12. Gyakorlat
Példaprogramok Khoros Cantata-ban

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 main(void)
{

   /* -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 */
```

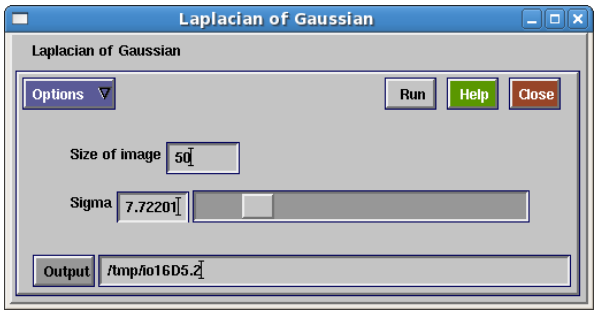
```
/* -main_library_call */
  for (ct = 0; ct < t; ct++) {
            min = 255;
            max = 0;
    kpds_set_attribute(in_object, KPDS_VALUE_POSITION, 0, 0, 0, ct, 0);
    kpds_get_data(in_object, KPDS_VALUE_PLANE, (kaddr)plane);
    kmemset(res_plane, 0, w*h*sizeof(char));

    /* 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 */
   kexit( KEXIT_SUCCESS );
}
```

Laplacian of Gaussian Filter:

A szűrő képlete a következőképpen adható meg:

$$\mathrm{LoG}_{\sigma}(x,y)=\frac{x^2+y^2-2\sigma^2}{\sigma^4}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

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!!

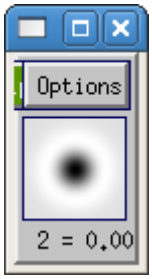


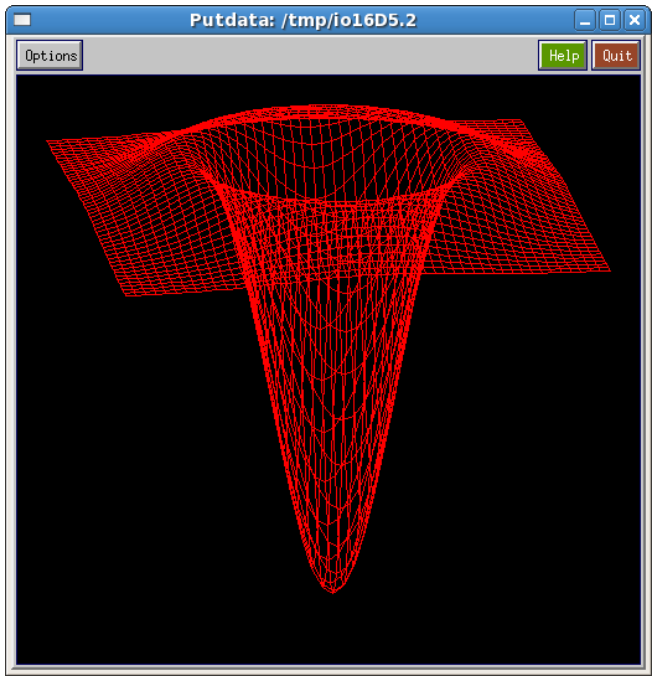

```
int main(void)
{

/* -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 sigma4;  /* új változó: a szigma a 4. hatványon  */
/* -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->size_int;    /* a GUI-ból származik */
      h = w;
      d = 1;
      t = 1;
      e = 1;

      sigma2 = clui_info->sigma_float * clui_info->sigma_float;
      sigma4 = sigma2 * sigma2;

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


  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 */
  cx = w / 2;   /* a kép középső pixele */
  cy = h / 2;   /* a kép középső pixele */


