Projet final

Soutenance finale présentée par ATMANI Hicham et KHERCHOUCH Kamal

Safran & AJC Formation - Juillet 2023

Déroulé de la présentation

01
Sujet choisi

Sujet 1 : Détéction de pts d'intérêt
Sujet 2 : Pyramide de résolutions

02
Filtres

Présentation succincte :
- *filtre de Gauss*
- *filtre de Sobel*

03
Plan de Validation

Explication du plan de validation proposé pour le sujet choisi

04
Résultats

Présentation des résultats de simulation et d'implémentation de la solution

05
Retour xp

Retour d'expérience du dernier projet et de la formation

06
Question(s) - *10min*

À vous de jouer …

01

Sujet Choisi

Détection de points d'intérêt ou pyramide de résolutions

Sujet 1 : Détection de points d'intérêt
VS.
Sujet 2 : Pyramide de résolutions

Sujet 1

- *Application du filtre de Sobel*
- *Détection des points en H*
- *Détection des points en V*
- *Application d'un seuil de détection*

→ *détection des points de croisement entre H et V*

Sujet 2

- *Application du filtre Gauss*
- *Redimensionnement de l'image en 320x240*
- *Réitération du principe pour obtenir une image format 160x120*

→ *Obtention d'une image nette avec un format réduit*

Sujet 1 : Détection de points d'intérêt
VS.
Sujet 2 : Pyramide de résolutions

Sujet 1

+ *Application d'un filtre de Sobel*
+ *Compréhension d'un nouveau filtre*
+ *Calcul de gradient*

Sujet 2

- *Filtre de Gauss déjà implémenté dans le projet VGA*

02

Filtres Sobel / Gauss

Explication du filtre de Sobel et de Gauss

Filtre de Gauss

Filtre de Gauss est un filtre d'atténuation
Il permet d'uniformiser les images
(Côté flouté / harmonisation de l'image)

Exemple d'application : redimensionnement de l'image

Avant

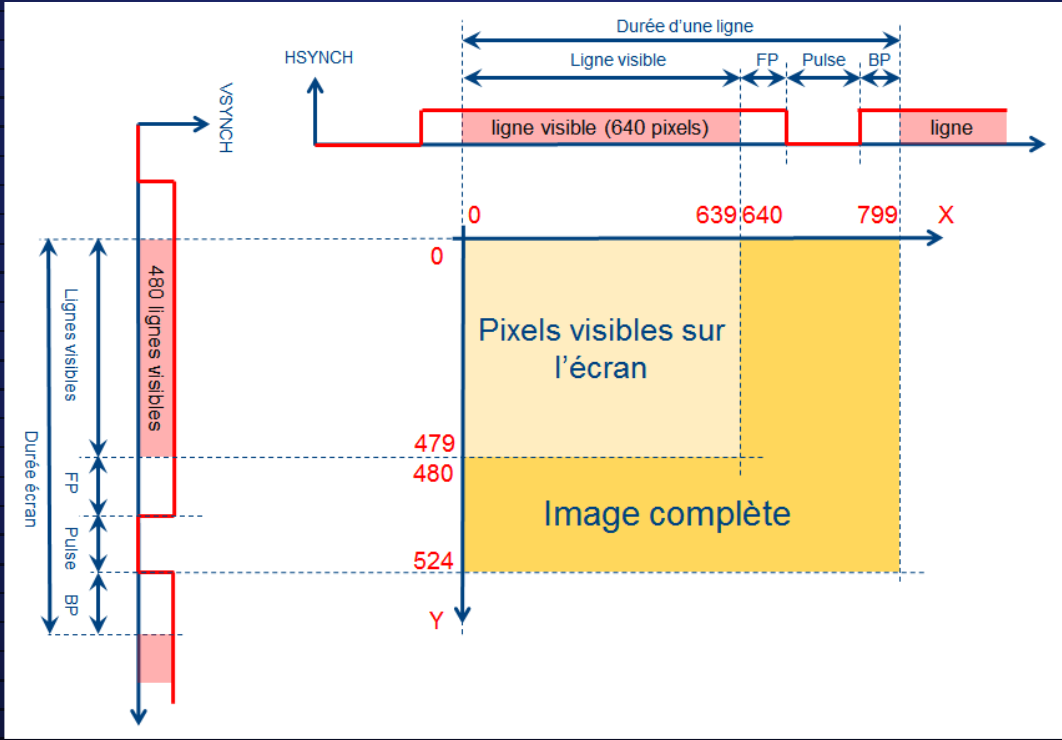
Après

Kernel / Coeur de Convolution

1	2	1
2	4	2
1	2	1

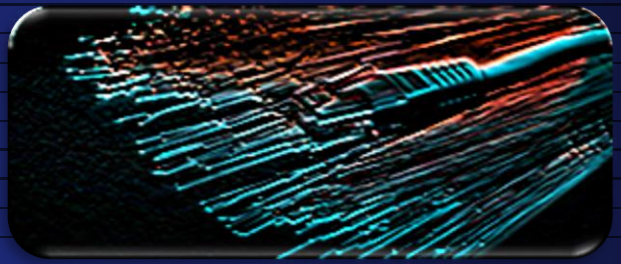
Filtre de Gauss

Application du filtre de Gauss à la norme VGA



Filtre de Sobel

Filtre de traitement d'image
Il permet de faire de la détection de point d'intéret en H et V
Filtre sensible au bruit donc utilisé avec d'autres techniques de traitement d'image
Exemple d'application : détection de contour



Avant

Après

Kernel / Coeur de Convolution en H et V

-1	0	1
-2	0	2
-1	0	1

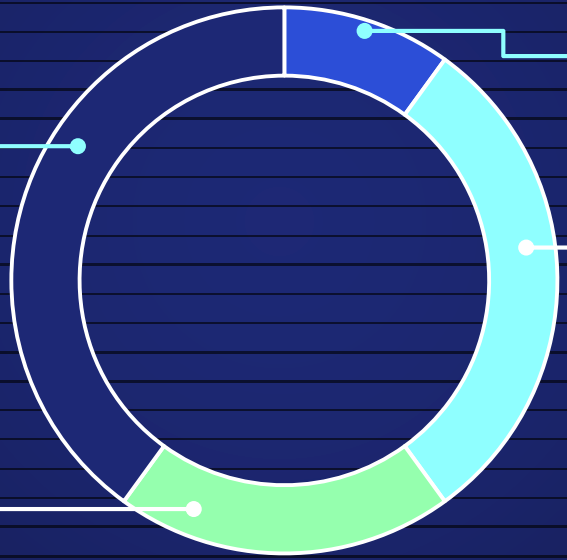
1	2	1
0	0	0
-1	-2	-1

03

Plan de Validation

Plan de validation de la détection de points d'intérêt

Plan de Validation



40%

4 - Validation du filtre de Sobel

10%

1 - Validation d'un passe plat

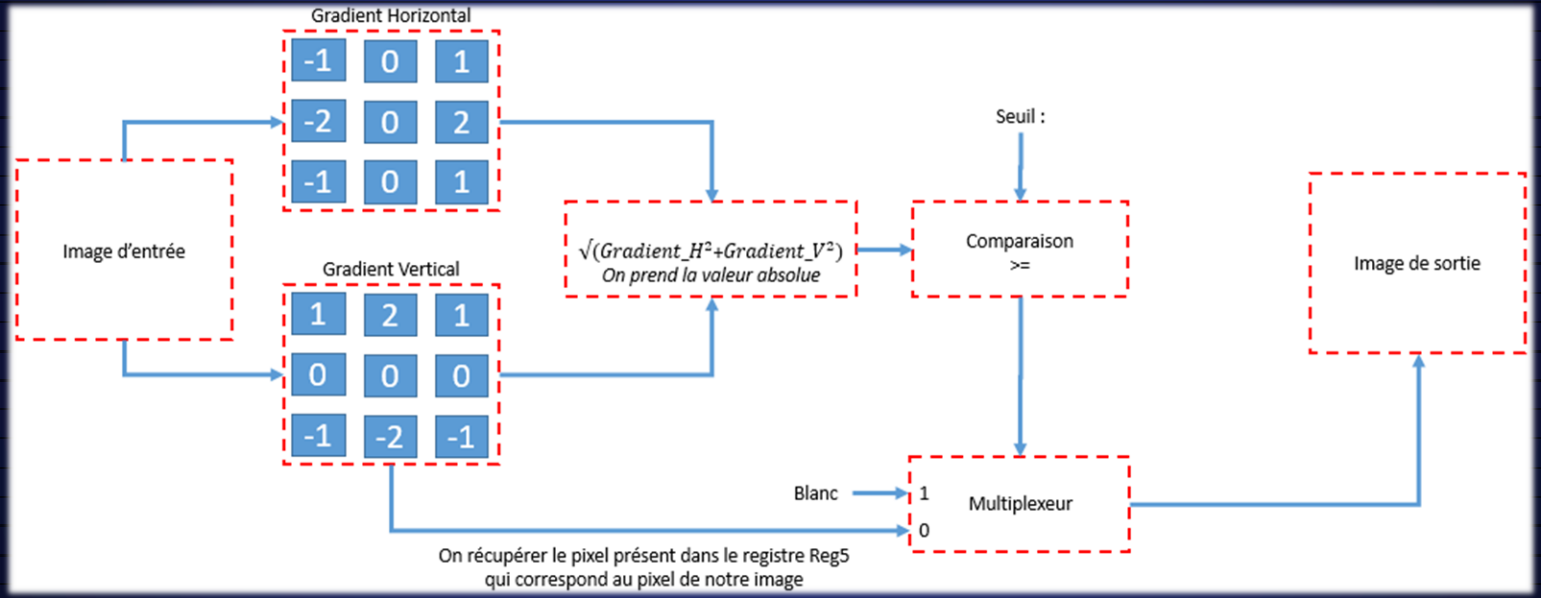
20%

3 - Validation du filtre de Gauss

30%

2 - Validation du sliding window

Plan de Validation



04

Résultat Solution

Résultat de la solution développée pour le sujet 1

Image d'entrée




1	21	22	22	22	23	23	23	23
2	18	19	19	19	19	20	20	20
3	17	17	17	17	17	18	18	18
4	18	18	18	19	19	19	19	19
5	20	20	20	20	20	20	20	20
6	20	20	20	19	19	19	19	19
7	19	19	19	19	18	18	18	18
8	20	19	19	19	18	18	18	18
9	19	19	19	19	19	19	19	19
10	18	18	18	18	18	18	18	18
11	19	19	19	19	19	19	19	19
12	20	20	20	20	20	20	20	20

Image originelle

On a sélectionné deux images au format 680x400

- *New-York (représentation de bulding)*
- *Son Goku (représentation d'un personnage)*

Conversion 8 bits

Conversion en 8 bits à l'aide du logiciel ImageJ
L'image de sortie sera en noir et blanc au même format que l'image d'entrée

Conversion .txt

Conversion de l'image au format texte pour pouvoir réaliser la lecture des pixels sous Vivado

Passe plat

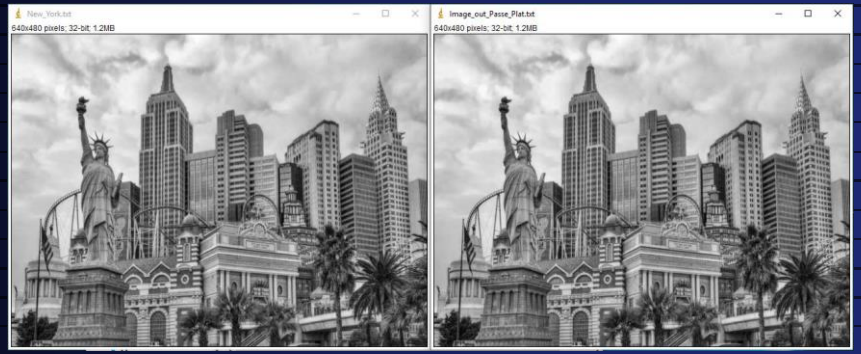
Synoptique

Code VHDL

```
process(clk, resetn)
begin
    if(resetn='1') then
        output_data <= (others => '0');
        output_data_valid <= '0';
    elsif(rising_edge(clk)) then
        if(input_data_valid = '1') then
            output_data <= input_data;
            output_data_valid <= '1';
        end if;
    end if;
end process;
```

