INFO-F106 : PROJETS D'INFORMATIQUE 1

Vandy Berten Jérôme Dossogne Vincent Ho Markus Lindström
Catharina Olsen Alessia Violin

version 3.0 - 8 février 2011
http://www.ulb.ac.be/di/info-f106/

Addendum pour les parties 3 et 4 (version 3.0)

Quelques consignes et conseils supplémentaires pour la dernière partie.

1. Si la plupart des rapports de la deuxième partie étaient tout à fait recevables sur le fond, il n'en allait pas de même en ce qui concerne lait forme, ce principalement pour deux raisons :
   - L'orthographe de la majorité des rapports était déplorable ! Nous n'exigeons certes pas que vous ayez une orthographe parfaite (cet énoncé contient certainement des fautes), mais une faute d'orthographe ou de grammaire dans chaque paragraphe, ce n'est pas acceptable. Nous serons plus exigeants pour le rapport final. N'hésitez pas à vous servir d'un dictionnaire, Bescherelle, voire même simplement du correcteur orthographique et grammatical de votre traitement de texte.
   - Tout rapport recevable exempt de fautes d'orthographe (y compris les accents sur les majuscules et règles de typographie) sera récompensé d'un bonus de 10 points. Il sera donc possible d'avoir 30/20 pour le rapport final !
   - Le rapport DEVRA contenir une table des matières générée AUTOMATIQUEMENT. Toute table des matières créée manuellement sera considérée comme manquante. Pour les utilisateurs d'un traitement de texte, cela impose donc l'utilisation des styles pour la gestion des titres.
2. Nous vous demandons de faire particulièrement attention à l'impression de votre code. Un code imprimé sans coloration syntaxique est *illisible*. De nombreux éditeurs, sous Windows, Linux ou MacOS, permettent, par exemple, de mettre en gras les mots-clés et en grisé les commentaires. Il n'est pas nécessaire d'imprimer en couleurs. Par ailleurs, n'hésitez pas à imprimer votre code en *paysage* si l'impression en portrait coupe régulièrement les lignes de code.
3. Toujours en matière de lisibilité, pensez à aérer votre code de façon appropriée. Une ou deux lignes vides entre deux fonctions est raisonnable. Cinq lignes blanches entre deux instructions, c'est beaucoup trop !
4. Le projet devra continuer à fonctionner via la ligne de commande, même lorsqu'il sera intégré à l'interface web. Pour faciliter les corrections, il faudra que votre projet contienne les deux fichiers suivants (avec les noms indiqués) :
   - `monProjet.py`, qui doit encore être utilisable avec la même syntaxe que les exemples donnés pour la partie 1 et 2,
   - `index.py`, qui génèrera la page HTML décrite dans l'énoncé.

   Il vous est bien entendu loisible (et vivement conseillé !) d'utiliser d'autres fichiers, que ces deux scripts Python invoqueront.
5. La partie 4 du projet a été mise à jour, et est maintenant complète.

Addendum pour la partie 2 (version 2.0)

Suite aux questions posées durant la partie 1 et aux corrections de cette même partie, voici quelques commentaires, précisions, ou ajouts, qui devront être pris en compte pour la partie 2 et les suivantes.

1. L'intensité maximale ne désigne pas la valeur maximale présente dans la matrice, mais la plus grande valeur possible. Par exemple, si cette valeur maximale est 100, un pixel blanc sera représenté par le triplet [100, 100, 100], mais rien ne dit qu'un tel pixel est présent dans l'image. Cette valeur ne peut par conséquent pas être déduite de la matrice lors de la création du fichier, et doit donc être mémorisée lors de la lecture du fichier originel.
2. Pour mieux correspondre au format tel qu'il est défini par les standards, nous vous demandons de considérer les commentaires. Un commentaire est une (partie de) ligne débutant par '#', qui peut apparaître à n'importe quelle position, y compris en toute première ligne. Ceux-ci ne doivent toutefois pas être conservés dans l'image résultat.
3. Les séparateurs entre les nombres peuvent être des espaces, tabulations, retours à la ligne, ou toute combinaison de ces « blancs ».
4. Votre code devra accepter des images ayant des commentaires ou des blancs avant le « P3 ». Cependant, pour rendre les images que vous produirez plus compatibles avec des outils dont nous nous servirons pas la suite, nous vous suggérons, lorsque vous générez une image, de ne rien faire apparaître (ni blancs ni commentaires) avant le « P3 ».
5. Votre code doit faire preuve d'une certaine robustesse. En particulier, il doit être capable de gérer une image PPM produite par le logiciel GIMP, mais aussi une image telle que le fichier de test donné sur la page Web du projet. Par ailleurs, si le nombre de paramètres que votre script reçoit n'est pas correct, ou que l'opération spécifiée n'existe pas, il affichera un message d'erreur clair.
6. Comme précisé dans l'énoncé, nous attendons pour chaque partie une version papier ET une version électronique. Pour la suite, tout projet pour lequel un des parties manque ne sera pas corrigé.
7. L'exemple de la stéganographie a été corrigé. Un paragraphe a par ailleurs été ajouté dans cette section.
8. Une correction de la partie 1 est disponible sur la page du projet. Nous vous suggérons cependant de continuer avec votre code, mais de tenir compte des commentaires qui auront été faits. Cette correction tient déjà compte des éléments présentés ci-dessus.

1 Idée générale

Le but de ce projet est de réaliser, en Python, un outil permettant d'effectuer du traitement d'images. Il faudra réaliser des rotations, des images miroir, des filtres divers (floutage, etc.), mais également des outils cryptographiques tels que de la stéganographie (cacher un texte dans une image). L'application sera disponible au travers d'une page Web : sur la page en question, on trouvera un formulaire permettant de sélectionner sur l'ordinateur local l'image à envoyer, le type d'outil à appliquer, éventuellement l'une ou l'autre option, puis un bouton « Traiter ». La page suivante affichera ensuite l'image traitée.

