Épreuve d’admissibilité de conception préliminaire d'un système, d'un procédé ou d'une organisation

A. Présentation de l’épreuve

Arrêté du 19 avril 2013 et arrêté du 19 avril 2016

- Durée totale de l'épreuve : 6 heures
- Coefficient 1

L'épreuve est spécifique à l'option choisie.

À partir d'un dossier technique comportant les éléments nécessaires à l'étude, l'épreuve a pour objectif de vérifier les compétences d'un candidat à synthétiser ses connaissances pour proposer ou justifier des solutions de conception et d'industrialisation d'un système technique dans le domaine de la spécialité du concours dans l'option choisie.

B. Sujet

Le sujet est disponible en téléchargement sur le site du ministère à l’adresse :
http://media.devenirenseignant.gouv.fr/file/agregation_externe/27/6/s2018_agreg_externe_sii_informatique_3_917276.pdf

Ce sujet porte sur l’étude de solutions informatiques de contrôle et de maintenance dans un entrepôt logistique.

C. Éléments de correction

Question 1.

Caractéristiques du capteur CMOS	
Taille d'un pixel du capteur	1.4 µm x 1.4 µm
Hauteur, largeur du capteur L_s x H_s	Hs=1.4*1944=2.7216 mm Ls=1.4*2592=3.6288 mm
Angle d'ouverture horizontale de la lentille	53.50 degrés
Angle d'ouverture verticale de la lentille	41.41 degrés
Distance focale L_f	3.6 mm
Capacité de charge à saturation	4300 e-
Rapport signal sur bruit (Signal to Noise Ratio : SNR)	36 dB
Échelle de mesure	68 dB
Plancher de bruit par seconde	16 mV
Tension électrique d'un pixel pour 1 lx.s (1 lux.s)	0.68 V
Nombre d'images par seconde avec une résolution maximale	15 fps pour une résolution de 2592 x 1944

Question 2.

Longueur L_o et largeur L_a du colis.

$$L_o = \left(\frac{H_p - H_c}{L_f}\right) \times H_s = \frac{1000 - 400}{3.6} \times 2.7216 \cong 454\ mm$$

$$L_a = \left(\frac{H_p - H_c}{L_f}\right) \times L_s = \frac{1000 - 400}{3.6} \times 3.6288 \cong 605\ mm$$

Question 3.

Définition minimale de l'image à la hauteur Hc.

Avec une définition de d_fL=2592 x d_fH=1944 et en considérant la taille d'un pixel t_{px}=1.4µm, les résolutions dans les deux axes sont égales :

$$S_H = \frac{d_f H.\, t_{px}}{L_o} \times \frac{1}{t_{px}} = 4.28\ pixels/mm$$

$$S_L = \frac{d_l L.\, t_{px}}{L_a} \times \frac{1}{t_{px}} = 4.28\ pixels/mm$$

La tolérance est de 5 mm, avec un coefficient de sécurité de 5, cette tolérance est ramenée à 1 mm, ce qui signifie S_H=S_L=1 pixel/mm, d'où la définition minimale d_fH_{min}, d_fL_{min}:

$$d_f H_{min} = S_H L_o = 454\ pixels$$

$$d_f L_{min} = S_L L_a = 605\ pixels$$

Question 4.

Poids de l'image couleur de résolution 605 x 454 avec 1 octet par canal => 605*454*3=921 ko et poids de l'image couleur de résolution 2592 x 1944 =>2592*1944*3=15.1 Mo. Il est donc inutile de choisir la définition maximale, ce qui permet de réduire la tailledes variables du type image dans l'application de contrôle vision.

Question 5.
D'après la relation (2) du sujet et avec SNR=36 dB et V_{noise}=16^{e}-3V, V_{max}=1V.

$$V_{max} = V_{noise}.10^{\frac{SNR_{max}}{20}} = 16e - 3 * 10^{\frac{36}{20}} \cong 1V$$

