Image Processing and Computer Vision (IPCV)

Prof. J. Weickert
Mathematical Image Analysis Group
Summer term 2017
Saarland University

Example Solutions for Homework Assignment 5 (H5)

Problem 1: (Keys Interpolation)

We want to show that the Keys synthesis function satisies

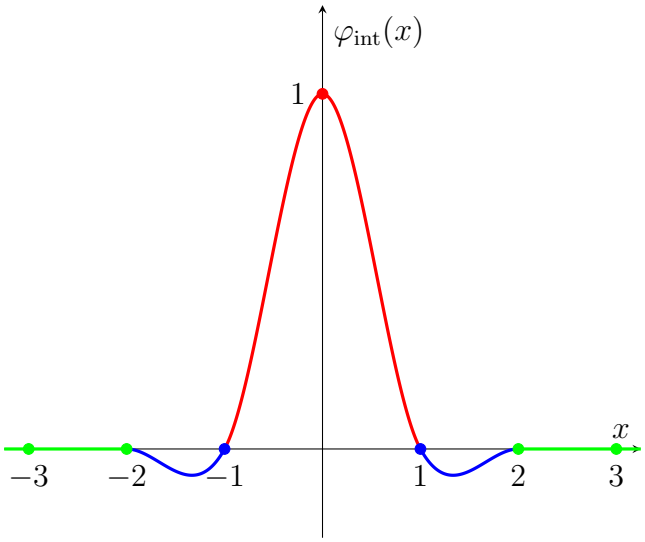
$$1 = \sum_{k \in \mathbb{Z}} \varphi_{\text{int}}(x - k) \qquad \text{for all } x \in \mathbb{R}\,.$$

At first glance, we have to compute an infinite sum. However, the Keys function has a limited support, which means that the summands will be zero for almost all k.
Furthermore, neighbouring positions at which we have to evaluate φ_{int} have a distance of exactly 1, since $k \in \mathbb{Z}$. We distinguish two different cases for the values of x, namely $x \in \mathbb{Z}$ and $x \notin \mathbb{Z}$.

Case 1: $x \in \mathbb{Z}$
This case needs to be treated specifically, since the $x - k$ for which we have to evaluate φ_{int} lie exactly on the boundaries of the definition intervals. Let us have a look at the Keys function

$$\varphi_{\text{int}}(x) := \begin{cases} \frac{3}{2}|x|^3 - \frac{5}{2}x^2 + 1, & \text{if } |x| < 1, \\ -\frac{1}{2}|x|^3 + \frac{5}{2}x^2 - 4|x| + 2, & \text{if } 1 \leqslant |x| < 2, \\ 0, & \text{else.} \end{cases}$$

and a corresponding sketch including evaluation points for $x \in \mathbb{Z}$:



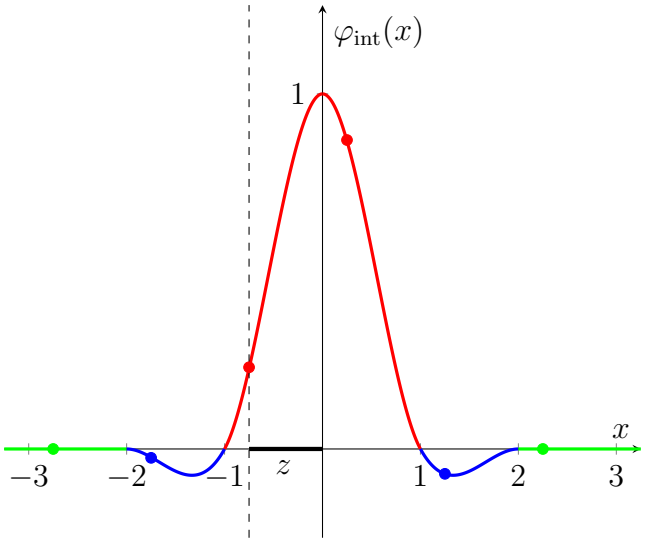
Formally, we obtain for $x \in \mathbb{Z}$:

$$\begin{aligned}\sum_{k\in\mathbb{Z}} \varphi_{\text{int}}(x-k) &= \varphi_{\text{int}}(-3) + \varphi_{\text{int}}(-2) + \varphi_{\text{int}}(-1) + \varphi_{\text{int}}(0) + \varphi_{\text{int}}(1) + \varphi_{\text{int}}(2) + \varphi_{\text{int}}(3) \\ &= 0 + 0 + 0 + 1 + 0 + 0 + 0 = 1\end{aligned}$$

Case 2: $x \notin \mathbb{Z}$
We define $z \in (0,1)$ as

$$z := \lceil x \rceil - x$$

which means that z is the distance of x to the next larger integer value. The following figure shows the Keys function with evaluation positions for an $x \notin \mathbb{Z}$ and the same colour coding as before.