2 Objectifs pédagogiques

Ce projet « transdisciplinaire » permettra de solliciter les compétences des étudiants selon différents aspects.

- Des connaissances vues aux cours de programmation, langages, ou algorithmique seront mises à contribution, avec une vue à plus long terme que ce que l'on retrouve dans les divers petits projets de ces cours. L'ampleur du projet requerra une analyse plus stricte et poussée que celle nécessaire à l'écriture d'un projet d'une page ainsi qu'une utilisation rigoureuse des différents concepts vus aux cours ;
- Des connaissances non vues aux cours seront nécessaires et les étudiants seront invités à les étudier par eux-mêmes, soit de façon totalement autonome, soit grâce aux « tuyaux » fournis par l'équipe encadrant le cours. Il s'agit entre autres d'une connaissance de base des langages HTML et CSS de présentation de pages Web, ainsi que de l'utilisation de Python pour réaliser un script générant ce code HTML en fonction d'une image et de paramètres reçus au travers d'un formulaire ;

- Des compétences de communication seront également nécessaires : un peu avant Noël et un peu avant Pâques, les étudiants remettront un rapport expliquant leur analyse, les difficultés rencontrées et les solutions proposées. Une utilisation correcte des outils de traitement de texte (utilisation des styles, homogénéité de la présentation, mise en page, etc.) sera attendue de la part des étudiants. Une orthographe correcte sera bien entendu exigée ;
- En plus des deux rapports, une présentation orale, avec « transparents » ou « slides » (produits par exemple avec PowerPoint ou OpenOffice) sera organisée, ainsi qu'une démonstration de l'outil développé. À nouveau, on attendra des étudiants une capacité à présenter et à vulgariser (c'est-à-dire rendre compréhensible pour des non informaticiens) leur projet.

En résumé, on demandera aux étudiants de montrer qu'ils sont capables d'appliquer des concepts vus aux cours, de découvrir par eux-mêmes des nouvelles matières et enfin de communiquer de façon scientifique le résultat de leur travail.

3 Règles du jeu

Il est vivement conseillé de relire ces règles avant chaque remise d'une partie de projet !

- Ce projet est INDIVIDUEL ! Il est bien entendu acceptable d'en discuter entre collègues, mais deux copies trop similaires seront considérées comme frauduleuses et seront sanctionnées ;
- Le projet sera organisé autour de quatre grandes étapes. Il y aura également un rapport intermédiaire à remettre au milieu du projet ainsi qu'un rapport final à la fin. Chaque partie comptera pour 20 points et le rapport comptera également pour 20 points, ce qui nous fait un total de 100 points ;
- L'énoncé fournit une description de l'ensemble du projet. Comme dans la « vraie vie » d'un informaticien, celui-ci évoluera au cours du temps : des précisions seront données, des aides fournies, éventuellement des modifications apportées. Chaque mise à jour sera annoncée aux valves, il est donc vivement recommandé de les consulter régulièrement ;
- Chaque partie devra être remise sur deux formats :
  - par courriel, à l'adresse infof106@lit.ulb.ac.be : tous les fichiers créés (sauf les images), scripts, rapports, fichiers HTML et CSS. Le courriel aura pour sujet « <Nom> <Prénom> - <Matricule> », avec les informations adéquates, et devra OBLIGATOIREMENT être envoyé depuis une adresse ULB,
  - sur papier, au secrétariat étudiants du département d'Informatique (local 2.N8.104 - Plaine), une version imprimée du code source de chacun des fichiers envoyés par courriel. Chaque feuille reprendra le nom et le prénom de l'étudiant, ainsi que son numéro de matricule ;
- Les dates et heures de remises sont à comprendre au sens strict. Tout retard sera sanctionné. Un retard de 24 heures sera toléré une fois sur l'ensemble des quatre parties, avec une pénalité de 20 % pour la partie concernée. Si un étudiant remet pour la deuxième fois en retard, la partie sera annulée. Une remise avec plus de 24 heures de retard annulera également la partie concernée ;
- Après chaque partie, un correctif sera proposé. Il vous est loisible de continuer sur base de ce correctif, mais nous conseillons aux étudiants de continuer avec leur travail en tenant compte des remarques qui auront été adressées ;
- En plus des quatre parties à remettre, les étudiants devront préparer une « démo » en salle machine de cinq minutes par personne, durant la semaine 19 (entre le 21 et le 26 mars). Par ailleurs, une défense orale de dix minutes par personne sera organisée pendant la semaine cours 20 (du 4 au 9 avril) ;
- En dehors des cas où l'énoncé le spécifie explicitement, l'utilisation de librairies (traitement d'image, cryptographie, manipulation de matrices, etc.) n'est pas autorisée ;
- L'ensemble du projet sera à réaliser en Python 2 ;
- Les parties 3 et 4 de ce projet devront *impérativement* pouvoir tourner sur la machine virtuelle fournie pour VirtualBox (voir l'énoncé de la partie 3), sans modification (changement de configuration, installation de logiciels. . .).

4 Planning

Le projet sera organisé autour de quatre grandes étapes.

4.1 Partie 1 : Manipulation d'une image PPM

L'ensemble de ce projet se basera sur le format d'image PPM (Portable Pixel Map). Ce format, standard, peut être produit et visualisé à l'aide d'un logiciel comme « Gimp », logiciel *open source* de manipulation d'image (`www.gimp.org`). Dans cette première partie de projet, on demande de réaliser quatre opérations sur de telles images.

Il existe deux variantes du format PPM : la binaire (identificateur P6) et la brute, parfois appelée « ASCII » (identificateur P3). Nous utiliserons dans ce projet la version brute, car les valeurs y sont représentées par des décimaux en ASCII. Voici un exemple d'une image en couleur de 4 colonnes et 2 lignes.

```
P3
4 2
29
0 0 0     1 2 3       2 2 2       3 6 4
4 4 4     10 10 10    20 29 20    20 20 20
```

Le fichier commence avec 4 données séparées par un séparateur (retour à la ligne, espace ou tabulation).

- Identificateur du fichier ; dans le cas de PPM brute, on a toujours P3 ;
- Nombre de colonnes de l'image ;
- Nombre de lignes de l'image ;
- Valeur maximale dans les matrices des intensités.

L'image est ensuite représentée par une matrice de pixels et chaque pixel de couleur est composé de trois valeurs RVB (Rouge-Vert-Bleu, ou RGB en anglais). Par exemple, les 5^e, 6^e et 7^e valeurs du fichier sont les intensités RVB du premier pixel. Dans l'exemple, il s'agit d'un pixel noir (toutes les composantes à 0). Le parcours commence du coin supérieur gauche, et il visite l'image ligne par ligne, de haut en bas. Votre opération pour la lecture du fichier obtiendra comme résultat une matrice de triplets. En $M[i][j]$, on obtiendra donc le triplet représentant le pixel situé en ligne i et colonne j, comprenant les trois coordonnées RVB.

Après la lecture, vous pourrez manipuler des matrices. La deuxième étape est l'implémentation des opérations suivantes :

- ROTG : Rotation de 90° vers la gauche (sens anti-horloger) ;
- ROTD : Rotation de 90° vers la droite (sens horloger) ;
- SYMV : Symétrie verticale (la gauche devient la droite) ;
- SYMH : Symétrie horizontale (le haut devient le bas).

Une fois une de ces opérations appliquée, on créera un fichier image, également au format PPM, contenant l'image modifiée.

Pour mettre toutes ces fonctionnalités ensemble, on vous demande d'écrire un script en Python qui prendra comme paramètre une commande et les noms de fichiers d'entrée et de sortie. Par exemple, pour faire une rotation vers la droite :

```
monProjet.py ROTD /chemin/imageIn.ppm /chemin/imageOut.ppm
```

4.1.1 Consignes

Remise : lundi 22 novembre à 13 heures.

À remettre :

- Au secrétariat étudiants : une version imprimée de votre (vos) script(s) Python ;
- Par courriel à infof106@lit.ulb.ac.be : ce(s) même(s) script(s).

4.2 Partie 2 : Filtres de floutage et stéganographiques

On rajoute au programme de la première partie trois filtres.

4.2.1 Floutage

Ce filtre réalisera un *floutage* d'une intensité donnée. Une façon simple de procéder (et que nous vous demandons d'implémenter) consiste simplement à remplacer la valeur de chaque *pixel* par la moyenne de sa valeur et de celles de tous les pixels qui l'entourent. Chaque pixel étant décrit par trois composantes RVB, il faut appliquer cette opération sur chaque composante (on floutera ainsi séparément les composantes rouge, verte et bleue).

Ainsi, pour un pixel de coordonnées (i, j) dans une image de hauteur h et de longueur ℓ (avec $i \neq 0 \neq j$, $i \neq h-1$ et $j \neq \ell - 1$), on remplacera par exemple sa valeur rouge (c'est-à-dire $M[i][j][0]$) par :

$$\frac{\sum_{k=i-1}^{i+1} \sum_{\ell=j-1}^{j+1} M[k][\ell][0]}{9}$$

Pour les pixels dans les coins de l'image, ou sur ses bords, il y aura lieu de modifier la formule afin de ne prendre en compte que des coordonnées valides (veillez bien à toujours diviser par le nombre de pixels concernés ; si vous divisez par 9 les valeurs dans un coin où il n'y a que 4 pixels à considérer, vous assombrirez le pixel concerné). L'intensité du floutage (par exemple 10) correspond au nombre de fois que cette opération est appliquée sur toute l'image.

Voici un exemple de commande qui réaliserait un floutage d'intensité 10 :

```
monProjet.py FLOU 10 /chemin/imageIn.ppm /chemin/imageOut.ppm
```

4.2.2 Encodage stéganographique

Ce filtre cache un message dans une image en utilisant l'algorithme stéganographique suivant. Partant du principe que chaque lettre du message à cacher est un caractère prenant 8 bits, que le format de l'image est une image de type PPM (P3) représentant chaque pixel à l'aide de 3 canaux de couleurs (rouge, vert, bleu) chacune codée sur 8 bits ; l'idée est alors de cacher un caractère (8 bits) dans un triplet de pixel ($= 3 * 3 = 9$ octets). Nous cacherons donc un bit d'information par octet d'image. Le dernier octet du triplet de pixel sera alors réservé pour indiquer si il reste des informations à encoder (ce qui sera très utile pour le système de déchiffrement). Les bits d'informations seront cachés du bit de poids le plus important (dans la représentation ASCII du caractère sur 8 bits) au bit le moins important, en progressant dans les canaux de couleurs ; si le bit d'information vaut 0, alors la valeur du canal doit être paire, sinon elle doit être impaire. Le 9^{e} et dernier canal de couleur de chaque triplet de pixels sera ajusté pour indiquer si il reste des caractères cachés dans l'image, sa valeur sera paire si c'est le cas, impaire sinon. Un message de n caractères sera ainsi caché dans les $3*n$ premiers pixels de l'image. Pour transformer une valeur paire en une valeur impaire, vous l'incrémenterez d'une unité. Pour transformer une valeur impaire en une valeur paire, vous la décrémenterez d'une unité.

[v 2.0] Dans le cas où l'intensité maximale est un nombre pair et un pixel d'intensité maximale doit changer de parité, il sera décrémenté.

Par exemple, si nous devions cacher la lettre « a » — dont le code ASCII est 97, soit 0110 0001 en binaire — dans le début de l'image de l'exemple de la première partie du projet, nous procéderions de la façon suivante. Au lieu de garder (0, 0, 0, 1, 2, 3, 2, 2, 2), il faudra que la première valeur soit paire (cf le premier bit de la représentation binaire du code ASCII de « a »), la 2^{e} et la 3^{e} impaire, les quatre suivantes paires, et l'avant-dernière impaire. S'il reste d'autres caractères à encoder, la dernière valeur sera paire. Les 9 premières valeurs seront donc (0, 1, 1, 0, 2, 2, 2, 3, 2). Le tableau ci-dessous illustre cet exemple.

9 premiers octets	0	0	0	1	2	3	2	2	2
Code binaire de « a »	0	1	1	0	0	0	0	1	(fin)
Résultat	0	1	1	0	2	2	2	3	2

Voici un exemple d'utilisation du script. L'utilisateur se verra demander le message à encoder.

```
monProjet.py ENC /chemin/imageIn.ppm /chemin/imageOut.ppm
> Message à cacher ? : blah blah blah
```

4.2.3 Décodage stéganographique

Ce filtre réalisera l'opération inverse : à partir d'une image cachant un message, il en extraira le contenu et l'affichera. Il s'utilisera de la sorte :

```
monProjet.py DEC /chemin/imageIn.ppm
> Message : blah blah blah
```

4.2.4 Rapport

En plus du code, on demande un rapport de 4 à 5 pages (page de garde et table de matière non comprises). Plus en détails, ce rapport doit comprendre :
- Une page de garde qui reprend vos coordonnées, la date, le titre, etc. ;
- La table de matière ;
- Une introduction et une conclusion ;
- Des sections, comprenant ce qui a été fait dans le projet jusqu'à ce point (la partie 1 comprise) ;
- Des références si nécessaire.

Le rapport doit détailler les difficultés rencontrées, l'analyse et les solutions proposées, ainsi que une bonne explication des algorithmes (pseudo-code avec des exemples, et pas le code source). Il doit être clair et compréhensible pour un non-informaticien, sans trop d'informations ni trop peu. Toutes les étapes doivent être détaillées et présentées dans une séquence logique. Expliquez toutes les notions et les terminologies que vous introduisez.

Le rapport peut être écrit avec des logiciels de traitement de texte comme ceux de Microsoft Office ou OpenOffice, mais il est obligatoire d'utiliser les outils de gestion automatique des styles, de la table de matière et de numérotation de sections. Nous vous conseillons cependant d'utiliser le système LaTeX, très puissant pour une mise en page de qualité.

On rappelle qu'une bonne orthographe sera exigée.

En annexe, un document donne des conseils sur la rédaction d'un bon rapport.

4.2.5 Consignes

Remise : lundi 13 décembre à 13 heures.
À remettre :
- Au secrétariat étudiants : une version imprimée de votre script et du rapport ;
- Par courriel à infof106@lit.ulb.ac.be : ce même script, et le rapport, dans son format source (.doc, .odt, ou .tex) et sa version PDF.

4.3 Partie 3 : Serveur Web

4.3.1 Intégration

Dans cette partie, vous allez construire un service Web basé sur les différentes transformations que vous avez implémentées. On attend une page Web similaire à la suivante celle présentée sur la Figure 1 :

Cliquer sur le bouton « OK » aura pour effet de rediriger l'utilisateur vers une nouvelle page, où seront affichées :
- Une image contenant le message caché dans le cas de l'encodage ;
- Le message caché dans le cas du décodage ;
- Deux images, avant et après transformation, dans le cas des filtres.

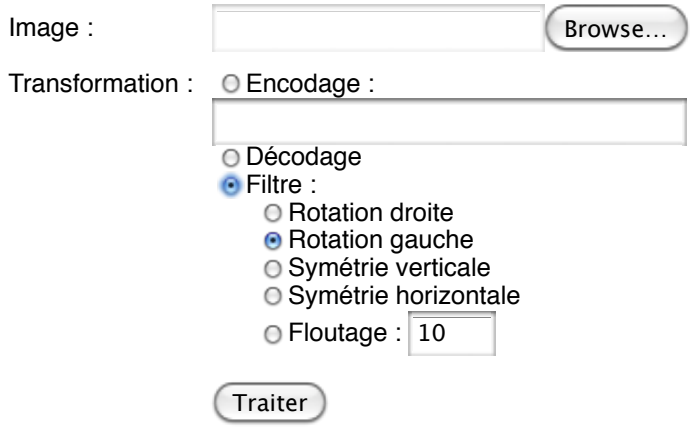

FIG. 1 – Exemple de page Web intégrant les outils développés

Il existe deux moyens d'utiliser Python dans un service Web : CGI et mod_python. Dans l'image VitualBox fournie (voir ci-dessous), vous avez tous les utilitaires nécessaires pour travailler avec le module mod_python. Celui-ci est basé sur l'interception de handler. Un handler n'est qu'une fonction avec un nom réservé, par exemple **index()**. Donc, pour afficher une page Web, on peut implémenter la fonction **index()** et renvoyer une chaîne de caractères contenant la source HTML de la page Web comme résultat. Vous pouvez trouver plus d'information dans : `http://webpython.codepoint.net/mod_python_tutorial`.

Dans l'image VitualBox fournie, vous trouvez un exemple de la génération et l'interaction entre un page Web en HTML et le code Python.

Notez que le format PPM n'est en général pas affichable dans les navigateurs. Pour rendre ces images affichables, il vous faudra les convertir vers un format tel que le JPEG ou le PNG. Vous aurez pour ce faire la possibilité d'utiliser soit la libraire Python Image Library, soit l'outil `convert` de ImageMagick en ligne de commande. Une petite recherche sur votre moteur de recherche préféré vous permettra d'en savoir plus.

4.3.2 Machine virtuelle *VirtualBox*

Par soucis de simplicité (pour vous éviter d'installer votre propre serveur Web), de sécurité (pour éviter les catastrophes en cas de fausse manœuvre) et d'uniformité (pour être sûr que tout le monde travaille dans le même environnement), nous vous fournissons une *machine virtuelle* (sous VirtualBox) avec toute la configuration nécessaire, et un petit exemple de script recevant des données d'un formulaire Web et affichant quelque chose de simple.

Une machine virtuelle est un logiciel qui permet de faire tourner un système d'exploitation au sein d'un autre. Dans le cas qui nous concerne, nous allons faire tourner le système Linux PepperMint sur votre ordinateur, sans toucher au système déjà présent.

Les parties 3 et 4 de ce projet devront *impérativement* pouvoir tourner sur la machine fournie, sans modification (changement de configuration, installation de logiciels. . .).

Une présentation sera faite de VirtualBox et de la machine virtuelle lors d'une séance en salle machine d'INFO-F101 et une communication reprenant des liens vers des tutoriaux et des informations complémentaires seront disponibles sur la page Web. Cette machine virtuelle est accessible par le réseau des salles du 4e étage du bâtiment NO. Vous trouverez les fichiers à importer dans VirtualBox, sous Linux dans le répertoire /home/samba/pub/INFOF106 et sous Windows dans le partage « pub\INFOF106 » (P :\INFOF106). Dès lors, trois choix s'offrent à vous[1] :

[1]Si vous ne disposez ni d'une clef USB de taille suffisante ni d'un ordinateur personnel, une quatrième option vous sera proposée lors de la démonstration.

1. Récupérer les fichiers sur une clef USB et les importer dans VirtualBox sur votre machine personnelle.

2. Importer les fichiers dans VirtualBox en indiquant comme répertoire de destination votre clef USB. Vous pourrez ainsi travailler avec votre machine virtuelle personnelle tant sur votre machine personnelle que dans les salles du NO4.

3. Importer les fichiers dans VirtualBox en indiquant comme répertoire de destination le dossier temporaire.

Les deux premiers choix permettent de conserver la machine virtuelle, les modifications que vous lui aurez apportées ainsi que les fichiers qu'elle contiendra. Le dernier choix vous contraint à réimporter la machine virtuelle dans VirtualBox à chaque redémarrage de la machine et ne garantit aucunement la sécurité ni la confidentialité des données contenues dans la machine virtuelle.

Pour importer une machine virtuelle dans VirtualBox, lancer VirtualBox et cliquez sur « Fichier > Importer application virtuelle. . . » et suivez la procédure.

Pour les choix 2 et 3, vous devez spécifier soit votre clef soit le dossier temporaire comme destination de l'application virtuelle. Pour ce faire, passez par « Fichier > Préférences. . . » et indiquez tant pour le disque dur que pour le dossier par défaut des machines le dossier correspondant à votre choix (voir Figure 2).

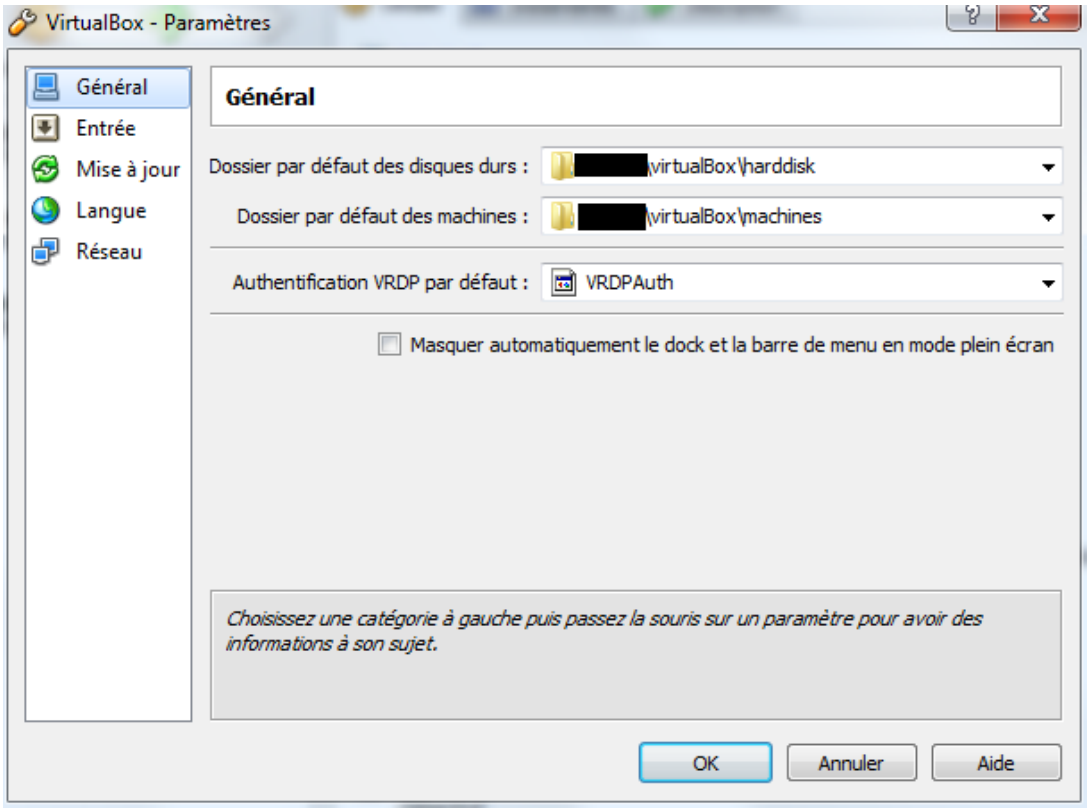

FIG. 2 – Configuration de VirtualBox

Le nom d'utilisateur et le mot de passe de la machine mise à disposition sont respectivement « utilisateur » et « utilisateur ». Vous pouvez changer ce mot de passe via le sous menu préférence du bouton menu ou via la commande `passwd` dans un terminal.