Pour H=1 lx.s, V=0.68V, la pente de la droite est égale à α=0.68, d'où :

$$H_{max} = \frac{V_{max}}{\alpha} = \frac{1}{0.68} = 1.47\ lx.s$$

Question 6.
La pente de la droite est égale àα=0.68 et V_{min} est égale au plancher de bruit soit 16 mV :

$$H_{min} = \frac{darkcurrent}{\alpha} = \frac{16e - 3}{0.68} \cong 23\ mlx.s$$

Question 7.
Les conditions sont optimales lorsque la tension est égale à 1V ; ce qui correspond à une exposition lumineuse de 1.47 lx.s ; or l'intensité lumineuse (E_{min}) minimum dans l'entrepôt est de 50 lx (cas le plus défavorable). La durée minimale d'exposition est donc égale à :

$$t_{emin} = \frac{H_{max}}{E_{min}} = \frac{1.47}{50} \cong 30\ ms$$

Question 8.
Pour une définition de 640x480 (VGA) la vitesse d'acquisition est égale à 90 images/seconde (fps).
Le choix de la valeur fps=30 images/seconde permetd'obtenir un temps d'intégration de 33 ms. Ce qui est cohérent avec la durée de la question précédente. Le choix d'une vitesse encore plus petite serait possible mais cela réduirait inutilement le temps d'exécution de l'application.

Question 9.
Valeur du bruit au niveau de la photodiode : d'après l'expression (3) du sujet, avec DR=68 dB et FWC=4300 e-, le bruit est égal à :

$$ReadNoise = \frac{FWC}{10^{\frac{DR}{20}}} = \frac{4300}{10^{\frac{68}{20}}} = 1.7\ e^{-}$$

Les critères de choix d'une caméra CMOS sont :
- -DR : plage d'exposition lumineuse ;
- définition de l'image en pixel (largeur x hauteur) ;
- résolution du CAN ;
- durée d'exposition minimale.

Question 10.
Configuration de la caméra avec la classe **Raspicam** :

```
RaspiCam camera;
    camera.setCaptureSize(640,480);
    camera.setFormat(RASPICAM_FORMAT_RGB);
    camera.setBrightness(70);
    camera.setISO(100);
    camera.setContrast(100);
    camera.setExposure(RASPICAM_EXPOSURE_FIXEDFPS);
    camera.setShutterSpeed(330000);
    camera.setAWB(RASPICAM_AWB_AUTO);
```

Question 11.
La taille de l'image est égale à 921 600 octets (N*K*3=640*480*3).

La variable « data » est par conséquent un tableau 1D (une dimension) dynamique de 921 600 octets. Les lignes sont stockées les unes à la suite des autres. Les composantes RGB sont stockées les unes à la suite des autres dans l'ordre des pixels d'une ligne.

En considérant un tableau à 2 dimensions de taille N x K, les éléments d'un pixel (R,G,B) sont rangés dans un tableau 1D selon les index p_R, p_G et p_B respectivement associés aux composantes R, G et B :

$$p_R = K \times i + 3 \times j$$
$$p_G = K \times i + 3 \times j + 1$$
$$p_B = K \times i + 3 \times j + 2$$

Avec :

$$i \in [0, N-1] et j \in [0, K-1]$$

Par exemple pour la première ligne N=0, (avec K=640, le nombre de colonnes) :
[R_0_0, G_0_0, B_0_0, R_0_1, G_0_1, B_0_1…R_0_(K-1), G_0_(K-1), B_0_(K-1)]

Et pour la dernière ligne N=479 :
[R_(N-1)_0, G_(N-1)_0, B_(N-1)_0,…R_(N-1)_(K-1), G_(N-1)_(K-1), B_(N-1)_(K-1)]

Question 12.

```
#Taille maxi du buffer circulaire
N=8
#Buffer circulaire
bCirc=[]
#K éléments
K=13
for i in range(K):
#Nombre d'éléments inférieur à la taille du buffer circulaire
if(len(bCirc)<N):
        bCirc.append(i)
#Lorsque Nombre d'éléments supérieur à la taille du buffer circulaire
#Suppression de l'élément le plus vieux
#Ajout du nouvel élément
else:
bCirc.pop(i%N)
bCirc.insert(i%N,i)
```

Question 13.

Ci-dessous, les sorties consoles associées au programme précédent lorsque i>7 :

[8, 1, 2, 3, 4, 5, 6, 7]
[8, 9, 2, 3, 4, 5, 6, 7]
[8, 9, 10, 3, 4, 5, 6, 7]
[8, 9, 10, 11, 4, 5, 6, 7]
[8, 9, 10, 11, 12, 5, 6, 7]

Question 14.

En utilisant un tri sur place par indexation :

```
#Tri par permutation circulaire
for k in range(i%N-1):
#Sauvegarde dernier élément
temp=bCirc[len(bCirc)-1]
for j in range(len(bCirc)-1,0,-1):
        bCirc[j]=bCirc[j-1]
    bCirc[0]=temp
```

Autres solutions utilisant les méthodes « append », « pop » etc… du type list de Python.

```
for i in range(K):
if(len(bCirc)>=N):
        bCirc.pop(i%N)
        bCirc.insert(i%N,i)
else:
bCirc.append(i)
```

Question 15.

Bibliothèque STL (standard template library C++) avec le type listou vector. Bibliothèque BOOST qui contient nativement un buffer circulaire (étudié dans la partie 3 du sujet).

Question 16.

Chaque niveau de gris est calculé suivant une moyenne pondérée par la proportion de la composante de couleur dans l'image initiale (source) :

$$I_g(i,j) = I_s(i,j)[0] \times W_{red} + I_s(i,j)[1] \times W_{green} + I_s(i,j)[3] \times W_{blue}$$

Question 17.

```
Mat imGray(480,640,CV_8UC1);
imGray.at<uchar>(0,0)=255;
```

Ou bien :

```
Mat imGray(480,640,CV_8UC1);
imGray.at<vec<uchar,1>>(0,0)=255;
```

Ou bien :

```
Mat imGray(480,640,CV_8UC1);
imGray.data[0]=255;
```

Question 18.

Accès par l'attribut data de la classe Mat par indexation :

```
imGray.data[480*640-1];
```

Ou bien suivant le déréférencement du pointeur associé :

```
*(imGray.data+480*640-1);
```

L'instruction imGray[H-1][L-1] sera rejetée car l'opérateur [ ] n'est pas surchargé et imGray est un objet composé de plusieurs données privées de différents types.

Question 19.

Code source permettant de déterminer l'image en niveaux de gris (8 bits) :

```
void grayScale_NEGG(Mat imSrc,Mat imGray)
{
int i,j;
int TRed=0,TGreen=0,TBlue=0;
float WRed,WGreen,WBlue;
for(i=0;i<imSrc.rows;i++)
{
for(j=0;j<imSrc.cols;j++)
{
            TBlue=TBlue+imSrc.at< Vec<uchar,3>>(i,j)[0];
            TGreen=TGreen+imSrc.at< Vec<uchar,3>>(i,j)[1];
            TRed=TRed+imSrc.at< Vec<uchar,3>>(i,j)[2];
}
}
    WBlue=(float)(TBlue)/(TBlue+TGreen+TRed);
    WGreen=(float)(TGreen)/(TBlue+TGreen+TRed);
    WRed=(float)(TRed)/(TBlue+TGreen+TRed);
```

```
for(i=0;i<imSrc.rows;i++)
{
for(j=0;j<imSrc.cols;j++)
{
            imGray.at<uchar>(i,j)=(uchar)(imSrc.at<
Vec<uchar,3>>(i,j)[0]*WBlue+imSrc.at< Vec<uchar,3>>(i,j)[1]*WGreen+imSrc.at<
Vec<uchar,3>>(i,j)[2]*WRed);
}
}
}
```

Question 20.

Expression des gradients suivants les deux directions :

$$G_x(i,j) = \sum_{l=-q}^{q} \sum_{k=-q}^{q} I_g(i+l, j+k) \times M_x(l+q, k+q)$$

$$G_y(i,j) = \sum_{l=-q}^{q} \sum_{k=-q}^{q} I_g(i+l, j+k) \times M_y(l+q, k+q)$$

$$\|G(i,j)\| = \sqrt{G_x(i,j)^2 + G_y(i,j)^2}$$

Question 21.

Valeur du gradient pour le pixel de l'image en niveaux de gris, Ig(245,557) :

$$G_x(245{,}557) = 91 \times -1 + 79 \times 0 + 47 \times 1 + 92 \times -2 + 84 \times 0 + 48 \times 2 + 93 \times -1 + 85 \times 0 + 48 \times 1$$
$$= -177$$

$$G_y(245{,}557) = 91 \times -1 + 79 \times -2 + 47 \times -1 + 92 \times 0 + 84 \times 0 + 48 \times 0 + 93 \times 1 + 85 \times 2 + 48 \times 1 = 15$$

$$\|G(245{,}557)\| = \sqrt{177^2 + 15^2} \cong 177$$

Les pixels de valeur 79, 84, 85 et 89 définissent le contour qui est une ligne verticale.

