

**Relatório Trabalho Prático 2 da UC**

**Processamento de Imagem e Biometria**

Grupo 14:

Miguel Lopes Nº40624

Miguel Pereira Nº40625

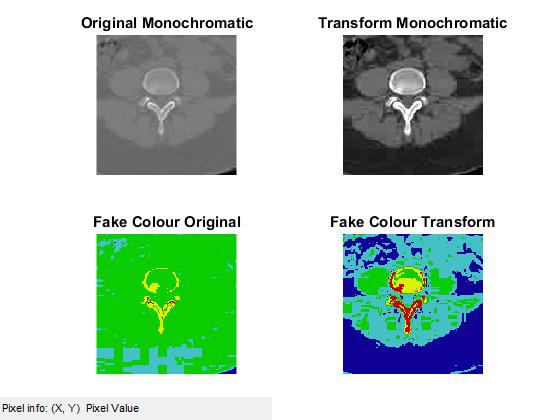
Docente: Artur Ferreira

**2016/2017 – SV**

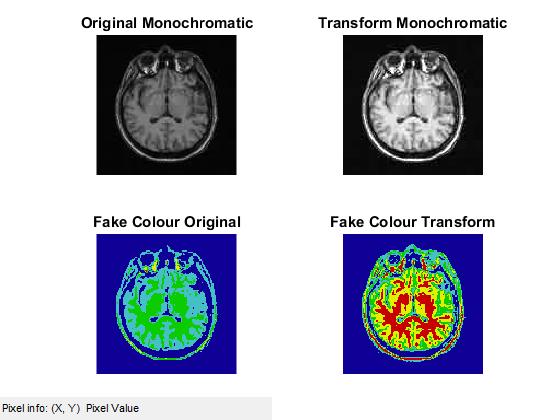
**Exercício 1.**

**Alínea b)**

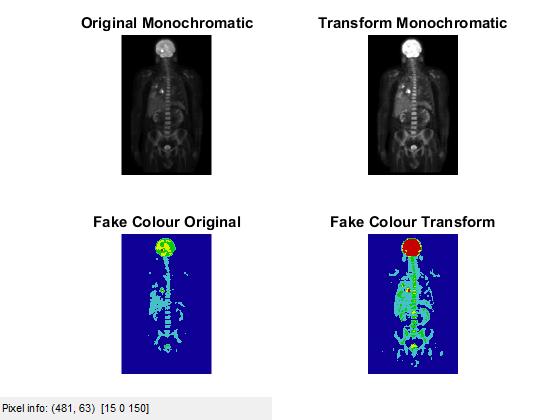
Resultados obtidos para a imagem ‘CT1.jpg’ original e transformada pela função *medical\_image\_enhancement.m*:



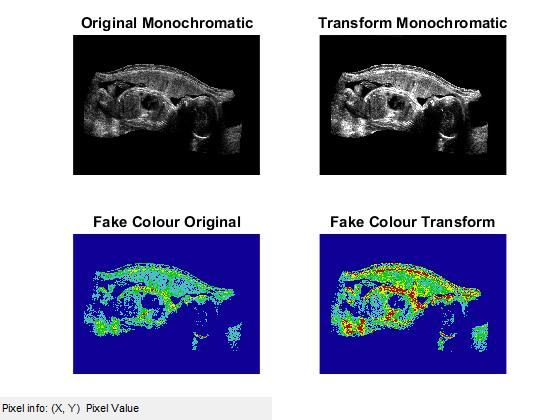
Resultados obtidos para a imagem ‘MR1.jpg’ original e transformada pela função *medical\_image\_enhancement.m*:



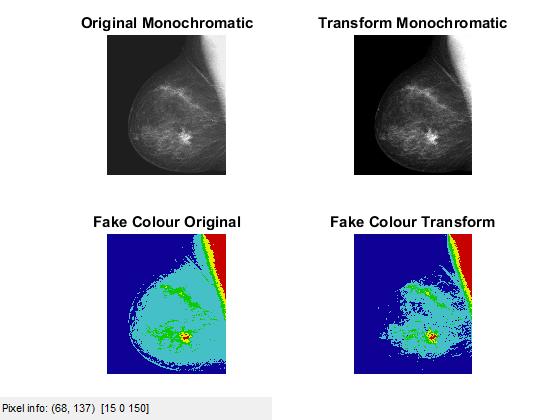
Resultados obtidos para a imagem ‘PET1.tif’ original e transformada pela função *medical\_image\_enhancement.m*:



