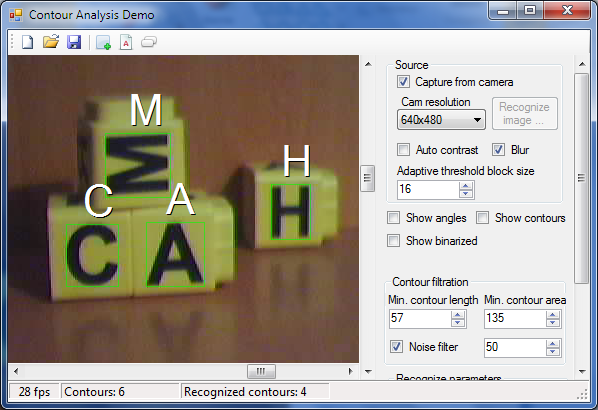
作业要求：

1. 从以下题目中任选一题，在已有代码的基础上，重新设计实现所有的功能之后，再结合自己的理解，进行功能或者界面的特色性工作；
2. 在提交原始代码的时候，还要提交一份说明文档，详细说明自己做的创新性工作，这是要重点阐述的内容。

# 作业三：Contour Analysis for Image Recognition in C#



## Part 1: Bases of the Contour Analysis

### What is Necessary for the Contour Analysis (CA)

The CA allows to describe, store, compare and find the objects presented in the form of the exterior outlines - contours.

It is supposed that the contour contains the necessary information on the object shape. Interior points of the object are not accepted to attention. It restricts area of applicability of algorithms of a CA, but reviewing only contours allows to pass from two-dimensional space of the image - to space of contours and by that to lower computing and algorithmic complexity.

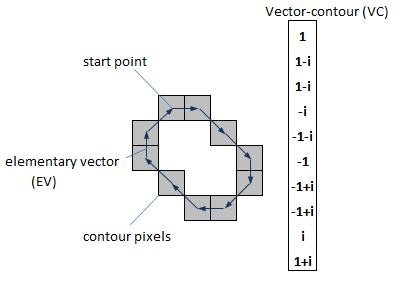
CA allows to effectively solve the main problems of a pattern recognition - transposition, turn and a rescaling of the image of object. CA methods are invariant to these transformations.

### The Main Concepts

At first, we define such an object contour. The contour is a boundary of object, a population of points (pixels), separating object from a background.

In systems of computer vision, some formats of coding of a contour are used - the code of Freeman, two-dimensional coding, polygonal coding are most known. But all these formats of coding are not used in a CA.

Instead, in a CA the contour is encoded by the sequence consisting of complex numbers. On a contour, the point which is called as starting point is fixed. Then, the contour is scanned (is admissible - clockwise), and each vector of offset is noted by a complex number a+ib. Where a - point offset on x axis, and b - offset on y axis. Offset is noted concerning the previous point.



Owing to the physical nature of three-dimensional objects, their contours are always closed and cannot have self-intersection. It allows to define unambiguously a way of bypass of a contour (to within a direction - on or counter-clockwise). The last vector of a contour always leads to the starting point.

Each vector of a contour we will name elementary vector (EV). And sequence of complex-valued numbers - vector-contour (VC).

Vectors-contours we will designate the big Greek letters, and their elementary a vector - small Greek letters.

Thus, vector-contour Γ of length k can be designated as:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f0.png

Why in a CA complex-valued coding is used? Because operations over a contour as over a vector of complex numbers possesses remarkable mathematical properties, in comparison with other modes of coding.

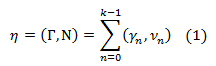
Basically, complex coding is close to two-dimensional coding where the contour is defined as a population of the EVs presented in the two-dimensional coordinates. But a difference between operation of scalar product for vectors and for complex numbers - are various. This circumstance also gives priority to CA methods.