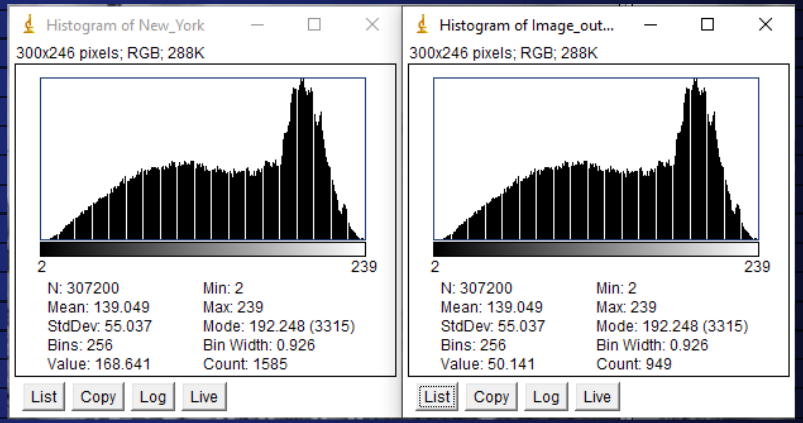
Passe plat

Simulation



Entrée Sortie

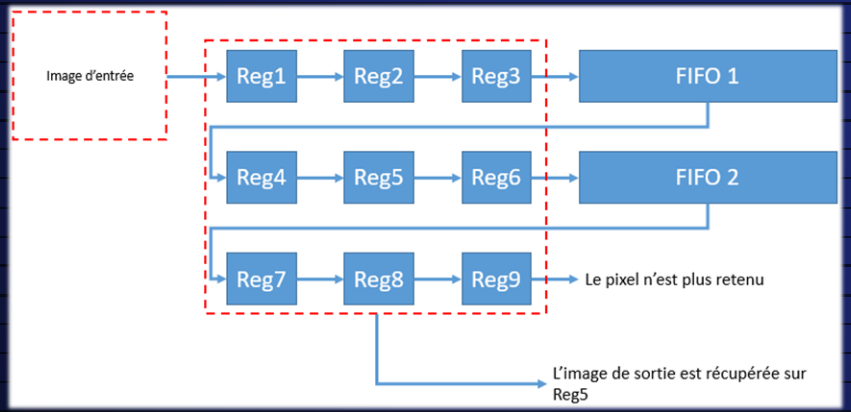
Histogramme



Entrée Sortie

Sliding Window

Synoptique



Code VHDL

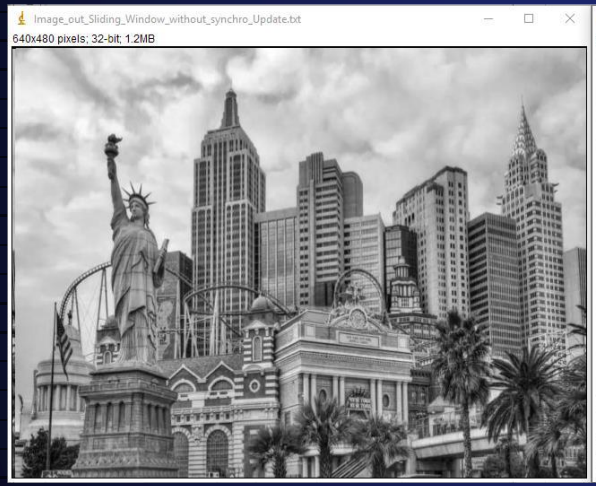
```
if (Compteur_projet = 3) then
    activation_ecriture_fifo_1 <= '1';
elsif (Compteur_projet = 640) then
    activation_lecture_fifo_1 <= '1';
elsif (Compteur_projet = 643) then
    activation_ecriture_fifo_2 <= '1';
elsif (Compteur_projet = 1280 ) then
    activation_lecture_fifo_2 <= '1';
end if;
```

```
reg1 <= input_data;
reg2 <= reg1;
reg3 <= reg2;
reg4 <= Out_Registre_Fifo1;
reg5 <= reg4;
reg6 <= reg5;
reg7 <= Out_Registre_Fifo2;
reg8 <= reg7;
reg9 <= reg8;

output_data <= reg5;
output_data_valid <= '1';
```

Sliding Window

Simulation sans / avec SW



Sortie sans SW

Sortie avec SW

MàJ signal de sortie

```
if(Compteur_projet > 643) then
    output_data <= reg5;
    output_data_valid <= '1';
else
    output_data_valid <= '0';
end if;
```

