Jéssica Araújo Parente

Desenho Generativo de Tipos de Letra

Relatório de Dissertação/Estágio
Mestrado em Design e Multimédia
orientada por Tiago Martins e João Bicker
e apresentada ao Departamento de Engenharia Informática
da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Setembro de 2018

DESENHO GENERATIVO DE TIPOS DE LETRA

Jéssica Parente

ORIENTADA POR:

Tiago Martins
João Bicker

À FAMÍLIA
Aos meus pais e meu irmão,
Aos meus avós, meus padrinhos e à minha prima Andreia
Obrigada pela preocupação e pelo apoio sempre presente.

AOS MEUS ORIENTADORES
Obrigada pela disponibilidade e apoio até ao fim.

AOS 'MENINOS DO CORO'
Obrigada por me alegrarem os fins de semanas
com sorrisos e boa disposição.

AOS AMIGOS 'ZAINERS'
À malta dos casamentos, ao Miguel e à Andreia.
Aos cafés e conversas longas.
Obrigada por terem enriquecido esta experiência!

Ao Luís.
Obrigada palavras bonitas
e pelas palavras sinceras nos momentos certos.

RESUMO

A tipografia é uma forma de visualizar a linguagem (Cheng, 2006). Para os designers, ganha ainda mais importância, pois acrescenta uma camada de conteúdo. A escolha e utilização de um tipo de letra dáindicações relativas ao assunto que estamos a abordar. Com o intuito de utilizar a tipografia da melhor forma, muitos designers estudam a sua anatomia e formas de a categorizar.

Com a revolução digital surgiram os computadores pessoais e as ferramentas que vieram facilitar o desenho de tipos de letra. Consequentemente, surgiram mais tipos de letra, mas com eles a incerteza da sua qualidade.

É necessário criar ferramentas que estabeleçam um equilíbrio entre o que o utilizador pode determinar e o que o sistema faz autonomamente. Por outro lado, a sociedade em que vivemos está dependente da personalização. Sugestões de acordo com as nossas visualizações tornam-se cada vez mais um requisito nas redes sociais.

Com o objetivo de responder a essas questões surgiu-nos a ideia de um sistema computacional que permitisse o desenho de tipos de letra de forma generativa. Numa primeira fase foi realizada uma investigação sobre (i) anatomia e classificação tipográfica e (ii) uma revisão do estado da arte relativamente ao desenho dinâmico e generativo de tipos de letra. Posteriormente, foi desenvolvido o referido sistema computacional. Para isto foram trabalhadas três vertentes: (i) o desenvolvimento da estrutura dos tipos de letra gerados e codificação dos diferentes elementos da anatomia da letra em diferentes camadas; (ii) a combinação de camadas de diferentes tipos de letra; e (iii) a criação de tipos de letra através da geração/modificação dos elementos destas camadas. Esta abordagem permitirá a geração de glifos de forma automática e coerente.

Palavras-chave

Desenho de tipos, Tipografia, Design Generativo, Design Computacional

ABSTRACT

Typography is a way of visualizing language (Cheng, 2006). As designers, it gains even more importance to us as it adds a layer of content. The choice and use of a font gives indications regarding the subject we are addressing. In order to use typography in the best way, many designers study their anatomy and ways to categorize it.

With the digital revolution appeared the personal computers and tools that came to facilitate the design of fonts. As a result, more fonts appeared, but with them the uncertainty of their quality.

It is necessary to create tools that established a balance between what the user can determine and what the system does autonomously. On the other hand, the society in which we live is dependent on personalization. Suggestions according to our views become increasingly a requirement on social networks.

In order to answer these questions we came up with the idea of a computer system that would allow the design of fonts in a generative way. In the first phase an investigation was made on (i) typographic anatomy and classification and (ii) a review of the state of the art in relation to the dynamic and generative drawing of typefaces. Subsequently, the computer system was developed. For this, three aspects were worked out: (i) the development of the structure of the typefaces generated and the codification of the different elements of the anatomy of the letter in different layers; (ii) the combination of layers of different typefaces; and (iii) the creation of fonts through the generation/ modification of the elements of these layers. This approach will allow the generation of glyphs automatically and coherently.

Keywords:

Type design, Typography, Generative Design, Computational design

I. INTRODUÇÃO

Text is everywhere.
It is a medium and a message.
It is a noun and a verb.
Ellen Lupton (citada em Willen, 2009)

A forma como comunicamos é uma das características que mais nos definem como seres humanos. No nosso dia-a-dia utilizamos as duas vertentes da linguagem, escrita ou falada, por necessidade ou conveniência.

> *Type is the visual manifestation of language. It is instrumental in turning characters into words, and words into messages.*
> Karen Cheng, Designing Type (2006)

Como Cheng refere, a tipografia é uma forma de visualizar a linguagem. Adrian Schaughnessy acrescenta que, no mundo moderno, esta é também uma forma de dar "significado". Já não é suficiente que a tipografia seja tipografia. Tem que transmitir ressonância e profundidade às mensagens que está a transmitir (Schaughnessy, 2009; Cheng, 2006)

> *I've found that typography is a pretty reliable guide to a nation's character. Typography in Tokyo is feverish and urgent; typography in rural France is languid and traditional; Dutch typography is intellectual and restrained; typography in the United States is strident and omnipresent.*
> Adrian Schaughnessy, Graphic Design: a user's manual (2009: 284)

Um designer pode comunicar a imagem pretendida através da composição e dos tipos de letra que utiliza. O seu desenho pode, portanto, ser usado de forma a acrescentar camadas de conhecimento (Schaughnessy, 2009: 284).

Com o avanço tecnológico, impulsionado com o aparecimento dos computadores pessoais na década de 1980, as ferramentas de tipografia sofreram alterações. Os tipos têm agora de ser otimizados tendo em conta o local onde vão ser lidos e o público alvo. Contudo, tal como Adrian Schaughnessy refere, os designers de tipos de hoje em dia continuam a fazer aquilo que sempre fizeram: estão a mudar e adaptar-se aos desenvolvimentos na tecnologia, nos media e na literacia (Schaughnessy, 2009; Lupton, 2006). A tecnologia veio possibilitar novas formas de exploração e permitir ao designer de tipos a exploração de campos anteriormente impensáveis. A programação vem automatizar o design. Hoje em dia os computadores podem "desenhar" novos caracteres em segundos. Consequentemente, através destas novas possibilidades, surgem tipos de letra que se adaptam a diversos contextos (Lehni, 2011; Knuth, 1986).

Este projeto vem tirar partido dessas novas possibilidades. O objetivo é o desenvolvimento de um sistema computacional capaz de criar tipos de letra de forma generativa. Os tipos serão gerados através da codificação dos diferentes elementos da anatomia da letra em diferentes camadas que serão combinadas posteriormente. Pretende-se também que os tipos gerados possam ser modificados tendo por base *inputs* externos. Com a finalidade de gerar glifos de forma coerente será feito um estudo sobre a anatomia e a classificação tipográficas. Será também realizada uma revisão do estado da arte relativamente às novas tecnologias digitais e como afetaram o design generativo, mais especificamente o desenho generativo de tipos de letra.

MOTIVAÇÃO

Enquanto designers lidamos constantemente com tipografia. A forma como compomos uma página dá indicações acerca do assunto que estamos a abordar. Consequentemente, a utilização de um tipo de letra específico acrescenta mais uma camada de conteúdo. Walter Tracy, no seu livro Letters of Credit, fala dos seus tempos de aspirante a tipógrafo. Com o acumular de conhecimento e de factos sobre os tipos de letra apercebeu-se de que teria de adquirir a sensibilidade necessária e a habilidade para fazer julgamentos acerca da qualidade dos tipos que usava. Tendo esta questão em mente, uma investigação nesta área torna-se bastante importante para um futuro designer (Tracy, 2003).

> *(...) you learn the rules and then you apply them. What could be simpler? Except there's nothing simple about typography. To become an expert, you have to learn the rules, but you also have to, at all times, exercise aesthetic judgement.*
> Adrian Schaughnessy, Graphic Design: a user's manual (2009: 284)

Por outro lado, a revolução tecnológica deu a possibilidade da experimentação em novas áreas. Surgiram abordagens algorítmicas e generativas para o design e, neste caso, para a tipografia, com a construção de tipos de letra dinâmicos que se adaptam a diversos contextos. Para os designers torna-se vantajoso utilizar essas novas possibilidades de experimentação, como a programação que permite automatizar e gerar tipos em pouco tempo.

Com a evolução das ferramentas de construção tipográfica surgem mais tipos de letra, mas com eles a incerteza da sua qualidade. É necessário criar sistemas que respeitem as regras tipográficas criando um equilíbrio entre o que o utilizador pode alterar e o que o sistema executa automaticamente. Além disso, na sociedade onde vivemos existe uma necessidade insaciável de personalizar tudo quanto possível: empresas investem em automóveis que fazem publicidade aos seus serviços; nas redes sociais as notícias aparecem de acordo com os nossos gostos no *Youtube* aparecem sugestões de acordo com as nossas visualizações.

Com o intuito de responder a estas questões surgiu-nos a ideia de um sister computacional que permitirá o desenho de tipos de letra de forma generativa. Os caracteres serão criados através da codificação dos diferentes elementos da anatomia da letra em diferentes camadas, estas poderão ser mais tarde modificadas tendo por base *inputs* externos. A fim de controlar a fiabilidade do sistema desenvolvido, as modificações ficarão restringidas, permitindo ao sistema, contudo, alguma abstração.

ENQUADRAMENTO

Ao longo do tempo, a criação tipográfica esteve sempre associada ao contexto social e aos movimentos artísticos. Mais especificamente neste último século, o design de tipos sofreu uma série de alterações que surgiram devido ao aparecimento das novas tecnologias digitais. Os tipos são agora trabalhados de outra forma e quem trabalha na área tem de se adaptar às novas ferramentas que vão aparecendo ao longo do tempo. Cada vez mais a tipografia se torna numa ferramenta fundamental no processo de design (Cheng, 2006; Lehni, 2011).

> *A história da tipografia reflete uma tensão contínua entre a mão e a máquina, o orgânico e o geométrico, o corpo humano e o sistema abstrato.*
> Ellen Lupton (2006: 13)

O surgimento dos computadores pessoais disponibilizou ferramentas tipográficas para um público mais alargado. Jürg Lehni afirma que o computador é uma máquina e o seu objetivo é simular outras máquinas ou processos. Portanto, temos de olhar para as novas tecnologias digitais como uma forma de automatização dos processos e tentar tirar o máximo partido dessa vantagem. Foi por isso que no início da década de 1990, depois de séculos à procura da perfeição, surgiu uma série de tipos de letra que procuravam a irregularidade e tinham a aleatoriedade como padrão (Lehni, 2011).

O projeto prático desta dissertação pretende criar um equilíbrio entre a tipografia tradicional, a tipografia como narrativa e a generatividade para o design e neste caso mais específico, para os tipos de letra.

ÂMBITO E OBJETIVOS

Este projeto tem como área de estudo o desenho generativo de tipos de letra. O que se pretende é o desenvolvimento de um sistema que seja capaz de criar tipos de letra de forma automática.

Para isso, será feita uma investigação acerca dos termos relativos a tipografia, a sua anatomia, assim como uma investigação relativa aos projetos que já foram desenvolvidos nesta área. Nesta revisão vai ser possível entender a evolução dos tipos de letra ao longo dos anos e de que forma as novas tecnologias digitais afetaram o desenho generativo de fontes.

Relativamente ao projeto, o que se pretende é a codificação computacional dos diferentes elementos da anatomia das letras em diversas camadas. Essas camadas serão combinadas e modificadas. Essas alterações poderão ter como base *inputs* externos (e.g. som, caligrafia, períodos e circunstâncias históricas, sociais ou culturais). O sistema generativo de desenho terá depois em conta esses *inputs*. O principal objetivo é a geração de glifos coerentes e que sejam propícios a serem utilizados.

ESTRUTURA DO DOCUMENTO

Esta dissertação está organizada em cinco capítulos: Introdução, Plano de trabalhos, Estado da Arte, Desenvolvimento do Projeto, e Conclusão.

No primeiro capítulo é apresentado o tema da dissertação e o que motivou para o desenvolvimento do projeto em questão. São também apresentados os objetivos desta dissertação.

O segundo capítulo, plano de trabalhos, tem como função a estruturação e calendarização as diferentes tarefas deste trabalho.

O Estado da Arte tem como objetivo aprofundar conhecimentos relativos aos temas abordados. Numa primeira instância será exposto uma secção sobre a forma das letras, isto é, serão expostos termos associados à terminologia e anatomia da letra e classificações. A segunda secção deste capítulo apresentará projetos desenvolvidos no âmbito da tipografia e da generatividade na tipografia.

O quarto capítulo desta dissertação é sobre o projeto prático. Neste capítulo o projeto é conceptualizado e explicado com maior detalhe, além disso são também exemplificadas algumas experimentações. São também apresentados todas as abordagens e métodos desenvolvidos ao longo deste projeto.

O quinto capítulo finaliza este projeto apresentando as conclusões retiradas e o trabalho futuro.

II. PLANO DE TRABALHOS

IDENTIFICAÇÃO DAS TAREFAS

Inicialmente tinhámos estabelecido uma série de tarefas para que os objetivos desta dissertação fossem cumpridos. O plano de trabalhos teve como função identificá-las e prever o tempo de duração de cada uma delas. As tarefas foram divididas em seis grandes temas apresentados a seguir.

1. Escrita da dissertação
2. Experimentação
3. Desenvolvimento do projeto
4. Aplicações

1. Escrita da dissertação

Esta tarefa encontra-se dividida em três etapas: Estado da Arte, Escrita intermédia e Finalização da escrita da dissertação.

Setembro — Fevereiro

ESTADO DA ARTE

O Estado da Arte consiste na recolha de informação relativa à evolução do desenho generativo de tipos de letra ao longo do tempo. Em primeira instância pretendia-se uma revisão dos termos relativos à anatomia da letra, a relação entre caracteres e a classificação. Estava previsto também uma análise de trabalhos criados no contexto do desenho algorítmico de tipos de letra.

Abril — Junho
Julho — Setembro

ESCRITA INTERMÉDIA E FINALIZAÇÃO DA ESCRITA DA DISSERTAÇÃO

A segunda e terceira etapa dizem respeito à escrita dos detalhes desenvolvidos durante a experimentação e à finalização da escrita da dissertação. Foi criada uma etapa intermédia para prevenir atrasos.

2. Experimentação

Outubro — Dezembro
Fevereiro — Junho

Esta tarefa foi calendarizada em dois períodos referentes às primeiras experiências, antes da entrega intermédia, e da experimentação ao longo do desenvolvimento do projeto. Como era de esperar o que se pretendia nesta tarefa era a experimentação e desenvolvimento de pequenas abordagens que ajudassem no crescimento do projeto prático.

3. Desenvolvimento do projeto

Esta tarefa foi dividida em duas etapas: definições iniciais e desenvolvimento final da ferramenta.

Fevereiro — Março

DEFINIÇÕES INICIAIS

Para esta primeira etapa pretendia-se o desenvolvimento de uma série de tarefas necessárias para a base do sistema de geração de tipos. Esperava-se a definição (i) do(s) esqueleto(s) base do tipo de letra gerado; (ii) das camadas (layers) e (iii) dos *inputs*.

Março — Junho

DESENVOLVIMENTO DA FERRAMENTA

A segunda etapa era referente ao desenvolvimento da ferramenta em si.

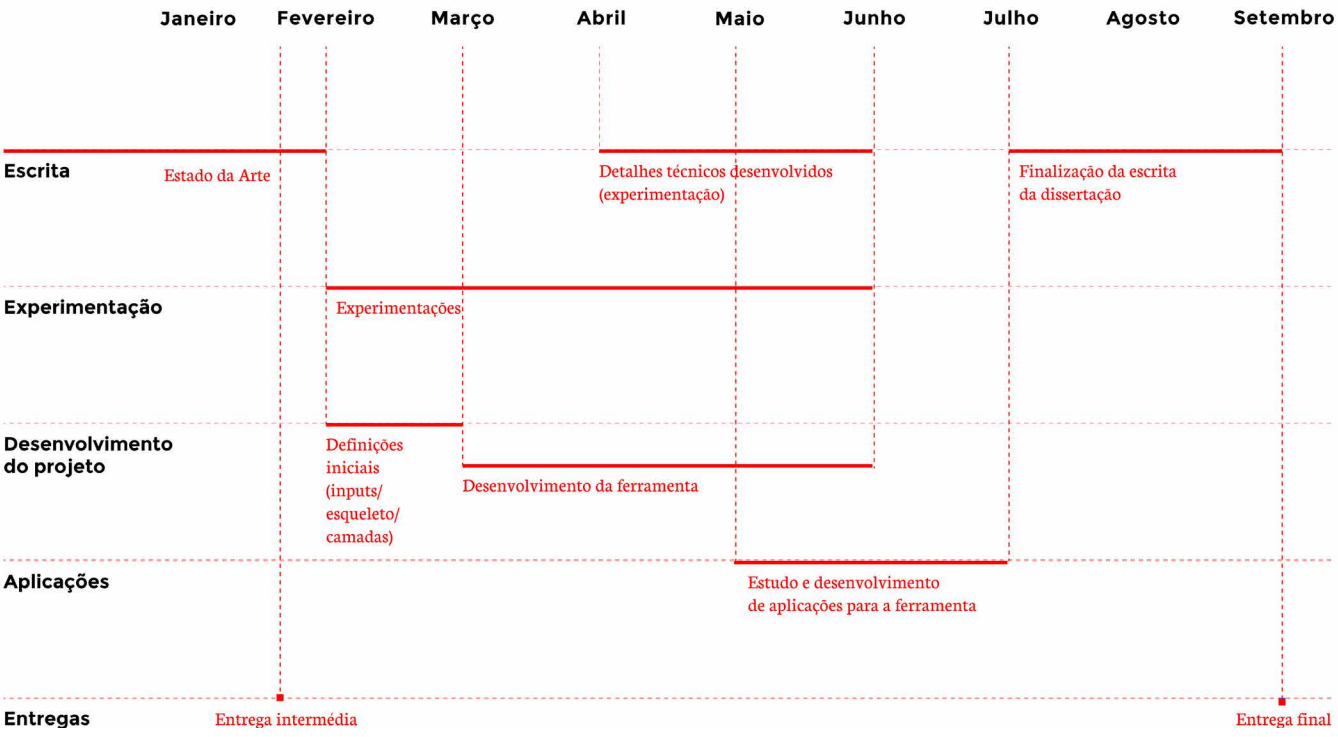
4. Aplicações da ferramenta

Maio — Junho

Esta tarefa pode ser vista como outra fase de experimentação. O que se pretendia eram testes de aplicação dos tipos de letra desenvolvidos e do sistema em si.

DIAGRAMA DE GANTT

Versão mais detalhada entre Janeiro e Setembro:



METODOLOGIA

Inicialmente o processo de desenvolvimento foi muito livre, pretendiamos obter o máximo de opções possível em aberto para decidir o caminho a escolher. Depois, quando nos apercebiamos do método que parecia mais promissor tendo em conta os nossos objetivos optavamos por esse. Se necessário a meio do caminho voltávamos atrás.

ALTERAÇÕES NO PLANO DE TRABALHO

No final, o calendário sofreu algumas alterações causadas pelos atrasos no desenvolvimento das tarefas. A escrita do artigo sobre os logotipos gerados para as Faculdades da Universidade de Coimbra e o problema inicial para a extração dos esqueletos fizeram com a fase das experimentações se prolonga-se até Agosto.

III. ESTADO DE ARTE

No desenvolvimento de um projeto é importante aprofundar conhecimentos na área de estudo e conhecer os projetos que foram desenvolvidos nesse contexto. Assim, poderemos avaliar a relevância do nosso projeto para a área. O objetivo do Estado da Arte é a contextualização acerca dos temas da dissertação e, portanto, no final deste capítulo deveremos ter os conhecimentos necessários para aplicação no projeto prático.

Para a criação de um tipo de letra é necessário conhecer as partes constituintes da letra para se poderem replicar. Além disso, é necessário saber que partes da letra são semelhantes entre caracteres e como são divididas as famílias de fontes existentes dependendo das suas características.

No presente capítulo são abordados termos associados à anatomia dos tipos, as relações existentes entre caracteres, uma das classificações tipográficas existentes e projetos que foram desenvolvidos neste contexto.

III.I A Forma da Letra

Desde o surgimento dos primeiros tipos de letra até aos dias de hoje muitos foram aqueles que se interessaram pelo tema. Enquanto designers temos a necessidade de adquirir a sensibilidade necessária para distinguir que tipos deverão ser usados e em que circunstâncias. No contexto deste projeto a necessidade é acrescida, de forma a criar tipos de letras viáveis.

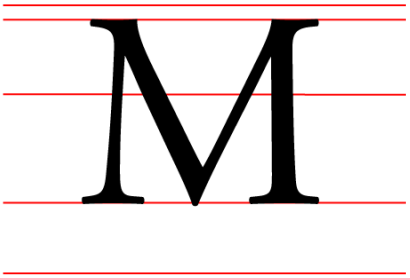
A ANATOMIA E TERMINOLOGIA

No século XV, Gutenberg revolucionou a escrita ocidental com a invenção dos tipos móveis. Anteriormente, estes já teriam sido usados na China, mas o alfabeto latino, devido ao seu reduzido conjunto de sinais, foi mais facilmente mecanizado. Assumindo que os tipos móveis foram o nascimento da tipografia, existem mais de 500 anos de história e consequentemente imensos nomes técnicos.

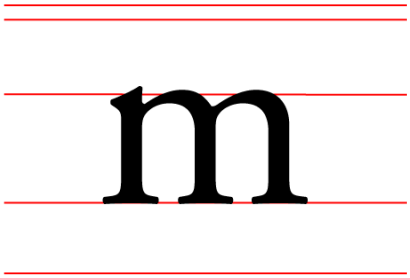
Para melhor compreensão serão apresentados, de seguida, alguns termos básicos. Carácter e glifo são dois termos que são constantemente confundidos nesta área. Carácter é um elemento tipográfico que pode ser uma letra um número, ou um sinal de pontuação. Glifo é a variação da mesma letra, do mesmo carácter. A combinação de um acento com um carácter cria um novo glifo.

Fonte ou tipo de letra são dois termos que se referem a um conjunto de caracteres alfabéticos, numéricos e sinais de pontuação com o mesmo desenho e estilo. Uma família tipográfica é um conjunto de fontes relacionadas e projetadas para trabalhar em conjunto (um peso romano, itálico e bold associado a um tipo de letra).

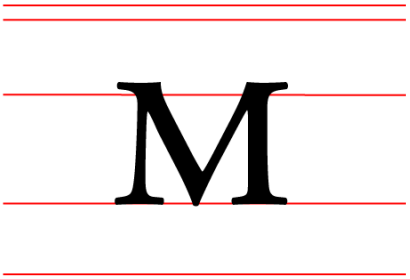
As letras maiúsculas podem também ser chamadas caixa-alta ou uppercase e, consequentemente, as letras minúsculas podem ser classificados como caixa baixa ou lowercase. Versaletes são letras maiúsculas desenhadas para combinar com o peso e tamanho das minúsculas.



Caixa-alta/ maiúscula — *Uppercase*



Caixa-baixa/ minúscula — *Lowercase*

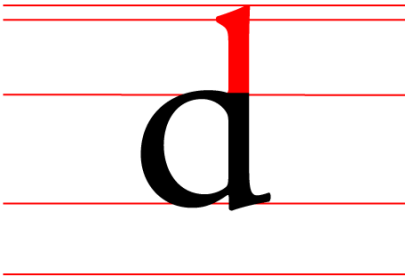


Versalete — *Small cap*

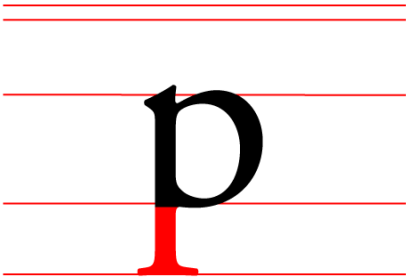
Fig. 1
Anatomia tipográfica
(adaptado de Amado & Silva, 2011)

Anatomia

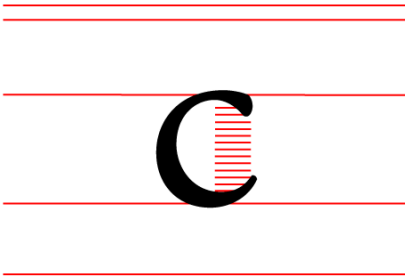
Todos os tipos de letra partilham partes anatómicas. São indiscutíveis as semelhanças encontradas entre o "m" e o "n" ou entre o "h" e o "n", mas existirá um capítulo para aprofundar essas relações. Na Fig.2 estão representados os principais termos.



Ascendente — *Ascender*



Descendente — *Descenter*

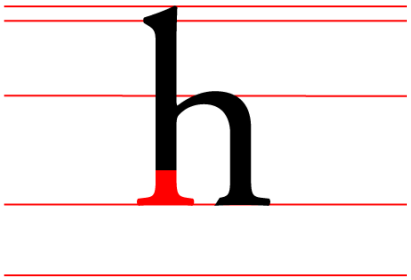


Abertura — *Aperture*

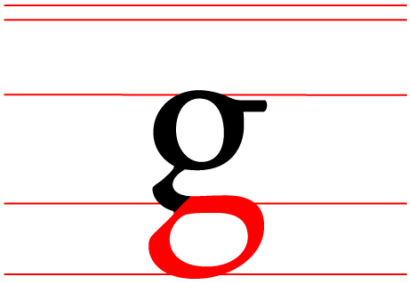
Contraforma/ espaço interno — *Counter*

Fig. 2
Anatomia tipográfica:
principais termos (adaptado
de Amado & Silva, 2011)

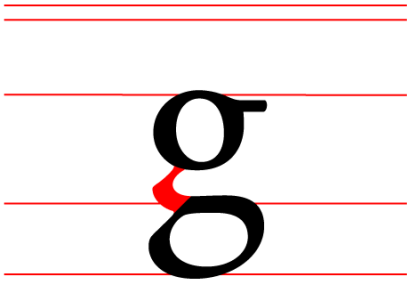
Ombro — *Shoulder*



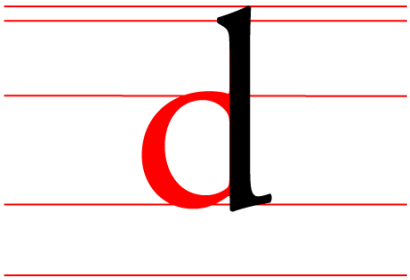
Pé — *Foot*



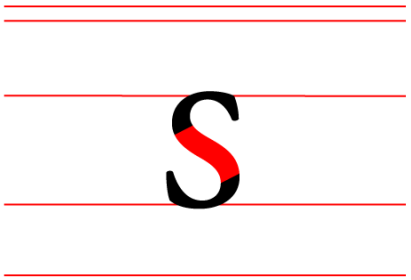
Contracurva/ laço/ olhal — *Loop*



Pescoço — *Link/ neck*



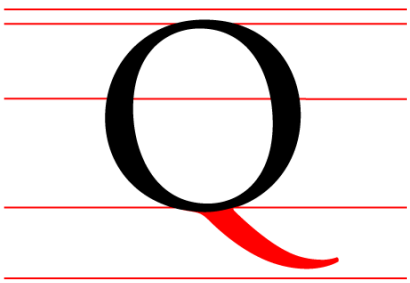
Barriga — *Bowl*



Espinha/ arco duplo — *Spine*

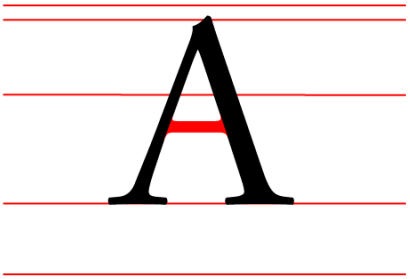
Perna — *Leg*

Braço — *Arm*

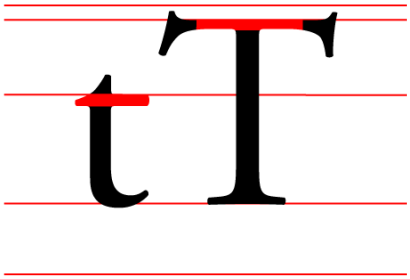


Cauda — *Tail*

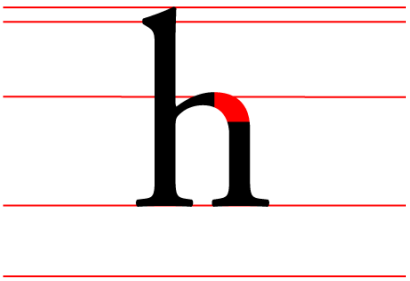
Filete/ perfil — *Hairline*



Barra transversal/ travessão — *Crossbar/ bar*

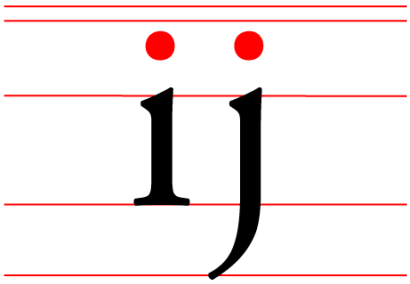


Cruz/ travessão — *Cross Stroke*

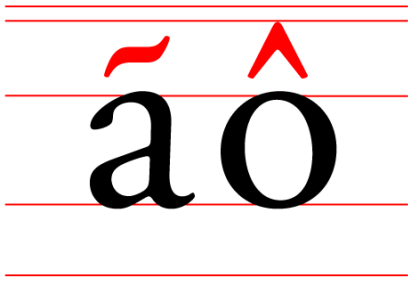


Arco da haste — *Arc of stem*

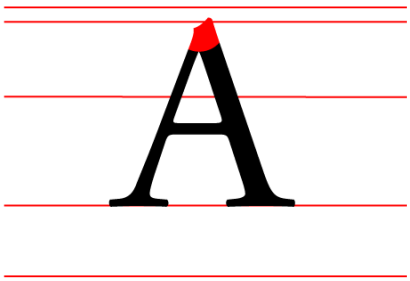
Haste — *stem*



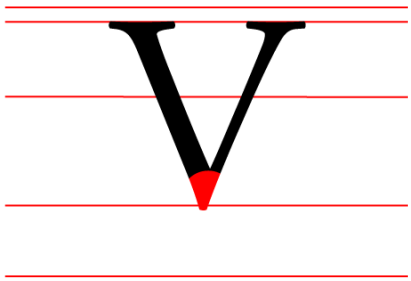
Pinta — *Dot/tittle*



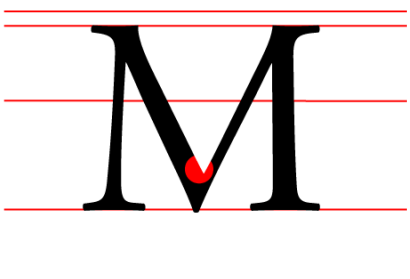
Diacrítico — *Diacritic*



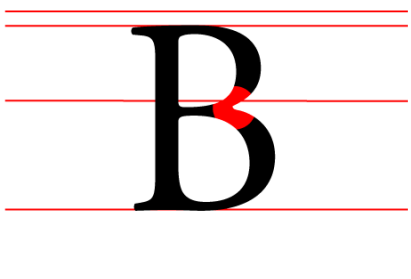
Ápice — *Apex*



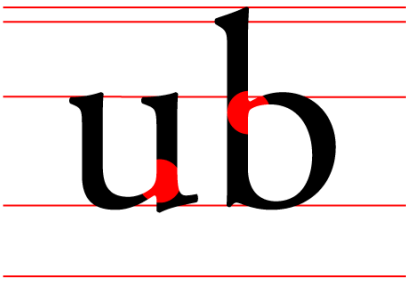
Vértice — *Vertex*



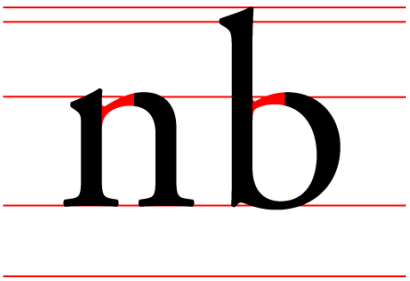
Virilha — *Crotch*



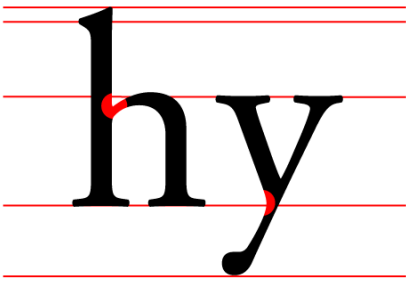
Cintura — *Waist*



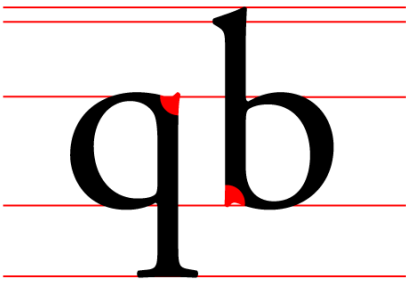
Junção — *Join*



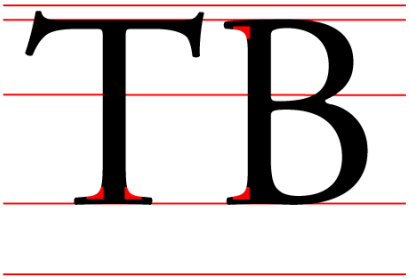
Afilamento — *Taper*



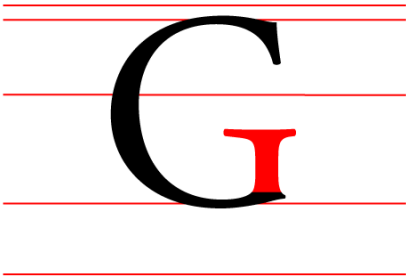
Incisão/ armadilha de tinta — *Ink tap*



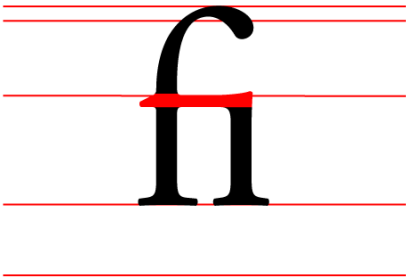
Esporão/ farpa — *Spur/ barb*



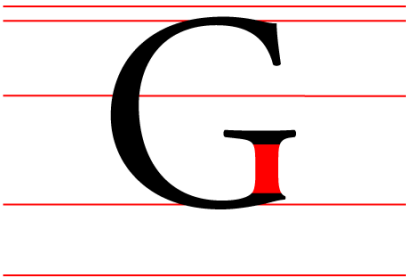
Enlance — *Bracket/ fillet*



Queixo — *Chin*

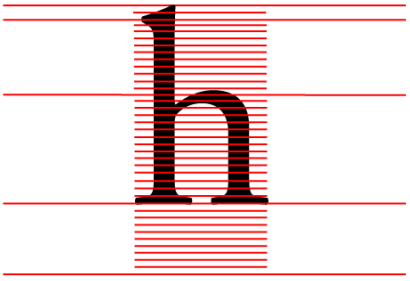


Ligadura — *Ligature*



Garganta — *Throat*

Floreado/ floreio — *Swash/ florish*



Corpo — *Body*

Arquitetura da Tipografia

Para que exista uma boa relação entre as letras num texto é necessário que os tipos de letra respeitem uma série de medidas. A linha de base é onde todas as letras pousam, é o eixo mais estável ao longo da linha de texto. A linha dos ascendentes define a altura dos ascendentes e, por consequência, a dos descendentes a altura dos descendentes. A altura das maiúsculas ou altura de versal é a distância entre a linha de base e o todo da maiúscula. Por fim, a altura x é a distância entre a linha de base até à altura do corpo principal da minúscula, excluindo os ascendentes e descendentes (Lupton, 2006).

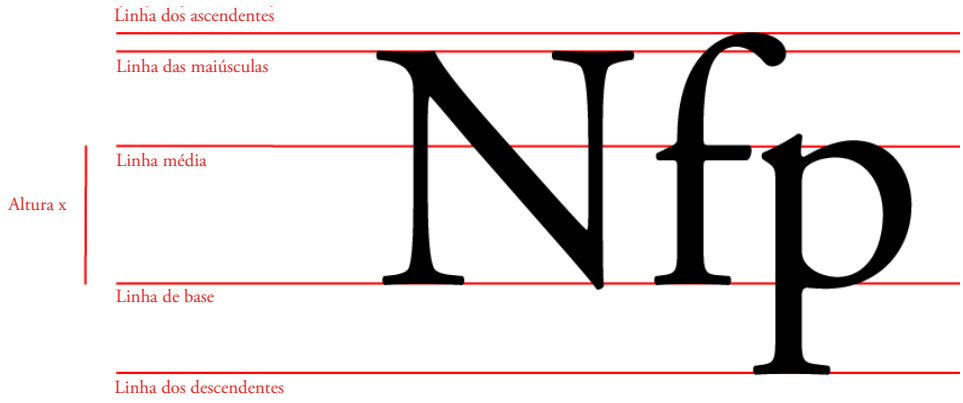


Fig. 3
Arquitetura da tipográfica
(adaptado de Amado & Silva, 2011)

Propriedades da tipografia

A tipografia tem também uma série de propriedades. O eixo é uma delas e define o ângulo de orientação da letra. O contraste é a relação entre traços finos e grossos e o peso ou espessura é a mancha de tinta de um tipo de letra. Geralmente um tipo de letra tem pelo menos três pesos: *light*, *medium* e *bold*. O peso medium normalmente é chamado de roman ou normal. Existem depois também outras propriedades como o ângulo, a inclinação e modulação, esquematizadas na Fig.4 (Cheng, 2006).

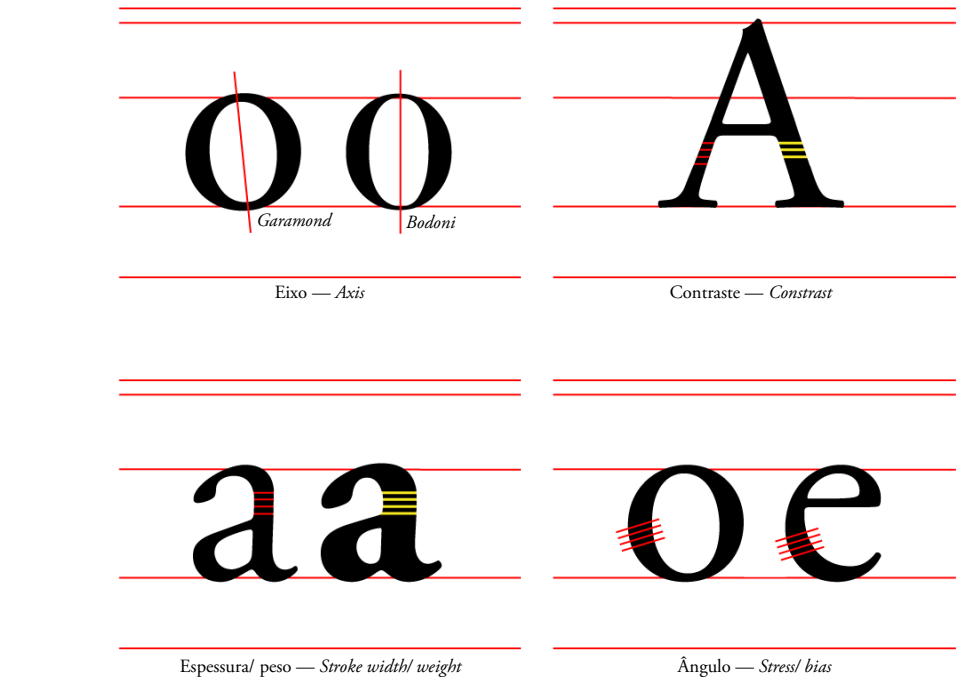
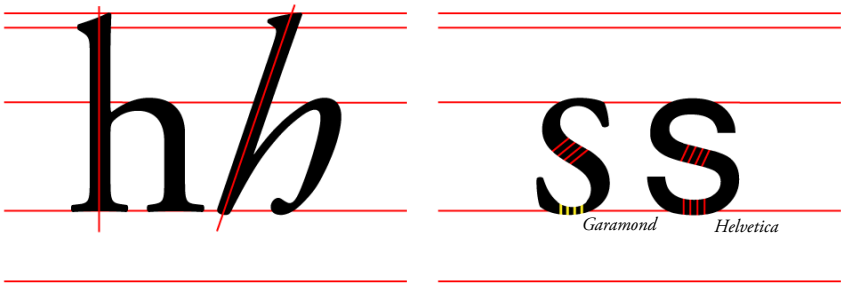


Fig. 4
Propriedades da tipográfica
(adaptado de Amado & Silva, 2011)



Inclinação — *Slope*

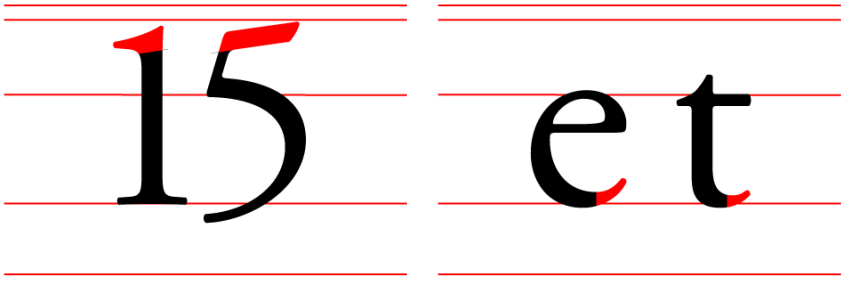
Modulação — *Modulation*

Terminais e serifas

Os terminais e serifas representam as marcas de entrada e saída da caneta. A existência de várias formas de serifa está associada aos diferentes estilos de escrita, assim como a diversas ferramentas, ângulos da caneta e pressão. As serifas são pequenas linhas nas extremidades dos traços horizontais e verticais. Os tons e os tamanhos das serifas podem variar consideravelmente (Cheng, 2006).

