HECTOR RODRIGUEZ

Paper: **Flowpoints**: A Generative Image Creation Method Using Optical Flow

Topic: Algorithmic Art

Author:
Hector Rodriguez
City University of Hong Kong,
School of Creative Media
Hong Kong

Abstract:
Flowpoints is a research-creation art project about the experimental exploration of algorithmic processes. Its main focus is the Lucas-Kanade (LK) optical flow algorithm, a technique widely used in computer vision for such purposes as motion tracking and the computation of stereo disparity. Instead of regarding the internal structure of the algorithm as an opaque black box, the Flowpoints project proceeds by opening up the black box and analyzing how LK actually works. The aim is to discover unforeseen effects that can only arise through the experimental visualization/sonification of the data structures and operations constitutive of the algorithm. [1]

This presentation will describe in detail the internal structure of the algorithm and the ways it can be subverted for artistic purposes. The outcomes will include abstract line renderings as well as sounds. The latter are generated by computing the formant shifts occuring across two different spectral magnitude envelopes and using this information to deform the original sound samples. [2]

This creative methodology thus refuses to regard technology as the transparent conduit of some previously fixed intention. To use the terminology of actor-network theory, technologies now function as mediators rather than intermediaries. [3] The assumption at the heart of the Flowpoints project is that the detailed and systematic exploration, modification, and visualization of an algorithm like LK will suggest novel artistic ideas.

Image from Flowpoints.

References:
[1] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of Imaging Understanding Workshop*, 1981, pp. 121-30
[2] Tony Ezzat et al., Morphing Spectral Envelopes Using Audio Flow. http://cbcl.mit.edu/publications/ps/audioflow.pdf
[3] Bruno Latour, Pandora's Hope: Essays on the Reality of Science Studies. Cambridge, Mass.: Harvard University Press, 1999.

Contact:
smhect@cityu.edu.hk

Keywords:
Algorithmic art, optical flow, audio flow, experimental art

Flowpoints: A Generative Method Using Optical Flow

Dr. Hector Rodriguez, PhD.
School of Creative Media, City University of Hong Kong, Hong Kong
http://sweb.cityu.edu.hk/flowpoints
e-mail: smhect@cityu.edu.hk

Abstract

Flowpoints is an experimental research-creation art project about algorithmic processes. Its main focus is the Lucas-Kanade (LK) optical flow algorithm, a technique widely used in computer vision for such purposes as motion tracking and stereo disparity. Instead of regarding the internal structure of the algorithm as an opaque black box, the Flowpoints project proceeds by opening up the black box and analyzing how LK actually works. The aim is to discover unforeseen effects that can only arise through the experimental visualization/sonification of the data structures and operations constitutive of the algorithm. This paper describes in detail the internal structure of the algorithm and the ways it can be subverted for artistic purposes. It proposes a creative methodology that refuses to regard technology as the transparent conduit of some previously fixed intention. To use the terminology of actor-network theory, technologies now function as mediators rather than intermediaries. The assumption at the heart of the Flowpoints project is that the detailed and systematic exploration, modification, and visualization of an algorithm like LK will suggest novel artistic ideas.

Intermediaries and Mediators

My recent artistic work explores the artistic possibilities of algorithms. To clarify the method, I shall draw a distinction between two ways of looking at algorithms, which can be termed instrumental and procedural.

The instrumental perspective is concerned with the algorithm's outcome. This is the perspective of a "user" or "client" who needs a practical solution for the problem that the algorithm was designed to tackle: data compression, motion tracking, anti-aliasing, etc. The user only cares about the algorithm's accuracy and efficiency. The details of the procedure are from this standpoint not important. The algorithm is thus viewed as a means to an end, a black box whose internal operation need not be known. Many digital artists use commercial software libraries this instrumental way.

In contrast, the procedural perspective is concerned with the internal structure of the algorithm. This approach constitutes the standpoint of the scientist or programmer who wants to know the procedural details, possibly with the aim of writing her/his own implementation. A programmer often regards an algorithm as the expression of a way of thinking whose logic has to be understood. To understand the algorithm is not only to understand how it works but also to appreciate those features that make it elegant and ingenious. Thus the programmer who knows the inner working of the technology also comes to admire it as a paradigm of intelligence and precision.

The distinction between instrumental and procedural standpoints corresponds to the distinction between intermediaries and mediators proposed by sociologist Bruno Latour in the tradition of Actor-Network Theory [1]. An intermediary is any structure that transports or fulfills some prior meaning or intention without changing it. Its behavior is completely determined by its function: inputs generate outputs in a reliable and predictable manner. Examples include any machine whose operation has become routine. An intermediary regularly behaves as expected, and so can be used as the transparent conduit of some stable intention. It gives a technical repertoire of possible functions that can be safely taken for granted. An intermediary whose internal composition is opaque to users is a black box. The competent user of a black box need not understand how the device actually works. Most computational tools used by artists, such as for instance standard graphics software packages, are black boxes in this sense. They provide a stock of already implemented algorithms for image (or audio) processing that can be approached from a purely instrumental standpoint.

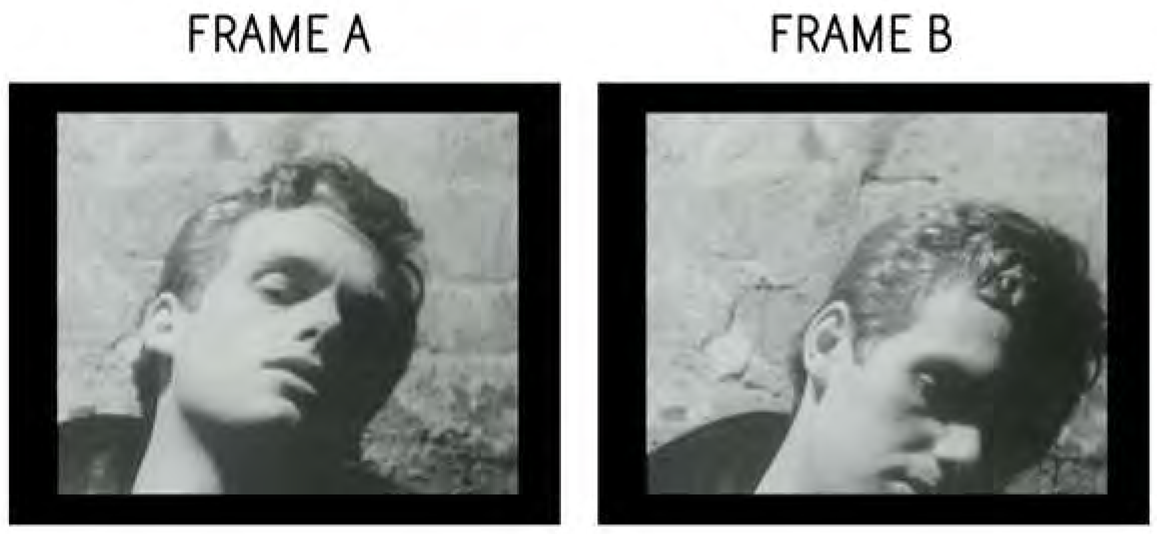
In contrast, a mediator does not just execute the intentions of its users but transforms them in unpredictable ways. A technology can become a mediator, for instance, when it breaks down. The mediator is not a transparent channel for the realization of predefined objectives. It cannot be safely taken for granted as a stabilized repertoire of techniques ready-to-hand. Its structure and behavior interferes with the execution of prior plans and often leads to the formation of unforeseen goals.