Pour plus d'information sur VirtualBox, des liens additionnels sont fournis sur la page Web du cours vers divers sites, vidéos, documentations et tutoriaux.

4.3.3 Introduction au langage de balisage HTML

Il vous sera demandé de construire des pages Web à l'aide du langage de description HTML (*HyperText Markup Language*) et plus particulièrement sa variante « XHTML 1.0 Transitional ». Un fichier HTML (d'extension typique `.html`) est un document texte qui a pour objectif de décrire le *contenu* d'une page Web (texte, formulaires, images, etc.) tout en évitant de comprendre des informations sur la manière dont ce contenu est *présenté* ; ce dernier aspect sera plutôt confié à un langage de présentation dénommé CSS (*Cascading Style Sheets*) qui sera décrit à la section suivante. Cette séparation entre contenu et présentation offre plusieurs avantages, dont une plus grande simplicité pour changer l'apparence d'une page ou d'un site Web entier en modifiant un seul fichier.

Un fichier HTML utilisant la variante « XHTML 1.0 Transitional » débute par l'en-tête suivant :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

La suite d'un fichier HTML est composée d'un ensemble de *balises* décrivant le contenu du document. Il existe trois catégories de balises :

- Les balises *ouvrantes* : `<html>`, `<h1>`, `<p>`, etc.
- Les balises *fermantes* : `</html>`, `</h1>`, `</p>`, etc.
- Les balises *autofermantes* : `<br/>`, `<hr/>`, etc.

Une balise ouvrante doit être suivie, plus loin dans le document, d'une balise fermante correspondante. Comme leur nom l'indique, les balises autofermantes n'ont pas de balise fermante correspondante. Une première règle à respecter scrupuleusement en HTML est de respecter le « parenthésage » des balises :

- `<div><p>Blabla</p></div>` est **correct** : une balise `<p>` est introduite au sein d'une balise `<div>`.
- `<div><p>Blabla</div></p>` est **incorrect** : on ne ferme pas la bonne balise en premier.

Quelques règles supplémentaires sont à suivre :

- Le document (hormis la déclaration `DOCTYPE` qui commence le fichier) doit être compris dans une balise `<html>`, aussi appelée balise *racine*. Dans le format « XHTML 1.0 Transitional », il est imposé de plus que cette balise porte un *attribut* `xmlns` avec une valeur particulière (voir exemple ci-dessous) pour que le document soit considéré valide.
- Des informations concernant le document (tel le titre à afficher dans le navigateur, l'encodage utilisé par le fichier, la feuille de style CSS à employer) doivent être comprises dans une balise `<head>` (voir exemple ultérieur).
- Les informations contenues dans le corps du document (et donc affichés dans le navigateur) doivent être comprises dans une balise `<body>`.

Un exemple de document HTML ressemble à ceci :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Ma première page Web</title>
    <meta http-equiv="Content-Type" content="text/html; charset: utf-8"/>
    <link rel="stylesheet" type="text/css" href="style.css"/>
  </head>
  <body>
    <h1>Hello, world!</h1>
    <p>Ceci est un paragraphe.</p>
  </body>
</html>
```

Nous retrouvons la déclaration du type de document (`DOCTYPE`) suivi d'une balise `<html>` qui englobe le reste du document, dans lequel on distingue d'abord une balise `<head>` dans laquelle on précise :

- Le titre du document via le contenu de la balise `<title>` ;

– L'encodage du document via la balise <meta> (dans l'exemple, l'encodage est utf-8, qui peut être remplacé par exemple par iso-8859-1 selon l'encodage du fichier) ;
– La feuille de style à utiliser pour cette page via la balise <link> : on précise ici qu'il faut utiliser un fichier nommé style.css.

Cet en-tête est suivi du corps du document, contenu dans la balise <body>, où un titre de niveau 1 (<h1>) est suivi d'un paragraphe.

HTML propose de nombreuses balises et nous vous invitons à expérimenter avec celles-ci. En particulier, vous devrez manipuler des *formulaires* (balise <form>). Le Web est rempli d'informations au sujet des diverses balises HTML, nous vous invitons à faire quelques recherches ou à vous tourner vers l'équipe du cours pour avoir des informations.

4.3.4 Introduction aux feuilles de style CSS

Une fois un fichier HTML créé, on peut utiliser le langage de présentation CSS pour dire au navigateur comment présenter la page. Dans l'exemple précédent, nous avons donné un exemple de document HTML précisant qu'il faut utiliser une feuille de style (nom qu'on donne typiquement aux fichiers CSS) répondant au nom de style.css qui pourrait ressembler à ceci :

