12. Gyakorlat
Példaprogramok VisiQuest-ben

Hisztogramszéthúzás:

A hisztogram operátoroknál már esett szó arról, hogy hisztogram széthúzással javítjatjuk a kép kontrasztját. A hisztogram széthúzás három lépésből áll:

- meghatározzuk a hisztogram szélességét
- eltoljuk a hisztogramot a 0-ba (hogy a bal széle a 0-ban legyen)
- transzfolmáljuk a hisztogramot 0-255 intenzitástartományba

Lássuk, hogyan is működik:

```
int run_myhistogramstretch(void)
{
   /*-- Put Your Code Here --*/
       /* -main_variable_list */
   char *lib = "myhistorgramstretch_obj";
   char *rtn = "main";
   kobject in_object = NULL;
   kobject out_object = NULL;
   unsigned char *plane;
   unsigned char *res_plane;
   int w, h, d, t, e;
   int cw, ch, ct, pos;
   int histw, min, max; /* segédváltozók a hisztogram szélességének
                           meghatározásához */

/* -main_variable_list_end */

/* -main_before_lib_call */
   if ((in_object = kpds_open_input_object(clui_info->i_file))
          == KOBJECT_INVALID) {
     kerror(lib, rtn, "Can not open input object %s.\n", clui_info->i_file);
     kexit(KEXIT_FAILURE);
   }
   if ((out_object = kpds_open_output_object(clui_info->o_file))
        == KOBJECT_INVALID) {
     kerror(lib, rtn, "Can not open output object %s.\n", clui_info->o_file);
     kexit(KEXIT_FAILURE);
   }
   if (!kpds_copy_object(in_object, out_object)) {
     kerror(lib, rtn, "Can not copy input object to output object.\n");
     kexit(KEXIT_FAILURE);
   }
   kpds_set_attribute(in_object,  KPDS_VALUE_DATA_TYPE, KUBYTE);
   kpds_set_attribute(out_object, KPDS_VALUE_DATA_TYPE, KUBYTE);

  kpds_get_attribute(in_object, KPDS_VALUE_SIZE, &w, &h, &d, &t, &e);

  plane = (unsigned char *)kmalloc(w*h*sizeof(unsigned char));
  res_plane = (char *)kmalloc(w*h*sizeof(unsigned char));
  if (!plane || !res_plane) {
     kerror(lib, rtn, "Could not allocate memory for the image\n");
     kexit(KEXIT_FAILURE);
  }
/* -main_before_lib_call_end */
```

```
/* -main_library_call */
  for (ct = 0; ct < t; ct++) {
            min = 255;
            max = 0;
    kpds_set_attribute(in_object, KPDS_VALUE_POSITION, 0, 0, 0, ct, 0);
    kpds_get_data(in_object, KPDS_VALUE_PLANE, (kaddr)plane);
    kmemset(res_plane, 0, w*h*sizeof(char));

    /* meghatározzuk a hisztogram szélességét */
    for (ch = 0; ch < h; ch++) {
      for (cw = 0; cw < w; cw++) {
        pos = ch*w + cw;

            if ( min > plane[pos] ) min = plane[pos];
            if ( max < plane[pos] ) max = plane[pos];

      }
    }
    histw = max    min; /* a szélesség a maximum és minimum különbsége lesz */

    /* eltoljuk a hisztogramot a 0-ba */
    for (ch = 0; ch < h; ch++) {
      for (cw = 0; cw < w; cw++) {
        pos = ch*w + cw;
        res_plane[pos] = plane[pos] - min;

      }
    }

    /* transzformáljuk a hisztogramot 0-255 közé */
    for (ch = 0; ch < h; ch++) {
      for (cw = 0; cw < w; cw++) {
        pos = ch*w + cw;
        res_plane[pos] = res_plane[pos] * 255 / histw;

      }
    }

    kpds_set_attribute(out_object, KPDS_VALUE_POSITION, 0, 0, 0, ct, 0);
    kpds_put_data(out_object, KPDS_VALUE_PLANE, (kaddr)res_plane);
  }
/* -main_library_call_end */

/* -main_after_lib_call */
   if (!kpds_set_attribute(out_object, KPDS_HISTORY, kpds_history_string())) {
     kerror(lib,rtn,"Unable to set history on the destination object");
     kexit(KEXIT_FAILURE);
   }
   if (plane)
     kfree(plane);
   if (res_plane)
     kfree(res_plane);
   kpds_close_object(in_object);
   kpds_close_object(out_object);
/* -main_after_lib_call_end */
   return TRUE;
}
```