My artistic agenda involves transforming algorithms from intermediaries into mediators by adopting a procedural point of view. I typically begin by choosing an existing algorithm and studying its internal logic without forming any plans as to how I will eventually come to use it in my artistic practice. I do not attempt to form a preconceived idea of what the end result will look like. This refusal to specify an a priori outcome prevents me from adopting an instrumental viewpoint. The algorithm is not a means to solve some pre-defined design problem. Instead, I aim to discover unforeseen artistic effects that can only arise through the experimental visualization, sonification, and manipulation of the data structures and operations constitutive of the algorithm. I thus allow my procedural understanding of the algorithm to suggest novel artistic ideas that take the project in potentially unexpected directions. The algorithm in a sense leads the creative process.

Flowpoints is an ongoing experimental research-creation project whose subject is the Lucas-Kanade (LK) optical flow algorithm, a technique widely used in computer vision for such purposes as motion tracking and stereo disparity. Instead of regarding the internal structure of the algorithm as an opaque black box, the Flowpoints project proceeds by opening up the black box and analyzing how LK actually works. The algorithm is then used as a source of artistic ideas. This paper describes the artistic results achieved thus far in the area of image and sound synthesis, but the current examples are only meant as illustrations of the creative potential unleashed whenever algorithmic thinking is viewed as a topic of artistic exploration. I begin by giving a non-technical overview of LK.

The Lucas-Kanade Algorithm

LK is one of several optical flow algorithms widely used in computer vision [2]. Optical flow methods aim to align two images, a task frequently described in the technical literature as “image registration”. A typical image registration task is motion tracking. Given two images A and B, such as for instance two consecutive frames in a movie, the goal is to measure any motion from A to B.



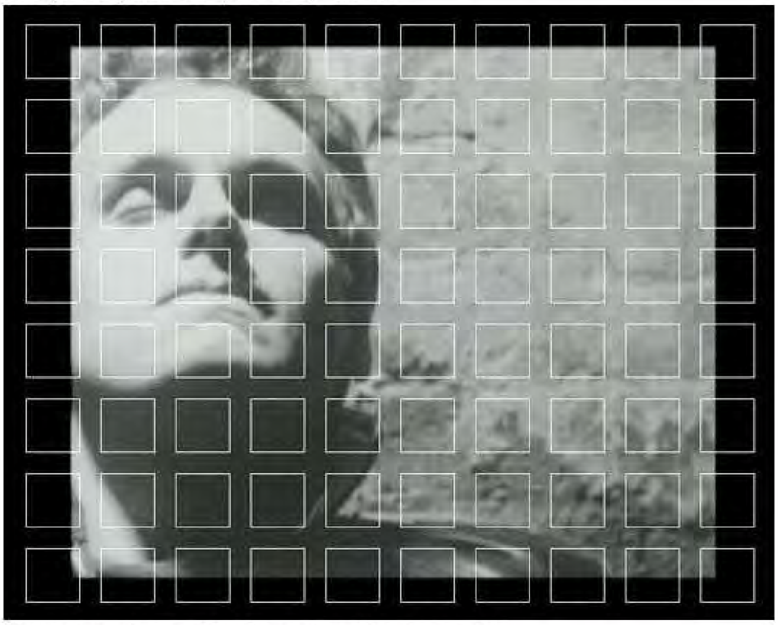
The algorithm makes several assumptions:

- Every image consists of a two-dimensional array of pixels.
- Each pixel is uniquely identified by an ordered pair of numbers (x, y) that give its location on the image.
- The relevant property of a pixel is its brightness.

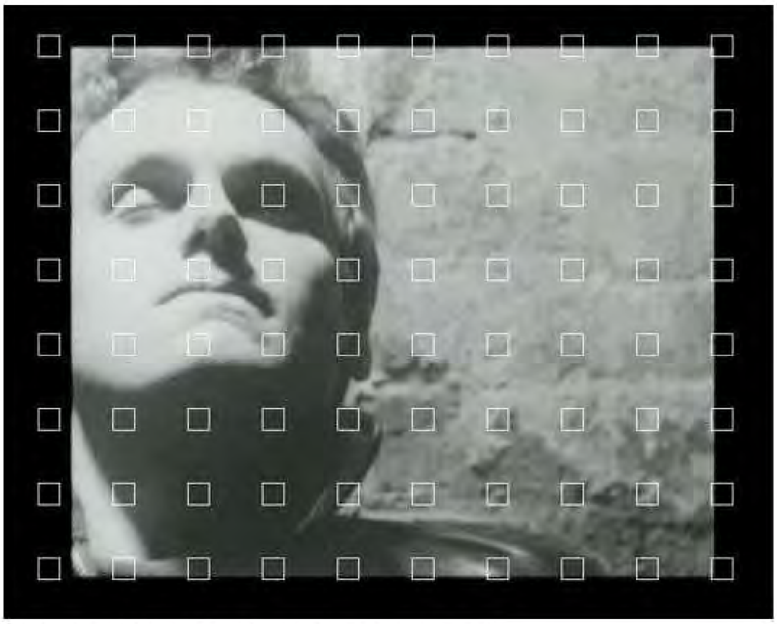
LK also assumes that clusters of contiguous pixels move together as a whole from one image to the next. Thus the procedure defines a set of subregions or “flowpoints” in the image. The goal is to track the motion of each of those flowpoints from one frame to the next.

The programmer has to choose the size of each flowpoint and the distance between them. Two different settings are shown here.

flowpoint grids on (720x576) frame:

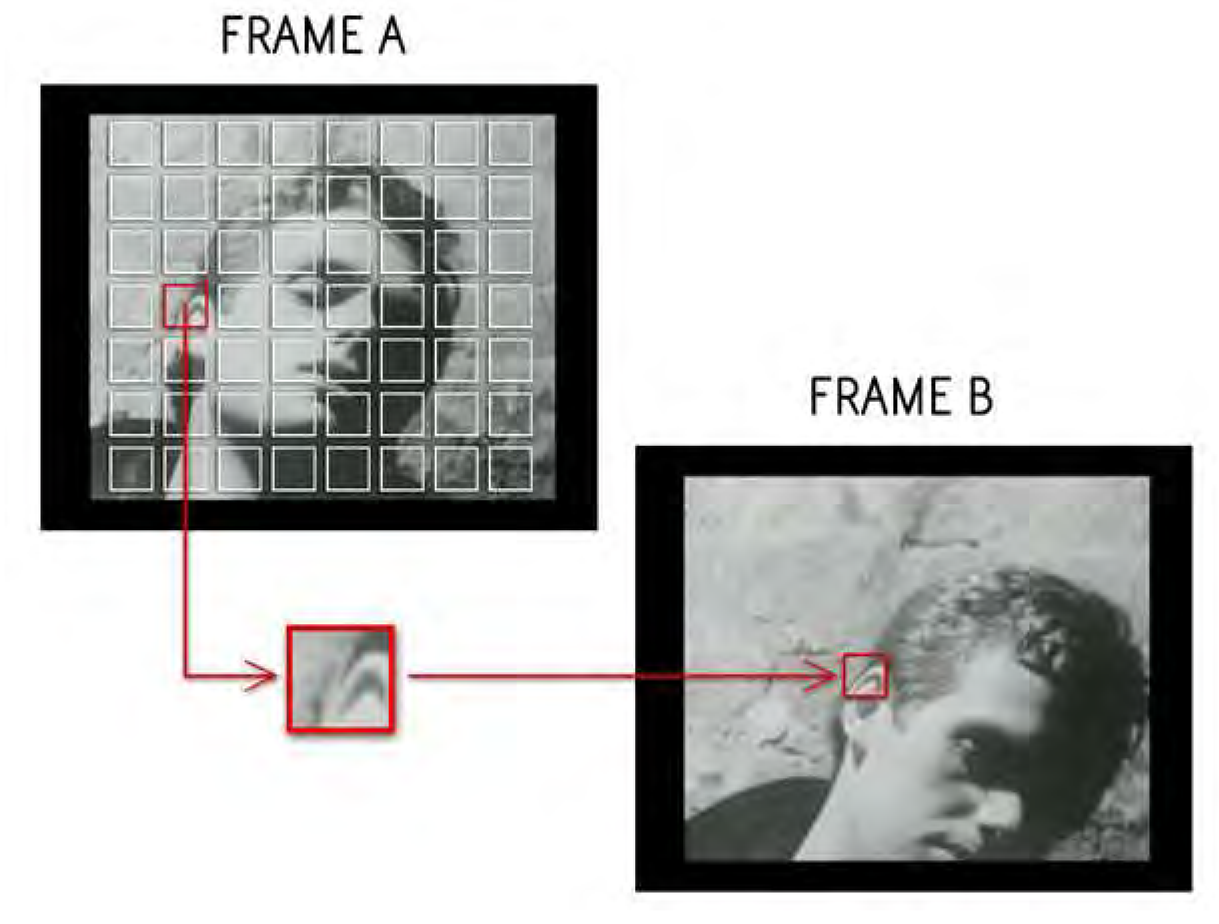


flowpoint size = 50, flowpoint distance = 20



flowpoint size = 20, flowpoint distance = 50

The objective of the algorithm is to align each flowpoint in frame A (or “template”) with some matching area of frame B.



Suppose that one flowpoint has moved 50 pixels along the x axis and 60 along the y axis. This displacement along two directions constitutes the motion vector for that flowpoint. The result of applying the LK algorithm to two distinct frames is a motion vector field correlating every template to some area of the second image (the “target”). This vector indicates a likely displacement of the pixels from frame A to frame B. The algorithm is not guaranteed to identify the correct motion vector in every case, and so only gives a hypothetical estimate.

How does the procedure decide that a match has been found? To compare two pixels is to compute the difference in brightness between them. Suppose that the brightness values of the pixels, on a scale of 0 to 255, are 130 and 134. The difference in brightness would in this case be 4. Call this the margin of error between the two pixels. The idea is that the error is directly proportional to the disparity between the two pixels. To compute the error between two flowpoints, we add up the errors of the corresponding pixels in the two flowpoints. (In effect, the algorithm uses a slightly more complicated measure known as the sum of squared error) [3]. This approach suggests that if the brightness of each corresponding pixel is identical (i.e., if the sum of all the differences is 0) then a match has been found. But such a perfect match is unlikely to happen, so a more realistic criterion of success is that the error taken across all corresponding pixels is “small enough”, in the sense that it falls below a certain threshold chosen by the programmer. The goal of image registration, then, is to minimize the error between the pixels of the template and those of the target.

How to go about searching for a match? A simple method would involve searching through all possible areas of frame B to identify the likeliest match. Assuming that each flowpoint is 5 x 5 pixels in size for instance, this approach would begin with one flowpoint in frame A (say, the top left flowpoint) and compare this template against every possible 5 x 5 subregion of frame B, selecting the one with the smallest error. If this error falls below the chosen threshold, then the target has been found. The pixels in the template have probably moved to the target region of frame B. The algorithm would then repeat the same procedure for every other flowpoint in frame A.

This method is simple to understand and implement, but its actual execution would in most cases be hugely inefficient, since time would often be wasted comparing each template against subregions of frame B where it is unlikely to have moved. An alternative method would avoid looking through every possible area of frame B by making the search somehow more targeted and intelligent. LK gives such a method. Its beauty lies in the manner in which it tackles this problem.

Suppose that we are attempting to find the target in frame B for one given template in frame A. The algorithm begins by proposing a hypothesis about its possible displacement in frame B. It then computes the margin of error between the template and its hypothetical target in frame B, and uses this error to generate a new hypothesis with a smaller error. Every iteration attempts to reduce the error previously made, using a technique known as gradient descent optimization, which relies on spatial gradient information. The previously hypothesized motion vector becomes an input parameter into the algorithm, which will produce a new and more exact vector in the next iteration. The procedure continues until a match is found (i.e., the error falls below the specified threshold).

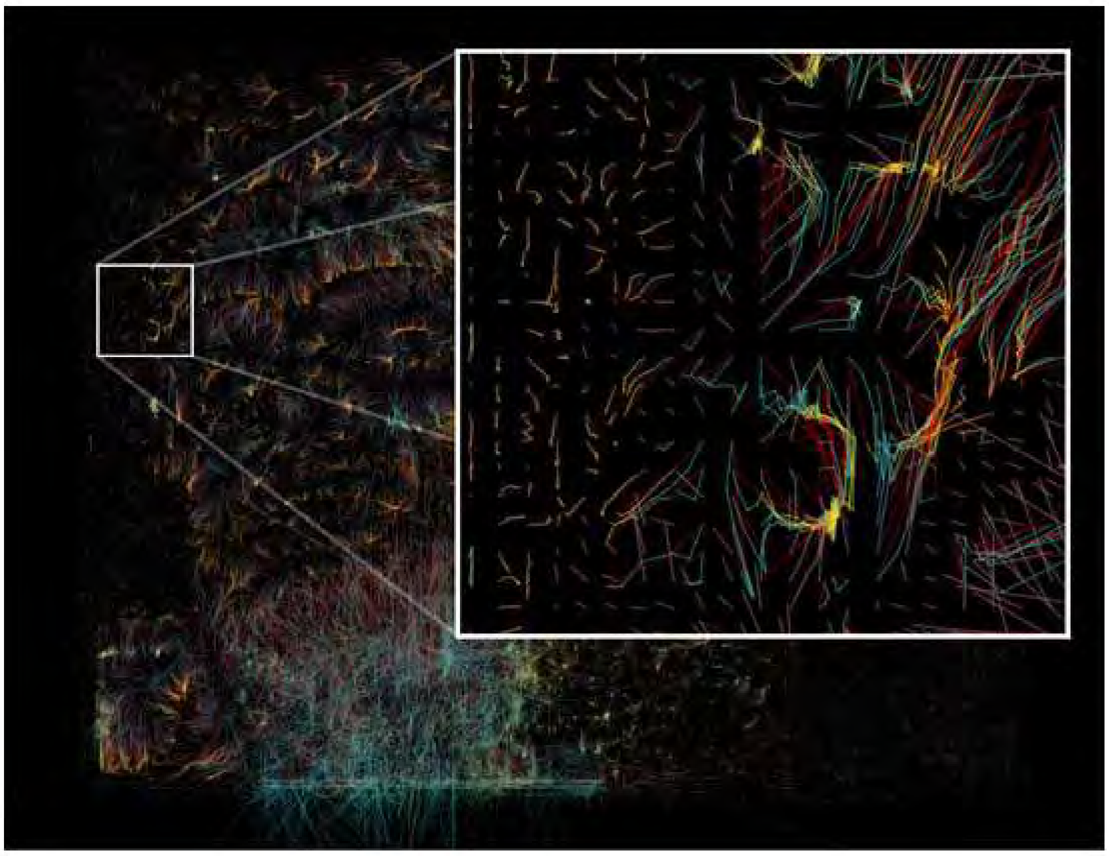
Every successive hypothesis is essentially a probe. The result of every step typically contains some margin of error, but the method is capable of measuring the error and refining its hypothesis in the next iteration. The intelligence and beauty of the algorithm does not consist in its immunity from error, but in its ability to identify the current error and use it to generate a better hypothesis in the next step. It unfolds a process of probing and testing that uses error as an enabling element. This approach somewhat resembles the theory of induction of C. S. Peirce: “Induction is the experimental testing of a theory. The justification of it is that, although the conclusion at any stage of the investigation may be more or less erroneous, yet the further application of the same method must correct the error.” [4]

An important parameter, chosen by the programmer, is the maximum number of iterations that the algorithm is allowed to continue probing and testing. If its value is 50, for instance, the algorithm will stop looking for a match after 50 attempts. The same procedure is applied in turn to all flowpoints in frame A. The outcome is a vector field that indicates the probable motions of all flowpoints.

Image Synthesis

The execution of LK proceeds by generating, testing, and refining hypotheses about the possible displacement of a template. Every iteration proposes a possible motion vector, computes the error, and generates a refined motion vector. Thus the algorithm continually produces and discards hypothetical motion vectors. The user who relies on some existing implementation of LK does not see this underlying interplay of probing and testing. The process is invisible to the user, who only cares about the algorithm's results. The data produced along the way is simply discarded. In contrast, the artist who writes her/his own implementation of the algorithm can take into consideration every motion vector produced in the iterative execution of the algorithm. The data that is normally discarded now becomes a material for the artist to work with.