  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 képletet beírom */
      res_plane[pos]=((x*x+y*y  2.0*sigma2)/sigma4)*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;
}
```

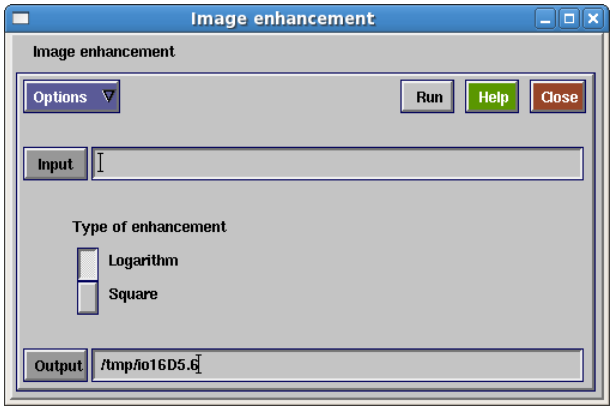
Az eredmény:





Képkiemelés nemlineáris függvényekkel:

Az alábbi példában láthatjuk, hogy hogyan választhatunk több opció közül rádiógomb segítségével. A rádiógombokat (*Flag*) a GUISE felületén a ***Toggle Variables*** menüben találjuk.



A grafikus panelen a rádiógombok változójának azonosítója *enhancement* legyen!

```
int main(
   int  argc,
   char **argv)
{

/* -main_variable_list */
  char *lib = "myimageenhancement_obj";
  char *rtn = "main";
  kobject in_object = NULL;
  kobject out_object = NULL;
  double *plane;
  double *res_plane;
  int w, h, d, t, e;
  int cw, ch, ct, pos;
/* -main_variable_list_end */

      khoros_init(argc, argv, "PROBA", PRODUCT_RELEASE_DATE,
                  PRODUCT_RELEASE_NAME, PRODUCT_RELEASE_VERSION,
                  PRODUCT_RELEASE_MAJOR, PRODUCT_RELEASE_MINOR,
                  "$PROBA/objects/kroutine/myimageenhancement");
      kexit_handler(myimageenhancement_free_args, NULL);

/* -main_get_args_call */
      kclui_init("PROBA", "myimageenhancement", KGEN_KROUTINE,  &clui_uis_spec,
               myimageenhancement_usage_additions, myimageenhancement_get_args,
               myimageenhancement_free_args);
/* -main_get_args_call_end */

/* -main_before_lib_call */
  /* megnyitom az input es az output képet */
  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);
  }


  /* átmásolom az input képet az outputba */
  if ( !kpds_copy_object( in_object, out_object ) ) {
    kerror(lib, rtn, "Can not copy input into the output object\n");
    kexit(KEXIT_FAILURE);
  }

  /* mindkét kép adattípusát DOUBLE-ra állítom */
  kpds_set_attribute( in_object, KPDS_VALUE_DATA_TYPE, KDOUBLE );
  kpds_set_attribute( out_object, KPDS_VALUE_DATA_TYPE, KDOUBLE );

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

  /* lefoglalom a segédtömböket */
  plane = (double* ) kmalloc ( w*h *sizeof( double) );
  res_plane = ( double* ) kmalloc ( w*h *sizeof( double ) );
  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++ ) {
    kpds_set_attribute( in_object, KPDS_VALUE_POSITION, 0,0,0, ct, 0 );
    kpds_get_data( in_object, KPDS_VALUE_PLANE, (kaddr) plane );

    if ( clui_info->enhancement_toggle == 1 ) {    /* GUI elem értékét
                                                      kérdezzük le */
      /* ha az első gomb van benyomva, akkor logaritmust csinálunk */
      for ( ch = 0; ch < h; ch++ ) {
        for ( cw = 0; cw < w; cw++ ) {
          pos = ch * w + cw;

          res_plane[ pos ] = log( plane[pos] );

        }
      }
    } else  {
      /* ha a második gomb van benyomva, akkor az
         intenzitások négyzetét számoljuk */
      for ( ch = 0; ch < h; ch++ ) {
        for ( cw = 0; cw < w; cw++ ) {
          pos = ch * w + cw;

          res_plane[ pos ] = plane[pos] * plane[pos];

        }
      }

    }
    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 */


    kexit(KEXIT_SUCCESS);
}
```