Filtre Gauss

Synoptique

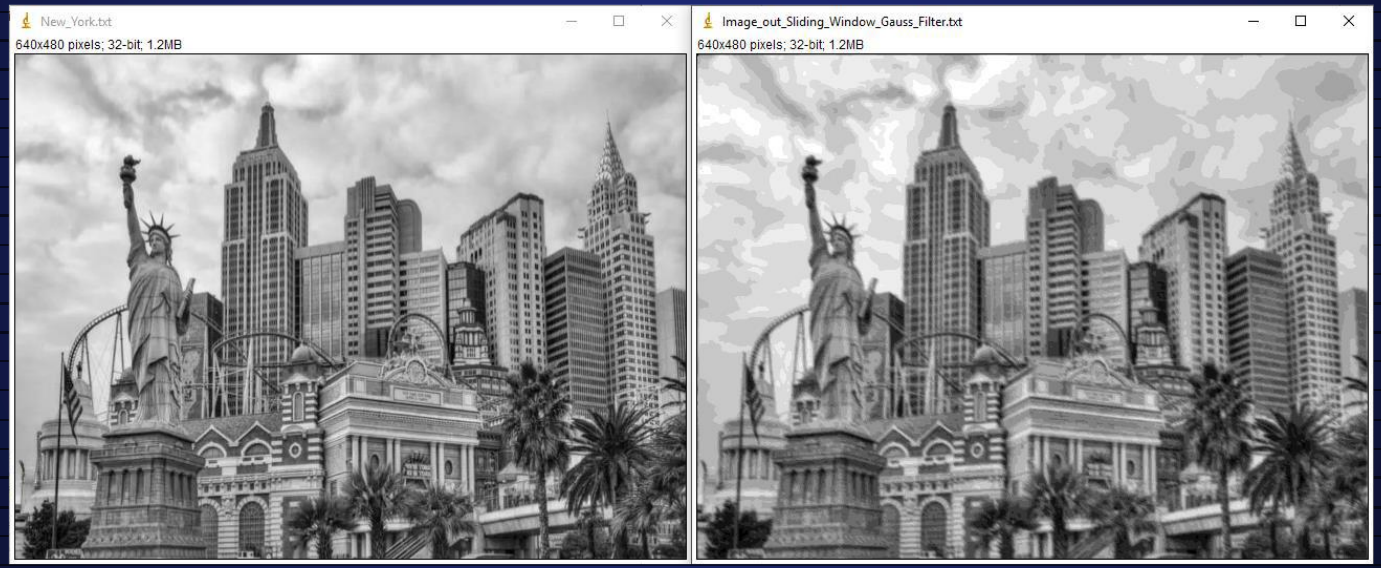

```
constant K1_Gauss : std_logic_vector(7 downto 0) := "00000001";
constant K2_Gauss : std_logic_vector(7 downto 0) := "00000010";
constant K3_Gauss : std_logic_vector(7 downto 0) := "00000001";
constant K4_Gauss : std_logic_vector(7 downto 0) := "00000010";
constant K5_Gauss : std_logic_vector(7 downto 0) := "00000100";
constant K6_Gauss : std_logic_vector(7 downto 0) := "00000010";
constant K7_Gauss : std_logic_vector(7 downto 0) := "00000001";
constant K8_Gauss : std_logic_vector(7 downto 0) := "00000010";
constant K9_Gauss : std_logic_vector(7 downto 0) := "00000001";
```

Code VHDL

```
if(Compteur_projet > 643) then
    Calcule_Gauss <= (reg1*K1_Gauss) + (reg2*K2_Gauss) + (reg3*K3_Gauss) + (reg4*K4_Gauss) + (reg5*K5_Gauss) + (reg6*K6_Gauss) + (reg7*K7_Gauss) + (reg8*K8_Gauss) + (reg9*K9_Gauss);
    output_data <= Calcule_Gauss(15 downto 8);
    output_data_valid <= '1';
else
    output_data_valid <= '0';
end if;
```

filtre Gauss

Simulation entrée / Sortie



Filtre Sobel

Synoptique


```
constant K1_H_Nat : integer := -1;
constant K2_H_Nat : integer := 0;
constant K3_H_Nat : integer := 1;
constant K4_H_Nat : integer := -2;
constant K5_H_Nat : integer := 0;
constant K6_H_Nat : integer := 2;
constant K7_H_Nat : integer := -1;
constant K8_H_Nat : integer := 0;
constant K9_H_Nat : integer := 1;
```

```
constant K1_V_Nat : integer := 1;
constant K2_V_Nat : integer := 2;
constant K3_V_Nat : integer := 1;
constant K4_V_Nat : integer := 0;
constant K5_V_Nat : integer := 0;
constant K6_V_Nat : integer := 0;
constant K7_V_Nat : integer := -1;
constant K8_V_Nat : integer := -2;
constant K9_V_Nat : integer := -1;
```

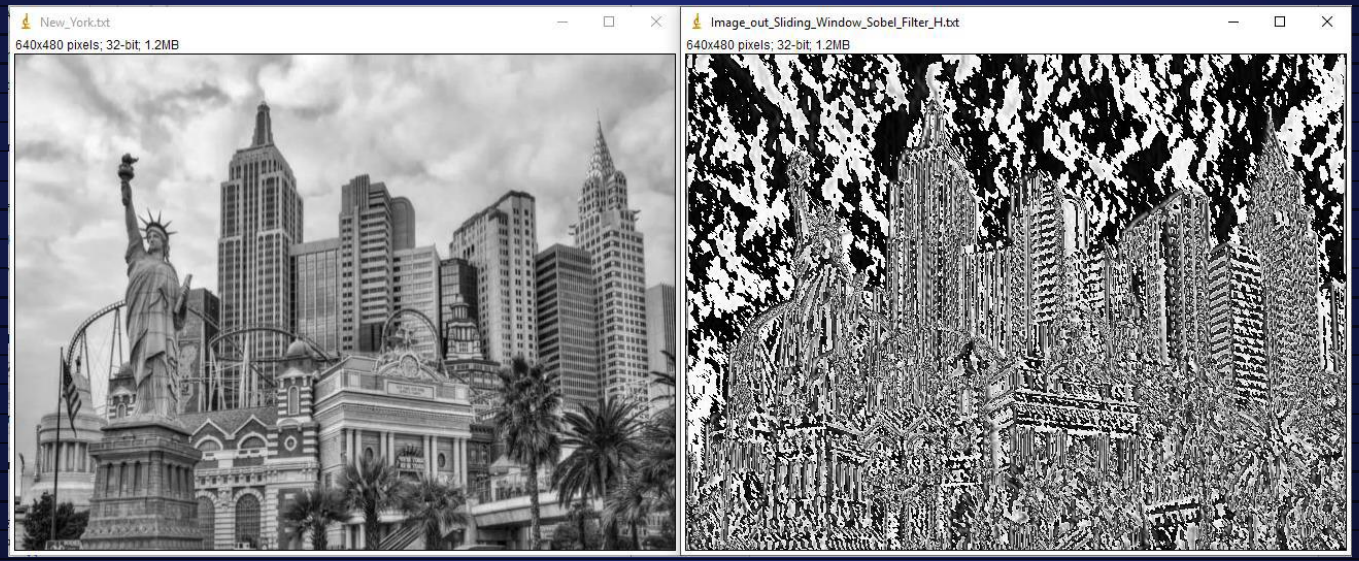
Code VHDL

```
if(Compteur_projet > 643) then
    Calcule_Sobel_H_Nat <= to_integer(signed(reg1))*K1_H_Nat + to_integer(signed(reg2))*K2_H_Nat + to_integer(signed(reg3))*K3_H_Nat +
    output_data <= std_logic_vector(to_unsigned(integer(Calcule_Sobel_H_Nat), 8));
    output_data_valid <= '1';
else
    output_data_valid <= '0';
end if;
```

```
if(Compteur_projet > 643) then
    Calcule_Sobel_V_Nat <= to_integer(signed(reg1))*K1_V_Nat + to_integer(signed(reg2))*K2_V_Nat + to_integer(signed(reg3))*K3_V_Nat +
    output_data <= std_logic_vector(to_unsigned(integer(Calcule_Sobel_V_Nat), 8));
    --output_data <= std_logic_vector(to_signed((to_integer(signed(Calcule_Sobel_V))/4),8));
    output_data_valid <= '1';
else
    output_data_valid <= '0';
end if;
```