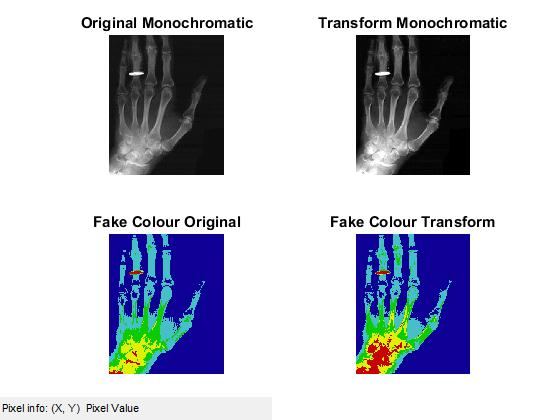
Resultados obtidos para a imagem ‘US1.tif’ original e transformada pela função *medical\_image\_enhancement.m*:

****

Resultados obtidos para a imagem ‘XRay1.tif’ original e transformada pela função *medical\_image\_enhancement.m*:

****

Resultados obtidos para a imagem ‘XRay2.tif’ original e transformada pela função *medical\_image\_enhancement.m*:

****

**Alínea c)**

A melhoria referente à coloração das imagens antes de depois da transformação pela função de *medical\_image\_enhancement.m*, deve-se à correção do histograma de intensidades. Isto é, devido às imagens médicas serem (geralmente) muito escuras ou muito claras, acontece que as intensidades estão pouco dispersas (pouca variação), o que resulta numa coloração má pois os valores de intensidade não diferem muito, ficando a imagem com muitas zonas com a mesma cor.

Depois da melhoria (ajuste de contraste), como passa a existir maior variação de intensidades, já é possível definir bem as cores para cada nível diferente de intensidade.

Usámos a técnica de *Intensity Slicing* para a coloração.

**Exercício 2.**

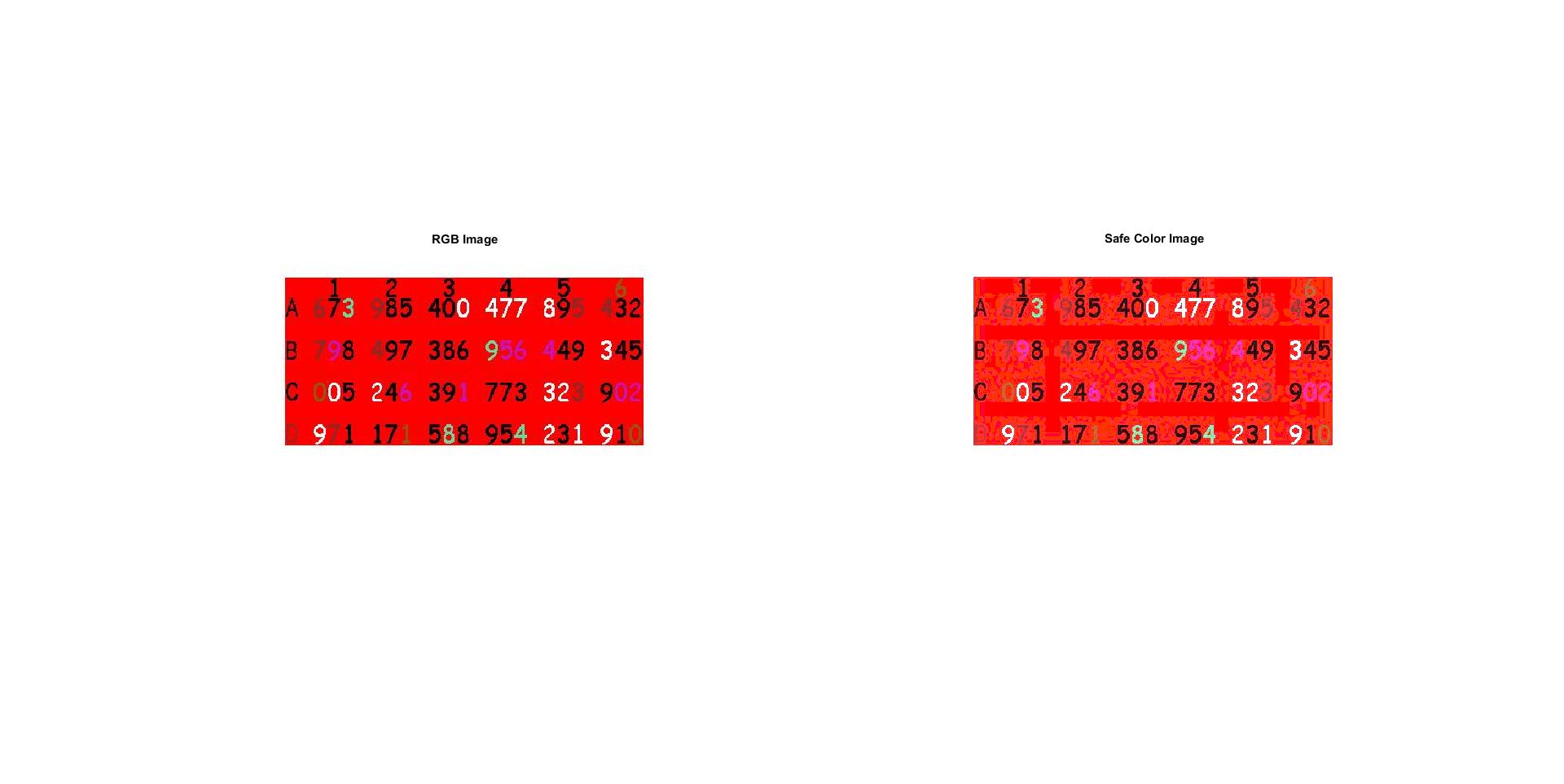
**Alínea a)**

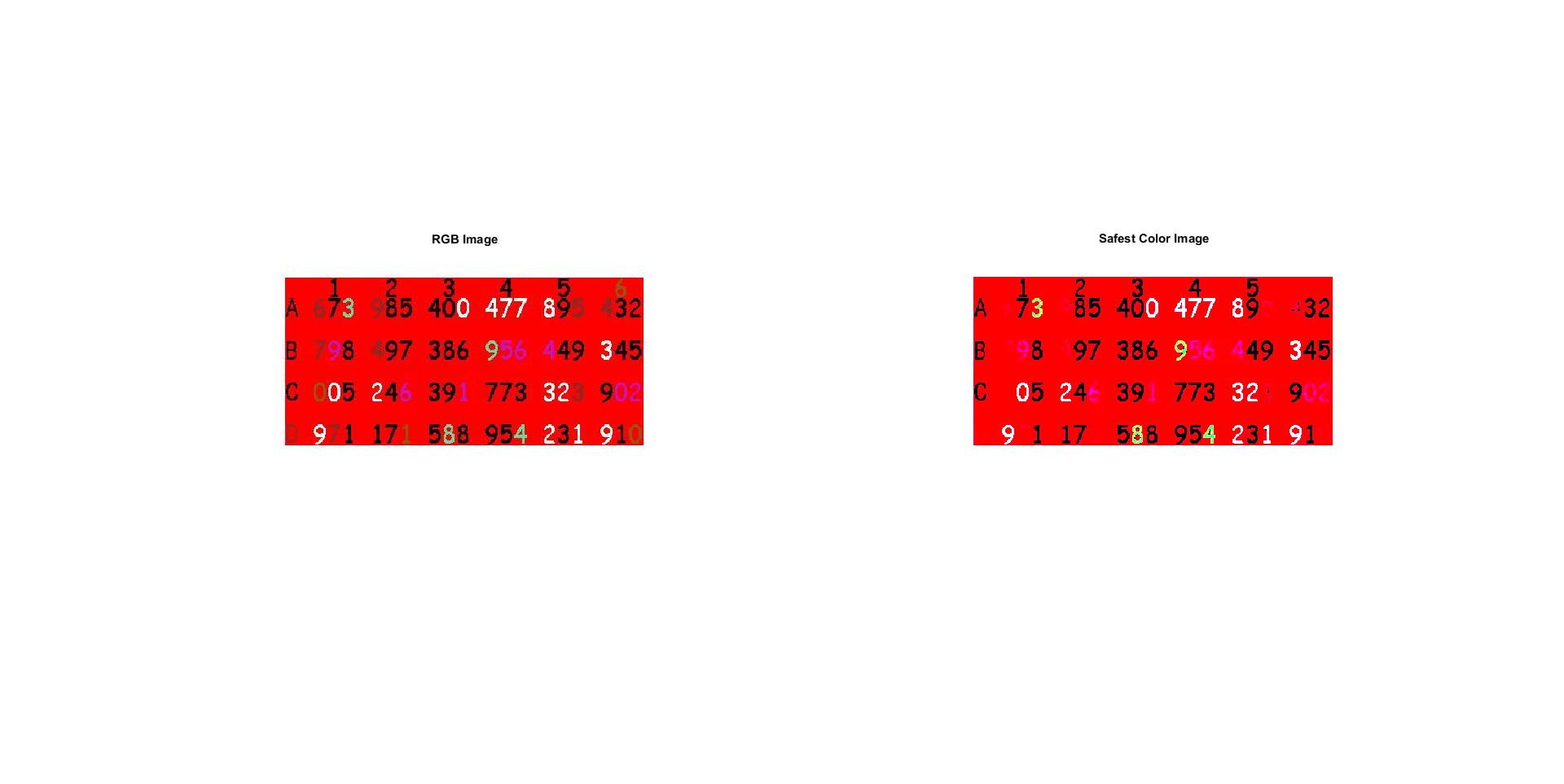
*Safe* – Substituir todos os valores dos pixéis da imagem pelo valor da safe color mais próxima. (diferença entre o valor de cada componente RGB de cada pixel, por cada valor do *array* de *safe* *colors* e escolher o valor de diferença mínimo.)

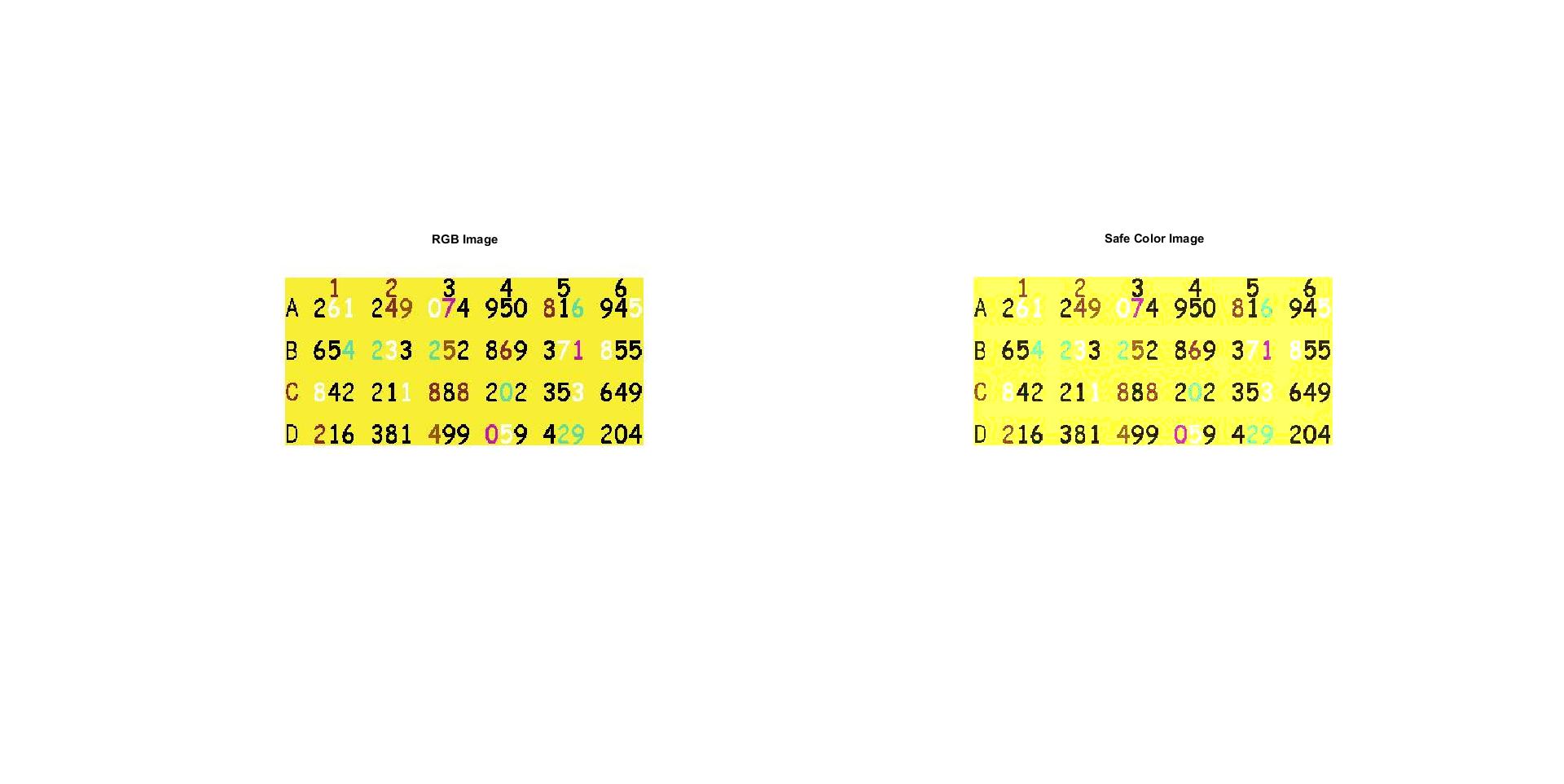
*Safest* – Distância absoluta entre o valor do pixel e cada cor do espaço de cores *safest* *color*. É escolhida a cor com o valor de diferença mínimo.