Using this sketch, we can now compute

$$
\begin{aligned}
&\sum_{k\in\mathbb{Z}} \varphi_{\text{int}}(x-k) \\
&= \varphi_{\text{int}}(-2-z) + \varphi_{\text{int}}(-1-z) + \varphi_{\text{int}}(-z) + \varphi_{\text{int}}(1-z) + \varphi_{\text{int}}(2-z) + \varphi_{\text{int}}(3-z) \\
&= 0 - \frac{1}{2}|-1-z|^3 + \frac{5}{2}(-1-z)^2 - 4|-1-z| + 2 + \frac{3}{2}|-z|^3 - \frac{5}{2}(-z)^2 + 1 \\
&\quad + \frac{3}{2}|1-z|^3 - \frac{5}{2}(1-z)^2 + 1 - \frac{1}{2}|2-z|^3 + \frac{5}{2}(2-z)^2 - 4|2-z| + 2 + 0 \\
&= -\frac{1}{2}(z+1)^3 + \frac{5}{2}(z+1)^2 - 4(z+1) + 2 + \frac{3}{2}z^3 - \frac{5}{2}z^2 + 1 \\
&\quad + \frac{3}{2}(1-z)^3 - \frac{5}{2}(1-z)^2 + 1 - \frac{1}{2}(2-z)^3 + \frac{5}{2}(2-z)^2 - 4(2-z) + 2 \\
&= -\frac{1}{2}(z^3+3z^2+3z+1) + \frac{5}{2}(z^2+2z+1) - 4z - 4 + 2 + \frac{3}{2}z^3 - \frac{5}{2}z^2 + 1 \\
&\quad + \frac{3}{2}(-z^3+3z^2-3z+1) - \frac{5}{2}(z^2-2z+1) + 1 \\
&\quad - \frac{1}{2}(-z^3+6z^2-12z+8) + \frac{5}{2}(z^2-4z+4) + 4z - 8 + 2 \\
&= 1
\end{aligned}
$$

This concludes the proof.

Problem 2 (Quadratic B-Spline Interpolation)

(a) The synthesis function for quadratic B-splines is defined as

$$\beta_2(x) = \begin{cases} \frac{3}{4} - x^2, & |x| < \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2} - |x|\right)^2, & \frac{1}{2} \leqslant |x| < \frac{3}{2} \\ 0, & \text{else} \end{cases}$$

Evaluating this function at the positions $0, 1$ and 2 yields

$$\beta_2(0) = \frac{3}{4}, \quad \beta_2(1) = \frac{1}{8}, \quad \beta_2(2) = 0$$

Thus, the following linear system has to be solved in order to obtain the interpolation coefficients:

$$\begin{pmatrix} \frac{3}{4} & \frac{1}{8} & 0 \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ 0 & \frac{1}{8} & \frac{3}{4} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 15 \\ 5 \\ 15 \end{pmatrix}$$

Computing the matrix vector product

$$\begin{pmatrix} \frac{3}{4} & \frac{1}{8} & 0 \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ 0 & \frac{1}{8} & \frac{3}{4} \end{pmatrix} \begin{pmatrix} 20 \\ 0 \\ 20 \end{pmatrix}$$

verifies that $(20, 0, 20)^\top$ indeed solves the linear system of equations.

(b) The interpolant is given by

$$\begin{aligned} u(x) &= \sum_{k=1}^{3} c_k \beta_2(x - k)) \\ &= 20\beta_2(x-1) + 0\beta_2(x-2) + 20\beta_2(x-3) \end{aligned}$$

Let us now have a look at the domains where each summand is defined and which value it yields there. First of all, it is easy to see that

$$\begin{aligned} |x-1| \in \left[0, \frac{1}{2}\right] \quad &\Leftrightarrow x \in \left[\frac{1}{2}, \frac{3}{2}\right] \\ |x-3| \in \left[\frac{1}{2}, \frac{3}{2}\right] \quad &\Leftrightarrow x \in \left[-\frac{1}{2}, \frac{1}{2}\right] \cup \left[\frac{3}{2}, \frac{5}{2}\right] \end{aligned}$$

and

$$|x-3| \in \left[0, \frac{1}{2}\right] \qquad \Leftrightarrow x \in \left[\frac{5}{2}, \frac{7}{2}\right]$$
$$|x-3| \in \left[\frac{1}{2}, \frac{3}{2}\right] \qquad \Leftrightarrow x \in \left[\frac{3}{2}, \frac{5}{2}\right] \cup \left[\frac{7}{2}, \frac{9}{2}\right]$$

It follows, that $\beta_2(x-1)$ is nonzero on $\left[-\frac{1}{2}, \frac{5}{2}\right]$ and $\beta_2(x-3)$ is nonzero on $\left[\frac{3}{2}, \frac{9}{2}\right]$. We don't have to evaluate β_2 at $(x-2)$ as coefficient $c_2 = 0$. Now it is enough to insert the correct expression in the corresponding interval, as it is done in the following table.

	$\left[-\frac{1}{2}, \frac{1}{2}\right]$	$\left[\frac{1}{2}, -\frac{3}{2}\right]$	$\left[\frac{3}{2}, \frac{5}{2}\right]$	$\left[\frac{5}{2}, \frac{7}{2}\right]$	$\left[\frac{7}{2}, \frac{9}{2}\right]$
$20\beta_2(x-1)$	$10\left(\frac{3}{2} - \lvert x-1\rvert\right)^2$	$15 - 20(x-1)^2$	$10\left(\frac{3}{2} - \lvert x-1\rvert\right)^2$	0	0
$0\beta_2(x-2)$	0	0	0	0	0
$20\beta_2(x-3)$	0	0	$10\left(\frac{3}{2} - \lvert x-3\rvert\right)^2$	$15 - 20(x-3)^2$	$10\left(\frac{3}{2} - \lvert x-3\rvert\right)^2$

Thus, we obtain

$$u(x) = \begin{cases} 10\left(\frac{3}{2} - |x-1|\right)^2, & x \in \left[-\frac{1}{2}, \frac{1}{2}\right] \\ 15 - 20(x-1)^2, & x \in \left[\frac{1}{2}, \frac{3}{2}\right] \\ 10\left(\frac{3}{2} - |x-1|\right)^2 + 10\left(\frac{3}{2} - |x-3|\right)^2, & x \in \left[\frac{3}{2}, \frac{5}{2}\right] \\ 15 - 20(x-3)^2, & x \in \left[\frac{5}{2}, \frac{7}{2}\right] \\ 10\left(\frac{3}{2} - |x-3|\right)^2, & x \in \left[\frac{7}{2}, \frac{9}{2}\right] \\ 0, & \text{else} \end{cases}$$

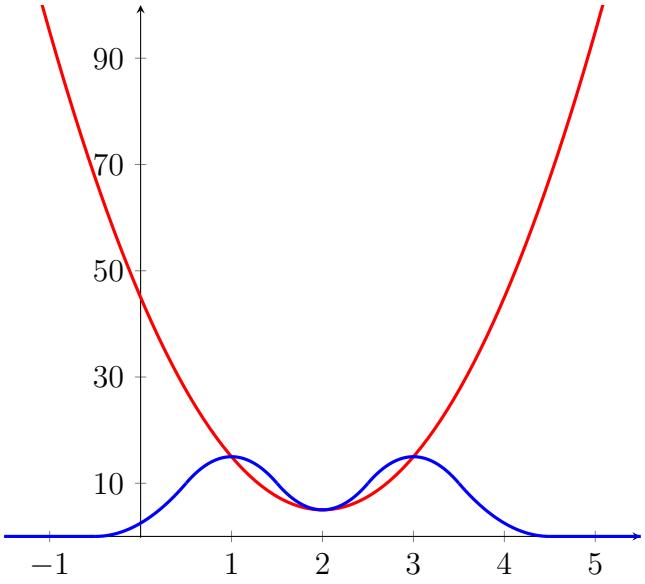
Evaluating this function at the desired points yields

$$u(0) = 2.5, \quad u\left(\frac{3}{2}\right) = 10, \quad u(3) = 15, \quad u(5) = 0$$

(c) The exact values at these points would be

$$f(0) = 45, \quad f\left(\frac{3}{2}\right) = 7.5, \quad f(3) = 15, \quad f(5) = 95$$

Consider the following graph that depicts the interpolant $u(x)$ in blue and the original function $f(x)$ in red, since this helps to understand the observed differences between the original and the interpolated values.



There is a fundamental difference between the approximation quality inside of the interpolation domain $[1, 3]$ and the points outside of this interval. Inside the interpolation domain, the interpolation is far from perfect (we only used a low number of samples), but the general shape of the interpolant fits fairly well. Outside of the interpolation domain, there is no similarity between the interpolant and the original function anymore. Since the synthesis functions have limited support, the interpolant also vanishes outside of this support.

These observations can also be seen from the four values that had to be evaluated in (b) and (c). The only location where the interpolation is perfect is at $u(3) = f(3) = 15$, which is clear, since the interpolation condition is met there. For $0, \frac{3}{2}$ and 5, the interpolation error (i.e. the difference between original and interpolated value) is increasing with the distance from $[1, 3]$. For $\frac{3}{2}$, two of the shifted synthesis functions contribute to the interpolated value, for 0 only 1 and 5 is already outside of the support of the interpolant.

(d) In practice, quadratic splines are usually not used. The computational cost is comparable to cubic splines, while the quality is significantly worse.

Problem 3 (Affine Rescaling)

(a) In order to apply the affine transformation $f(x) = mx + n$, we have to determine the slope m and the offset n first. Since we know that $F(x_{min}) = a$ and $F(x_{max}) = b$ we have two linear equations for the two unknowns. Solving this 2×2 linear system of equations yields $m = \frac{b-a}{x_{max}-x_{min}}$ and $n = a - x_{\min}\frac{b-a}{x_{max}-x_{min}}$. This allows us to implement the `rescale` method via

```
/*-------------------------------------------------------*/
long    i, j, k;        /* loop variables */
double  min, max;       /* extrema of u */
double  factor;         /* time saver */

/* determine extrema of u */
min = max = u[1][1];
for (i = 1; i <= nx; i++)
{
  for (j = 1; j <= ny; j++)
  {
    if (u[i][j] < min) min = u[i][j];
    if (u[i][j] > max) max = u[i][j];
  }
}

/* rescale */
factor = (b-a) / (max-min);
for (i = 0; i <= 255; i++)
  g[i] = factor*i + a-min*factor;
/*-------------------------------------------------------*/
```

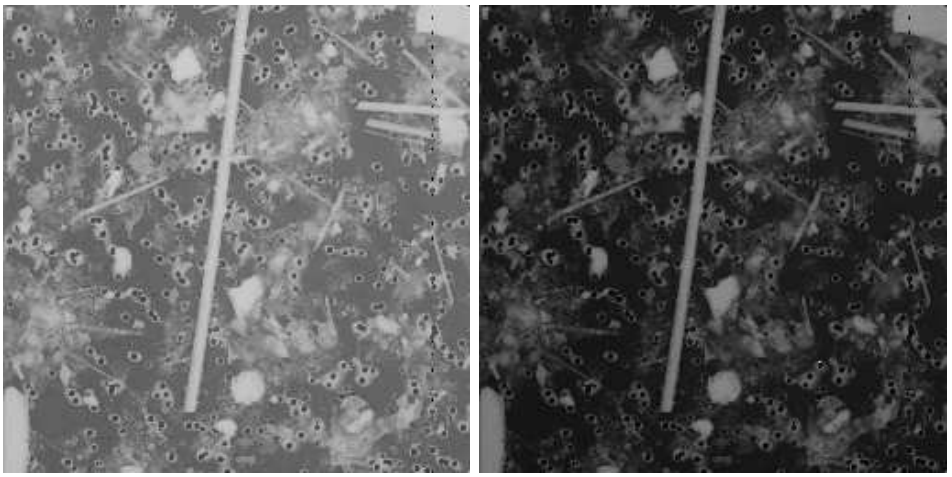
(b) Applying the affine transformation to the image `machine.pgm` yields (left: original, right: transformed image ($a = 0$, $b = 255$)).