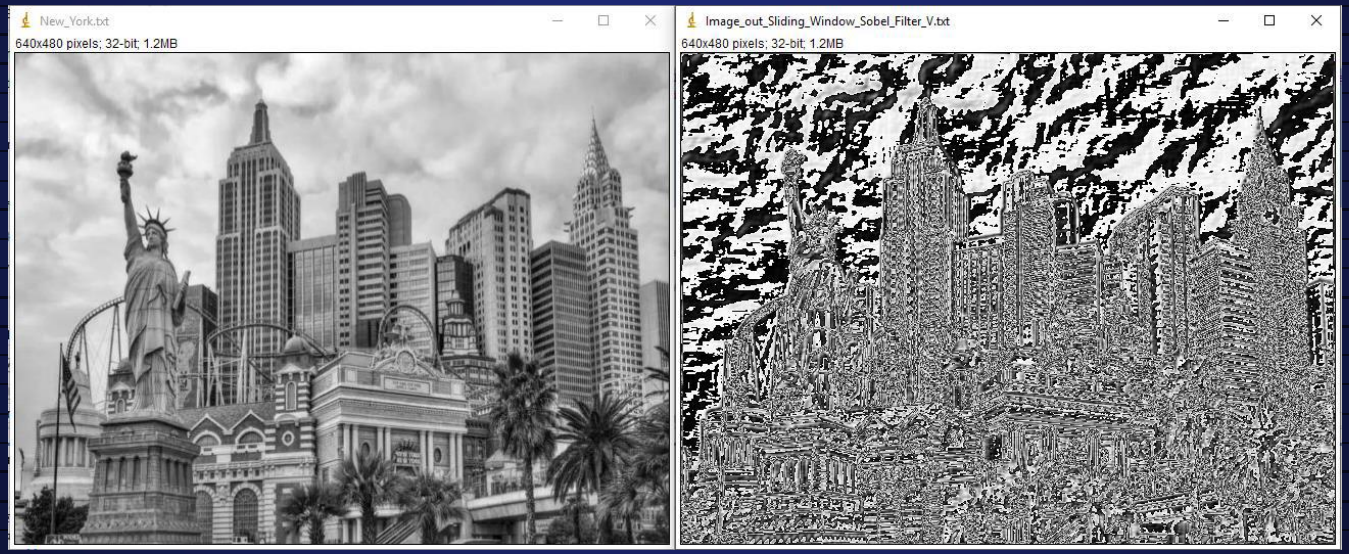
filtre Sobel

Simulation Sobel en H



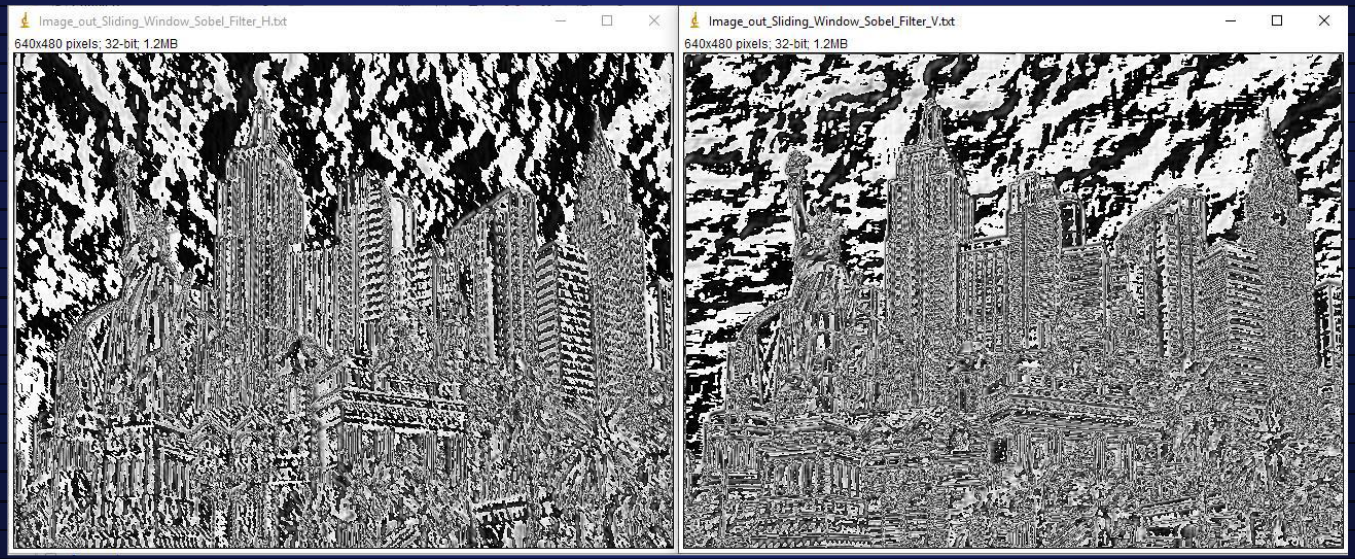
filtre Sobel

Simulation Sobel en V



filtre Sobel

Simulation Sobel en H et en V

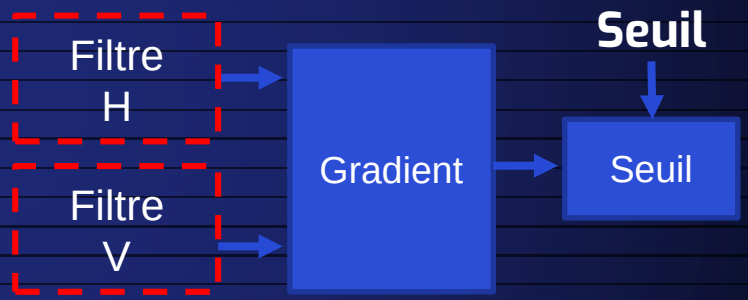


filtre Sobel

Code VHDL

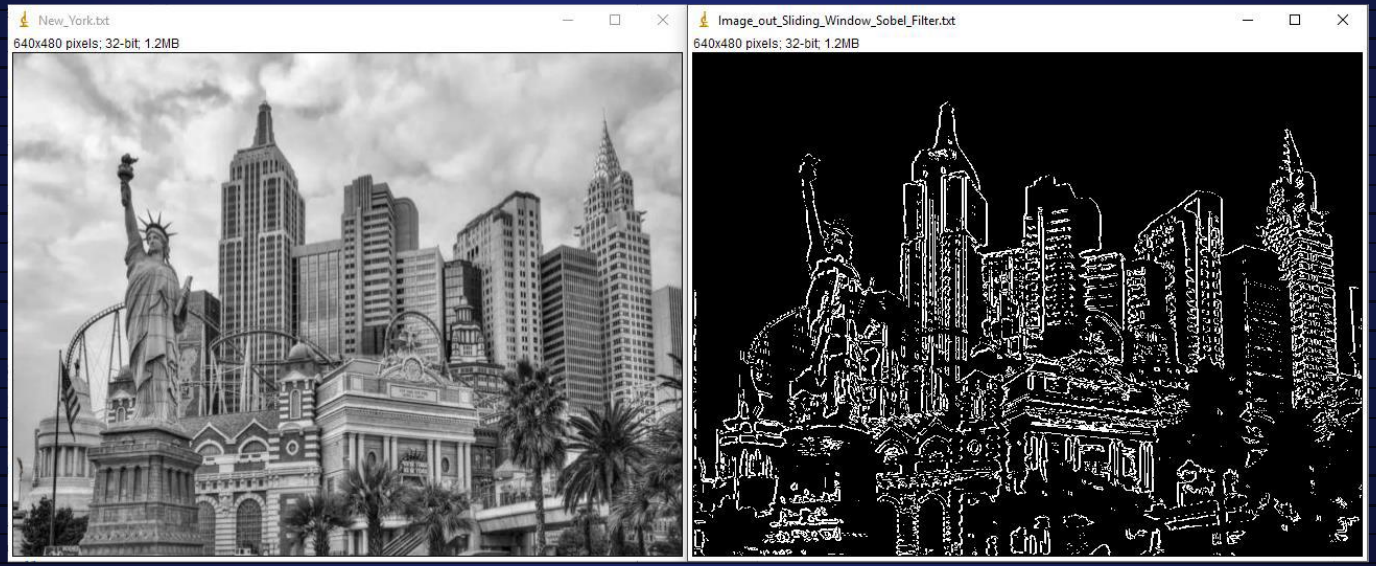
```
if(Calcule_Gradiant_Sobel_Nat > Seuil_Sobel) then
     output_data <= (others => '1');
else
     output_data <= (others => '0');
     --output_data <= reg5;
end if;
output_data_valid <= '1';
```