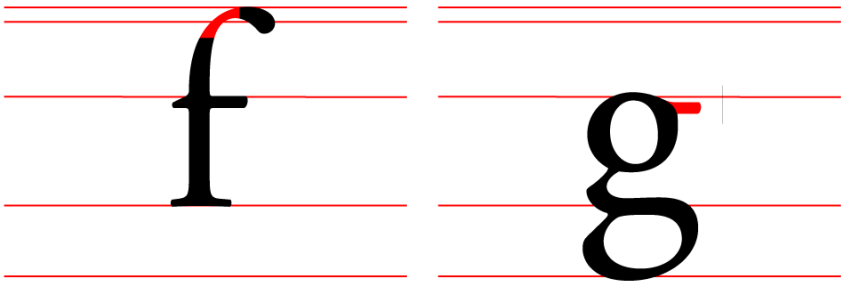
Terminais

Fig. 5
Terminais (adaptado de Amado & Silva, 2011)



Bandeira — *Flag*

Finial — *finial*



Gancho — *hook*

Orelha — *ear*

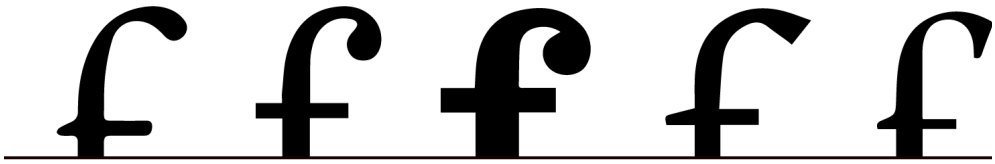


Fig. 6
Terminais (adaptado de Amado & Silva, 2011, e Willem, 2009)

Terminal em gota/ lágrima
—
Teardrop/ lachrymal terminal

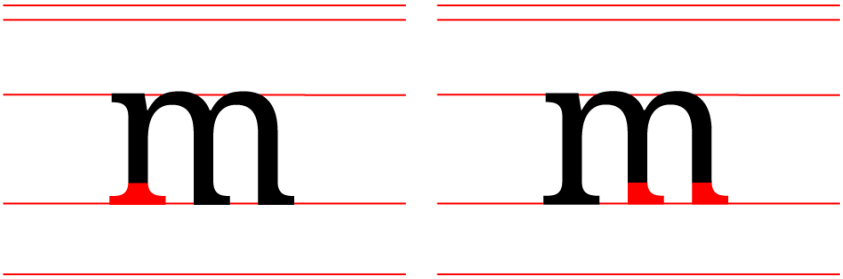
Terminal em bola (enlaçado)
—
Ball terminal (bracket)

Terminal em bola (abrupto)
—
Ball terminal (unbraket)

Terminal em corte/ cisanha
—
Sheared terminal

Terminal em bico
—
Beak terminal

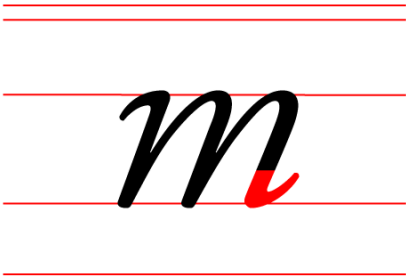
Serifas



Serifa reflexiva — *Reflexive serif*

Serifa unilateral — *Unilateral serif*

Fig. 7
Tipos de serifa (adaptado de Amado & Silva, 2011, e Willem, 2009)



Serifa em gancho — *Hooked serif*

Serifa caligráfica/ assimétrica — *Calligraphic serif/ assimetrical*

Serifa em copa — *Cupped serif*

Serifa filiforme — *Hairline serif*

Serifa mistiforme/ enlaçada — *Bracketed serif/ adnate serif*

Serifa quadrangular/ egípcia — *Slab serif*

Serifa rectiforme/ abrupta — *Unbracked serif/ abrupt serif*

Fig. 8
Tipos de serifa (adaptado de Amado & Silva, 2011 e Willem, 2009)

Relações entre caracteres

Desde o surgimento da tipografia muitos foram aqueles que se interessaram pelo seu desenho. Além de interesse, surgiram também muitas abordagens diferentes na forma de desenhar. Segundo Laura Meseguer existem três formas diferentes de construir tipos digitalmente: por partes, modular ou por derivação de formas. A construção por partes consiste na separação do caracter em partes segundo a lógica dos traços da caligrafia. Essa divisão deve ser mantida até ao desenho final para que cada parte possa ser alterada em separado sem quebrar a fluidez. A construção via modular é baseada na repetição de formas na construção de todos os caracteres, por exemplo a haste do "h" repete-se no "l". A derivação de formas segue a lógica da construção em sequência, umas formas são derivadas das outras para manter a coerência (Henestrosa, Meseguer, Scaglione, 2015).

Todos os tipos de letra têm partes anatómicas comuns. Neste subcapítulo serão apresentadas algumas relações entre caracteres para facilitar e tornar coeso o desenho de um tipo.

> *A test of the excellence of any type is this – that whatever the combination of letters, no individual character stands out from the rest – a severe requirement to which all permanently successful types conform.*
>
> Updike (citado em Tracy, Letters of credit: a view of type design, 2003)

As letras de caixa alta são as formas mais antigas do alfabeto romano e as mais simples de serem desenhadas. Cada caracter é constituído por uma variedade de formas lineares (linhas verticais, horizontais, diagonais e curvas) que lhe permite a diferenciação e interligação com os restantes caracteres. Cheng refere que para facilitar o desenho de um tipo devemos agrupar as letras de caixa alta em três grupos de acordo com as formas semelhantes (redondas, triangulares e quadradas). Contudo algumas letras são demasiado complexas e necessitam da combinação de diferentes grupos, por exemplo circulares-quadradas (D, B, P) (Samara, 2004; Cheng, 2006:20).

Formas redondas: O Q C G S
Formas redondas-quadradas: B P R D J U
Formas quadradas: E F L H I T
Formas diagonais: V A W X
Formas diagonais-quadradas: M N K Z Y (Cheng, 2006: 20)

Karen Cheng, no seu livro Designing type, refere ainda que a divisão das letras pode ser por outros parâmetros.

Dois andares: E F H B P R S K X Y
Lados abertos: L T X K Z J
Extra-largas: M W
Extra-estreitas: I J
(Cheng, 2006: 20)

Samara também apresenta uma abordagem para agrupar as letras semelhantes de um tipo que se aproxima do primeiro sistema de Cheng.

Traços horizontais e verticais: E F H I J L T
Traços diagonais: A K M N V W X Y Z
Curvas e as vezes combinação com traços verticais: C D O P S Q U Traços verticais, curvas e diagonais: B G R
(Samara, 2004)

Em *Letters of credit: a view of type design*, Walter Tracy aborda uma outra forma de divisão das maiúsculas. Tracy cria quatro grupos que permitem a repetição de letras, por exemplo a letra "D" encontra-se na categoria de letras com traço vertical reto e com forma redonda.

letras com um traço vertical reto: B D E F H I J K L M N P R U
letras com forma redonda: C D G O P Q
letras triangulares: A V W X Y
letras estranhas/singulares: S T Z
(Tracy, 2003)

As minúsculas apareceram muito mais tarde entre o sétimo e nono século e são assimiladas em linguagem escrita durante centenas de anos. São caracterizadas por um desenho mais complexo e por traços arredondados.

Apresentam uma maior variedade e distinção visual que as letras de caixa alta. Devido à sua estrutura mais complexa requerem o acrescento de categorias (formas verticais, verticais com gancho e ramificadas) (Samara, 2004; Cheng, 2006:74).

Formas redondas: o c e
Formas redondas-quadradas: b d p q g
Formas redondas-diagonais: a s Formas verticais: i l
Formas verticais com gancho: f t j
Formas ramificadas: n h m u r
Formas diagonais: v w y x
Formas diagonais-quadradas: k z (Cheng, 2006: 74)

Cheng apresenta uma outra abordagem para agrupamento das minúsculas de acordo com a largura ocupada e com a existência de ascendentes ou descendentes. Neste caso existem letras que se repetem entre as categorias.

Largura expandida: m w
Largura média: o b d p q g n h u k v x y z
Largura estreita: c e a r s
Largura extra-estreita: i j f t
Com ascendente: b d h f l k t
Com descendente: p q g y
(Cheng, 2006: 74)

Tracy, pelo contrário, mantém a mesmos grupos para a divisão das letras de caixa baixa.

letras com um traço vertical reto: b d h i j k l m n p q r u
letras com forma redonda: b c d e o p q
letras triangulares: v w x y
letras estranhas/singulares: a f g s t z (Tracy, 2003)

No desenho das letras de caixa-alta a maior decisão que se tem de tomar é provavelmente as suas proporções, estas devem seguir o sistema clássico ou o sistema moderno. O sistema clássico ou old style foi criado por razões práticas e estéticas. O sistema é caracterizado pela utilização de quadrados geométricos para a largura das letras, sendo estas agrupadas de acordo com a largura que ocupavam: quadrado inteiro (A, C, D, G, H, K, N, O, Q, T, V, X, Y e Z) e meio quadrado (B, E, F, L, P, R e S). O I e o M são exceções, pois o primeiro é extra-estreito e o segundo extra-largo. O U, J e W foram adicionados mais tarde ao alfabeto romano, portanto não pertencem ao sistema (Cheng, 2006:20).

Na prática muitos tipos de letra que são descritos como tendo proporções clássicas não encaixam exatamente com as divisões de um quadrado. Embora o seguimento deste sistema crie letras bonitas e elegantes, estas tornam-se impraticáveis, pois resultam em tipos de letra que necessitam de um grande espacejamento. Tipos condensados não podem ser desenhados neste sistema, pois os caracteres não podem ser produzidos com proporções quadradas. Devido a estas desvantagens foi criado outro sistema chamado Moderno (Cheng, 2006:20). No sistema Moderno é criado um equilíbrio entre a cor do caracter e o espaço negativo.

> *Typographic color is similar to chromatic color — like red, blue, or orange — but deals only with changes in lightness and darkness, or value, not hue. It is also different from the qualities of chromatic color in that it describes changes in rhythm and texture.*
>
> Timothy Samara, Typography workbook: a real-world guide to using type in graphic design (2004)

Relativamente ao modelo clássico este sistema tem a vantagem de poder ser alargado a fontes condensadas e promover a readability, pois variações aleatórias na cor da letra cansam o leitor (Cheng, 2006:20).

B CLASSIFICAÇÃO

É um fenómeno natural a organização de grandes quantidades de informação para mais facilmente encontrar o que pretendemos. Agrupar torna o processo de procura menos trabalhoso e isso tornou-se necessário com o rápido crescimento de novos tipos de letra ao longo dos anos (Kupferschmid, n.d.).

Nos primeiros 400 anos da tipografia os tipos eram chamadas consoante o seu tamanho (Paragon, Great Primer, Nonpareille). Com a revolução digital surgiram mais tipos de letra e com eles a necessidade de aplicação de nomes para os dar a conhecer. A criação tipográfica era normalmente associada aos modelos históricos, portanto, a terminologia poderia surgir a partir deles (Kupferschmid, n.d.).

Até ao final do século XIX a tentativa de classificar tipos era considerada, além de redundante, impossível e inconveniente. Francis Thibaudeu, no entanto, em 1921 propôs um sistema baseado na forma das serifas, como mais tarde, em 1964, fez Aldo Novarese (Kupferschmid, n.d.).

> *Not only the designs were becoming more creative and individual but also the terminology, resulting in the problem that genre names were not universally understood anymore.*
>
> Indra Kupferschmid (n. d.)

Devido às imensas variáveis, foi necessário estabelecer sistemas para encontrar uma solução que servisse universalmente para classificar os tipos de letra. Em 1954 surge o sistema Thibaudeau desenvolvido por Maximilian Vox. Os nomes dos grupos foram derivados a partir do nome dos tipógrafos mais icónicos, por exemplo, o grupo de fontes constituído pela Didot e pela Bodoni é chamado de *Didone* (Kupferschmid, n.d.).

Segundo Vox os nomes poderiam ser alterados se as pessoas assim o desejassem, pois o importante eram os grupos. Walter Tracy afirma ainda que qualquer classificação é uma ajuda para estudar e não um fim em si, pois a partir do momento em que um aluno é capaz de determinar que tipo de letra encaixa em cada grupo, os nomes reais dos grupos e a sua precisão deixam de ser importantes (Tracy, 2003).

Desde o surgimento do sistema de classificação de Vox apareceram muitas mais formas de classificação e muitas delas foram baseadas em parte pelo sistema Thibaudeau. Por esse motivo, nesta dissertação será utilizado o sistema de classificação criado por Vox (*Humanes, Garaldes, Reáles, Didones, Mécanes, Lineáles, Incises, Scriptes* e *Manuaires*). Serão abordadas mais a fundo seis das nove categorias, excluindo as categorias *Incises, Scriptes* e *Manuaires (display)*. Os grupos apresentados por Vox também se distinguem, porque são baseadas

[1] *readability*
Se as colunas de um jornal, de uma revista ou de uma página de um livro podem ser lidas durante muito tempo sem tensão ou dificuldade podemos dizer que o tipo tem boa readability (Tracy, 2003).

legibility
É referente à perceção e a medida é a velocidade com que cada caracter consegue ser compreendido (Tracy, 2003).

não só na história da tipografia, mas também nas características visuais como o contraste, a forma da serifa e a inclinação.