```
body
{
  background-color: red;
  color: yellow;
}

p
{
  color: rgb(0, 0, 255);
  font-family: monospace;
}
```

Ce fichier décrit une série de règles de présentation (ou propriétés) à appliquer sur les balises correspondantes en HTML. Dans l'exemple donné, il est demandé que la balise <body> (en d'autres termes, l'ensemble du contenu du document HTML) soit affiché avec une couleur de fond (propriété background-color) en rouge (red) et que la couleur du texte (propriété color) soit jaune (yellow). Ensuite, une règle de présentation est définie pour les paragraphes <p> où il est demandé que la couleur du texte soit en bleu (ce qui correspond aux coordonnées RVB $(0, 0, 255)$) et affichée avec une police *monospace*. Les accolades après chaque nom de balise ainsi que les points-virgules au bout de chaque ligne sont nécessaires.

La notion *d'héritage des propriétés* est importante en CSS : dans l'exemple ci-dessus, nous avons demandé que les paragraphes <p> soient en bleu ; or, les paragraphes sont compris dans la balise <body> pour laquelle il est demandé que les textes soient en jaune : quel style sera alors appliqué ? Quand le navigateur se retrouvera confronté à une balise, par exemple <p>, il regardera d'abord dans quelles balises elle se trouve (dans l'exemple que nous avons donné, la seule balise <p> a pour « grand-père » la balise <html> et pour « père » la balise <body>). Il appliquera ensuite successivement les propriétés depuis l'ancêtre le plus lointain jusqu'à la balise en question. Dans cet exemple, il verra d'abord qu'il n'y a aucun style particulier défini pour <html>, puis il verra que la balise <body> demande un fond rouge et un texte jaune, puis finalement que le paragraphe <p> demande un texte bleu et une police *monospace*. Au final, le paragraphe de l'exemple se retrouve donc avec un fond rouge, une couleur de texte bleue et une police *monospace*.

Autre exemple, nous verrions que le titre de niveau 1 (balise <h1>) de notre fichier HTML serait, quant à lui, affiché en jaune ; en effet, cette balise est comprise dans la balise <body> pour laquelle un style a été défini, et elle hérite donc de ses propriétés.

Enfin, notons qu'en absence de feuille de style CSS (ou simplement si aucune propriété n'est donnée), le navigateur appliquera une feuille de style par défaut (typiquement, fond blanc, texte noir, polices plus grandes pour les titres, etc.).

Comme pour le HTML, nous vous invitons à rechercher des informations sur le Web au sujet des diverses propriétés qui peuvent être données aux balises ; nous vous **demandons** par ailleurs d'illustrer

que vous avez bien compris les principes du HTML et du CSS en égayant comme se peut votre page Web (polices différentes, utilisation de couleurs, d'images, etc.).

Enfin, nous n'avons ici gratté que la surface du HTML et du CSS ; nous vous encourageons à vous informer sur les autres possibilités de ces langages (il peut par exemple être intéressant d'utiliser la notion de *classes* de CSS).

4.3.5 Validation HTML et CSS

Vous pouvez vérifier la validité de votre code HTML et CSS grâce aux outils gratuitement fournis en ligne par le W3C (*World Wide Web Consortium*) qui sont par ailleurs les éditeurs des normes HTML et CSS. L'adresse suivante vous permet par exemple d'envoyer un fichier HTML et de recevoir un rapport sur sa validité avec une explication des erreurs éventuelles :

`http://validator.w3.org/#validate_by_upload`

Un autre tel validateur existe pour les feuilles de style CSS :

`http://jigsaw.w3.org/css-validator/#validate_by_upload`

Nous vous demandons d'assurer que vos pages HTML et vos styles CSS sont bien conformes selon ces validateurs.

4.3.6 Consignes

Remise : lundi 21 février à 13 heures.
À remettre :
- Au secrétariat étudiants : une version imprimée de votre (vos) script(s) Python, des fichiers .html et .css ;
- Par courriel à infof106@lit.ulb.ac.be : les mêmes fichiers.

4.4 Partie 4 : Un filtre complexe, rapport final et présentation

Pour cette partie on vous demande d'implémenter un filtre de détection de contours. Pour ce faire, nous allons nous concentrer sur des images en niveaux de gris. Pour avoir un filtre qui peut détecter des contours sur une image en niveaux de gris, il faut implémenter les parties suivantes :
- convertir une image de RVB en niveaux de gris et la stocker dans une matrice d'entiers ;
- écrire un filtre qui détermine le gradient de l'intensité de chaque pixel ;
- écrire un filtre qui détermine le pixel maximal dans le voisinage ;
- écrire un filtre qui supprime toutes les valeurs plus petites qu'une valeur donnée.

Pour le rapport, il faudra montrer les effets de ces quatre parties sur une image de votre choix.

4.4.1 Conversion en gris

Une image en couleur RVB contient trois matrices, une pour chaque couleur : rouge (R), vert (V) et bleu (B). Afin d'avoir une seule matrice en niveaux de gris, il faut calculer la valeur en gris $\mathcal{I}$ pour chaque pixel (i, j) :

$$\mathcal{I}(i, j) = round(0.3 * R(i, j) + 0.59 * V(i, j) + 0.11 * B(i, j)).$$

L'image en niveaux de gris peut être stockée comme une matrice d'entiers. Pour visualiser l'effet de la transformation, il suffit de créer une image en couleur dans laquelle chaque coordonnée de $\mathcal{I}$ est répétée pour chaque canal (rouge, vert, bleu).

4.4.2 Détermination du gradient

Dans cette partie, on recherche essentiellement les contours où l'intensité en niveaux de gris de l'image change le plus. Ces zones sont déterminées par les gradients de l'image. La première étape

FIG. 3 – Image représentant $\mathcal{I}$

consiste à estimer le gradient dans les directions de x et y en se servant des matrices suivantes :

$$K_{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$K_{G_y} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Considérons d'abord le gradient dans la direction x. Pour chaque pixel (i, j), on multipliera chaque élément de la sous-matrice allant de $(i-1, j-1)$ à $(i+1, j+1)$ par l'élément correspondant dans K_{G_x}, puis on sommera chaque produit. Le nombre obtenu, dénommé $G_x(i, j)$, est le gradient dans la direction x du pixel (i, j).