Problem 4 (Gamma Correction)

(a) The second task is to implement the gamma corrections from the lecture. The corresponding code is given by

```
/* ------------------------------------------------------------ */
void gamma_correct

     (double  gamma,      /* gamma correction factor */
      double  *g)         /* transformed grey levels */

/*
 applies gamma correction to the 256 grey levels that may appear
 in byte wise coded images
*/

{
long  k;   /* loop variable */

for (k = 0; k <= 255; k++)
    g[k] = 255.0 * pow(k/255.0, 1.0/gamma);

}
/* ------------------------------------------------------------ */
```

(b) Applying the gamma correction to the image `asbest.pgm` yields (left: original, right: transformed image ($\gamma = 0.4$)).



Problem 5 (Histogram Equalisation)

(a) Here, we implement the histogram equalisation from the lecture. Two algorithms are presented. The first one maps the grey values directly if the inequality holds (early map). The second one is closer to the lecture and maps the grey values *en block* after the inequality is violated (late map).

```
/* ---------------------------------------------------------- */
void hist_equal_early_map

     (double  **u,        /* input image, range [0,255] */
      long    nx,         /* size in x direction */
      long    ny,         /* size in y direction */
      double  *g)         /* transformed grey levels */

/* performs histogram equalization on u. */

{
long    i, j, k;          /* loop variables */
long    r;                /* current summation index r */
long    k_r;              /* current summation index k_r */
double  hist[256];        /* histogram */
long    n;                /* pixel number */
double  psum, qsum;       /* sums in equalisation algorithm */

/* initialise histogram with zero */
for (k = 0; k <= 255; k++)
    hist[k]=0;

/* create histogram of u with bin width 1 */
for (i = 1; i <= nx; i++)
  for (j = 1; j <= ny; j++)
    hist[(long) u[i][j]]++;

/* initialise psum and k_r */
k_r = 1;
psum = hist[0];

/* total number of pixels */
n = nx*ny;
```

```
for (r = 1; r <= 256; r++)
{
  /* determine right hand side */
  qsum = r * n / 256.0;

  /* increase left hand side as long as it is smaller or equal */
  /* the right hand side; note that each k_r is only used once.*/
  for (; (k_r <= 256) && (psum <= qsum); k_r++)
  {
    /* perform mapping : we have to substract 1 from the grey */
    /* value index r and k_r to get the actual grey values */
    g[k_r-1] = r - 1;

    /* increase left hand side */
    psum += hist[k_r];

  }
}
}

/* ------------------------------------------------------------ */


/*--------------------------------------------------------------*/

void hist_equal_late_map

     (double  **u,        /* input image, range [0,255] */
      long    nx,         /* size in x direction */
      long    ny,         /* size in y direction */
      double  *g)         /* transformed grey levels */

/* performs histogram equalization on u. */

{
long    i, j, k;          /* loop variables */
long    r;                /* current summation index r */
long    k_r;              /* current summation index k_r */
long    n;                /* pixel number */
double  hist[256];        /* histogram */
double  psum, qsum;       /* sums in equalisation algorithm */

/* initialise histogram with zero */
for (k = 0; k <= 255; k++)
```

```
    hist[k]=0;

/* create histogram of u with bin width 1 */
for (i = 1; i <= nx; i++)
  for (j = 1; j <= ny; j++)
    hist[(long) u[i][j]]++;

/* initialise psum and k_r */
k_r  = 1;
psum = hist[0];

/* total number of pixels */
n = nx*ny;

/* j is the index of the last grey value that has been mapped */
j   = 0;

for(r=1; r<=256; r++)
{
  /* determine right hand side */
  qsum = r * n / 256.0;

  /* search the largest index k_r such that the inequality */
  /* is satisfied */
  while(k_r <= 256 && psum <= qsum)
  {
    psum += hist[k_r];
    k_r  = k_r+1;
  }

  /* perform mapping from j which corresponds to k_{r-1} to k_r-1.*/
  /* we have to subtract 1 from the grey value index r and k_r */
  /* to get the actual grey values */
  while(j < k_r-1)
  {
    /* perform mapping */
    g[j] = r-1;

    /* store last index that has been mapped (=k_{r-1}) */
    j    = j+1;
  }
}
}
```

```
/*-------------------------------------------------------------*/
```

(b) Applying the histogram equalisation to the image `office.pgm` yields (left: original, right: transformed)