Question 22.

Le type float ou double tout d'abord, puis une conversion explicite à réaliser en niveaux de gris, i.e. vers le type prédéfini CV_8UC1.

Question 23.

D'après la documentation associée, une binarisation est réalisée avec une méthode à deux seuils :
Si la valeur du pixel > threshold2, alors le pixel appartient au contour (valeur forcée à 255).
Si la valeur du pixel <threshold1, alors le pixel n'est pas un contour (valeur forcée à 0).
Si la valeur du pixel est comprise entre les deux seuils, il appartiendra au contour s'il est connecté à au moins un pixel dont la valeur est supérieure à threshold2.

Question 24.

D'après la documentation : la variable threshold2 correspond à la valeur du gradient la plus élevée ; à la question 21 : threshold2=177.
Le rapport threshold1/threshold2 préconisé est compris entre 1/3 et 1/2. Par exemple threshold1=(177/3)=60.

```
Mat imCanny(480,640,CV_8UC1);
Canny(imGray,imCanny,60,177,3,true);
```

Question 25.

En remarquant :

$$\omega_0(k) + \omega_1(k) = 1$$
$$\omega_1(k) = 1 - \omega_0(k)$$

Question 26. Variance inter-classe :

Variance inter-classe :

$$\sigma_B^2(k) = \omega_0(k)\big(1-\omega_0(k)\big)\left[\frac{\mu_T - \mu(k)}{1-\omega_0(k)} - \frac{\mu(k)}{\omega_0(k)}\right]^2$$

Question 27.

Code source du calcul de la variance :

```
float varianceOtsu(float histoN[],uchar k)
{
float var2;
float w0,muT,mu;
int i;
//Probabilité de la classe C0 et valeur moyenne C0
for(i=0;i<k+1;i++)
{
        w0=w0+histoN[i];
        mu=mu+i*histoN[i];
}
//Valeur moyenne totale
for(i=0;i<256;i++)
{
muT=muT+i*histoN[i];
}
    var2=w0*(1-w0)*(((muT-mu)/(1-w0))-mu/w0)*(((muT-mu)/(1-w0))-mu/w0);
return(var2);
}
```

Exceptions à lever : w0=1 et w0=0.

Question 28.

Code source pour la recherche de threshold2 (variance maximum) et threshold1 (1/3 de la variance maximum) :

```
float maxVar=sigmaB[0];
float minVar;
uchar iMax=0;
for(i=0;i<256;i++)
{
if(sigmaB[i]>maxVar)
{
      maxVar=sigmaB[i];
iMax=i;
}
}
minVar=maxVar/3;
```

Question 29.

D'après le graphique : threshold2=70, threshold1=23 à 35.

Question 30.

Template de classe vector.

```
std::vector< std::vector<cv::Point>> contours;
```

std::vector<**typX**> : tableau de variable du type « typX »,

std::vector<**cv**::**Point**> : tableau de point (x,y) associé à un contour unique.

std::vector< **std**::vector<**cv**::**Point**>> : tableau contenant l'ensemble des contours,

D'après la documentation, l'index de la première régionest définie par l'expression : contours[0].

Question 31.

Code source permettant de déterminer la région associée à la surface maximale.

```
void findMaxArea(std::vector< std::vector<cv::Point>> contours,int&iMaxArea)
{
int i;
double maxArea;
    iMaxArea=0;
if(contours.size()>0)
{
        maxArea=cv::contourArea(contours[0],false);
for(i=0;i<(int)(contours.size());i++)
{
if(cv::contourArea(contours[i],false)>maxArea)
{
                maxArea=cv::contourArea(contours[i],false);
                iMaxArea=i;
}
}
}
}
```

Question 32.