### Properties of Contours

1. The sum of an EV of a closed contour is equal to zero. It is trivial - as the elementary vectors result in starting point, their sum is equal to a zero-vector.
2. The contour-vector does not depend on parallel transposition of the source image. As the contour is encoded relative to starting point, this mode of coding is invariant to shift of an initial contour.
3. Image turn on certain angle is equivalent to turn of each EV of a contour on the same angle.
4. The starting point modification conducts to VC cycle shift. As EVs are encoded concerning the previous point, it is clear that at a modification of starting point, the sequence of an EV will be the same, but the first EV will be what begins in the starting point.
5. The source image rescaling can be considered as multiplication of each EV of a contour to scale factor.

### Scalar Product of Contours

As scalar product of contours, Γ and N are called such complex number:



Where k - dimensionality of a VC, γn - n the elementary vector of contour Γ, νn - n EV of contour N. (γn, νn) - the scalar product of complex numbers calculated as:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f2.png

Let's pay attention to that in a CA the scalar product only a VC of identical dimensionality is supposed. That is the number of the elementary vectors in contours should coincide.

The scalar product of usual vectors and scalar product of complex numbers - differ. If we multiplied an EV as simple a vector, their scalar product would look so:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f3.png

Compare this formula to the formula (2) and you note that:

* Outcome of scalar product of vectors is the real number. And outcome of product of complex numbers - a complex number.
* The real part of scalar product of complex numbers coincides with scalar product of appropriate vectors. That is complex product includes vectorial scalar product.

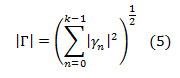
And now let's remember linear algebra. To be exact - physical sense and properties of scalar product. The scalar product is equal in the linear algebra to product of lengths of vectors on a cosine of the angle in between. It means that two perpendicular vectors will always have zero scalar product, collinear a vector - opposite, will give maximum value of scalar product.

These properties of product allow to use it as a certain measure of closeness of vectors. If it is more - the less angle between vectors, the "more close" they are to each other. For perpendicular vectors - it is lowered to zero, and further becomes negative for the vectors directed every which way. It appears, scalar product (1) also possesses similar properties.

Let's introduce one more concept - the normalized scalar product (NSP):

https://www.codeproject.com/KB/graphics/ContourAnalysis/f4.png

Where |Γ| and |N| - the norms (length) of contours calculated as:



The NSP in space of complex numbers, also is a complex number.

Thus, unity is greatest possible value of norm of NSP (it follows from a Cauchy–Bunyakovsky–Schwarz inequality: |ab| <= |a||b|), and it is reached only if...

https://www.codeproject.com/KB/graphics/ContourAnalysis/f6.png

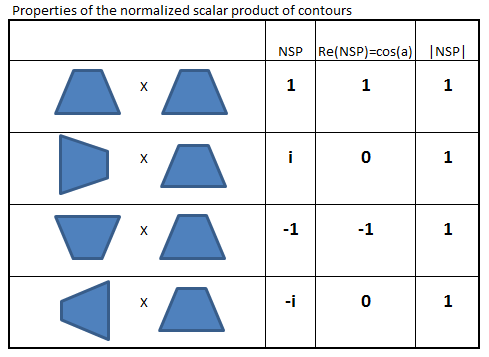
...where μ - the arbitrary complex number.

What does it mean in practice? We recall physical sense of multiplication of complex numbers. At multiplication of complex numbers, their lengths are multiplied, and arguments (angles) - are added. The contour μN means it is the same contour N, but turned both scaled. The scale and turn is defined by a complex number μ.

So, norm of NSP reaches maximum value - unity, only if contour Γ is the same contour N, but turned on some angle and scaled on certain coefficient.

For an example, we consider a scalar multiplication of a contour most on themselves, but turned on a certain angle.

So if to count a NSP of a vector most on itself, we receive NSP=1, if to turn a contour on 90 degrees, we receive NSP=0+i, turn on 180 degrees gives a NSP=-1. Thus, the real part a NSP will give us a cosine of the angle between contours, and the norm of NSP will be always equal to 1.



Similarly, if we increase a VC by some real coefficient (scale) we also receive NSP=1 (it simply to see from the formula (4)).

https://www.codeproject.com/KB/graphics/ContourAnalysis/e.gif **Note**: Norm of NSP is invariant to transposition, rotation and scaling of contours.

