precision**. Down the line, it could be found that adjustments to these would provide **improvements**, and they can be made and tested.

Central Difference:

$$A_u = \frac{\partial A}{\partial u} = \frac{A(u + \Delta u) - A(u - \Delta u)}{2\Delta u}$$

Note, derivatives of order  $> 1$  can simply **iterate** on this and timesteps may be reduced:

$$A_{uv}(u, v) = \frac{\partial A_u}{\partial v} = \frac{A_u(u, v + \Delta v) - A_u(u, v - \Delta v)}{2\Delta v}$$

$$A_{uv}(u, v) = \frac{A(u + \Delta u, v + \Delta v) - A(u - \Delta u, v + \Delta v) - A(u + \Delta u, v - \Delta v) + A(u - \Delta u, v - \Delta v)}{4\Delta v \Delta u}$$

$$A_{uu}(u) = \frac{A_u(u + \Delta u) - A_u(u - \Delta u)}{2\Delta u} = \frac{A(u + 2\Delta u) - 2A(u) + A(u - 2\Delta u)}{4(\Delta u)^2} = \frac{A(u + \Delta u) - 2A(u) + A(u - \Delta u)}{(\Delta u)^2}$$

Replacing  $I$  with  $H$  in each of these equations produces the **discretized central differences** for the partial derivatives of  $H$  as well.

$$I_x(x, y) = \frac{I(x + \Delta x, y) - I(x - \Delta x, y)}{2\Delta x}$$

$$I_y(x, y) = \frac{I(x, y + \Delta y) - I(x, y - \Delta y)}{2\Delta y}$$

$$I_{xx}(x, y) = \frac{I(x + \Delta x, y) - 2I(x, y) + I(x - \Delta x, y)}{(\Delta x)^2}$$

$$I_{yy}(x, y) = \frac{I(x, y + \Delta y) - 2I(x, y) + I(x, y - \Delta y)}{(\Delta y)^2}$$

$$I_{xy}(x, y) = \frac{I(x + \Delta x, y + \Delta y) - I(x - \Delta x, y + \Delta y) - I(x + \Delta x, y - \Delta y) + I(x - \Delta x, y - \Delta y)}{4\Delta x\Delta y}$$

$$I_{xxx}(x, y) = \frac{I(x + 3\Delta x, y) - 3I(x + \Delta x, y) + 3I(x - \Delta x, y) - I(x - 3\Delta x, y)}{8(\Delta x)^3}$$

$$I_{yyy}(x, y) = \frac{I(x, y + 3\Delta y) - 3I(x, y + \Delta y) + 3I(x, y - \Delta y) - I(x, y - 3\Delta y)}{8(\Delta y)^3}$$

$$I_{xyy}(x, y) = \frac{I(x + \Delta x, y + \Delta y) - 2I(x + \Delta x, y) + I(x + \Delta x, y - \Delta y) - I(x - \Delta x, y + \Delta y) + 2I(x - \Delta x, y) - I(x - \Delta x, y - \Delta y)}{2\Delta x(\Delta y)^2}$$

$$I_{xxy}(x, y) = \frac{I(x + \Delta x, y + \Delta y) - I(x + \Delta x, y - \Delta y) - 2I(x, y + \Delta y) + 2I(x, y - \Delta y) + I(x - \Delta x, y + \Delta y) - I(x - \Delta x, y - \Delta y)}{2(\Delta x)^2\Delta y}$$

$$I_{xxxx}(x, y) = \frac{I(x + 2\Delta x, y) - 4I(x + \Delta x, y) + 6I(x, y) - 4I(x - \Delta x, y) + I(x - 2\Delta x, y)}{(\Delta x)^4}$$

$$I_{yyyy}(x, y) = \frac{I(x, y + 2\Delta y) - 4I(x, y + \Delta y) + 6I(x, y) - 4I(x, y - \Delta y) + I(x, y - 2\Delta y)}{(\Delta y)^4}$$

$$I_{xxyy}(x, y) = \frac{I(x + \Delta x, y + \Delta y) - 2I(x + \Delta x, y) + I(x + \Delta x, y - \Delta y) - 2I(x, y + \Delta y) + 4I(x, y) - 2I(x, y - \Delta y) + I(x - \Delta x, y + \Delta y) - 2I(x - \Delta x, y) + I(x - \Delta x, y - \Delta y)}{(\Delta x)^2(\Delta y)^2}$$



Let  $\tilde{I}$  be the **time evolving image**. This is different from  $I$  which is a variable parameter input for the energy functional.  $\tilde{I}(0, x, y)$  is the initial image. The **update function** for this image utilizes the PDE:

$$\tilde{I}(t + \Delta t, x, y) = \tilde{I}(t, x, y) + I_t(\tilde{I}(t), x, y)\Delta t$$