Principe de la fonction drawBoundingRectangle() rappelée ci-dessous :

```
void drawBoundingRectangle(Mat src,std::vector< std::vector<cv::Point>>
contours,cv::Point2f pt[],int iMaxArea,RotatedRect &rect)
{
unsignedchar j;
    rect=cv::minAreaRect(contours[iMaxArea]);
    rect.points(pt);
//Dessin du bounding rectangle
for(j=0;j<4;j++)
{
        line(src,pt[j],pt[(j+1)%4],Scalar(255,0,0),1);
}
}
```

Cette fonction prend comme paramètre, le tableau des contours et la valeur de l'index associé à la plus grande région.
La fonction minAreaRect(…) retourne le rectangle englobant la région,
L'instruction rect.points(pt) permet d'affecter au point **pt**les coordonnées de **rect.**
La boucle FOR permet de tracer le rectangle avec la fonction line(…).
j=0, line1 : pt[0]—pt[1]
j=1, line2 : pt[1]—pt[2]
j=2, line2 : pt[2]—pt[3]
j=3, line2 : pt[3]—pt[0]

L'opérateur modulo permet de revenir au premier point afin de fermer le rectangle.

Question 33.

Détermination des dimensions et de l'orientation du colis :

```
int iMaxArea;
findMaxArea(contours,iMaxArea);
cv::Point2f pt[4];
RotatedRect rect;
drawBoundingRectangle(imSrc,contours,pt,iMaxArea,rect);
float hauteur=rect.size.height;
float largeur=rect.size.width;
float deviation=rect.angle;
```

Dans le cas d'un colis parfaitement rectiligne, le rectangle englobant au plus près permettra de déterminer les dimensions. Si le rectangle englobant est mal positionné, l'erreur pourra être supérieure à celle tolérée.
Autre solution : Déterminer les dimensions du contour englobant au plus près du colis.
Enfin le contrôle dimensionnel sera correct si le colis n'est pas déformé car les dimensions sont celles du rectangle englobant.

Question 34.
Identification des paramètres de la fonction approxPolyDP(…) rappelée ci-dessous :

void cv::**approxPolyDP**(InputArray **curve**, OutputArray **approxCurve**,double **epsilon**,
bool closed)

-curve : l'élément de la variable contour associé à la plus grande région,
-approxCurve : contient les points de la ligne polygonale simplifiée,
-epsilon : valeur maximale de l'erreur de la ligne polygonale simplifiée par rapport à la ligne polygonale initiale,
-si vraie, la ligne polygonale sera fermée : connecte le premier point avec le dernier point.

Question 35.
Équation de la première droite entre P0=(x_0,y_0) et P_{N-1}=(x_{N-1},y_{N-1}) :

$$b \times y_i - a \times x_i - c = 0 \text{ avec } b = 1$$
$$a = 0.0125 \text{ et } c = y_0 = 6.2$$

En remplaçant a, b et c dans l'expression suivante :

$$d_i = \frac{|-ax_i + by_i - c|}{\sqrt{x_i^2 + y_i^2}}$$
$$d_{max} = 1.13$$

Question 36.
La récursivité s'arrête lorsque d_{max} est inférieur ou égal à epsilon. Les points sont stockés dans la liste tR.

Question 37.
Dans le meilleur des cas :
Si le premier d_{max} est inférieure ou égale à epsilon, les points sont considérés alignés sur la première droite. Il y a N-2 itérations pour la boucle FOR puis arrêt de la récursivité. Complexité en O(N).

Dans le pire des cas :
À chaque appel récursif, le deuxième point est choisi comme pivot. Le coût de la boucle FOR au premier appel est égal à N-2, puis celui de la boucle FOR du deuxième appel est égal à N-3 et ainsi de suite. Cette analyse aboutit à l'expression du coût total et de la complexité associée :

$$\sum_{2}^{N-2} N - i = \sum_{j=2}^{j=N-2} j = \frac{(N-2)\times(N-1)}{2} \Rightarrow O(N^2)$$
$$avec\ j = N - i$$

Question 38.
Le paramètre epsilon représente la précision de la ligne polygonale par rapport au contour réel. Epsilon est une dimension relative à la taille de l'image en pixels. D'après les images test, la valeur devrait se situer entre 0 et 5.

Question 39.
Il est possible de comparer la surface du rectangle englobant et la surface délimitée par les points de la ligne polygonale. Une autre solution serait de compter le nombre de coins de la forme détectée (méthode de Harris). Pour déterminer la déformation sur un seul côté, l'algorithme de Ramer-Peucker-Douglas est également utilisable. Il suffira de définir au préalable, une distance maximum entre la ligne polygonale et le rectangle englobant au-delà de laquelle le colis sera considéré déformé.