So, the norm of the normalized scalar product of contours gives unity only in the event that these two contours are equal to within turn and a scale. Otherwise, the norm of NSP it will be strict less unity.

It is a central conclusion of a CA. Actually, the norm a NSP is an invariant on transposition, rotation and scaling of contours. If there are two identical contours their NSP always gives a unity, is not dependent on where contours are, what their angle of rotation and a scale. Similarly, if contours are various, their NSP will be strict less 1, and also independent of a place, rotation and a scale.

https://www.codeproject.com/KB/graphics/ContourAnalysis/e.gif **Note**: Norm of NSP is measure of contours closeness.

The norm gives a measure of a likeness of contours, and argument a NSP (equal atan(b/a)) - gives us an angle of rotation of contours rather each other.

### Correlation Functions of Contours

In the previous chapter, we ascertained that the NSP is extremely useful formula for search of contours similar among themselves. Unfortunately, there is one circumstance not allowing to use it directly. And this circumstance - a starting point choice.

The matter is that the equality (6) is reached only if starting points of contours - coincide. If contours are identical, but the EV reference begins with other starting point the norm the NSP of such contours will not be equal to a unity.

The matter is that the equality (6) is reached only if starting points of contours - coincide. If contours are identical, but the EV reference begins with other starting point the norm the NSP of such contours will not be equal to a unity.

Let's introduce the concept of intercorrelation function (ICF) of two contours:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f7.png

Where N(m) - a contour received from N by cycle shift by its EV on m of elements.

For an example, if N = (n1, n2, n3, n4), N(1) = (n2, n3, n4, n1), N(2) = (n3, n4, n1, n2) and so on.

What does intercorrelation function show? Values of this function show contours Γ and N are how much similar if to shift starting point N on m positions.

ICF it is defined on all set of integral numbers but as cycle shift on k leads us to an initial contour the ICF is periodic, with phase k. Therefore us will interest values of this function only in limits from 0 to k-1.

Let's discover the magnitude having the maximum norm among values an ICF:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f8.png

From determinations a NSP and an ICF, it is clear that τmax is a measure of similarity of two contours, invariant to transposition, scaling, rotation and starting point shift.

Thus, the norm |τmax| shows a level of similarity of contours, and reaches unity for identical contours, and the argument arg(τmax) gives an angle of rotation of one contour, concerning another.

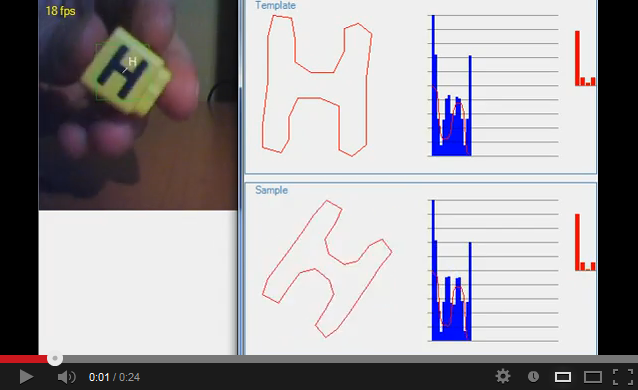
https://www.codeproject.com/KB/graphics/ContourAnalysis/e.gif **Note**: Maximum of norm of ICF is a measure of similarity of two contours.

https://www.codeproject.com/KB/graphics/ContourAnalysis/e.gif **Note**: Maximum of norm of ICF is invariant to transposition, scaling, rotation and starting point shift.

Let's introduce one more concept - an autocorrelation function (ACF). The Autocorrelation function is an ICF for which N=Γ. As a matter of fact is a scalar product of a contour most on itself at various shifts of starting point:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f9.png

Let's consider some properties an ACF:

[](http://www.youtube.com/watch?v=UvuPaUE1cGs)

1. The ACF does not depend on a choice of starting point of a contour. Really, we look at determination of a scalar product (1). As we see, the starting point modification leads simply to a modification of the order of summable elements and does not result to a sum modification. This conclusion is not so obvious but if to ponder upon sense an ACF it is clear.
2. The norm an ACF is symmetric concerning a central reference k/2. As the ACF is the total of pairwise products of an EV of a contour each pair meets two times on an interval from 0 to k.

Let, for example, N = (n1, n2, n3, n4), we write values an ACF for different m:

ACF(0)=(n1,n1)+(n2,n2)+(n3,n3)+(n4,n4)ACF(1)=(n1,n2)+(n2,n3)+(n3,n4)+(n4,n1)ACF(2)=(n1,n3)+(n2,n4)+(n3,n1)+(n4,n2)ACF(3)=(n1,n4)+(n2,n1)+(n3,n2)+(n4,n3)ACF(4)=(n1,n1)+(n2,n2)+(n3,n3)+(n4,n4)

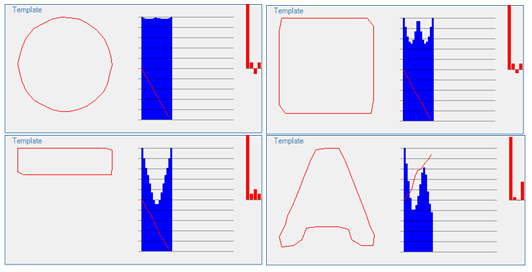
Let's note that items in an ACF(1) same, as in an ACF(3), to within permutation of factors. And recalling that for complex numbers (a, b) = (b, a)\*, we receive that an ACF(1) = ACF(3)\*, where \* - a conjugate complex number sign.

And as |a\*| = |a| turns out that norms of ACF(1) and an ACF(3) - coincide.

Similarly, norms an ACF(0) and an ACF(4) coincide.

Further, everywhere under an ACF we will understand only a function part on an interval from 0 to k/2 as the remaining part of function - is symmetric to the first part.

1. If the contour has any symmetry to turn its ACF has similar symmetry. For an example, we bring graphics an ACF for some contours:



In pictures, the norm the ACF is represented by dark blue color (an ACF it is represented only for an interval from 0 to k/2).

As we see, all contours, except last have symmetry to turn that the ACF leads to symmetry. The last contour of such symmetry has no, and the graph its ACF - is not symmetric.

1. In a sense, it is possible to consider a contour ACF as characteristic of the shape of a contour. So, shapes, close to a circle have the uniform values of the norm an ACF (see a picture for a circle). Shapes strongly prolated in one direction - have a dip in a central part an ACF (see a rectangle picture). The shapes passing in at turn, have a maxima an ACF in an appropriate place (the quadrate ACF see).
2. Normed the ACF does not depend on a scale, position, rotation and a choice of starting point of a contour. It follows from point of 1st and from properties a NSP.

Video showing this fact:

On it, we finish a theoretical part of a CA. In this part, I brought only the main concepts and theoretical calculations which I will immediately apply for pattern recognition in the second part.

## Part 2: Practical Application of the Contour Analysis

### The General Algorithm of Recognition

So, we will solve the pattern recognition task on the image.

The general sequence of an operation at recognition looks so:

1. Preliminary handling of the image - smoothing, a filtration of noise, a contrast raise
2. Binarization of the image and selection of contours of objects
3. Initial filtration of contours on perimeter, squares, to a crest factor, fractality and so on
4. Coercion of contours to uniform length, smoothing
5. Search of all discovered contours, searching of the template maximum similar to the given contour

We will not consider points 1 and 3, they are specific to application area, and have the small relation to a CA.

Further we consider an algorithm body - searching and comparing of contours with templates. And then - we stop on binarization, coercion to uniform length and smoothing of contours a little.

### Estimation of Performance of Algorithms in a CA

At once, we make an estimation of high-speed performance of the algorithms based on a CA.

Let the image already binarized and on it contours are selected.

As further we will work only with points of contours, we estimate their general amount on the image.

For this purpose, we take the image a size n\*n pixels. Then breed its uniform grid with a step s. The total length of all grid lines is:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f10.png

It turns out that passage from the plane two-dimensional image to contours does not reduce dimensionality of the task. We as before work in complexity O(n2).

However, here it is possible to make some notes:

1. The brought estimation is extremal. In the real image, not all contours have the minimum size, and they do not cover all square of the image. Hence, the number of contours and their total perimeter can be much less 2n2/s. Ideally - if the image contains one object, without noise the CA will work only with one contour, and its number of pixels makes not so square, and the linear association from n. In a case of operation with the image as with a two-dimensional array of pixels (for example, applying the cascade filters of Haar) - we always work with all pixels of images irrespective of, how many objects on it are represented.
2. As the image in the form of contours already has natural segmentation - is divided into contours it is possible to carry out a filtration of parts of the image to simple indications. Among them - contour square, perimeter, the ratio of quadrate of perimeter to squares. Thus, there is enough simple and effective mechanism of a preliminary filtration of parts of the image.
3. Basic algorithms of two-dimensional methods, as a rule, are not invariant to a scale (SURF, SIFT) or to rotation (Haar). Therefore, actually, they work in three-dimensional space where one more measurement appears because of necessity to sort out scales or angles of rotation. As CA methods are invariant to a scale and rotation here such problem does not exist (elimination - uninvariance to starting point, but more low we consider effective methods of struggle against this complexity).
4. The CA allows to process the image in a progressive mode. It means that we can sort contours on any to an indication (for example, by square or on a gradient of boundaries, or on brightness, etc.). And then to treat the first contour, and to produce outcome. Remaining contours to process in a background mode. It means that the first outcome (and in many application-oriented tasks it and it is necessary) can be received for O(n) that is an excellent estimation for algorithms of pattern recognition.
5. As contours are independent from each other algorithms of a recognition it is easy to parallelize. Besides, algorithms are very simple and can be executed on graphic processors.

All above-stated speculations concern only handlings of contours. Certainly, the stage of obtaining of contours does not disappear anywhere, and it demands at least O(n2). However, in practice, this stage occupies no more than 10-20 % from the general time of handling.

The population of these factors essentially reduces a constant of computing complexity, and at the modern progress of computers, the CA can quite be used as algorithm of real time.

Let's up reasons of complexity O(n2). The contour, as a matter of fact, is one-dimensional object. It is a vector of complex-valued numbers. However the count of contours grows proportionally to image size quadrate. From here also, there is complexity O(n2). From this, it is possible to draw a conclusion that the most effective mode of performance gain is reduction of count of contours by a preliminary filtration.

Now we estimate processing complexity separately the taken contour.

In the previous part, we became clear that for comparing of contours, it is necessary to discover a maxima an ICF (formulas 7, 8). We estimate complexity of an evaluation. The scalar product demands O (k) operations, where k - length of a contour. And the ICF needs to be produced for an evaluation k evaluations of scalar products. Means, the ICF demands evaluations of order O (k2). Besides, considering that comparing goes not with one template, and with set of templates the total time of searching of a template for a contour can be estimated as

https://www.codeproject.com/KB/graphics/ContourAnalysis/f11.png

Where k - length of a contour, and t - number of template contours.

It is not such a good estimation. In the following chapter, we will struggle with it, and we will manage to reduce essentially a constant at this estimation.

Total, complexity of identification of all contours of the image makes:

https://www.codeproject.com/KB/graphics/ContourAnalysis/f12.png

Where - n - the linear image size, k - length of a contour, t - count of templates.

Complexity of identification of the first contour (in a progressive mode):

https://www.codeproject.com/KB/graphics/ContourAnalysis/f13.png

### Contour Descriptor

As it has been shown in the previous chapter, complexity of an evaluation of ICF is O(k2). Despite that k - a constant (to uniform length we consider process of coercion of contours further), all the same, an evaluation an ICF - labor-intensive process. At k=30 on an evaluation the ICF is required 900 operations of multiplying of complex numbers. And at presence t=1000 templates, searching of a template for a contour demands about one million multiplyings of complex numbers.

For fast searching of templates, it is necessary to introduce the certain descriptor characterizing the shape of a contour. Thus, close among themselves contours should have the close descriptors. It would save us the procedure of an evaluation an ICF of a contour with each template. Would be to compare only descriptors and if they are close - only in that case enough - to calculate an ICF. Comparing of descriptors should be fast. Ideally, one number should be a descriptor.

Unfortunately, in the literature, I did not discover explicit exposition of the suitable descriptors describing the shape of a contour. Nevertheless, such descriptor exists.

In the first part, we explicitly considered properties of an autocorrelation function. In the same place, we marked that an ACF invariantly to transposition, rotation, scaling and a starting point choice. And besides, the ACF is a function of one contour, instead of two, as an ICF.

Proceeding from it, the ACF can be selected as the descriptor of shape of a contour. The close contours will always have the close values an ACF.

https://www.codeproject.com/KB/graphics/ContourAnalysis/e.gif **Note**: ACF is shape descriptor of a contour.

Comparing two ACF has complexity O(k) that already considerably it is better than O(k2) for an ICF. Besides, as we remember, an ACF because of symmetry, is defined only on an interval from 0 to k/2 that else twice reduces time of evaluations.

If the base of templates stores their ACF, searching of a template for a contour, by comparing the ACF, makes O(kt). It is already good estimation. But also to us it was to be refined.

For reduction of time of comparing an ACF, I apply wavelet convolution an ACF. I will omit description of this process, it is possible to look at it in source codes.

Let's mark the following:

1. Comparing an ACF, generally, does not save us of the necessity of an evaluation an ICF. Only the ICF states an exact estimation of closeness of contours. The ACF, generally, can coincide for various contours. But, thus, preliminary selection of templates on an ACF essentially narrows down count of candidates on comparing on an ICF.
2. In some cases, comparing the ACF can be enough for identification of contours. If on the image markers with the simple geometrical shape are searched, and the probability of failed recognitions is not essential, it is possible to omit process of comparing an ICF, having restricted only to comparing of convolutions an ACF. Such algorithm gives the greatest system performance.

### Equalization of Contours

As it has already been marked in part 1, CA methods assume identical length of contours. In the real image contours have arbitrary length.

Therefore, for searching and comparing of contours, all of them should be led to uniform length. This process is called equalization.

At first, we fix length of a VC which we will use in our system of a recognition. We designate it k.

Then, for each initial contour A we create vector-contour N in length k. Further probably two variants - or the initial contour has greater number of an EV than k, or smaller number than k.

If an initial contour more necessary it is sorted out all by its EV, and we consider elements N as the total of all EVs, as follows (C#):

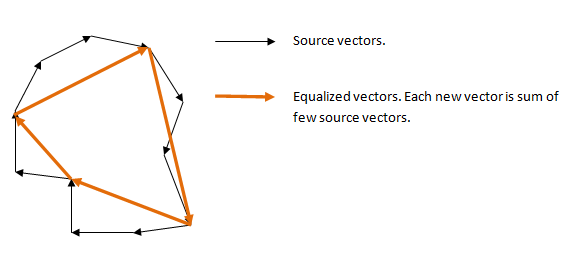
Hide   Copy Code

Complex[] newPoint = new Complex[newCount];

for (int i = 0; i < Count; i++)

newPoint[i \* newCount / Count] += this[i];

The picture shows the meaning of equalization:



This algorithm is rough enough, especially for lengths the little big k, however it quite puts into practice. If the initial contour is less k we produce interpolation, and is considered approximately so:

Hide   Copy Code

Complex[] newPoint = new Complex[newCount];

for (int i = 0; i < newCount; i++)

{

double index = 1d \* i \* Count / newCount;

int j = (int)index;

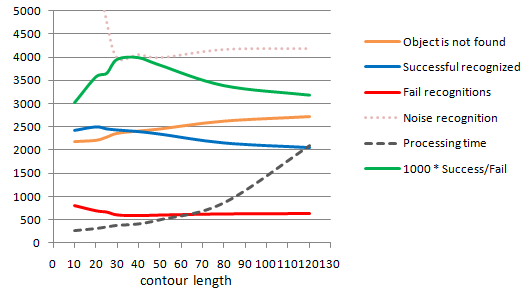
double k = index - j;

newPoint[i] = this[j] \* (1 - k) + this[j + 1] \* k;

}

There is a problem which needs to select value k, what value is optimal? The answer to this problem is completely defined by specificity of application area. On the one hand, the big length k means the big expenditures on evaluations. On the other hand - small values k carry less information, and accuracy of a recognition decreases, and noise recognition - increases.

The diagram of the main parameters of system of a recognition, depending on the selected length of a VC:



This diagram was under construction for system of a recognition of Latin letters, on the basis of tests [ICDAR](http://algoval.essex.ac.uk/icdar/Datasets.html).

As we see, at great values (from 80 and above) lengths of a contour, system parameters are stabilized and change (except time of handling which, naturally, increases) a little.

At small values k (less than 30) - the number of noise recognitions (a recognition of noise or other not character image elements - as letters), decreases number of successful recognized, and the number of false recognitions (fail recognitions) increases.

Thus, value k=30 is optimal for the given system of recognition.

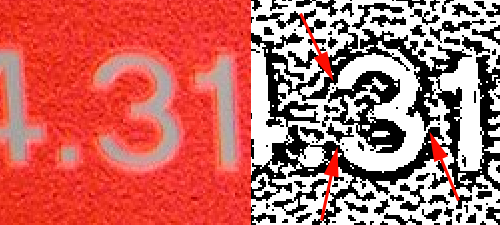
Pay attention that magnification of length of a contour, after certain level (25 and above) at all does not lead to improving of quality of a recognition. It is connected by that in the described method, equalization is led simultaneously with smoothing of contours. At great values of length, smoothing becomes more and more small-scale, and the contour becomes too detailed, and therefore, more distinct from template contours (in the brought test numerals of learning sampling have no exact coincidence to test images).

### CA Limitation

The CA has two groups of factors negatively influencing outcomes of a recognition.

The first group of factors is connected to a problem of selection of a contour on images. The contour is strictly certain discrete structure. However the objects feeblly expressed on an environmental background have a great number of real images. The object cannot have a clear boundary, it can be identical on brightness and color with a background, it can be noised and so on. All these factors lead to that the contour or cannot be selected generally, or it is selected incorrectly, and does not correspond to boundary of objects.

In a picture on the right, it is shown as a binarized image. Small "bridges" between the image of digit and an environmental background makes a digit contour unrecognizable by CA methods.



Such cases are very heavy for a CA. After all, the CA makes sense, only in that case when the object contour is defined unambiguously correctly in all points.

The second group of the factors complicating a CA, is connected to principles of the contour analysis. CA methods assume that the contour describes all object bodily, and does not suppose any intersections with other objects or incomplete visibility of object.

On a picture on the right - binarized image. It is visible that the photo flare, going horizontal line, does letters indiscernible for a CA.



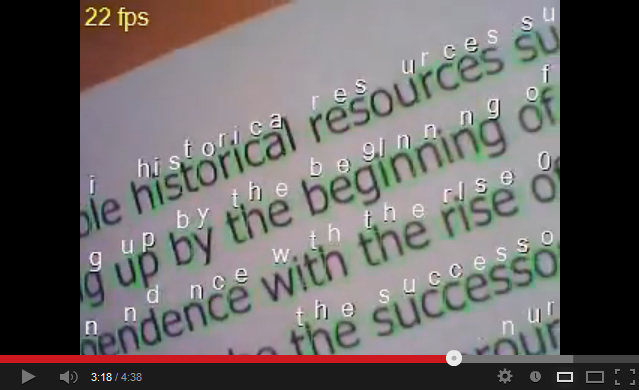
Thus, the CA has feeble stability to parasites, does not suppose intersection or the partial visibility of object.