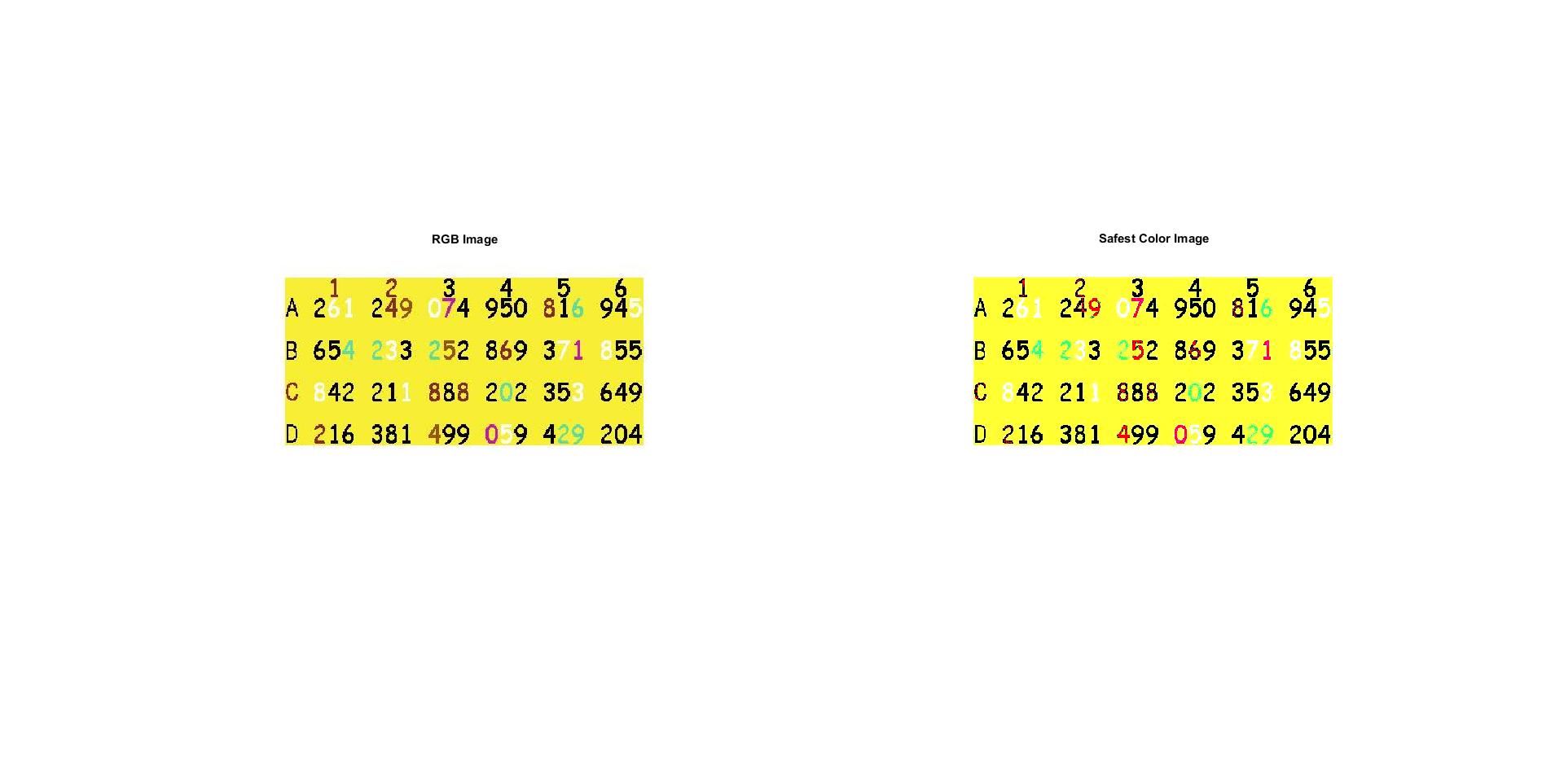
**Alínea b)**

Resultados:

cardCode1 – Versão *safe*

cardCode1 – Versão *safest*

cardCode2 – Versão *safe*

cardCode2 – Versão *safest*

Conclusão: A partir dos resultados destas duas imagens, consegue-se concluir que para a versão *safe*, a imagem fica muito semelhante à original. Apresenta todas as cores muito idênticas, notando-se apenas diferença na intensidade. A imagem correspondente à versão *safe* fica mais baça, enquanto que a imagem original tem as cores mais vivas (com mais brilho).

Para a versão *safest*, as diferenças já são mais visíveis. Com estas imagens do exemplo verifica-se que alguns caracteres da imagem correspondente à versão *safest* apresentam uma cor diferente em relação à imagem original. Como a conversão é realizada pixel a pixel e devido à reduzida gama de cores, existem mesmo alguns caracteres que “perdem” parte da sua cor.

Como era expectável, devido ao menor número de cores nas versões *safe* e *safest* *color*, a qualidade das imagens convertidas é inferior à qualidade das imagens originais, sendo mais acentuada na versão *safest* (por vezes inelegível) que na versão *safe* (na maioria dos casos, legível).