> *(...) the Venetian, Garalde and Transational classifications refer to typefaces designed in the fifteen, sixteen and seventeenth centuries. (...) some typographers use the term 'Old Roman' to refer to these styles collectively. However, the three separate classifications are in fact important, sinceeach group defines a specific step within a larger typography movement: the evolution of letters from written, calligraphic forms to drawn and designed constructions.*
>
> Karen Cheng (2006)

De seguida são expostas as categorias, juntamente com as características e alguns exemplos de tipos de letra que fazem parte dos grupos.

Humanes *(venetian/humanist)*
Nas *Humanes* o eixo oblíquo é severo, o contraste é pequeno e os componentes da letra, como as serifas e os arcos, exibem um modelo abrupto. Além disso, nos tipos de letra humanistas existe sempre uma inclinação do braço do "e" (Costa, 2013).
ex: *Jenson, Véronèse, Centaur, Perpetua*

Garaldes *(old roman/old style)*
Nas Garaldes as marcas da caligrafia são atenuadas a favor de influências estéticas, como os movimentos Barroco e Maneirista. As fontes que encaixam neste grupo têm de médio a alto contraste e largura variável. Apesar de existir uma ténue diferença entre *Humanes* e *Garaldes*, nesta categoria o braço do "e" deixa de ser inclinado e o eixo do "o" é só ligeiramente inclinado para a esquerda (Costa, 2013).
ex: *Bembo, Garamond, Granjon, Caslon*

Reáles (transitional/neoclassical/rationalist)
As Reáles fazem a ligação entre as *Garaldes* e as *Didones* e são influenciadas pelo neoclassicismo e pela filosofia racionalista. Esses movimentos afetam os tipos de letra deste grupo através do eixo totalmente vertical, pela construção sistemática e pelo alto contraste (Cheng, 2006; Costa, 2013).
ex: *Baskerville, Cochin, Times*

Didones *(new roman/modern)*
Embora as *Didones* tenham sido inspiradas nas Transicionais, ao contrário delas estas refletem os ideais expressivos do Romantismo. As letras são desenhadas com eixos totalmente verticais, larguras uniformes e contrastes extremos (Cheng, 2006; Costa, 2013).
ex: *Didot, Bodoni, Walbaum, Falstaff*

Mécanes *(slab serif/ square serif/ mechanistic/ egyptians)*
Antes do século XIX os tipos de letra eram criados para a produção de livros. Com o surgimento da revolução industrial o campo tipográfico foi alargado e começaram a surgir tipos de letra com serifas quadrangulares e grossas (Cheng, 2006; Costa, 2013).
ex: *Clarendon, Ionic, Rockwell*

Lineáles *(sans serif/ grot/ grotesque/ grotesk/modernist/ lineal)*
As *Lineáles* são a categoria mais associada aos modernistas devido à sua simplicidade e ligação com a "idade da máquina". Surgiram no século XIX, mas é em 1957 com o aparecimento da *Helvetica* e da *Univers* que ganham importância. As humanistas contemporâneas sem serifa têm também uma estrutura humanística, mas as maiúsculas são normalmente modernas em proporção com eficiência no texto (Cheng, 2006; Costa, 2013).
ex: *Helvetica, Akzidenz-Grotesk, Univers, Futura*

O sistema criado por Vox responde a muitos dos problemas da classificação, pois tem em conta a forma do tipo e a história da tipografia. Contudo, desde o surgimento do sistema, o design de tipos tem evoluído, principalmente com o avanço tecnológico. Para encaixar estes novos tipos na classificação, são feitas algumas tentativas de expandir o sistema de Vox criando categorias híbridas (por exemplo: *Demi-Didone*). Contudo, estas categorias não são conhecidas e aceites universalmente (Cheng, 2006).

As limitações das categorias de Vox devem-se ao facto de os tipos agora serem mais complexos e o aspecto visual e o contexto histórico já não serem suficientes. A notabilidade, a função e a intenção são factores a ter em conta para a classificação. Idealmente as fontes que são criadas para um media específico ou aquelas que são criadas tendo em conta um movimento artístico ou social deveriam ser separadas (Cheng, 2006).

> *The original Vox classification system also fails to account for important geographic and cultural differences that influence the design of type.*
> Karen Cheng, Designing type, 2005

Cheng sugere ainda a criação de um sistema de classificação onde seja possível ordenar os tipos por várias escalas (visual, histórica, tecnológica, funcional, cultural e geográfica). Kupferschmid tem uma opinião semelhante e afirma que o ideal seria a combinação das diferentes abordagens num sistema flexível que funcione tanto para iniciantes como para especialistas. Abandonar os sistemas e terminologias estabelecidas ao longo do tempo não vai trazer nada de útil e o ideal é encontrar uma forma de integrar tudo e explicar de uma forma compreensível. Segundo Indra Kupferschmid o desenvolvimento de uma terminologia inequívoca poderá ser mais importante que uma abordagem universal de classificação de tipos de letra (Cheng, 2006; Kupferschmid, n.d.).

III.II Trabalho relacionado

REFORMA E REVOLUÇÃO

O início do século XX foi um momento de grande mudança na criação tipográfica. Surgiram experiências modernistas que reduziram o alfabeto às formas geométricas básicas — círculo, quadrado e triângulo — e que se afastaram das origens manuscritas do alfabeto (Lupton, 2006; Willen, 2009).

No entanto alguns designers consideravam estas distorções na construção do alfabeto grosseiras e imorais. Na busca e reanimação de um alfabeto constituído por letras puras e não corrompidas, em 1906, Edward Johnston propõe um diagrama de caracteres essenciais (Fig.9) (Lupton, 2006).

Fig. 9
Caracteres romanos essenciais,
Edward Johnston,1906

TIPOGRAFIA COMO SISTEMA DE RELAÇÕES ABSTRATAS

Os artistas de vanguarda rejeitavam as formas históricas e olhavam para o alfabeto como um sistema de relações abstratas. Esse aspecto veio a intensificar-se, em 1909, com a publicação do manifesto futurista por Marinetti. Aspectos ultrapassados sobre visão e linguagem foram destruídos com o surgimento do Futurismo. Projetos como *Parole en Libertá* ou *Les mots en liberté* (Fig.10) introduziram uma energia ruidosa na vida do século XX. As composições tipográficas dinâmicas criadas pretendiam criar uma reação emocional no leitor. As palavras eram parte da composição e não apenas usadas para transmitir pensamentos. Filippo Marinetti, além do design, está também associado à poesia concreta e tornou-se um ponto de referência para os artistas de vanguarda que aproveitaram a tipografia e a composição da página como forma de se expressarem.

Fig. 10
Parole en Libertá
e *Les mots en liberté*,
Filippo Marinetti, 1919

Em 1916, surge o Dadaísmo que é marcado por ser um movimento anti-arte e é associado ao futurismo na luta pela mudança da estética tradicional. Raoul Hausmann é um dos artistas de destaque nesta altura devido à forma como utilizava as letras como sua matéria prima. Em 1918, Hausmann desenvolve Kp'erioum (Fig.11) onde são representadas variações na respiração e na voz através do tamanho da fonte (Flask, n.d.; Bargues, 2016).

> *Revealed the surprise and joy of Dada dancing with typography (sobre Raoul Hausmann)*
>
> Brion Gysin (citado em Bargues, 2016)

Em 1923, também Kurt Schwitters e Theo Van Doesburg marcam a uso da tipografia como narrativa. O trabalho Kleine DADA (Fig.12) resulta de uma imagem impressionante, mas também confusa e intrigante. Apesar de optarem pela utilização do erro como estratégia, também são transmitidas informações úteis para o utilizador. O facto do leitor tentar decifrar o texto — estratégia dadaísta — aproxima-o do cartaz.

Mais tarde, aspectos da herança dadaísta ressurgem em convenções gráficas da pop art e em trabalhos de artistas do final da década de 90 como David Carson (Collection, n.d.).

Fig. 11
À esquerda:
Kleine DADA soirée,
Theo van Doesburg
e Kurt Schwitters, 1923

Fig. 12
À direita: *Kp'erioum*,
Raoul Hausmann,
1886 — 1971

TIPOS DE LETRA MODULARES

O início do século XX é também uma época onde surgem projetos com componentes modulares. Designers como Jan Tschichold e Herbert Bayer na Bauhaus criaram tipos de letra altamente geométricos e racionalizados. Theo Van Doesburg foi fundador do *De Stijl* — movimento que apoiava que as letras deveriam ser reduzidas a elementos essenciais. Neste contexto, em 1919, desenvolveu um alfabeto através de elementos perpendiculares (Fig.13) (Willen, 2009; Lupton, 2006) .

Neste contexto, Vilmos Huszár desenhou um logótipo para a revista *De Stijl* (Fig.14) utilizando módulos retangulares, mas, ao contrário de Theo Van Doesburg, os módulos aproximam-se a pixeis (Lupton, 2006).
Ainda com a ideia de oferecer alternativas na forma de construção de tipos, Herbert Bayer e Josef Albers construíram alfabetos apenas com formas geométricas. Bayer, em busca de um tipo de letra universal, criou, na Bauhaus, um alfabeto apenas com letras minúsculas (Fig.15) (Lupton, 2006).

> *Why should we write and print with two alphabets?*
> *We do not speak with a capital A and a small a.*
>
> Herbert Bayer, (citado em Tracy, Letters of credit: a view of type design, 2003)

Fig. 13
Em cima, Square alphabet,
Theo van Doesburg, 1919

Fig. 14
À esquerda, Logótipo para a revista
De Stijl, Vilmos Huszár, 1917

Fig. 15
Universal,
Herbert Bayer, 1925

Nos anos 20 surge também outro sistema modular constituído por um conjunto de vinte formas geométricas. Este sistema é apelidado de *Fregio Mecano* (Fig.15), contudo não existe nenhum designer associado à sua autoria.

Este tipo de letra tem também uma versão digital de nome Section Bold Condensed (Typefaces., n.d.).

Fig. 16
Fregio Mecano,
autor desconhecido, 1920s

Em 1967, em resposta ao progresso na comunicação eletrónica, Wim Crowel criou um alfabeto que denominou *New Alphabet* (Fig.17) e que é baseado num sistema de grelha bastante rigoroso. Este alfabeto foi pragmático, pois foi projetado para funcionar bem em écrans de vídeo. Crowel, designer gráfico e tipógrafo nascido na Holanda, tinha fundado o seu estúdio — Total Design — quatro anos antes e o seu trabalho mais conhecido foi para o *Museu Stedelijk* (Fig.18) (Flask, n.d; Lupton, 2006).

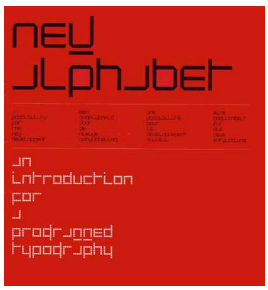
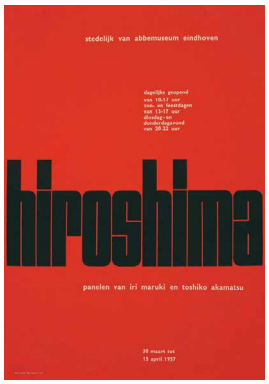


Fig. 17
New Alphabet,
Wim Crouwel, 1967



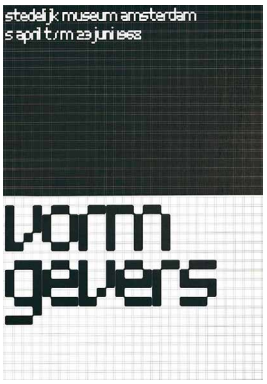


Fig. 18
Cartazes para o *Museu Stedelijk.*
Da esquerda para a direita:
Hiroshima(1957) e Vorm Gevers (1968); Wim Crouwel.

Com o surgimento de ferramentas digitais o processo de criação de fontes tem sido facilitado, o que tem como consequência um aumento dos tipos de letra existentes. Regina (Fig.19) é um tipo de letra modular que nasceu de quatro módulos retirados de partes do tipo de letra *Baskerville*. Foi desenvolvido por Vanessa Garcia, Audrey Devantay, Philippe Egger e Adeline Mollard no workshop de type design dado por Jean-François Porchez, presidente da *ATypI (Association Typographique Internationale)* em 2005 na Suiça. É caracterizada por um aspecto invulgar — resultado da utilização de apenas um módulo curvo — e pelos seus espaços brancos que fazem lembrar stencil (Devantay, n.d.).

Fig. 19
Regina typeface, Vanessa Garcia, Audrey Devantay, Philippe Egger e Adeline Mollard, 2005

TIPOGRAFIA COMO NARRATIVA

Com a chegada dos anos 80 é iniciada uma revolução direta contra as ideias de legibilidade e racionalidade. Muitos designers e tipógrafos adotaram essa abordagem pragmática o que permitiu o aparecimento de alfabetos conceptuais.

> *Don't confuse legibility with communication. Just because something is legible doesn't mean it communicates and, more importantly, doesn't mean it communicates the right thing.*
>
> David Carson (Hustwit,2007)

A introdução do computador pessoal disponibilizou novas abordagens no design tipográfico. Uma nova linguagem visual foi criada a partir da desconstrução do modernismo e surgiram designers e artistas como David Carson e Ed Fella. Em 1984, nasceu a *Emigre*[2] e com ela uma série de novas abordagens para a tipografia.

Na criação de um tipo de letra conceptual a ideia é o mais importante. A ênfase é dada ao processo criativo e é retirada importância ao resultado final, pois o essencial é a forma como lá chegamos. Ed Fella destacou-se pela sua obra de tipografia experimental que influenciou o design de tipos na década de 90. Como artista, passou 30 anos em Detroit, onde experimentou tipografia e deu palestras a estudantes da *Cranbrook Academy of Art*. A interação com os alunos levou-o a fazer uma pós graduação e graças a isso ainda hoje ensina desenho no Instituto das Artes da Califórnia. Os cartazes que fez para a *Detroit Focus Gallery* com a utilização de formas danificadas ou com defeito — desenhadas à mão, recolhidas de fotocopiadoras ou láminas de letragem — fazem parte do conjunto das suas obras com mais influência. A sua forma de abordar a tipografia — estilo caprichoso e extremamente detalhado — influenciou grande parte da tipografia moderna (Willen, 2009; Lupton, 2006; Flash, n.d.).

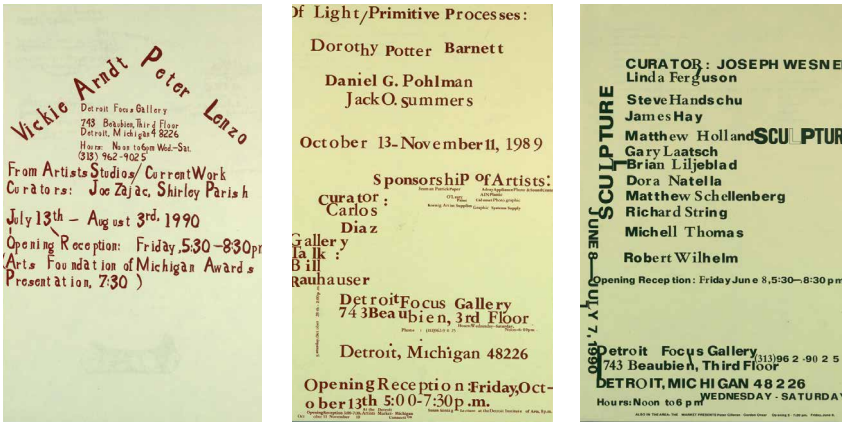


Fig. 19
Gift of Edward Fella,
cartazes para a Detroit Focus Gallery, Ed Fella, 1995

Em 1991 nasceu uma revista que marca a história do design gráfico. A componente visual da revista é associada a David Carson e é caracterizada pela ilegibilidade tipográfica como forma de expressão. A *Ray Gun* (Fig. 20) ajudou a estabelecer uma linguagem visual à música alternativa. A forma de comunicação rompia com os princípios existentes nos media o que dava à revista uma essência inovadora (Flask, n.d; Westgate, 2007).

> *To start designing, I have to read the article, or brief it or listen to the music, to see where it takes me visually and emotionally. It was [a] bit funny, maybe, that at Ray Gun some of the writers complained early that their articles were hard to read. But then by the 30th issue, the same writers would complain if they thought their articles were too easy to read!*
>
> David Carson (Westgate, 2007)

Segundo Carson, ao colocarmos um pouco de nós no nosso trabalho, como a educação e as experiências de vida, ganhamos duas coisas: aproveitamos e

[2]Emigre foi uma revista nascida em 1984 e uma das primeiras a usar computadores Macintosh, influenciando o uso da publicação digital na comunidade de design gráfico. A revista foi responsável pela publicação e promoção de artigos variados sobre o design por muitos autores diferentes. O formato da revista foi muitas vezes alterado passando até pelo suporte CD. (Flash, n.d.)

criamos melhores projetos. David Carson acrescenta que para desenvolver projetos razoáveis e seguros não são necessários designers (Flask, n.d; Layers magazine, n.d.).

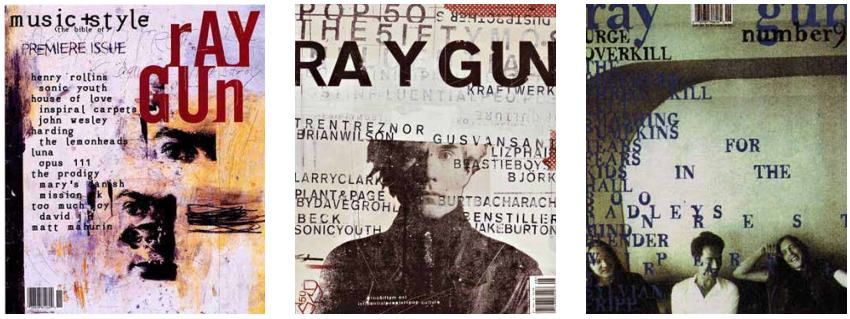


Fig. 2o
Capas *Ray Gun*, David Carson

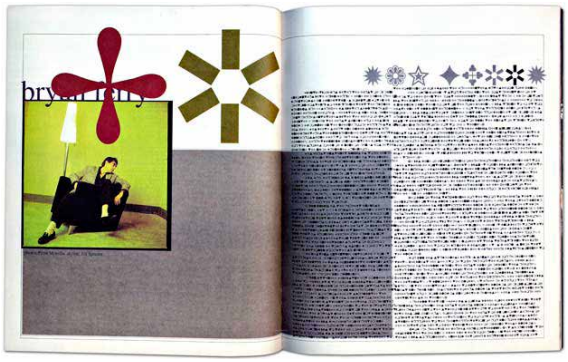


Fig. 21
Detalhe do interior de um número da *Ray Gun.*David Carson achou o um artigo de Brian Ferry chato demais para ser lido. Então decidiu aplicar a fonte *Zapf Dingbats* para que ninguém se desse ao trabalho de ler.

Carson começou a sua carreira profissional como professor no ensino médio e só em meados da década de 80 começou a fazer experiências no design gráfico. O seu interesse no mundo do surf proporcionou oportunidades de experimentar o design, trabalhando em várias publicações diferentes relacionadas com a profissão. Ainda hoje o designer é conhecido como "herói da deconstrução" e o seu estilo de experimentação tipográfica influenciou uma nova geração de designers. A sua estratégia visual, associada aos sentimentos que queria transmitir para o público e caracterizada pelo "sujo", sugere a influência que o dadaísmo teve no designer (Flask, n.d; Westgate, 2007).

A TIPOGRAFIA E A INFLUÊNCIA DAS NOVAS TECNOLOGIAS

Com o aparecimento das novas tecnologias surgiram muitas fontes digitais que pretendiam também marcar o design de tipos. A *Template Gothic* (Fig.22) é um marco devido à sua popularidade e ao tipo de letra que serviu de inspiração para o designer no seu desenvolvimento — um sinal publicado na lavandaria do seu bairro. Este tipo de letra nasceu em 1990 desenhada por Barry Deck e caracteriza-se pelo desejo de abandonar a perfeição das formas modernas. Para Deck, a representação estética da imperfeição era o seu grande objetivo (Deck, 2011).

> *I was inspired to design a face that looked as if it had suffered the distortive ravages of photomechanical reproduction*
>
> Barry Deck (citado em Deck, 2011)

Também Makela via a criação tipográfica da mesma forma e foi nessa busca da imperfeição que, no mesmo ano, projetou a *Dead History* (Fig.23). O principal objetivo de P. Scoott Makela foi ignorar a história da tipografia existente.

> *(...) personifies a new attitude in type creation... the result of the computer's capabilities to function as the perfect assembling tool.*
>
> P. Scoott Makela (citado em Makela, 2011)

O projeto iniciou-se com a combinação de duas fontes — Linotype's Centennial e Adobe's V.A.G. Rounded — para criar um tipo de letra totalmente novo e inesperado (Makela, 2011).

No mesmo contexto surge, em 1991, a *FF Fudoni* (Fig.24). Este tipo de letra concebido por Max Kisman foi criado pela combinação de dois dos tipos de letra mais utilizados no mundo: Bodoni e Futura (Fontfont, n.d.c).

Fig. 22
Template Gothic,
Barry Deck, 1990

Fig. 23
Dead History,
P. Scoott Makela, 1990

Fig. 24
FF Fudoni,
Max Kisman, 1991

No início da década de 1990 a evolução das impressoras e das tecnologias de desenho abriram portas para novos métodos de reprodução de tipos. O tipo de letra *Beowulf* apareceu juntamente com a primeira série de fontes com contornos aleatórios e comportamento programado. Erik von Blokland e Just van Rossum, programador e designer, respectivamente, foram marcantes na forma como mudaram a programação em fontes *PostScript*. Inicialmente denominaram este tipo de letra aleatório de *RandomFont*. A invenção atraiu o interesse da *FontFont* que se ofereceu para vender e distribuir o tipo, o que levou a uma renovação da fonte (Lupton, 2006).

> *Os métodos industriais de produção tipográfica forçaram todas as letras a ser idênticas (...) Hoje, a tipografia é produzida com equipamentos sofisticados, que não impõem tais regras. As únicas limitações residem em nossas experiências.*
>
> Erik von Blokland e Just van Rossum (citado em Lupton, 2006)

A fonte *FF Beowolf,* já renovada, afligiu tipógrafos e designers gráficos com o seu grande grau de deformação, pois foi vista como prova dos perigosos efeitos dos computadores na tipografia. No entanto, rapidamente começou a ser usada em posters, capas de CD e logotipos de bandas. Uma das características mais peculiares da *FF Beowolf* é o facto de não existirem dois glifos idênticos, devido ao movimento aleatório de pontos dos contornos (FontFont, n.d.b).

O tipo de letra de Erik von Blokland e Just van Rossum funcionava bem, no entanto, não era o que os fabricantes de computadores e impressoras tinham em mente para a publicação, pois era um pouco lento. Com o passar dos anos, os problemas no tipo de letra aumentam, no entanto, a esperança foi restaurada com o surgimento de tecnologias como o *OpenType*[3] (FontFont, n.d.b.).

Em 2011, o *Museu de Arte Moderna de Nova York* (MoMA) adicionou os primeiros tipos de letra digitais à sua coleção permanente. A *FF Beowolf* foi um dos apenas 23 projetos selecionados e estreou-se no museu como parte da instalação *Standard Deviations*, na galeria de design contemporâneo (FontFont, n.d.b).

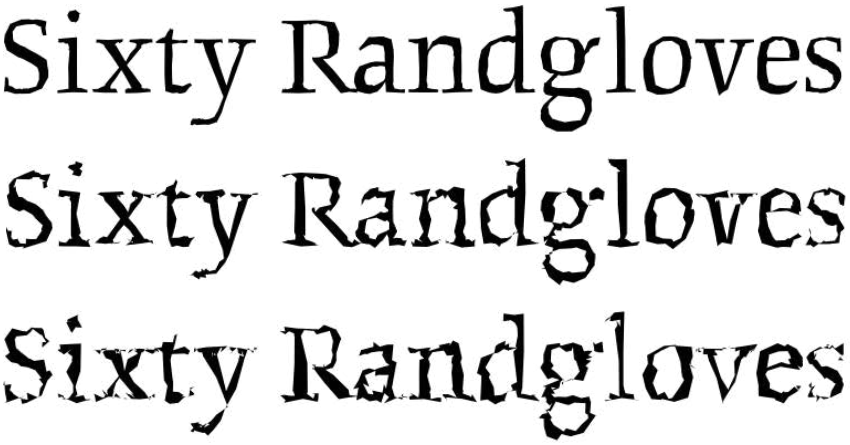


Fig. 25
Beowulf, Erik von Blokland e Just van Rossum, 1990

Em 2010, Paul McNeil e Hamish Muir decidem explorar o seu potencial e fundar o MuirMcNeil. Desde essa altura os seus projetos são identificados pela exploração de sistemas de tipos com a utilização de matemática e design generativo. Entre os projetos desenvolvidos pelo dupla, no contexto tipográfico, são de destacar o sistema *FF ThreeSix, Intersect* e *TwoPlus*.

O *FF ThreeSix* (Fig.26) é um sistema experimental de tipos de letra ópticos, composto por seis tipos, em oito pesos. Este projeto explora questões de legibility e readability no design de tipos de letra geométricos para uso em texto, além de utilizar métodos de design generativo. O sistema *ThreeSix* opera em cinco funções tipográficas: (i) contorno, a forma das letras individuais; (ii) modulação do traço, ou seja, o equilíbrio óptico entre traços horizontais e verticais; (iii) junções, os efeitos ópticos nas interseções dos traços; (iv) peso, ou seja, a acomulação de densidade nos esqueletos das formas das letras e (v) espaçamento, o ajuste das formas em sequências (FontFont, n.d.a; MuirMcNeil, n.d.c).

Este sistema tem uma série de estratégias ópticas compensatórias que são exploradas para criar a ilusão de uniformidade no tecido da matéria de leitura. Enquanto as modificações compensatórias são facilmente visíveis em tamanhos grandes, em tamanhos menores e a olho nu apenas são percebidos os efeitos gerais desses ajustes. Este sistema foi construído com restrições geométricas rigorosas, sendo que todos os caracteres são desenhados numa grelha de 36 quadrados subdivididos em nove unidades. Além disso, são construídos também a partir de um conjunto de módulos constituídos por linhas verticais ou horizontais e arcos. Quando é aumentado o peso, é diminuído os valores de suporte lateral o que enfatiza visualmente o aumento. Este sistema tem também uma ampla gama de opções visuais para designers. Utilizando *software* de desenho bitmap ou vetor, é possível aplicar contornos selecionados num registo preciso (FontFont, n.d.a; MuirMcNeil, n.d.c).

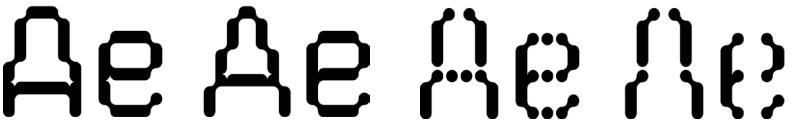


Fig. 26
Detalhes do sistema *ThreeSix,* MuirMcNeil

[3] *OpenType:*
Formato desenvolvido pela Apple e pela Microsoft para pôr fim à guerra *PostScript/ TrueType.* Neste formato existe apenas um arquivo único que contém os dados de estrutura e bitmap. Uma fonte neste formato é escalável e compatível para Mac e PC. Uma das vantagens deste formato é o facto de oferecer um conjunto de caracteres estendidos e controlos tipográficos mais avançados.

PostScript:
formato digital de alta qualidade, amplamente utilizado na composição profissional, desenvolvido em 1984. Os arquivos de uma fonte *PostScript* consistem em dois arquivos: uma fonte de ecrã com informações de bitmap e um arquivo com informações de estrutura para imprimir a fonte.

TrueType:
Formato originalmente desenvolvido pela *Apple* no final da década de 1980, mas adaptado pela *Microsoft*. O arquivo de uma fonte *TrueType* contém as informações de ecrã e estrutura, tornando-os facilmente portáteis, existindo apenas ligeiras diferenças na forma como é rasterizada a fonte em Mac e PC. Uma desvantagem deste formato é limitação na mudança de escala.

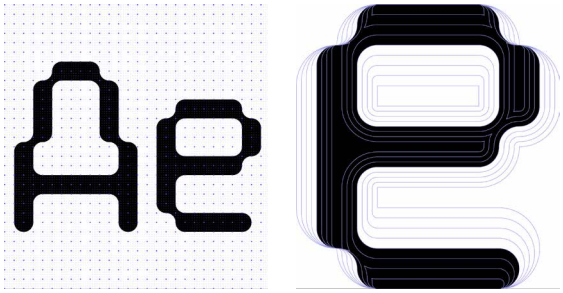


Fig. 26 (continuação)
Detalhes do sistema ThreeSix, MuirMcNeil

TwoPlus (Fig. 27) é outro sistema desenvolvido pela dupla MuirMcNeil. O projeto iniciou-se como uma coleção de tipos personalizados para uso na identidade visual da pós-graduação de 2015 do *London College of Communication*. Mais tarde, o sistema modular foi explorado mais amplamente para o desenvolvimento da identidade do *TypeCon* 2016. Este sistema é constituído por sete coleções de tipos monospaced. Cada letra individualmente funciona como componente variável dentro do sistema visual e a grelha comum determina a posição do contorno. Os tipos de letra do sistema *TwoPlus* são projetados para interagir uns com os outros. Com o software de design, o utilizador tem a possibilidade de aplicar estilos, cores, texturas e transparências. (MuirMcNeil, n.d.a)

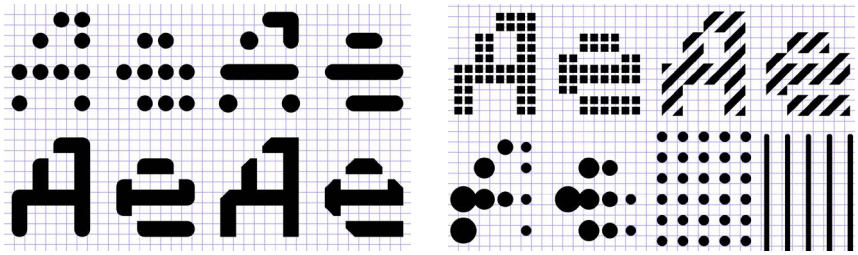


Fig. 27
TwoPlus, MuirMcNeil

Intersect (Fig. 28) é outro sistema tipográfico desenvolvido por Paul McNeil e Hamish Muir. Este sistema afasta-se da criação tipográfica tradicional no sentido em que o contraste deixa de ser binário (preto e branco; forma e contraforma). Os tipos de letra *Intersect* são constituídos por uma gama de pesos que cria a ilusão de densidade. Existem duas variantes de contorno (A e B) e estes partilham 16 padrões de tela dispostos em quatro grupos, cada um com quatro pesos. Este sistema foi projetado de forma a que as 256 configurações possam ser combinadas (MuirMcNeil, n.d.b).

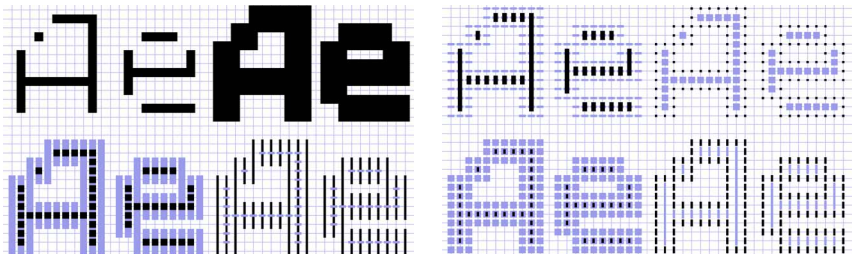


Fig. 28
Intersect, MuirMcNeil

Em 2011, a designer gráfica Dina Silanteva, graduada recentemente pela *Kingston University*, em Londres, iniciou um projeto de investigação em tipografia generativa. O projeto foi denominado *Typographic Music* (Fig. 29 e 30) e foi aplicado à teoria musical. O projeto, iniciado como pesquisa, levou à criação de um sistema de identidade para um festival de música. Utilizava uma grelha básica com três formas geométricas simples para o desenho das letras. De acordo com as regras aplicadas, eram modificados uma série de parâmetros nas formas (raio, saturação, cor, transparência) (Silanteva, n.d.; Sullivan, 2011).

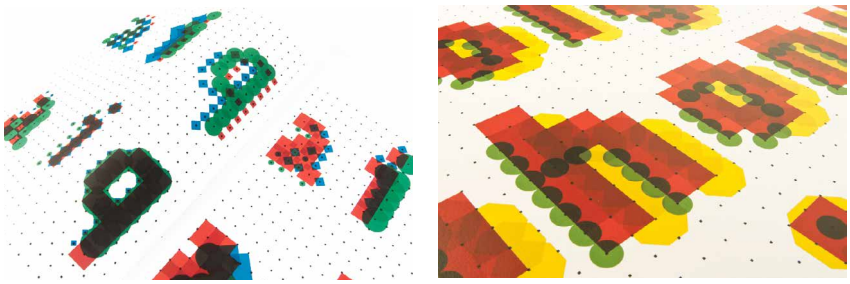


Fig. 29
Typographic Music — aplicações finais, Dina Silanteva, 2011

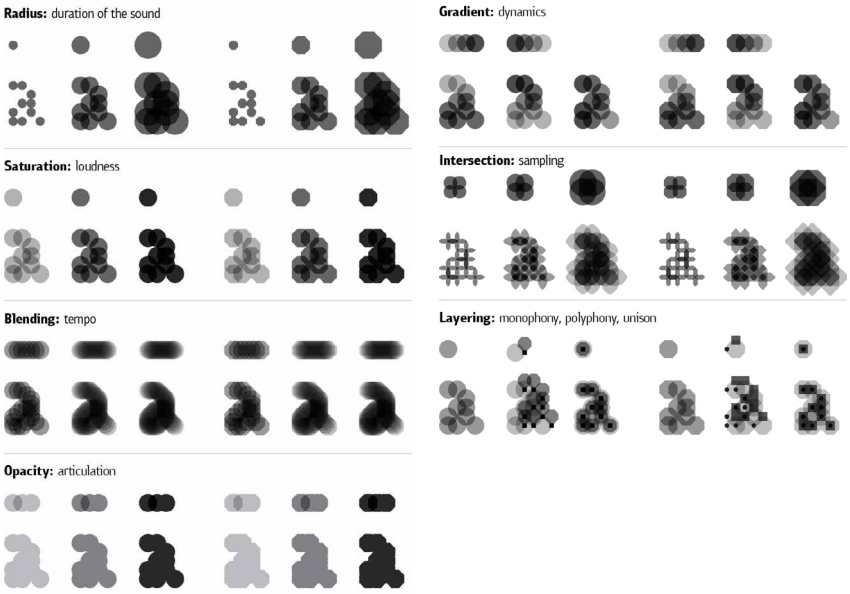


Fig. 30
Typographic Music — Diferentes parâmetros utilizados, Dina Silanteva, 2011

Em 2008, Denis Klein desenvolve uma fonte experimental e generativa em *Processing*. O tipo de letra foi denominado *Blast* (Fig. 31) e era gerada com música, caracterizando-se pelo dinamismo. O seu aspecto visual — forma e espessura — estava diretamente associado à análise da música, funcionando como uma interpretação visual da mesma. A música era analisada em tempo real e os dados obtidos fluíam diretamente para a forma visual da fonte. Através do tipo de letra é possível observar características da música (Klein, n.d.; Mainz, n.d.h).

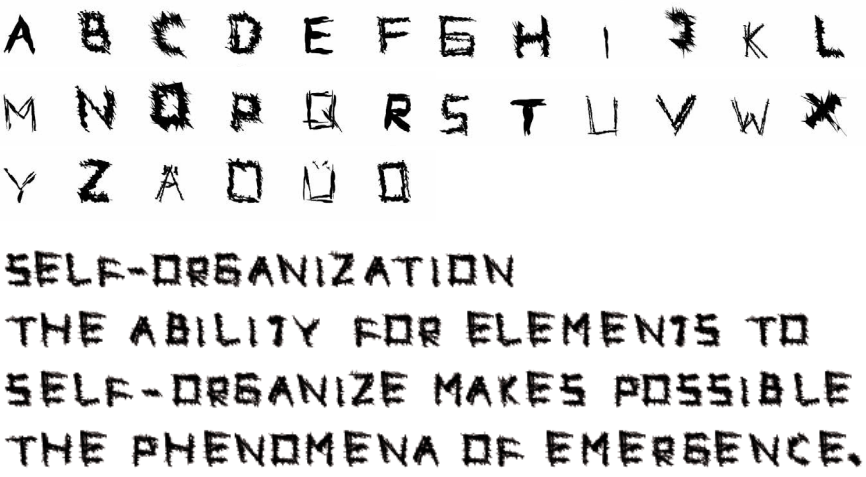


Fig. 31
Blast, Denis Klein, 2008

PDE Lubanoise (Fig. 32) foi um projeto de tipografia generativa criado por Il-Ho Jung em 2008. Neste sistema, os caracteres introduzidos são preenchidos por pontos. Outra característica é o facto dos caracteres terem um movimento ondular. Além de controlar esse movimento o utilizador tem também a

possibilidade de alterar o número de pontos que preenchem os caracteres, o seu tamanho e o rasto (Jung, n.d.).

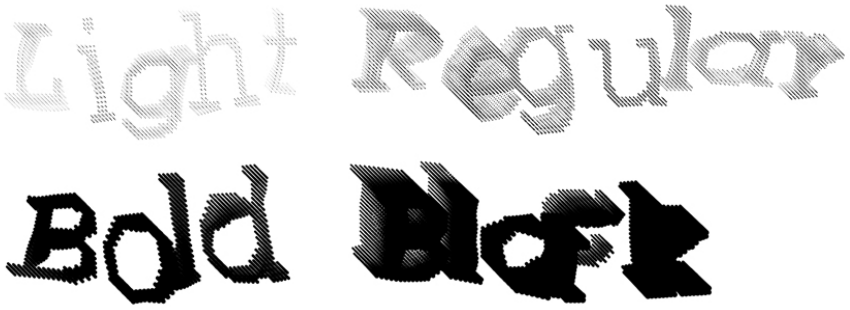


Fig. 32
PDE Lubanoise,
Il-Ho Jung, 2008

Também em 2008, Ingo Reinheimer desenvolveu a *Irratio* (Fig. 33), um tipo de letra gerado através do *Processing*. A fonte foi construída através de um tipo de letra com um esqueleto simples e sem serifa. A geração dos caracteres é feita através da criação de curvas Beziér que vão de um ponto de âncora ou vértice para outro. Considerando pontos de âncora sucessivos, é desenhada uma curva Beziér do primeiro ponto até ao terceiro, depois desde o segundo ao quarto e assim sucessivamente, percorrendo todos os pontos de ancoragem (Mainz, n.d.g).

Fig. 33
Irratio,
Ingo Reinheimer,
2008

Broken Grid (Fig. 34) é um tipo de letra criado por Nils Holland-Cunz que se assemelha a *Irratio*. Foi desenvolvido em *Processing* em 2008 e consiste no uso generativo de curvas Beziér nos pontos de âncora de uma fonte existente (Mainz, n.d.f).

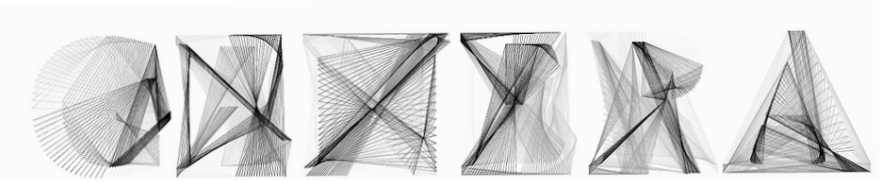


Fig. 34
Broken Grid,
Nils Holland-Cunz,
2008

Zwirn (Fig. 35) é uma aplicação em *Processing* desenvolvida por Lisa Reimann. É caracterizada por entrelaçar todas as letras da palavra inserida desenhando pontos não visíveis no contorno da fonte. Qualquer fonte no formato *TrueType* pode ser utilizada e o utilizador pode alterar vários parâmetros que influenciam o aspeto visual da fonte final, entre eles o número de linhas (Mainz, n.d.e).