Un opération similaire est effectuée pour le calcul de $G_y(i, j)$, le gradient dans la direction y du pixel (i, j), à l'aide de la matrice K_{G_y}.

On considère que le gradient est nul sur les bords de l'image.

On définit ensuite

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

FIG. 4 – Image représentant $\mathcal{G}$

4.4.3 Suppression non maximale

Le but de cette étape est d'isoler les contours de l'image. En gros, cela se fait en conservant tous les maxima locaux de l'image gradient et en supprimant tout le reste. On va créer une matrice C qui vaudra $\mathcal{G}(i, j)$ si (i, j) se trouve sur un contour de l'image, 0 sinon. L'algorithme est, pour chaque pixel de l'image gradient (excepté les bords) :

1. $\theta = arctan\dfrac{|G_y(i, j)|}{|G_x(i, j)|}$ arrondi aux 45 degrés le plus proche
2. Comparer le pixel courant $\mathcal{G}(i, j)$ avec le pixel dans les directions positives et négatives du gradient. Plus précisément :
   - Si $\theta = 45°$ ou $225°$: on compare $\mathcal{G}(i, j)$ avec $\mathcal{G}(i+1, j-1)$ et $\mathcal{G}(i-1, j+1)$;
   - Si $\theta = 135°$ ou $315°$: on compare $\mathcal{G}(i, j)$ avec $\mathcal{G}(i-1, j-1)$ et $\mathcal{G}(i+1, j+1)$;
   - Si $\theta = 90°$ ou $170°$: on compare $\mathcal{G}(i, j)$ avec $\mathcal{G}(i-1, j)$ et $\mathcal{G}(i+1, j)$;
   - Si $\theta = 0°$ ou $180°$: on compare $\mathcal{G}(i, j)$ avec $\mathcal{G}(i, j-1)$ et $\mathcal{G}(i, j+1)$;

FIG. 5 – Image représentant C

3. Si le pixel courant $\mathcal{G}(i, j)$ est plus grand que ses deux valeurs voisines, $C(i, j) = \mathcal{G}(i, j)$. Sinon, $C(i, j) = 0$

4.4.4 *Thresholding*

Pour obtenir des contours plus exacts, il faut supprimer toutes les valeurs inférieures à un seuil donné τ (déterminé par l'utilisateur). Écrivez une fonction qui reçoit un paramètre dans l'intervalle $[0, 255]$ et retourne une image dans laquelle toutes les valeurs plus petites que ce paramètre sont égales à zéro.

On définit donc :

$$\mathcal{T}(i, j) = \begin{cases} \mathcal{G}(i, j) & \text{si } C(i, j) \geq \tau \\ 0 & \text{sinon} \end{cases}$$

FIG. 6 – Image représentant $\mathcal{T}$

Le résultat final du filtre sera une image de type PPM P3, ne contenant que des pixels en niveaux de gris, correspondant à la matrice $\mathcal{T}$.

4.4.5 Rapport

En plus du code, on demande un rapport de 9 à 10 pages (page de garde et table des matières non comprises), avec au minimum 4000 mots. Comme pour le premier, ce rapport doit comprendre :

- Une page de garde qui reprend vos coordonnées, la date, le titre, etc. ;
- La table des matières ;
- Une introduction et une conclusion ;
- Des sections, comprenant ce qui a été fait dans le projet, donc les parties 1, 2, 3 et 4 ;
- Des références si nécessaire.

Il peut aussi comprendre des annexes (facultatif) si on pense qu'elles sont nécessaires pour des explications supplémentaires (comme des parties de code). Les annexes ne sont pas toujours lues, donc les informations indispensables à la compréhension du projet ne doivent pas s'y trouver. Les annexes ne comptent pas non plus dans les 9–10 pages de rapport.

Les instructions données pour le premier rapport s'appliquent également ici.

Vous pouvez reprendre des parties du premier rapport, mais faites attention à la continuité de l'ensemble du rapport. Il ne suffit pas de rajouter les parties 3 et 4 au document précédent, il faut retravailler le rapport dans son ensemble. Si dans les parties 1 et 2 il y avait des erreurs de contenu ou autre (orthographe, typographie. . .), elles doivent être corrigées.

Dans l'introduction et la conclusion, tout le projet doit être présenté avec ses objectifs et résultats.

Dans les sections, tout ce que vous avez fait doit être expliqué : les algorithmes, les fonctions Python, les pages Web et l'utilisation de la machine virtuelle. Vous devez décrire le projet dans son ensemble, pas partie par partie, donc n'utilisez pas comme noms de section « Partie 1 », « Partie 2 », etc.

On rappelle qu'un non-informaticien doit pouvoir comprendre au moins 90–95 % du rapport.

Pour des conseils sur la rédaction d'un bon rapport, nous vous renvoyons à nouveau vers le document en annexe.

4.5 Présentation

À coté d'un rapport écrit qui explique un projet en détail, il est souvent demandé, dans les travaux scientifiques, une petite présentation orale qui résume le projet. On vous demande donc une présentation de 10 minutes (pas plus !), où, à l'aide de transparents, vous nous expliquez votre projet.

La présentation ne doit pas aller dans les détails, mais expliquer l'idée générale et les objectifs du projet, les méthodes utilisées et les résultats obtenus. Des exemples et des figures aident beaucoup dans un exposé oral.

Pour des conseils sur la réalisation d'une bonne présentation, voir de nouveau le document en annexe. On remarque juste que pour un exposé de 10 minutes, il faudrait compter 10–12 transparents maximum (la règle empirique est 1 transparent par minute).

Le calendrier des démos et défenses apparaîtra en mars. Les démos auront lieu pendant la semaine de cours 19 (21–26 mars), les défenses pendant la semaine de cours 20 (4–9 avril).

4.5.1 Consignes

Remise : lundi 21 mars à 13 heures.
À remettre :
- Au secrétariat étudiants : une version imprimée de tous les scripts et fichiers HTML/CSS (code source), et du rapport final ;
- Par courriel à infof106@lit.ulb.ac.be : tous les scripts et fichiers concernés. Pour le rapport, on attend le fichier source (format source et PDF).

5 Organisation

Titulaire : Vandy Berten – vandy.berten@ulb.ac.be – N8.111
Assistants :
- Jérôme Dossogne – jdossogn@ulb.ac.be – N8.214
- Vincent Ho – vincent.ho@ulb.ac.be – N3.202
- Markus Lindström – mlindstr@ulb.ac.be – N8.211
- Catharina Olsen – catharina.olsen@ulb.ac.be – O8.213
- Alessia Violin – aviolin@ulb.ac.be – N3.204