Since realistic images have a finite pixel width, finite pixel height, and finite bit depth, some **conditions** must be set to handle cases in which the discretization above attempts to reach beyond these boundaries. For example, a central difference scheme may require a pixel outside of the image or an updated image is supposed to have a negative pixel value when only non-negative integer values are stored. The following **rules** cover these cases:

1. Queried positions that lie **outside** the image will be replaced with the location of the closest pixel to the queried position. For example, for a 100x100 pixel resolution image, the 1-indexed point (64,102) would use the pixel value of (64,100), and the point (103,-2) would use the pixel value of (100,1).
2. When updating the time evolving image, non-integral values will be **quantized** to the nearest unsigned integer, and values outside the bounds of the bit depth will be replaced with the closer of the minimum or maximum of the bit depth. For example, in an 8-bit single-band image with pixel values falling between 0 and 255 inclusive, attempting to update to a pixel value of 27.472 would instead be set to 27, and attempting to update to a pixel value of 283.903 would instead be set to 255.

# Recap

The goal to highlight changes between two images,  $H$  (before) and  $I$  (after), is attempted by using the **time-evolving** image  $\tilde{I}$  defined by starting at  $\tilde{I}(0, x, y) = I_0(x, y)$  and following:

$$\tilde{I}(t + \Delta t, x, y) = \tilde{I}(t, x, y) + I_t(\tilde{I}(t), x, y)\Delta t$$

The change of the image  $\tilde{I}$  follows the following single-function, multivariate, second-order derivative form of the **Euler-Lagrange** equation:

$$I_t(I, x, y) = -\nabla E = -\frac{\partial \mathcal{L}}{\partial I} + \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial I_x} + \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial I_y} - \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial I_{xx}} - \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial I_{yy}}$$

The changes over time seek to minimize the following parameterized **energy functional** which was designed to maintain changes in the second image while eliminating non-change related anomalies:

$$E(I) = \iint \mathcal{L} dy dx \quad \mathcal{L}(I, I_x, I_y, I_{xx}, I_{yy}, x, y) = \frac{a[(I_{xx}-H_{xx})^2 + (I_{yy}-H_{yy})^2] + b(I-H)^2}{1+c[(I_x-H_x)^2 + (I_y-H_y)^2]}$$

Order 1 to 4 partial derivatives of  $H$  and  $I$  present in the update equation of the image  $\tilde{I}$  are all approximated using the **central difference** scheme on the pixel values of the full images.

# Code Snippets

The code for this project was written in [MATLAB](#) as it is very intuitive to read for the non-programmer mathematician and it is quite easy and straight forward to translate these large equations into functions.

The image to the right shows the major functional [components](#) of the program, many of which match nearly 1-to-1 with one of the equations defined earlier in this presentation. Other functions are used to set up variables and aid in visualization of results.

A copy of this code in its entirety can also be sent in addition to this PowerPoint if desired.

```
function runFinalProjectDefault() ...

function defaultH = returnDefaultH() ...
function defaultI = returnDefaultI() ...
function diffImage = returnAbsDiffImage(a,b) ...
function diffImage = returnRelDiffImage(a,b) ...

function createPlot(inImage,id) ...
function createGIF(imageCellArray,filename) ...

function calculateDerivatives(inImage) ...
function calculatePDx(I,x,y,dx,~) ...
function calculatePDy(I,x,y,~,dy) ...
function calculatePDxf(I,x,y,dx,~) ...
function calculatePDyf(I,x,y,~,dy) ...
function calculatePDxx(I,x,y,dx,~) ...
function calculatePDyy(I,x,y,~,dy) ...
function calculatePDxy(I,x,y,dx,dy) ...
function calculatePDxxx(I,x,y,dx,~) ...
function calculatePDyyy(I,x,y,~,dy) ...
function calculatePDxxy(I,x,y,dx,dy) ...
function calculatePDxyx(I,x,y,dx,dy) ...
function calculatePDxxxx(I,x,y,dx,~) ...
function calculatePDyyyy(I,x,y,~,dy) ...
function calculatePDxxyy(I,x,y,dx,dy) ...

function result = evolveImage(H,I,params) ...

function It = calculateGradientDescent(I,H,a,b,c) ...

function dL_dI = calculate_dL_dI(I,H,~,b,c) ...
function d_dx_dL_dIx = calculate_d_dx_dL_dIx(I,H,a,b,c) ...
function d_dy_dL_dIy = calculate_d_dy_dL_dIy(I,H,a,b,c) ...
function d2_dx2_dL_dIxx = calculate_d2_dx2_dL_dIxx(I,H,a,~,c) ...
function d2_dy2_dL_dIyy = calculate_d2_dy2_dL_dIyy(I,H,a,~,c) ...
```

$$\tilde{I}(t + \Delta t, x, y) = \tilde{I}(t, x, y) + I_t(\tilde{I}(t), x, y)\Delta t$$

```
function result = evolveImage(H,I,params)
a = params.a;
b = params.b;
c = params.c;
Itilde = I.copy();
result = cell(params.numSteps+1,0);
result{1} = Itilde.copy();
for i = 1:params.numSteps
    Itilde.o = uint8(Itilde.o + params.timeStep.*(calculateGradientDescent(Itilde,H,a,b,c)));
    calculateDerivatives(Itilde);
    result{i+1} = Itilde.copy();
end
end
```

$$I_t(I, x, y) = -\frac{\partial \mathcal{L}}{\partial I} + \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial I_x} + \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial I_y} - \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial I_{xx}} - \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial I_{yy}}$$

```
function It = calculateGradientDescent(I,H,a,b,c)
% NOTE: It is _NOT_ an image, it is a matrix, so it does not have an .o ...
dL_dI = calculate_dL_dI(I,H,a,b,c);
d_dx_dL_dIx = calculate_d_dx_dL_dIx(I,H,a,b,c);
d_dy_dL_dIy = calculate_d_dy_dL_dIy(I,H,a,b,c);
d2_dx2_dL_dIxx = calculate_d2_dx2_dL_dIxx(I,H,a,b,c);
d2_dy2_dL_dIyy = calculate_d2_dy2_dL_dIyy(I,H,a,b,c);

It = - dL_dI + d_dx_dL_dIx + d_dy_dL_dIy - d2_dx2_dL_dIxx - d2_dy2_dL_dIyy;
end
```

$$I_{xxyy}(x, y)$$

```
function calculatePDxxyy(I,x,y,dx,dy)
I.xxyy = ( ...
    I(x+dx,y+dy) - 2.*I(x+dx,y) + I(x+dx,y-dy) ...
    - 2.*I(x,y+dy) + 4.*I(x,y) - 2.*I(x,y-dy) ...
    + I(x-dx,y+dy) - 2.*I(x-dx,y) + I(x-dx,y-dy) ...
    ) ./ (dx*dx*dy*dy);
end
```

$$\frac{\partial \mathcal{L}}{\partial I}$$

```
function dL_dI = calculate_dL_dI(I,H,~,b,c)
dL_dI = (2*b*(I-H)) ...
    ./ (1 + c*((I.xf-H.xf).^2 + (I.yf-H.yf).^2));
end
```

$$I_{xxy}(x, y)$$

```
function calculatePDxxy(I,x,y,dx,dy)
I.xxy = ( ...
    I(x+dx,y+dy) - I(x+dx,y-dy) - 2.*I(x,y+dy) ...
    + 2.*I(x,y-dy) + I(x-dx,y+dy) - I(x-dx,y-dy) ...
    ) ./ (2*dx*dx*dy);
end
```

$$\frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial I_{yy}}$$

```
function d2_dy2_dL_dIyy = calculate_d2_dy2_dL_dIyy(I,H,a,~,c)
d2_dy2_dL_dIyy = ...
    2.*a.* ( ...
        (...
            (1 + c.*((I.y-H.y).^2+(I.x-H.x).^2)).^2 .* ( ((1+c.*((I.y-H.y).^2+(I.x-H.x).^2)).*(I.yyyy-H.yyyy)+(I.yyy-H.yyy).*2.*c.*((I.y-H.y).*(I.yy-H.yy)+(I.x-H.x).*(I.xy-H.xy))) ...
                - ( (I.yy-H.yy).*2.*c.*((I.y-H.y).*(I.yyy-H.yyy)+(I.yy-H.yy).*(I.yy-H.yy)+(I.x-H.x).*(I.xyy-H.xyy)+(I.xy-H.xy).*(I.xy-H.xy)) ...
                + 2.*c.*((I.y-H.y).*(I.yy-H.yy)+(I.x-H.x).*(I.xy-H.xy)).*(I.yyy-H.yyy) ) ) ...
            - ( ((1+c.*((I.y-H.y).^2+(I.x-H.x).^2)).*(I.yyy-H.yyy)-(I.yy-H.yy).*2.*c.*((I.y-H.y).*(I.yy-H.yy)+(I.x-H.x).*(I.xy-H.xy))) ...
                .*2.*(1+c.*((I.y-H.y).^2+(I.x-H.x).^2).*2.*c.*((I.y-H.y).*(I.yy-H.yy)+(I.x-H.x).*(I.xy-H.xy))) ) ...
        ) ./ ( ...
            (1+c.*((I.yf-H.yf).^2+(I.xf-H.xf).^2)).^4 ...
        ) ...
    );
end
```

# 5 Experimental Results

---



# The Example Images

The following images below were used to **test** out the effectiveness of the PDE for change detection. They are identical apart from the bottom right dime which is synthetically **removed** from the second image. While the removal is not seamless, and there are slight effects left behind, the PDE was specifically designed to be able to ignore these small differences when compared to the more significant change in an image.

*H* (Before)



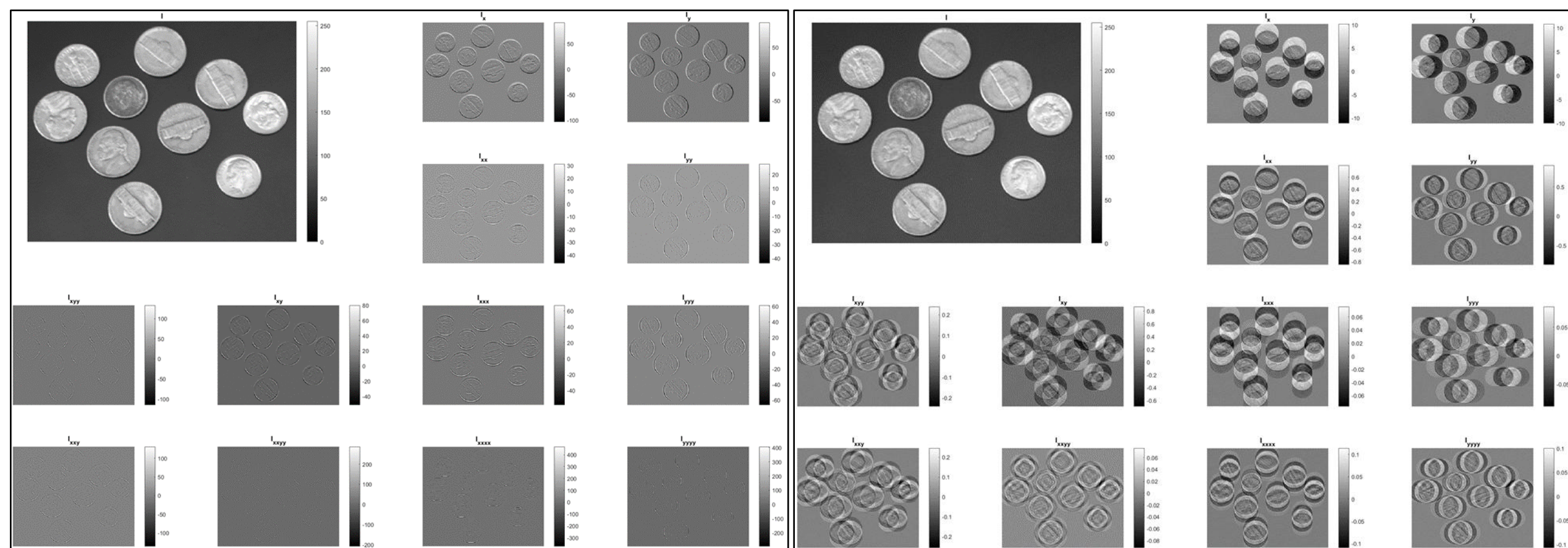
*I* (After)



Using identical images instead of real data provides the **control** to add synthetic white noise and alter image contrast and brightness in order to target certain capabilities when testing.

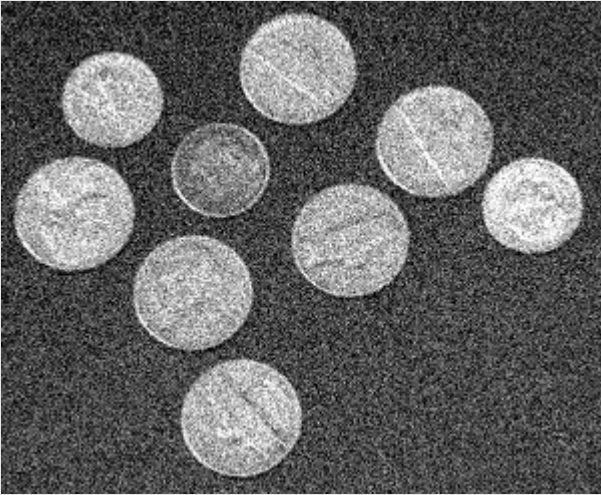
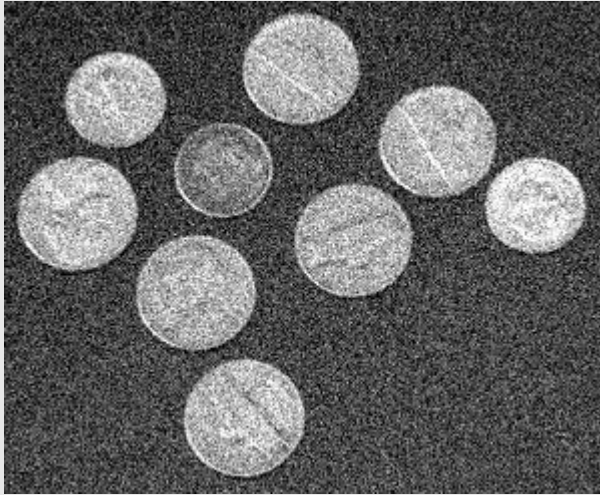
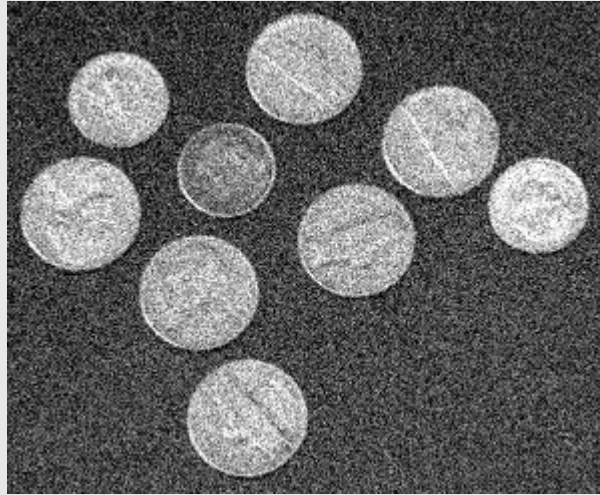
# Partial Derivative Discretizations

Plots for each of the **discretized derivatives** were made from the original image to confirm the proper implementation of those discretizations. The first image uses a step size of a single pixel and the second uses a **larger step size** to better see the higher-order derivatives. Here,  $x$  runs down while  $y$  runs to the right.



# Testing Each Energy Component

First, each of the three coefficients ( $a$ ,  $b$ , and  $c$ ) were set to positive values in **isolation** to confirm that each of the **intended behaviors** was at least approximately being exhibited. To properly test, significant white noise was added to the initial second image, and the first image stayed the same.

$a$	$b$	$c$
To <b>blur/denoise</b> the image.	To remove image <b>differences</b>	To scale changes by <b>pattern</b> difference
		
The image loses its sharpness, although some “large” noise remains. This is discussed in the final section.	The Missing coin reenters the scene and some of the higher-intensity noise recedes.	Nothing. This is expected as if both $a$ and $b$ are zero, so too is the energy functional, and no change occurs.

# The Time Step Without a CTF Condition

When considering the extensive task of algebraically performing [Von Neumann analysis](#) on this PDE and calculating the CTF condition, the alternative solution of [visually inspecting](#) the images for divergent behavior and adjusting the timestep to account for them appeared significantly more reasonable. The following is an example of the time-evolved image that has a timestep set [too low](#):



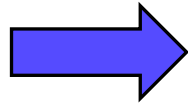
# Example Case with No Adjustments

Performing discretized gradient descent on the two images as is  
(without adding noise or adjusting brightness and contrast)  
produces the **desired result** where the missing dime is present in the  
difference image, and slowly disappears as the time evolved image  
becomes more similar to the first image.

$H$  (Before)



$I$  (After)



$\tilde{I}(t)$



$\tilde{I}(t) - H$





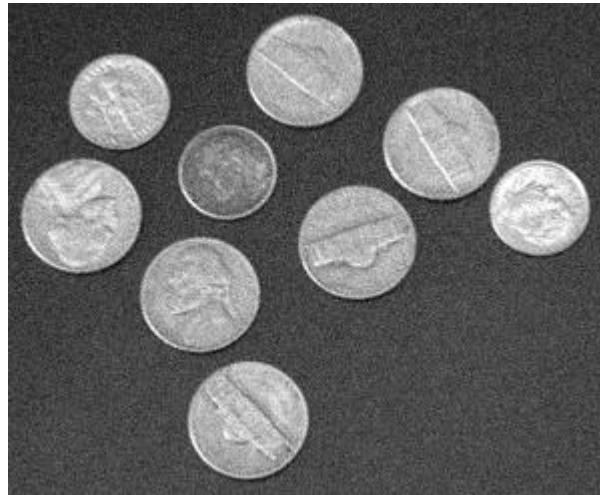
# Example Case with Added Noise

After adding noise to the image, the result is fairly **similar** to the last. The dime appears in the time-evolved image, and the noise is more **smoothed** to match the smooth background of the original image.

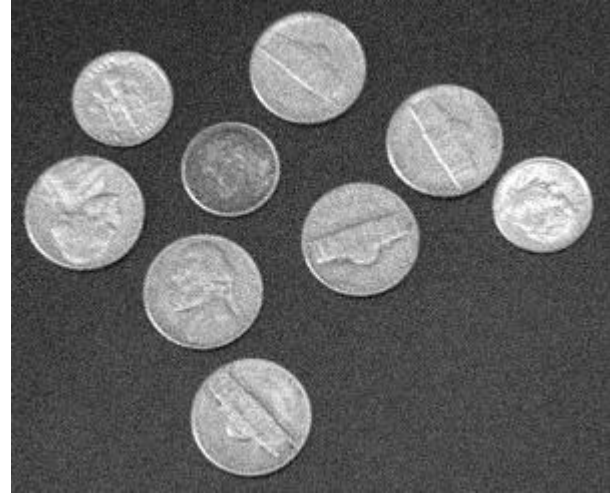
$H$  (Before)



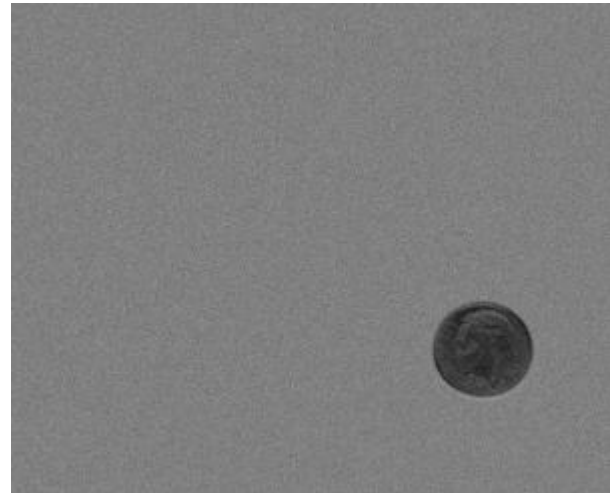
$I$  (After)



$\tilde{I}(t)$



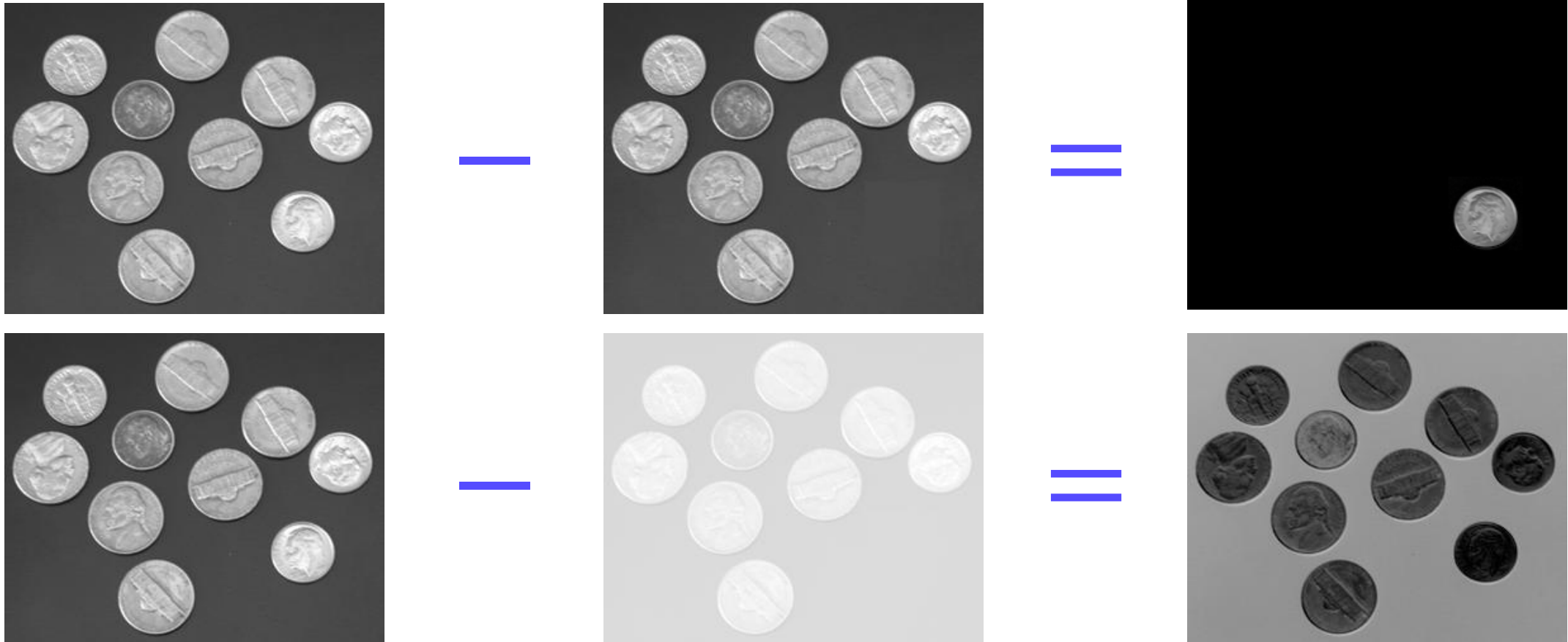
$\tilde{I}(t) - H$





# Addressing Simple Image Difference

One of the goals of this project is to produce a result that is **better** than simply performing an image difference and thresholding the result. While this can be a good first step, for real data, this often is **not sufficient** for detecting real changes that occur within a scene as shown in the second example below where the dime is not lit.



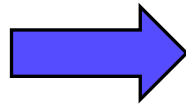
# Example Case with Noise and Contrast

While simply taking the difference between the two images fails to distinguish the change of the dime in comparison to the other coins, the time-evolved PDE does a **good job** at persisting this change even though the difference in intensity in the area of the dime is significantly less than the difference everywhere else in the image.

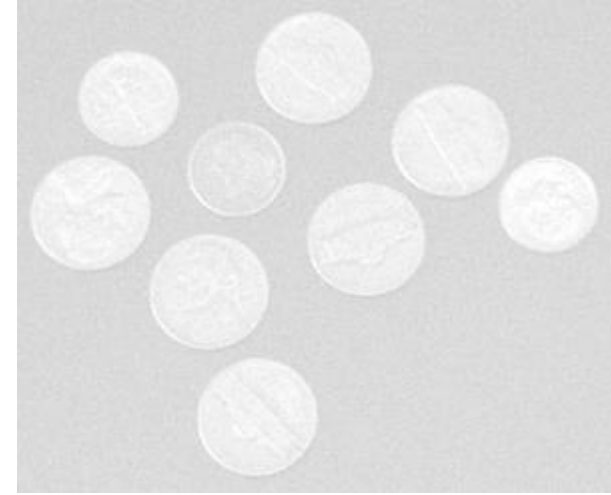
$H$  (Before)



$I$  (After)



$\tilde{I}(t)$



$\tilde{I}(t) - H$



# 6 Conclusion

---

# The Strengths of This PDE

The PDE for this project **exceled** in several areas compared to other change-detection approaches:

- This PDE outperformed the **absolute difference** for a high brightness and low contrast image.
- This PDE can perform **local** image analysis instead of a global image analysis such as generating intensity and oriented gradient histograms for adjusting the entire image.
- It did not simply take the easy route of being **difference focused**.
- The **scaling of the edges** appeared to work well for persisting the important change within the scene.

# Changes Made

These are changes that **were made** to the original model in response to testing results:

- The values for  **$a$** ,  **$b$** , and  **$c$**  were **tweaked** until they appeared to be appropriately balanced.
- The timestep was **nudged** lower and lower whenever divergent behavior appeared.
- The first-order derivatives in the denominator were switched from central difference to **forward difference**.

These are changes that **should be explored** if this project would be repeated in the future:

- Have the image difference portion of the energy functional be **less than linear** (logarithmic or inverse quadratic).
- Utilize a **different denoising algorithm** that produces better results for eliminating white noise.
- Explore having the image intensity **not quantized** and instead be represented by floats or doubles.
- Explore the effects of having all derivatives multiplied by  **$c$**  switched to **forward difference**.
- Increase the complexity of the **pattern detection component** in the energy functional.

# Further Exploration

Here are some ideas for how gradient descent for change detection can be explored in the future:

- Use real data:
  - VIRAT (<https://viratdata.org/>) has an excellent collection of video footage closely resembling security camera footage



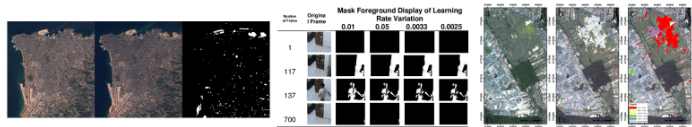
- Add more components to the energy functional to capture different phenomena in images.
- Explore other related problems such as image registration/alignment or depth estimation.
- Explore hooking up this technology to a real data stream. Can it process data in real-time? Is it efficient?



# Summary

## Image-to-Image Change Detection

This is the process of identifying differences in the state or contents of a scene captured in images at different times. Automating change detection in computer vision can be an extremely useful technology.



There are different modern approaches for designing an image processing solution to implement this functionality into a computer vision system. Many include more standard computer science methods of image processing such as calculating cross correlation, entropy and contour mapping, using histograms of pixel intensity and oriented gradients. Others leverage machine learning and neural networks, and many of these contain gradient descent or some similar energy-inspired metric for classifying changes within images.

Images: <https://cloud.githubusercontent.com/assets/12345678/12345678/12345678.png> | [https://www.researchgate.net/publication/312345678/Motion-Detection-Test-Results-on-Indian-Condoms\\_061\\_12345678](https://www.researchgate.net/publication/312345678/Motion-Detection-Test-Results-on-Indian-Condoms_061_12345678) | <http://www.mdpi.com/2072-4292/6/7/5976/htm>

To mathematically define a weighting scheme that exhibits the variable-speed image adjustments described in the previous section, the following energy functional was defined. It is characterized by three adjustable parameter coefficients  $a$ ,  $b$ , and  $c$ , and both the constant earlier image  $H$  and the time-evolving later image  $I$ :

$$E = \int_0^1 \int_0^1 a \frac{[(I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2] + b(I - H)^2}{1 + c[(I_x - H_x)^2 + (I_y - H_y)^2]} dy dx$$

This functional is composed of three main components.

Scaling Coefficient	Functional Structure	Purpose and Intent
$a$	$(I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2$	A simple way to capture differential noise between the two images and resolve it before they are classified as change.
$b$	$(I - H)^2$	The driving term to morph $I$ into an identical copy of $H$ .
$c$	$(I_x - H_x)^2 + (I_y - H_y)^2$	A dampening term to limit the effect of the other two components by reducing the energy consumption of portions of the image that contain different patterns and edges.

$$\frac{\partial \mathcal{L}}{\partial z_1} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial z_2} - \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial z_3} + \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial z_4} + \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial z_5} = 0$$

Applying gradient descent to this equation to receive the rate of change of the later image follows with:

$$\frac{\partial I}{\partial t} = I_t = -\nabla E = -\frac{\partial \mathcal{L}}{\partial I} + \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial I_x} + \frac{d}{dy} \frac{\partial \mathcal{L}}{\partial I_y} - \frac{d^2}{dx^2} \frac{\partial \mathcal{L}}{\partial I_{xx}} - \frac{d^2}{dy^2} \frac{\partial \mathcal{L}}{\partial I_{yy}}$$

Each of these terms can be calculated using the previously engineered function:

$$\mathcal{L}(I, I_x, I_y, I_{xx}, I_{yy}, x, y) = \frac{a[(I_{xx} - H_{xx})^2 + (I_{yy} - H_{yy})^2] + b(I - H)^2}{1 + c[(I_x - H_x)^2 + (I_y - H_y)^2]}$$

## Code Snippets

The code for this project was written in MATLAB as it is very intuitive to read for the non-programmer mathematician and it is quite easy and straight forward to translate these large equations into functions.

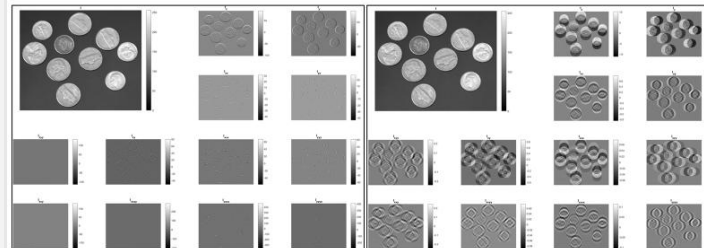
The image to the right shows the major functional components of the program, many of which match nearly 1-to-1 with one of the equations defined earlier in this presentation. Other functions are used to set up variables and aid in visualization of results.

A copy of this code in its entirety can also be sent in addition to this PowerPoint if desired.

```
function runFinalProjectDefault()
function defaultN = returnDefaultN()
function defaultI1 = returnDefaultI1()
function diffImage = returnABDiffImage(a,b)
function diffImage = returnABDiffImage(a,b)
function createPlot(inImage,d)
function createDiff([imageCellArray,filename])
function calculateDerivatives(inImage)
function calculatePxx(I,x,y,dx)
function calculatePyy(I,x,y,dy)
function calculatePxx(I,x,y,dx)
function calculatePyy(I,x,y,dy)
function calculatePxx(I,x,y,dx)
function calculatePyy(I,x,y,dy)
function calculatePxx(I,x,y,dx)
function calculatePyy(I,x,y,dy)
function calculatePxx(I,x,y,dx)
function calculatePyy(I,x,y,dy)
function calculatePxx(I,x,y,dx)
function calculatePyy(I,x,y,dy)
function calculatePxx(I,x,y,dx)
function calculatePyy(I,x,y,dy)
function result = evolveImage(I,params)
function It = calculateGradientDescent(I,H,a,b,c)
function dL_dI = calculate_dL_dI(I,H,a,b,c)
function dL_dIx_dIy = calculate_dL_dIx_dIy(I,H,a,b,c)
function dL_dIxx_dIyy = calculate_dL_dIxx_dIyy(I,H,a,b,c)
function dL_dIxx_dIyy = calculate_dL_dIxx_dIyy(I,H,a,b,c)
```

## Partial Derivative Discretizations

Plots for each of the discretized derivatives were made from the original image to confirm the proper implementation of those discretizations. The first image uses a step size of a single pixel and the second uses a larger step size to better see the higher-order derivatives. Here,  $x$  runs down while  $y$  runs to the right.



## Example Case with Noise and Contrast

While simply taking the difference between the two images fails to distinguish the change of the dime in comparison to the other coins, the time-evolved PDE does a good job at persisting this change even though the difference in intensity in the area the dime is located is significantly less than the difference everywhere else in the image.