**Exercício 3.**

**Alínea b)**

Resultados:

Uma imagem com pássaro

Descrição gerada com confiança altafinger1.tif

****finger2.tif

****finger3.tif

**Uma imagem com interior

Descrição gerada com confiança alta**finger4.tif

**Uma imagem com fotografia, edifício, parede

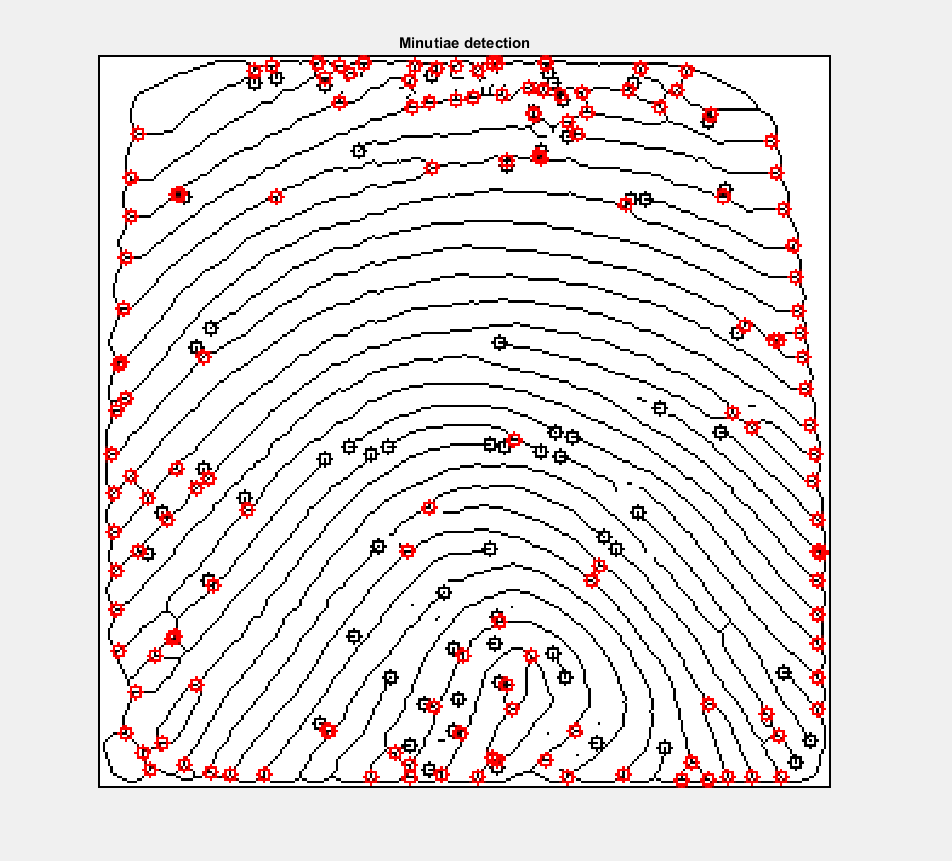
Descrição gerada com confiança alta**finger5.bmp

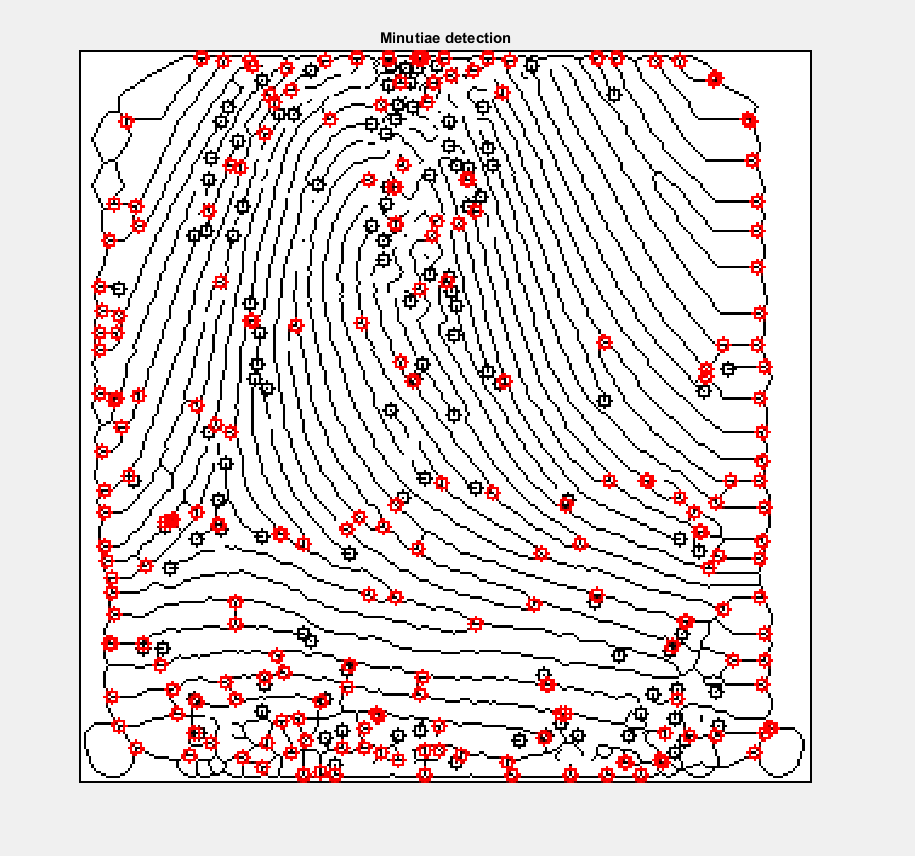
Conclusão: Uma vez que os resultados obtidos pela função *fingerprint\_enhancement* já foram bastante satisfatórios, apenas foi necessário, na função *fingerprint\_enhancement\_morph*, reduzir a espessura da linhas da impressão digital para apenas um pixel, de forma a que facilite o trabalho da deteção de minúcias e os resultados apresentados foram os pretendidos, com exceção dos contornos de fora da impressão digital, que não seria suposto existirem, uma vez que originam a que sejam detetadas minúcias inexistentes na imagem original.

**Alínea c)**

Referência para realização da função: <http://www.mathworks.com/matlabcentral/fileexchange/31926-fingerprint-minutiae-extraction?focused=5190983&tab=function>

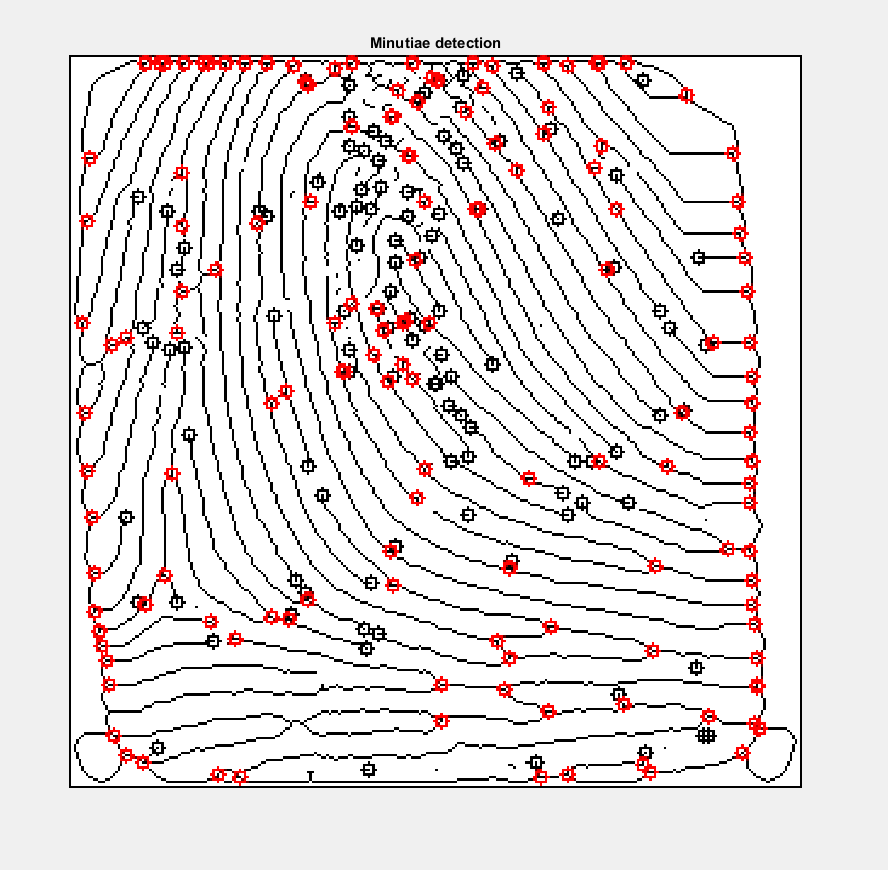
Resultados:

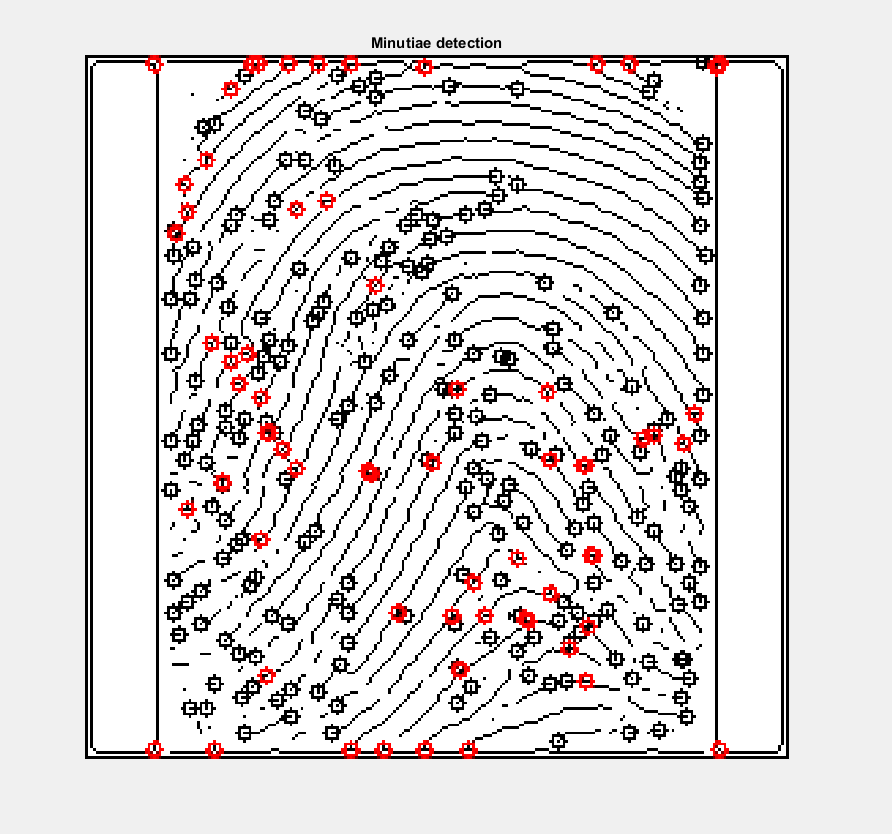
finger1.tif

****finger2.tif

**Uma imagem com texto

Descrição gerada com confiança alta**finger3.tif

finger4.tif

finger5.bmp

Conclusão: A função *minutiae\_detection* deteta e apresenta na imagem binária de entrada, as minúcias da impressão digital. As minúcias detetadas podem ser bifurcações – marcadas a vermelho – ou cristas (fim de linha) – marcadas a preto.

O resultado não foi exatamente o pretendido devido ao facto da imagem produzida pela função *fingerprint\_enhancement\_morph* apresentar linhas que limitam a impressão digital, originando minúcias indesejáveis. Para além deste caso, verifica-se que existem assinaladas falsas minúcias, pois a imagem da impressão digital contém algumas falhas nas linhas ou junções de linhas distintas que induzem o algoritmo em erro. Para tentar atenuar este problema, decidimos apagar uma das minúcias quando existem 2 minúcias praticamente sobrepostas ou apagar todas as minúcias que se encontravam a menos de 6 pixéis de distância, o que pode levar a que, excecionalmente, minúcias verdadeiras também sejam apagadas. No entanto e como se pode observar nos exemplos *finger2.tif* e *finger5.bmp*, ainda são assinaladas uma quantia significativa de falsas minúcias, mas no geral, praticamente todas as minúcias existentes nas imagens de impressões digitais usadas foram detetas e assinaladas.

**Exercício 4. (Foi realizado o face\_detector proposto na 1 serie, que ficou em falta)**

[**Base de dados**](http://www.anefian.com/research/face_reco.htm)[**Programa exemplo onde nos baseamos**](https://www.mathworks.com/videos/face-recognition-with-matlab-100902.html)

**Alínea a)**

Para o projeto proposto, um programa de reconhecimento facial através da extração de características da face, escolhemos utilizar a linguagem Mathlab.

A composição de cada pasta das imagens é feita da seguinte forma: 1 pasta contém 10 fotos do mesmo individuo, em que dessas 10 fotos são selecionadas 80% (8) para aprendizagem, e as restantes 20% (2) para treino.

Sendo assim, visto que temos 4 pastas, equivale a 4 indivíduos.

Utilizamos a função [extractHOGfeatures](https://www.mathworks.com/help/vision/ref/extracthogfeatures.html), para a extração de características.

**Aprendizagem**: Esta é a etapa onde são extraídas as características das 8 imagens (para este exemplo), e que no fim é feita uma média final para ficar com apenas um vetor de características desse individuo. Ao realizar este procedimento para os 4 indivíduos, no fim da aprendizagem ficaremos com 4 vetores, cada um deles com as características de 1 individuo.

**Teste:** Com as 2 imagens restantes de cada individuo, 8 no total, verificamos se a aprendizagem foi feita da melhor forma. Para isto, comparamos o vetor de características da face a testar com os 4 vetores resultantes da fase de aprendizagem, e comparamos com qual é que a distância euclidiana é menor. Onde for menor é um possível candidato a match.

**Falsos Positivos:** Após verificar qual dos vetores deu a menor distância euclidiana, utilizamos um valor de threshold (9 no nosso caso) para verificar se esse valor mínimo é superior. Caso seja é possível que seja um Falso Positivo. **(Para testar os falsos positivos, inserimos nas supostas imagens de teste, uma imagem semelhante á pessoa mas que não é a mesma.)**

**Falso Negativo:** Na amostra de imagens utilizada, para o individuo 5, aconteceu um falso negativo, isto é, o nosso sistema disse que não havia match, mas o individuo é o mesmo. Isto acontece possivelmente devido á diferença dessa fotografia com os restantes desse individuo.

**RESULTADOS**