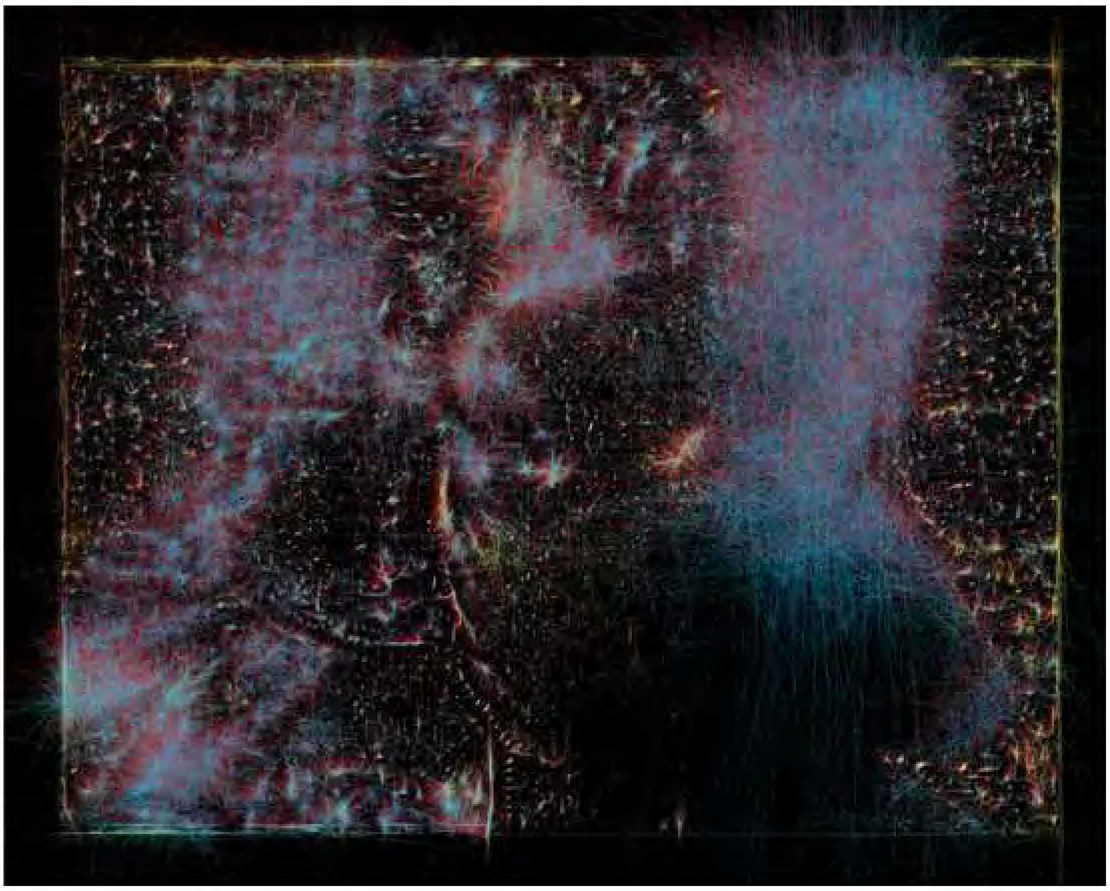
The latter becomes the essential approach taken in the Flowpoints project. It consists of a series of line renderings that visualize the execution of the LK algorithm. Suppose that one template is centered at point (10, 5) of frame A. The algorithm will first attempt to match it against the region centered at point (10, 5) of frame B. If the match fails, the algorithm might hypothesize that a more likely match can be found at point (15, 5). If the attempt is again unsuccessful, the next hypothesis might be at point (15, 3). My approach is to visualize the path taken by the search for an optimal match. This involves drawing a line segment from point (10, 5) to point (15, 5), and again from point (15, 5) to point (15, 3). The segments representing earlier iterations are drawn in warmer colors. The color becomes colder in the later iterations.



There are several parameters that determine the graphical appearance of each image, including:

1. The size of each flowpoint, which I have allowed to range from 4 by 4 to 50 by 50.
2. The distance between any two contiguous flowpoints.
3. The maximum number of iterations, which I have allowed to range from 1 to 50.

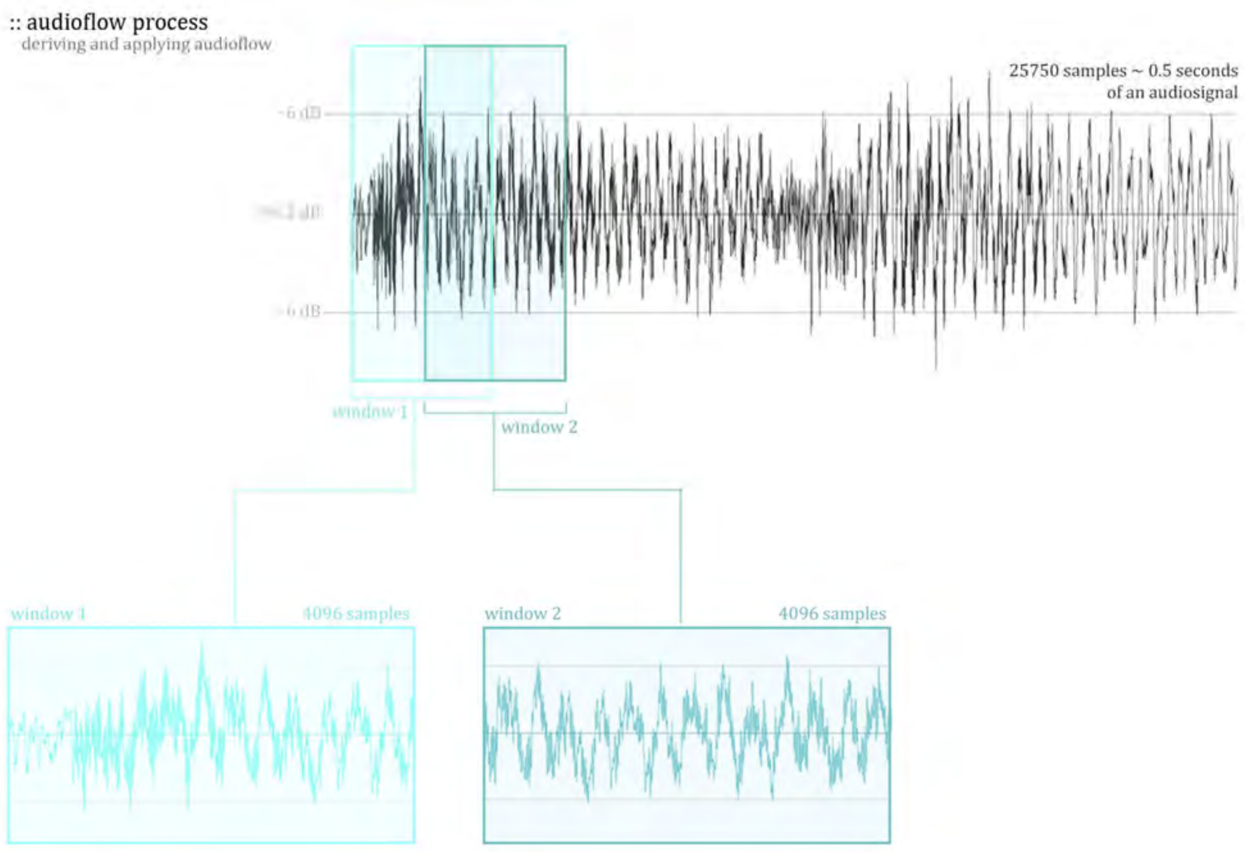
I proceed by selecting a movie clip and trying out different values of the above parameters to generate abstract linear renderings that can be viewed online or displayed as digital prints and video installations [5].



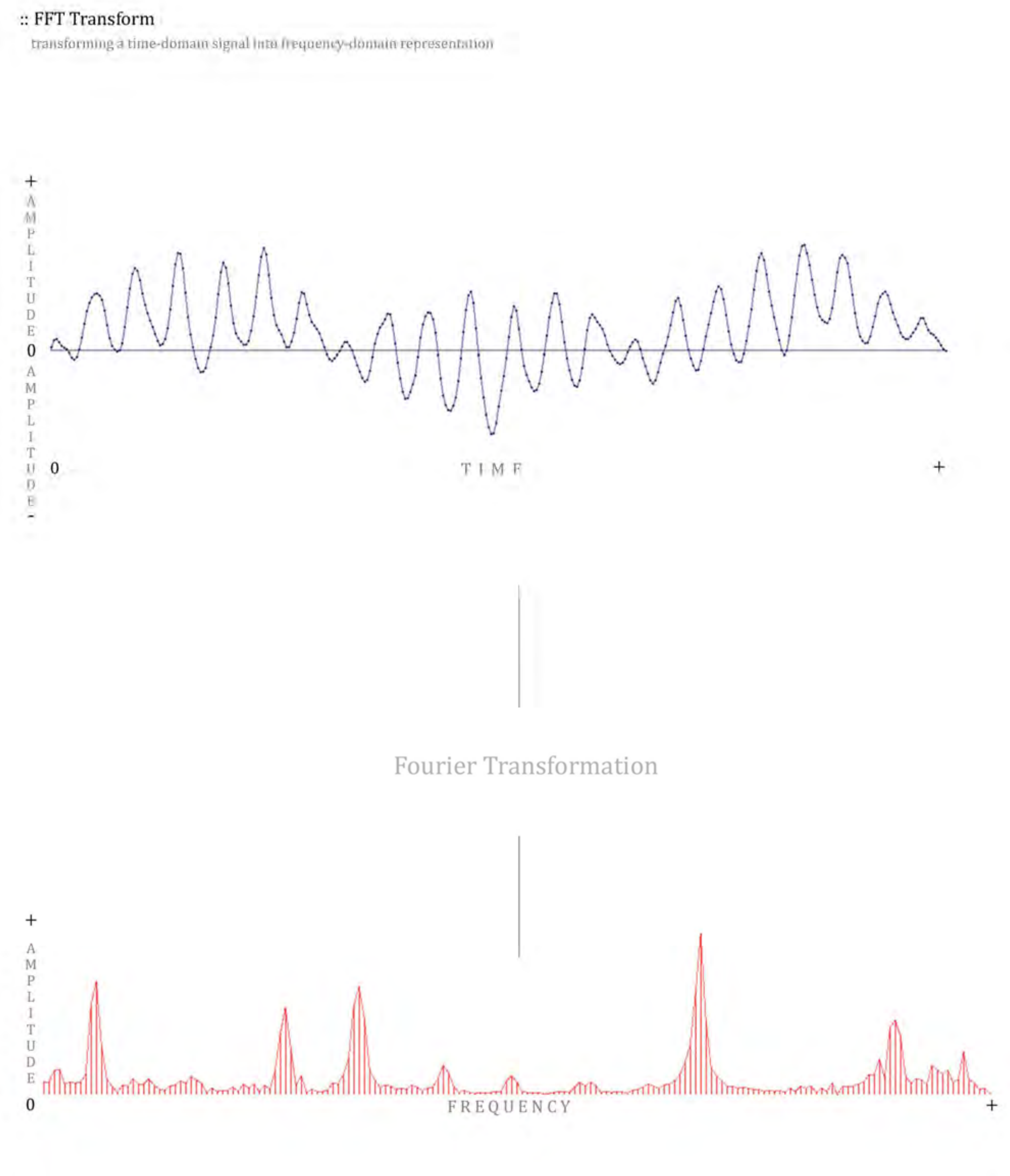
Audio Synthesis

The extension of this project to the sonic domain was motivated by a question: can the Lucas-Kanade algorithm be used as an experimental method to synthesize sounds? The approach followed here defines audio flow as the motion between two spectral magnitude envelopes [6]. Audio flow computes the motion across two (or more) frames of sound in the frequency domain. A more detailed explanation of the procedure now follows.

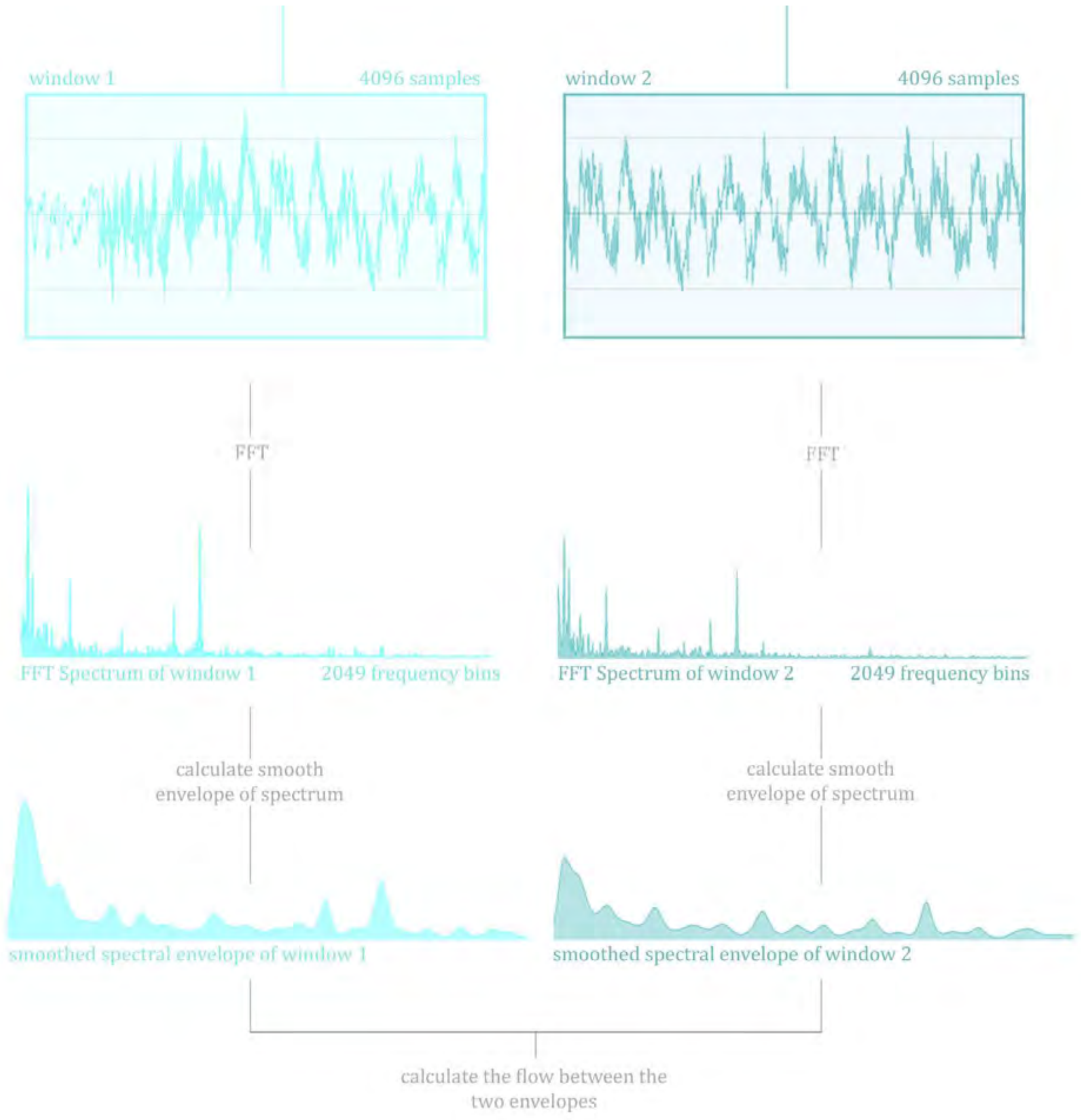
The first point to be made is that the term “frame” in sound synthesis does not mean quite the same as an image frame. A sonic frame is a small (for instance, 10 milliseconds) sequence of audio samples, which has typically been extracted from a longer audio clip. The sound synthesis algorithm I am proposing begins by selecting the two frames that are to be compared. Typically, it takes overlapping frames rather than successive frames, as shown in the figure below.



Since the audio flow computation will search for movement in the frequency domain, we must perform a transformation that takes the data for each frame from the time domain to the frequency domain. This is typically accomplished by a mathematical technique known as the Fourier Transform (FT), which computes the frequency distribution (or “spectral envelope”) of an audio frame. This envelope can be diagrammed as a two-dimensional graph with the frequencies along the x-axis and the respective magnitudes of those frequencies along the y-axis [7].

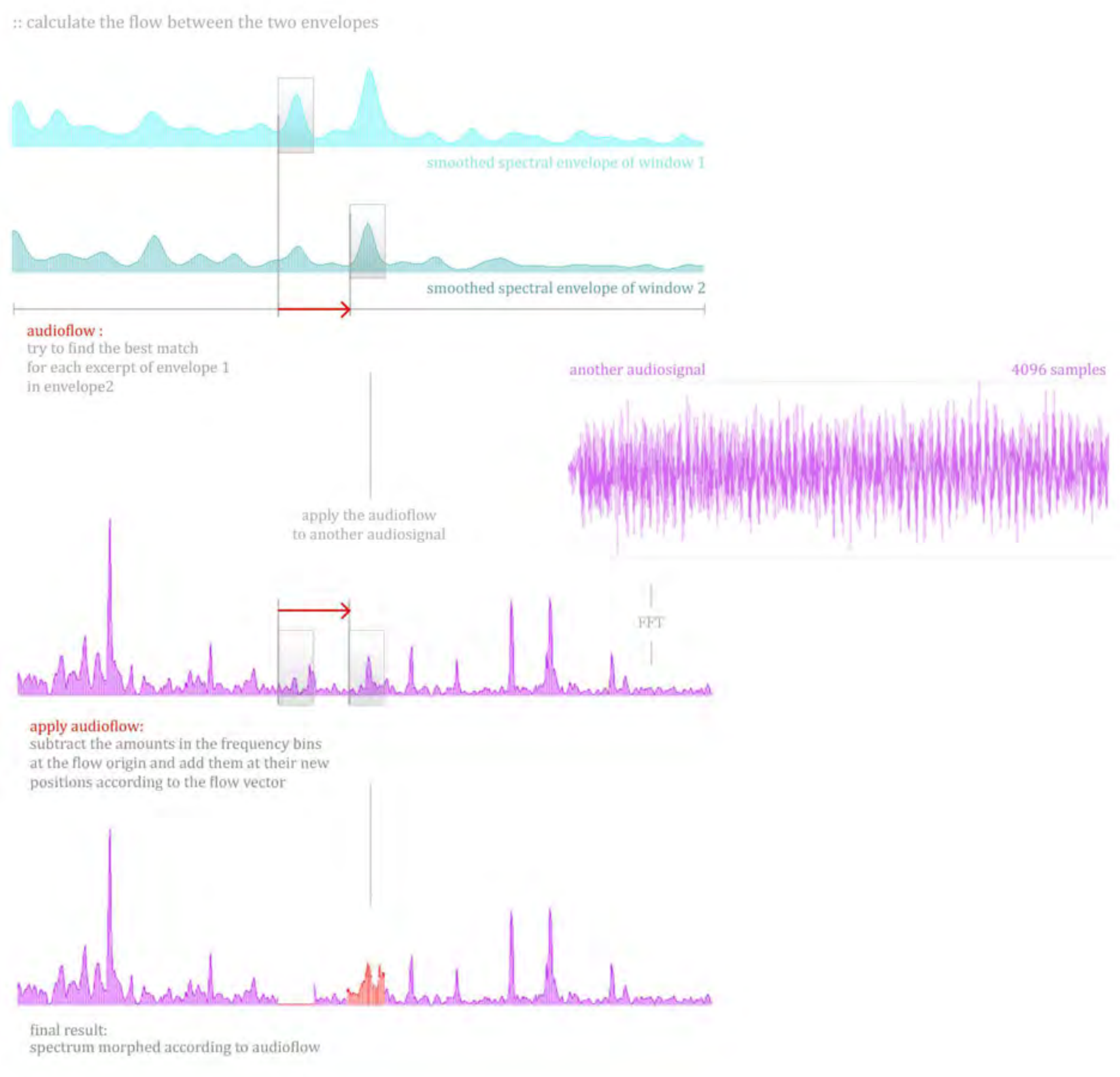


Each audio frame is transformed into the frequency domain, and then smoothed. Think of smoothing a spectral envelope as analogous to the blurring of an image. To blur an image is to eliminate irrelevant details so as to capture only essential positional information. In the same way, smoothing a spectral envelope removes its short-term fluctuations and captures its overall shape. Typically, the smoothing is performed using a low-pass filter, although other options are possible.

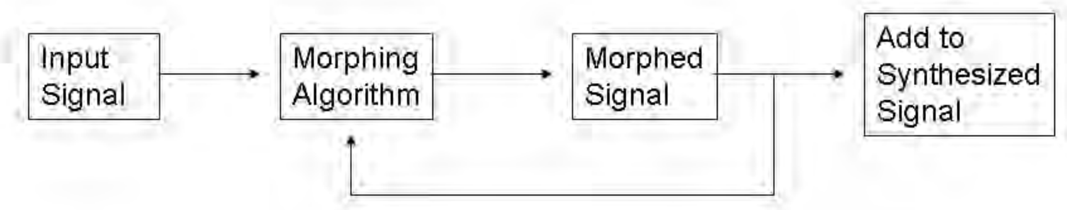

window 1
4096 samples
window 2
4096 samples
FFT
FFT
FFT Spectrum of window 1
2049 frequency bins
FFT Spectrum of window 2
2049 frequency bins
calculate smooth envelope of spectrum
calculate smooth envelope of spectrum
smoothed spectral envelope of window 1
smoothed spectral envelope of window 2
calculate the flow between the two envelopes

We are now going to compute the audio flow across the spectral envelopes of the two frames. To accomplish this computation, the first envelope is segmented into flowpoints [8]. We want to find out where each template has “moved” in the second envelope. The outcome of this calculation, which involves the same iterated procedure used with image frames, is a displacement vector for each flowpoint.

If this analysis is carried out on a sequence of frames, the result will be a sequence of vector fields. The audio synthesis method proposed here uses these vectors to deform the spectral envelope of another audio clip (the “seed”), which can be either a short section of the clip previously used to compute the audio flow or an extract from a different source. The synthesis procedure begins by obtaining the spectral envelope of the seed using an FT [9]. This envelope is then deformed by means of the first vector field in the sequence. In other words, the values of its frequencies are pushed along the corresponding vectors. The figure below illustrates how one single vector taken from two spectral envelopes (shown in green) deforms the envelope of a seed (shown in purple).


:: calculate the flow between the two envelopes
smoothed spectral envelope of window 1
smoothed spectral envelope of window 2
audioflow :
try to find the best match
for each excerpt of envelope 1
in envelope2
another audiosignal
4096 samples
apply the audioflow
to another audiosignal
FFT
apply audioflow:
subtract the amounts in the frequency bins
at the flow origin and add them at their new
positions according to the flow vector
final result:
spectrum morphed according to audioflow

The algorithm then transforms the deformed envelope back into the time domain as a new frame of the signal being progressively synthesized. The morphed audio signal is then fed back as an input of the next iteration of the same procedure, using the next vector field in the sequence [10].



12

The synthesized signal therefore comprises a sequence of progressive deformations of one short input signal. The result is experienced as a continuously varying sound-object whose component frequencies are progressively separated, each undergoing an independent evolution. The unity and differentiation of the sound object is thus the core subject of the aesthetic experience.

Conclusion

Media art confronts a fundamental choice between two opposing points of view. First of all, the instrumental standpoint considers technologies as direct channels for the transmission and execution of intentions. This is the standpoint of the artist-user who is concerned with effects rather than processes and procedures. Secondly, the procedural standpoint considers technologies as productive partners capable of interfering with and modifying the artist's intentions by suggesting new creative possibilities. Technologies are not mere intermediaries but complex mediators whose specificity must be acknowledged and engaged. The Flowpoints project illustrates the creative potential of this procedural standpoint.

References

[1]. Bruno Latour, *Pandora's Hope: Essays on the Reality of Science Studies*. Cambridge, Mass.: Harvard University Press, 1999.

[2]. B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of Imaging Understanding Workshop*, 1981, pp. 121-30.

[3]. To put the problem of image registration more formally, assume that there exists a function T(x) which gives the brightness values of the pixels in one subregion of the frame at a given time t, and another function I(x) which gives the brightness values of every pixel at time t+1. Here, x is taken as a vector, so it will typically consist of two numbers that represent the x and y location of the image. Every optical flow algorithm aims to find a disparity vector h, which minimizes the difference between T(x) and I(x+h). This difference measures the extent to which T(x) and I(x+h) are misaligned. More specifically, the aim is to minimize the sum of the squared error E between the two images. The difference measure is

$$E = \Sigma [I(x + h) - T(x)]2$$

where the sum ranges over x, the pixels in the template image.

[4]. Charles Sanders Peirce, *Collected Papers*, Cambridge: Harvard University Press, 1931-1958, 5.145.

[5]. The project website is http://sweb.cityu.edu.hk/flowpoints.

[6]. This approach is based on: Tony Ezzat et al., "Morphing Spectral Envelopes

Using Audio Flow," accessed November 10, 2010.
http://cbcl.mit.edu/publications/ps/audioflow.pdf.

[7]. In more technical terms, the procedure used is the spectral method, which is somewhat more complicated than a typical Fourier Transform (FT), since it involves several operations. First perform an FT on the audio signal and take the log of the square of the absolute value, then take another FT and finally compute the square of its absolute value. The outcome of this sequence of operations is the power spectrum of the audio signal. The process is as follows:

audio signal → FT → absolute value → square → log → FT → absolute value → square

[8]. The flowpoint length is a parameter whose value can be set by the programmer.

[9]. The envelope of the seed can be smoothed, but I have found that the results of this synthesis algorithm are more aesthetically complex if no smoothing is applied at this stage.

[10]. The above description has omitted many technical details. Since frequencies are multiplicative rather than additive, for instance, the procedure is carried out in the logarithmic scale. Moreover, audio frames are windowed before being transformed into the frequency domain, so as to ensure a more continuous synthesized signal. The purpose of this paper is not, however, to detail every step in the actual procedure but only to explain the overall concept.