Question 40.
Le masque est de 18 bits -> 214-2 machines possibles.
@min=10.239.64.1
@max=10.239.127.254
Le masque n'est pas en concordance avec la classe A mais cela n'a pas d'incidence depuis bien longtemps avec le classless (trop de limitations sans cela pour segmenter un réseau).

Question 41.
La segmentation d'un réseau se fait grâce à l'ajout d'un routeur.

Question 42.
Afin de pouvoir capturer les trames il est nécessaire d'utiliser un matériel qui diffuse les trames à toutes les stations sur le réseau. Il est donc nécessaire d'utiliser un Hub.

Question 43.
Le mode promiscuous permet de configurer l'interface réseau pour qu'elle ne filtre pas les trames dont les @ mac ne sont pas la sienne propre. En temps normal, les trames sont filtrées et ne remontent pas au noyau de l'OS. Grâce à un logiciel de type wireshark, ceci permet ensuite de visualiser toutes les trames capturées par l'interface réseau.

Question 44.
Une fonction de callback est une fonction passée en paramètre et destinée à être appelée (en général lorsqu'un certain type d'évènement survient). Cela permet de modulariser le traitement à réaliser en laissant au développeur le soin de ce traitement.

Question 45.
La fonction de callback doit respecter un prototype afin de pouvoir être passée en paramètre à pcap_loop(). En analysant les codes fournis, on constate que la signature de la fonction traiter_trame() est identique à celle qu'il faut fournir à pcap_loop().

Question 46.
Une solution possible.
```
void traiterTrame(...) //on ne rappelle pas les paramètres...
{
      if strstr(packet,"DEFAUT") Defaut=1;
}
```

Question 47.
Comme on a besoin de synchroniser deux tâches, il est beaucoup plus simple de le faire en multitâche plutôt qu'avec des applications dans des processus lourds différents. Les noyaux possèdent des moyens de synchronisation tels que les mutex/semaphores par exemple.

Question 48.
Le thread principal est le thread parent. S'il venait à terminer, tout son espace est libéré. Par conséquent, les threads enfants n'ont plus d'espace et sont forcés à terminer aussi.

Question 49.
Bien évidemment, ce type de compteur est beaucoup plus simple à gérer car il est déjà optimisé et implémente directement le buffer circulaire.

Question 50.
On a une complexité asymptotique constante, c'est à dire que le temps d'insertion ne dépend pas de ce qu'il y a à insérer ni où on l'insère. On a donc un temps garanti.

Question 51.
```
//on retient une capacité de 500
//car avec une image toutes les 100ms pendant 50 sec
//cela donne 500 images à stocker.

boost::circular_buffer<Mat> cb(500);
```

Question 52.
```
while(1)
{
	usleep(100000);
	vcap >> frame;  //lecture d'une image
	 cb.push_back(frame) ; //on pousse l'image dans le buffer
	                  //et ça écrase automatiquement au fur et à mesure.
	if (Defaut==1) //on a détecté un défaut
	{
		Defaut = 0;
		//sauvegarde des frames du buffer dans un fichier
		while( !cb.empty() )
		{
			video.write(cb[0]); //on sauvegarde la première frame pour
		avoir
		                        //la vidéo dans le bon sens
		            cb.pop_front() ; //puis on la détruit
		}
		//on enregistre ensuite les autres frames
		//pendant les 10 sec suivantes
		Temps=0 ;
		while(Temps <= 100)
		{
			vcap >> frame ;
			video.write(frame) ;
			usleep(100000) ; //on attends 100ms
			Temps += 1 ;
		}
	}
}
```

Question 53.
Les points suivants pouvaient être abordés sur le système étudié :

- l'architecture utilisée permet de répondre au cahier des charges. L'utilisation du multitâche permet d'éviter un appel bloquant et rend la modification de l'application plus souple en cas de changement ;
- le point concernant la détection du défaut en espionnant des trames réseau est intéressante car elle ne modifie pas le superviseur. Cependant cette solution est moins satisfaisante que la modification du superviseur utilisant un socket pour communiquer directement avec l'application gérant l'enregistrement vidéo (donc plus d'espionnage du réseau et plus de hub).