Laplacian of Gaussian Filter:

A szűrő képlete a következőképpen adható meg:

$$G_\sigma(x,y)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Nincs más dolgunk, minthogy ezt beírjuk a kódba. A konstans értékeket előre ki lehet számolni. A GUI-n lesz három paraméter, a kép szélessége és magassága, valamint a képletben szereplő σ érték. Fontos megjegyezni, hogy a szűrőnek nincs inputja!!


```
int run_mylaplacianofgaussian(void)
{
  /*-- Put Your Code Here --*/


        /* -main_variable_list */
  char *lib = "mylaplacianofgaussian_obj";
  char *rtn = "main";

  kobject out_object = NULL;

  double *res_plane;
  int w, h, d, t, e;
```

```
  int cw, ch, ct, pos, cx, cy, x, y;  /* x, y segédváltozó, cx, cy a kép közepe lesz */
            double sigma2;   /* új változó: a szigma négyzete  */
            double k;            /* ebbe fogjuk letárolni  a konstans értéket */
/* -main_variable_list_end */

/* -main_before_lib_call */

   /* nincs input kép, csak az output képet kell megnyitni */
   if ((out_object = kpds_open_output_object(clui_info->o_file))
       == KOBJECT_INVALID) {
     kerror(lib, rtn, "Can not open output object %s.\n", clui_info->o_file);
     kexit(KEXIT_FAILURE);
   }
  /* létre kell hozni a Value szegmenst */
  kpds_create_value( out_object );

  kpds_set_attribute(out_object, KPDS_VALUE_DATA_TYPE, KDOUBLE);   /* az adattípus KDOUBLE lesz */

  /* itt állítjuk be a méret attribútumokat */
            w = clui_info->width_int;    /* a GUI-ból származik */
            h = clui_info->height_int;    /* a  GUI-ból származik */
            d = 1;
            t = 1;
            e = 1;

  kpds_set_attribute(out_object, KPDS_VALUE_SIZE, w, h, d, t, e);

            cx = w / 2;   /* a kép középső pixele */
            cy = h / 2;   /* a kép középső pixele */
  res_plane = (double *)kmalloc(w*h*sizeof(double));
  if (!res_plane) {
     kerror(lib, rtn, "Could not allocate memory for the image\n");
     kexit(KEXIT_FAILURE);
  }
/* -main_before_lib_call_end */

/* -main_library_call */

  sigma2 = clui_info->sigma_double * clui_info->sigma_double;   /* a GUI-ból származik, négyzetet számolunk */
  k = 1.0 / sqrt( 2.0 * 3.14 * sigma2 ) ;          /* előre kiszámolható konstans érték */
  for (ct = 0; ct < t; ct++) {

    kmemset(res_plane, 0, w*h*sizeof(double));

    for (ch = 0; ch < h; ch++) {
      for (cw = 0; cw < w; cw++) {
        pos = ch*w + cw;

            x =  cw - cx;
            y =  ch - cy;

            /*  a kepletet beirom */
            res_plane[pos] = k * exp( - ( x * x + y * y  ) / ( 2.0 * sigma2 ) );

      }
    }

    kpds_set_attribute(out_object, KPDS_VALUE_POSITION, 0, 0, 0, ct, 0);
    kpds_put_data(out_object, KPDS_VALUE_PLANE, (kaddr)res_plane);
  }
/* -main_library_call_end */
```

```
/* -main_after_lib_call */
   if (!kpds_set_attribute(out_object, KPDS_HISTORY, kpds_history_string())) {
     kerror(lib,rtn,"Unable to set history on the destination object");
     kexit(KEXIT_FAILURE);
   }
   if (res_plane)
     kfree(res_plane);

   kpds_close_object(out_object);
/* -main_after_lib_call_end */

   return TRUE;
}
```

Az eredmény: