

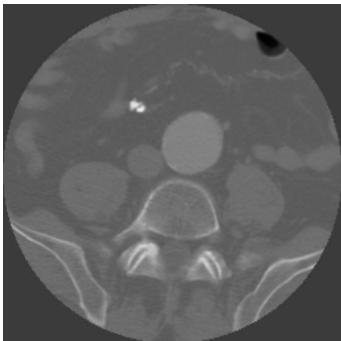


# Compression universelle d'images

Réalisé par Odorico Thibault.

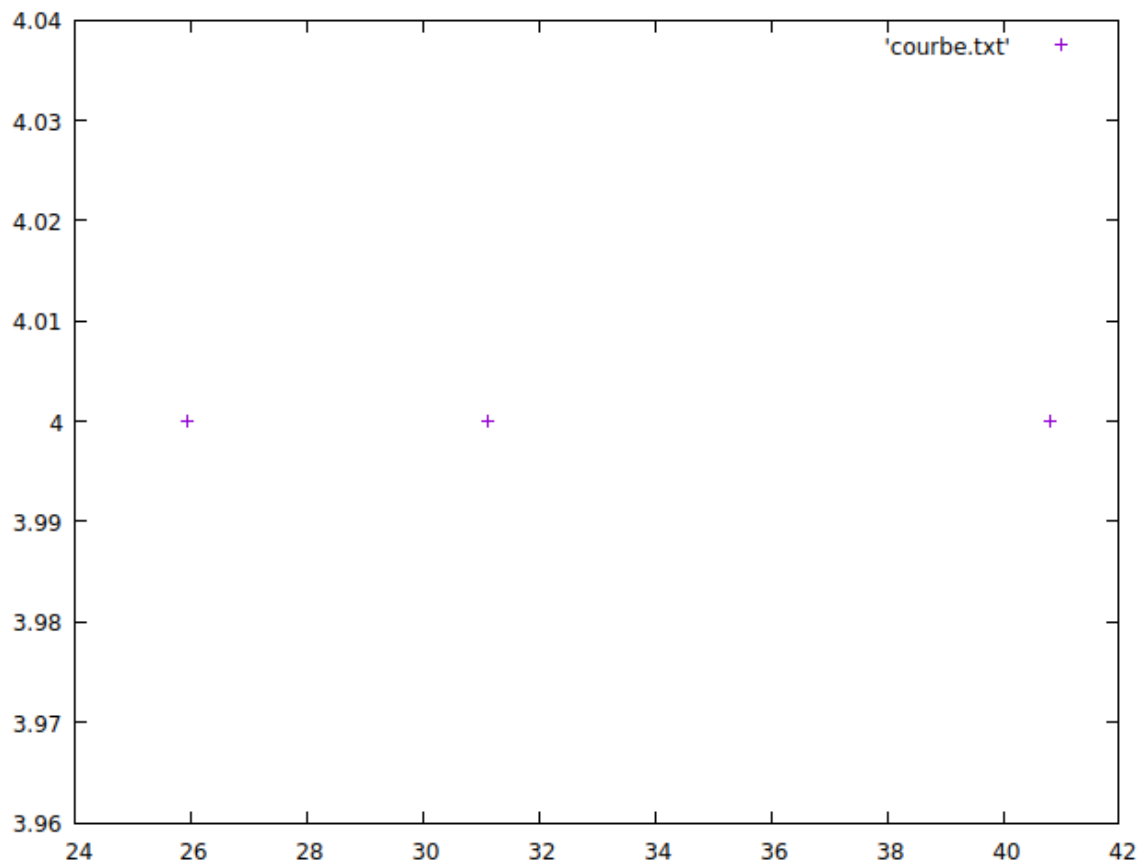
## Images de références

Image Médicale $I_M$	Image de Girafes $I_G$	Image de Casquettes $I_C$
		

## Image résultats

Compression $I_M$	Compression $I_G$	Compression $I_C$
		
Taille originale	Taille originale	Taille originale
262144 octets	262144 octets	786432 octets
Taille compressée	Taille compressée	Taille compressée
65536 octets	65536 octets	196608 octets
Taux de compression	Taux de compression	Taux de compression
4	4	4
PSNR	PSNR	PSNR
40.8132	25.9476	31.1263

Graph (Taux de compression en Y / PSNR en X)



## Algorithme

```
1 PGM reduced_by_2(const PGM& img)
2 {
3     PGM reduced(img.width() / 2, img.height() / 2);
4
5     for (size_t i = 0 ; i < reduced.height() ; ++i)
6     {
7         for (size_t j = 0 ; j < reduced.width() ; ++j)
8         {
9             reduced(i, j) = img(i * 2, j * 2);
10        }
11    }
12
13    return reduced;
14 }
15
16 PGM extended_by_2(const PGM& img)
17 {
18     PGM extended(img.width() * 2, img.height() * 2);
19
20     // Etant le pixel comme un block.
21
22     for (size_t i = 0 ; i < extended.height() ; i++)
23         for (size_t j = 0 ; j < extended.width() ; j++)
24             extended(i, j) = img(i / 2, j / 2);
25
26     // Moyenne des 4 pixels voisins de img
27 }
```

```

28     for (size_t i = 0 ; i < img.height() ; ++i)
29     {
30         for (size_t j = 0 ; j < img.width() ; ++j)
31         {
32             std::array<PGM::value_type, 4> pixels = {
33                 img(i      , j),
34                 img(i      , j + 1),
35                 img(i + 1, j),
36                 img(i + 1, j + 1),
37             };
38
39             double mean = 0;
40
41             for (const auto& pixel : pixels)
42                 mean += pixel;
43
44             mean /= pixels.size();
45
46             extended((i * 2) + 1, (j * 2) + 1) = round(mean);
47         }
48     }
49
50     // moyenne des pixels voisins
51
52     for (size_t i = 1 ; i < extended.height() - 1 ; i++)
53     {
54         for (size_t j = (i % 2 == 0 ? 1 : 2) ; j < extended.width() - 1 ;
j+=2)
55         {
56             std::array<PGM::value_type, 4> pixels = {
57                 extended(i - 1, j),
58                 extended(i      , j + 1),
59                 extended(i + 1, j),
60                 extended(i      , j - 1)
61             };
62
63             double mean = 0;
64
65             for (const auto& pixel : pixels)
66                 mean += pixel;
67
68             mean /= pixels.size();
69
70             extended(i, j) = round(mean);
71         }
72     }
73
74     return extended;
75 }

```