### Outcomes of Testing of a Method

Despite limitation, CA methods are attractive for the simplicity and high-speed performance. In the presence of accurately expressed object on a contrasting background and lack of parasites of a CA well copes with a recognition.

Testing of methods of a CA for tests ICDAR yields outcome of 48% of the recognized symbols. It is quite a good outcome, considering that this test is very uneasy, and contains a great number bad-read images of letters. Thus, the CA treated 249 images of a various size (from 400x400 to 1280x960) for 30 seconds.

Besides recognition of freeze frame images, high-speed performance of a CA allows to process video in a real-time mode. The video showing a possibility of a CA:

[](http://www.youtube.com/watch?v=BHsaZyCjb40)

## Part 3: ContourAnalysis Library

The library includes two projects. The first project ContourAnalysis - implements base functions of the contour analysis - creation of contours, a scalar product of contours, equalization, evaluation ICF and ACF, comparing and searching of templates.

The class Contour - creates and stores contours. It contains basic operations for contours - scalar product, scaling, equalization, normalization, a spectrum evaluation, evaluation ACF and ICF.

The class Template is used for creation of base of templates. This class stores a contour, it ACF, descriptors ACF, the linear parameters of an initial contour (area), norm of a contour. Also, the template has a name which is used as the recognized value.

Class TemplateFinder implements fast searching of a template for the given contour. Outcome of operation of this class is FoundTemplateDesc which contains an initial contour, and the template discovered for the given contour. Besides, FoundTemplateDesc contains similarity rate, angle of rotation and a scale of a contour, relative to a template.

The second project - ContourAnalysisProcessing - contains methods for preliminary handling of the image, selection of contours, their filtrations and a recognition. Also, it contains tools for automatic generation of templates for recognition of printing symbols.

Project ContourAnalysisProcessing uses library OpenCV (EmguCV .NET wrapper) for operation with the image.

The class ImageProcessor is used for image handling. It, also, stores base of templates.

Method ImageProcessor.ProcessImage() receives the image on an input. Outcome of operation are lists discovered contours (ImageProcessor.samples) and the list of the recognized contours (ImageProcessor.foundTemplates).

The class ImageProcessor contains also settings for searching of contours.

The ImageProcessor works as follows:

1. At first, the image will be transformed to gray-scale:



1. Then it is binarized by AdaptiveThreshold:



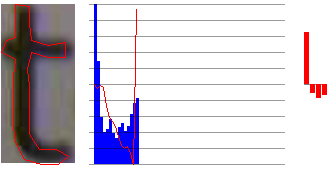
1. Contours are extracted:



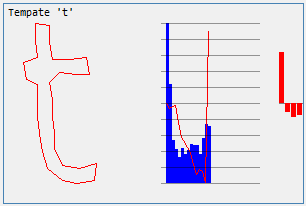
1. Contours are filtered on linear parameters (length, square, etc.):



1. Contours are equalized, there is calculation ACF and ACF descriptors:



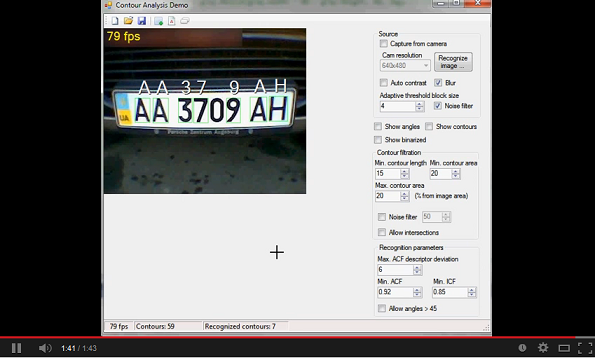
1. And then there is maximum suitable template:



The static class TemplateGenerator is used for automatic generation of templates of numerals of a certain font.

Except two library projects, there is a demo-example showing operation of library with the webcam. The demo contains tools for creation and editing of templates, recognition tuning, and allows to produce recognition of contours from the webcam, and also allows to create the augmented reality.

## How to "train" CA

[](http://youtu.be/xZI38ECeW0c)