Synoptique



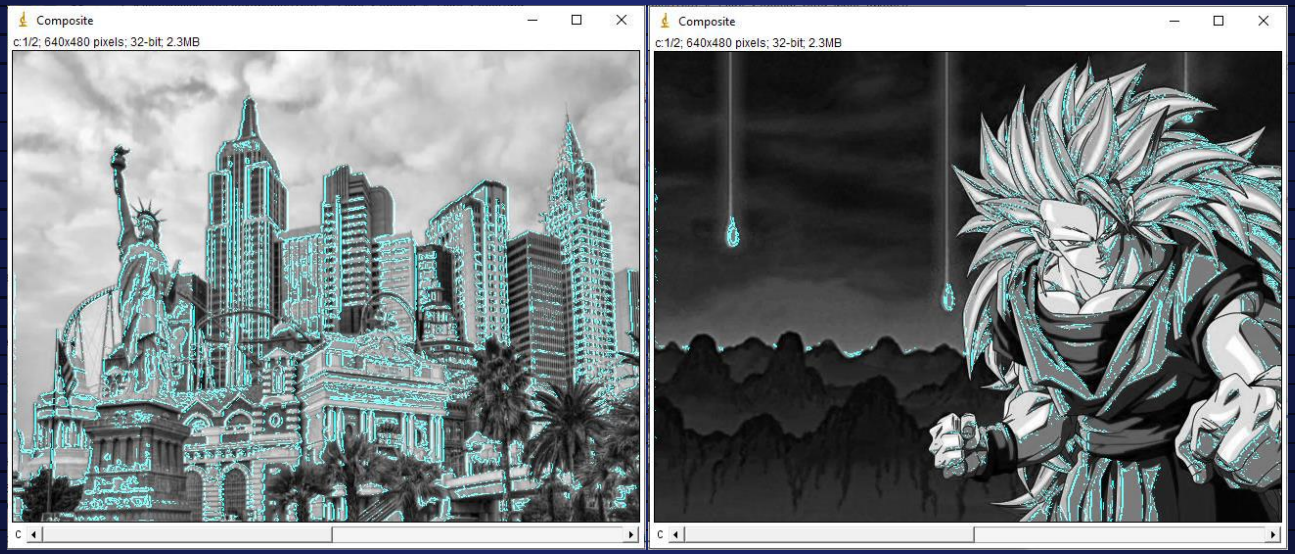
filtre Sobel

Simulation Sobel



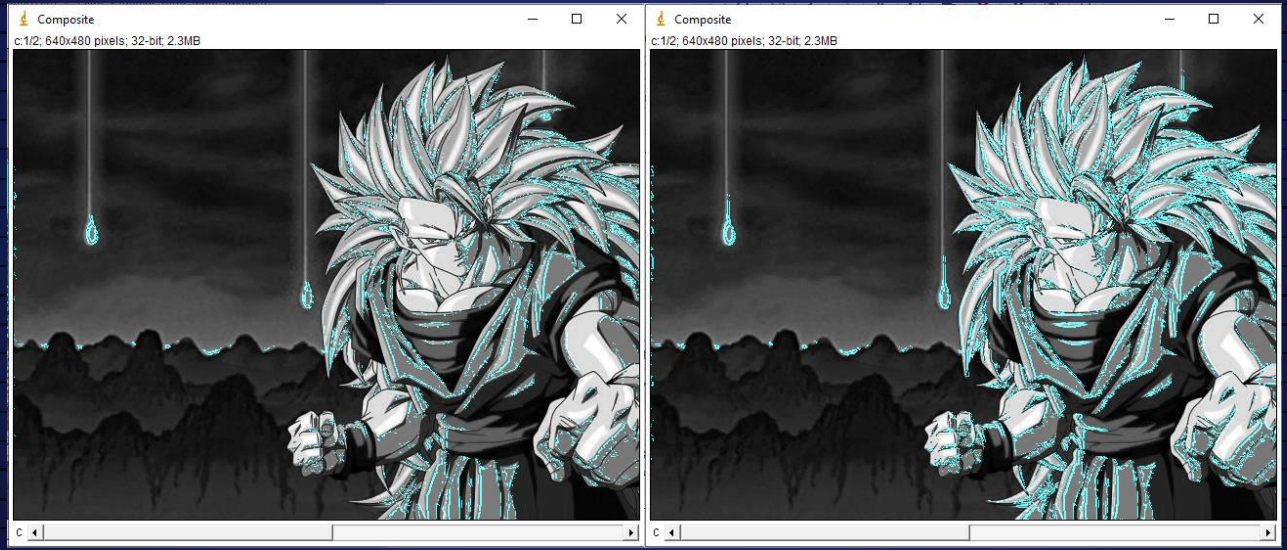
filtre Sobel

Simulation Sobel
seuil : 850



filtre Sobel

Simulation Sobel
seuil : 850 / 650



Synthèse Gauss vs Sobel

Filtre de Gauss

```
---------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
      2 Input   10 Bit       Adders := 2
+---XORs :
      2 Input      1 Bit         XORs := 40
+---Registers :
                  10 Bit    Registers := 4
                   1 Bit    Registers := 4
+---Muxes :
      2 Input    1 Bit        Muxes := 1
---------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------
```

Report Cell Usage:

	Cell	Count
1	LUT1	5
2	LUT2	6
3	LUT3	5
4	LUT4	24
5	LUT5	4
6	LUT6	6
7	MUXCY	20
8	RAMB18E1	1
9	FDRE	40
10	FDSE	4

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	32,624 ns	Worst Hold Slack (WHS):	0,042 ns	Worst Pulse Width Slack (WPWS):	2,000 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	454	Total Number of Endpoints:	454	Total Number of Endpoints:	280

All user specified timing constraints are met.

Filtre de Sobel

```
---------------------------------------------------------
Start RTL Component Statistics
---------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
      2 Input   10 Bit       Adders := 2
+---XORs :
      2 Input      1 Bit         XORs := 40
+---Registers :
                  10 Bit    Registers := 4
                   1 Bit    Registers := 4
+---Muxes :
      2 Input    1 Bit        Muxes := 1
---------------------------------------------------------
Finished RTL Component Statistics
---------------------------------------------------------
```

Report Cell Usage:

	Cell	Count
1	LUT1	5
2	LUT2	6
3	LUT3	5
4	LUT4	24
5	LUT5	4
6	LUT6	6
7	MUXCY	20
8	RAMB18E1	1
9	FDRE	40
10	FDSE	4

Design Timing Summary

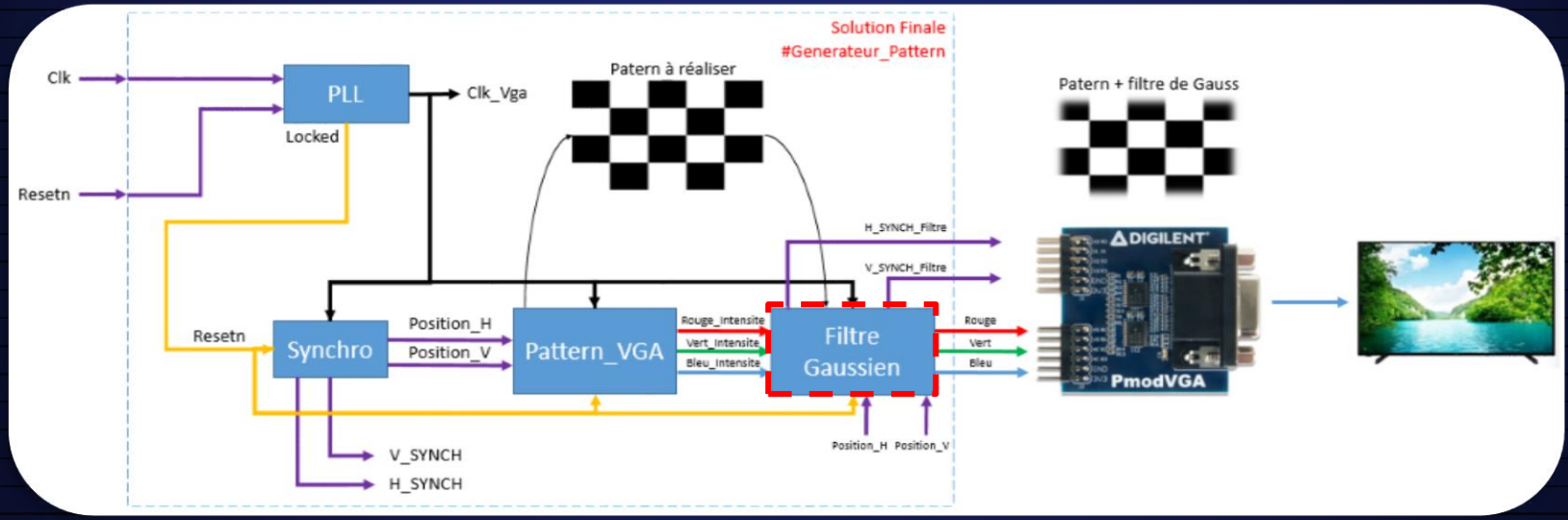
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0,211 ns	Worst Hold Slack (WHS):	0,056 ns	Worst Pulse Width Slack (WPWS):	3,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	465	Total Number of Endpoints:	465	Total Number of Endpoints:	268

All user specified timing constraints are met.

Même architecture entre le projet VGA et le projet de détection de point

→ Le projet est donc totalement implémentable sur Carte en y ajoutant la norme VGA

Synthèse Gauss vs Sobel



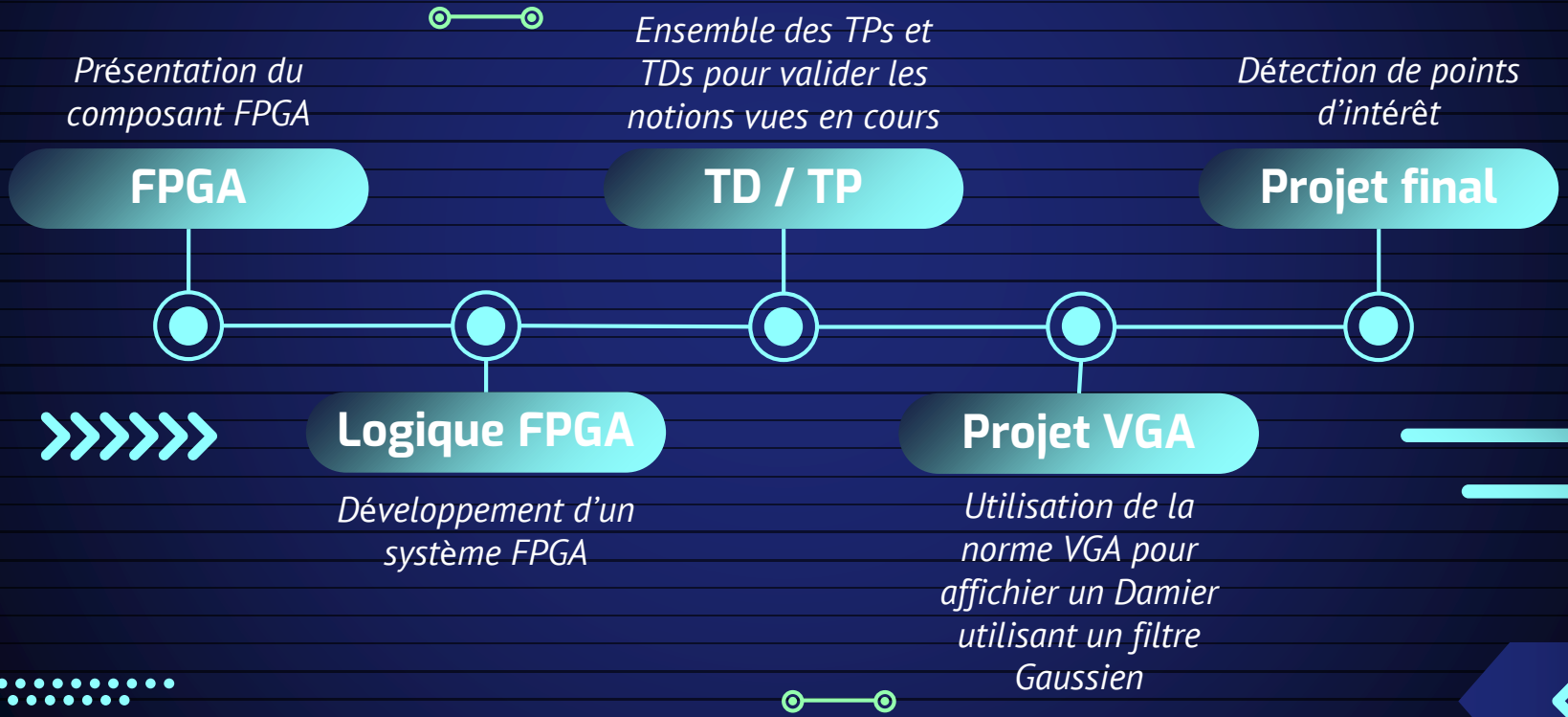
→ Possibilité de remplacer le filtre de Gauss par le filtre de Sobel

05

Retour d’expérience

Retour d’expérience sur la formation

Déroulement de la formation



Retour d'Expérience

Apprentissage du VHDL

Autonomie de codage

Travail d'équipe

Transmission de savoir

- *Validation de la notion de logique combinatoire*
- *Validation de la notion de process*
- *Création de schéma RTL*
- *Validation de la mise en place de simulation à l'aide de TB*
- *Mise en place de la norme VGA (25,175MHz) (640x480 pixels)*
- *Mise en place de filtre de Gauss et de Sobel*
- *Utilisation de la Carte Cora Z7 et validation sur oscilloscope (Hantek)*

Merci

de votre attention

Avez vous des questions?