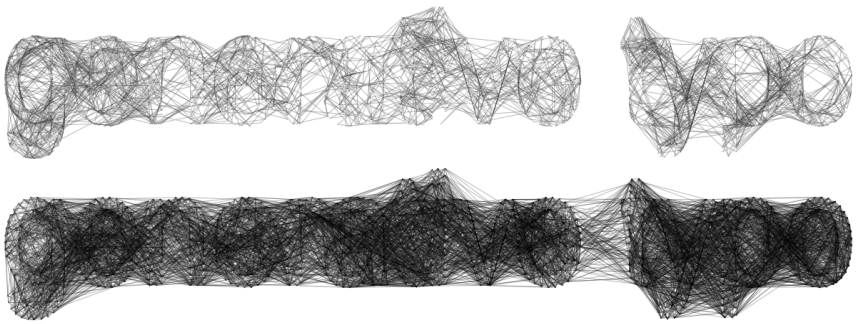


Fig. 35
Zwirn,
Lisa Reimann,
2008

Pong (Fig. 36) é outra fonte generativa desenvolvida em *Processing*. A autoria é associada a Kersten Stahl e baseia-se na metáfora de uma bola que desenha uma linha dentro de cada letra. Sempre que a aplicação é usada o utilizador pode determinar o texto que é desenhado, sendo que as linhas são variáveis na sua representação e podem variar em força e cor ou serem representadas por curvas. O nome deste projeto é uma homenagem ao jogo (Fig. 37), com o mesmo nome, em que é baseado (Mainz, n.d.d).

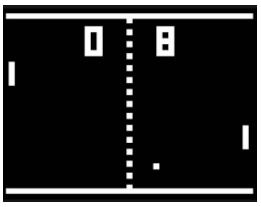
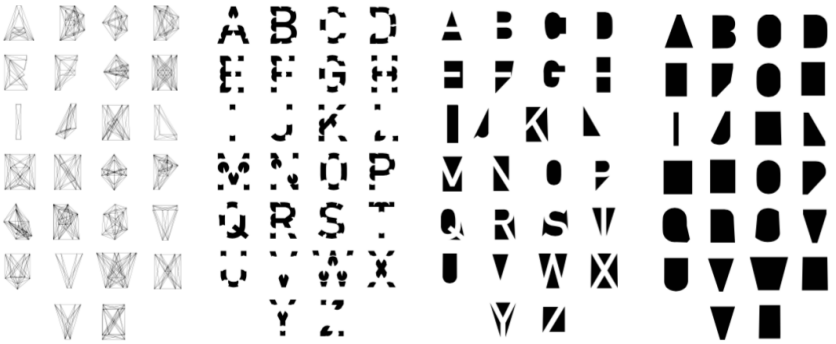


Fig. 36
À esquerda: *Pong*, Kersten Stahl, 2008

Fig. 37
À direita: *Pong* — jogo que serviu de inspiração a Kersten Stahl, Atari, 1972

Em 2002, Michael Stout cria uma família de fontes e denomina-a de *Imageability: Paths, Edges, Nodes, Districts, Landmarks* (Fig. 38). O projeto tinha como base o livro de planeamento urbano de Kevin Lynch, *Image of the City* (Fig. 39), e pretendia criar um alfabeto que respeitasse os conceitos de Lynch de planeamento e navegação no meio urbano (Willen, 2009).



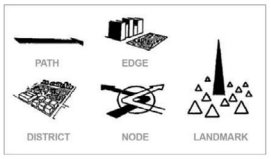


Fig. 38
Imageability: Paths, Edges, Nodes, Districts, Landmarks, Michael Stout, 2002

Fig. 39
Conceitos presentes no livro *Image of the City,* Kevin Lynch,1960

Typegalapos é outra ferramenta de criação de tipos de letra através do *Processing*. Foi desenvolvido por Ann Chen e Danne Woo e nasceu do interesse por tipografia, design evolutivo e algoritmos genéticos. O objetivo da dupla era a construção de de uma ferramenta que gerasse um novo tipo de letra personalizado a partir de fontes existentes (Chen & Woo, n.d.a.; Chen & Woo, n.d.b).

Fig. 40
Typegalapos, Ann Chen e Danne Woo

Em 2010, Tatevik Aghababyan desenvolveu a *Elien* (Fig. 41), uma fonte generativa monospace gerada em *Processing* a partir de metaballs. Estas últimas são objetos que podem conter até cinco níveis dos círculos configurados entre si e formam transições a uma pequena distância um do outro. As *metaballs* são criadas ao longo da linha do esqueleto das letras, formando novas formas entre *metaballs* individuais e letras vizinhas, e o seu tamanho é determinado aleatoriamente (Tatssachen, 2010 ; Mainz, n.d.c).

A aplicação desenvolvida permite ao utilizador a alteração de diversos parâmetros que mudam a forma da letra: (i) contraste, descreve a diferença nos tamanhos das metaballs; (ii) densidade, que determina o número de metaballs em cada linha; (iii) espacejamento, que define o espaço entre letras; e (iv) níveis, ou seja, o número de círculos nas metaballs. Este tipo de letra é constituído por 39 glifos, nomeadamente as letras maiúsculas do alfabeto latino e alguns sinais de pontuação (Elien, n.d.; Mainz, n.d.).

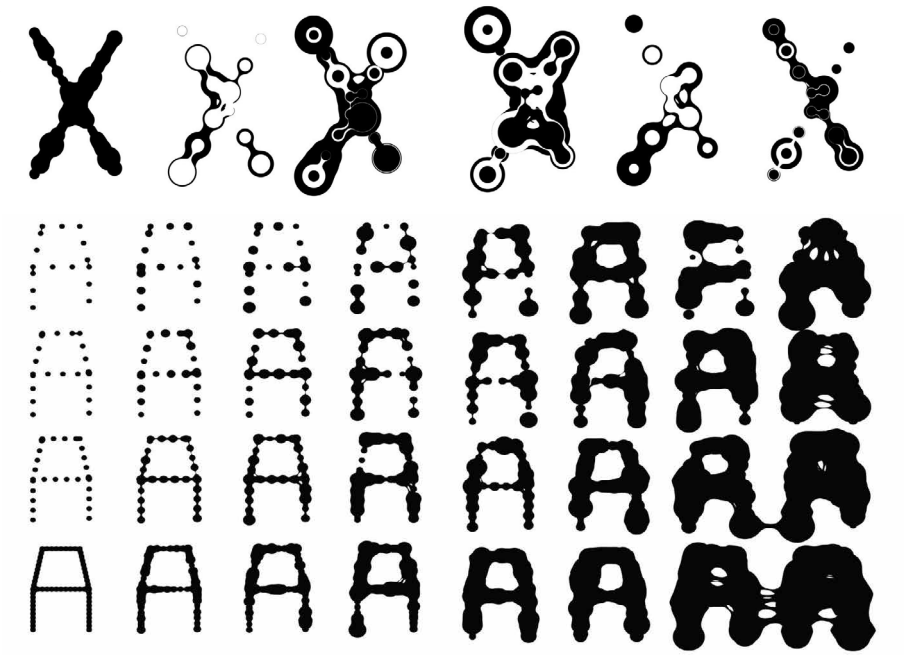


Fig. 41
Elien, Tatevik Aghababyan, 2010

No mesmo contexto, Stefanie Oppenhäuser desenvolveu um sistema que permite a combinação de três fontes distintas denominado *Fontmixer* (Fig. 42). A aplicação, desenvolvida em *Processing* em 2008, permite ao utilizador a seleção dos tipos de letra e determinar se as fontes são adicionadas ou subtraídas umas às outras.

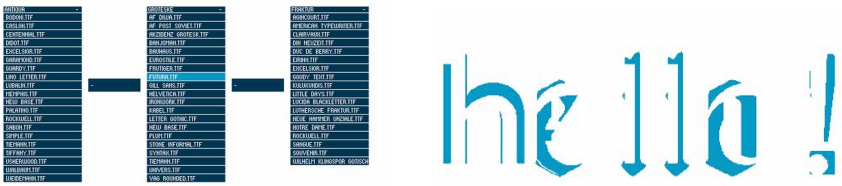


Fig. 42
Fontmixer, Stefanie Oppenhäuser, 2008

A combinação das fontes é sempre diferente e aleatória (Mainz, n.d.). *Bastard* (Fig. 43) é outra aplicação que gera tipos generativos. A ferramenta foi implementada em *Processing* com a biblioteca *Geomerative* (Marxer,n.d.) e a autoria é associada a Tobias Tshense. Os tipos são gerados através da combinação de fragmentos de fontes muito diferentes, portanto cada letra gerada é sempre diferente dos resultados anteriores. A *Geomerative* é uma biblioteca para *Processing* que facilita a geometria generativa (Marxer,n.d.). Inclui uma fonte *TrueType* e um intérprete SVG e facilita o acesso aos pontos de âncora, ajudando no desenvolvimento de peças de geometria generativas em processing (Mainz, n.d.; Geomerative, n.d.).

Fig. 43
Bastard, Tobias Tshense, 2008

GenoTyp (Fig. 44) é um sistema que gera fontes através da combinação de características genéticas de diferentes tipos de letra. As características das letras analisadas são codificadas através de software em sistemas hereditários. O sistema permite a combinação de fontes diferentes e manipulação dos seus genomas. Para combinar diferentes tipos de letra os seus códigos genéticos devem ser compatíveis. Na área de reprodução é apresentada uma árvore genealógica com as fontes originais e as gerações seguintes. A combinação de fontes diferentes pode resultar em mutações, no entanto a herança passada pode ser modificada manualmente. *GenoTyp* é também uma máquina que fornece ideias para a criação de novas fontes, devido ao carácter próprio e imprevisível das gerações. Este sistema foi publicado no livro *Generative Gestaltung* e apresentado em diversos eventos: *Typo-Berlin* (2004), *Viper Basel* (2004), *Generative Art Conference Milan* e *Transmediale* (2005) (GenoTyp, n.d.).

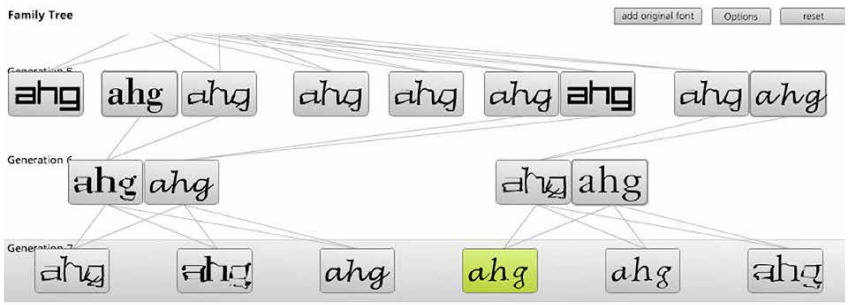


Fig. 44
GenoTyp, Michael Schimtz, 2004

Em 2009, Michael Flückiger e Nicolas Kunz desenvolveram um tipo de letra dinâmico, em Processing, denominado de *LAIKA* (Fig. 45). Esta fonte é caracterizada pela variação de peso, contraste, tamanho da serifa e inclinação de forma dinâmica. Michael e Nicolas acreditavam que o facto de um tipo de letra ser construído para tela significava que não precisava de ser rígido. Por esse motivo criaram uma fonte interativa que dá aos utilizadores a possibilidade de alterar parâmetros em tempo real (Flueckiger, n.d.; Flueckiger, 2009).

Este tipo de letra ganha potencial na aplicação em protótipos, com a combinação de entradas interativas, audiovisuais, sensores ou dados solicitados na Internet em tempo real (Fig. 46). Deste modo, a *LAIKA* pode ser utilizada em textos publicitários dinâmicos e inspirar uma nova abordagem de utilização dos tipos digitais (Flueckiger, n.d.; Flueckiger, 2009).

Fig. 45
LAIKA, Michael Flückiger e Nicolas Kunz, 2009

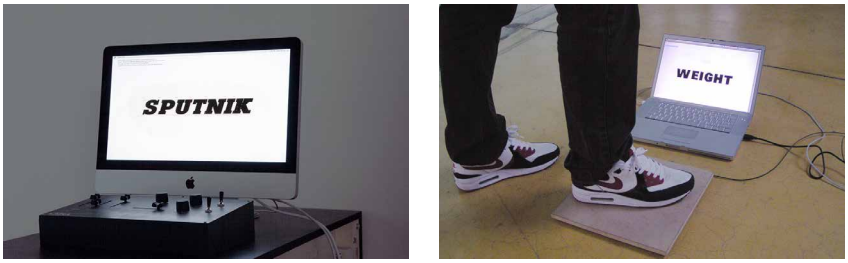


Fig. 46
LAIKA — protótipos Michael Flückiger e Nicolas Kunz, 2009

Metaflop (Fig. 47) é uma aplicação web de fácil utilização para modulação de fontes. Foi desenvolvido por Alexis Reigel e Marco Müller, em 2012, e utiliza a linguagem de programação metafont. A utilização desta linguagem permite a personalização de uma fonte facilmente dentro dos parâmetros disponibilizados (e.g. dimensões, proporções, forma, entre outros) e gerar uma grande variedade de famílias de fontes. Os tipos utilizados como base para a modulação são escolhidos da biblioteca disponível e não é necessário lidar com linguagens de programação. Os tipos de letra criados podem ser guardados como um pacote *webfont* ou como fonte *postscript* de tipo *opent* (.otf). A aplicação é de código aberto e portanto está disponível para qualquer interessado (Reigel & Müller, 2012).

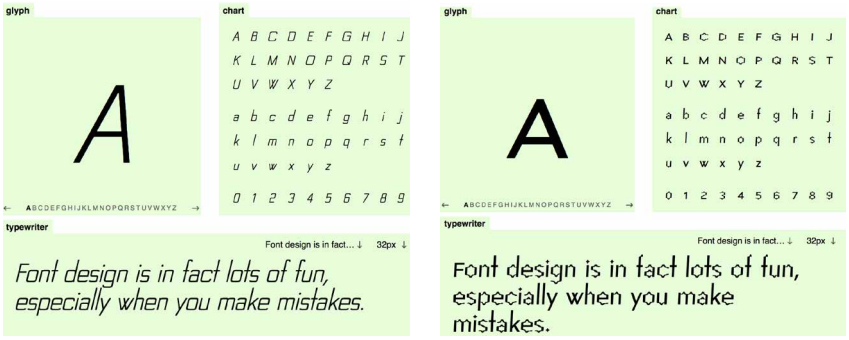


Fig. 47
Metaflop, Alexis Reigel e Marco Müller, 2012

Prototyp-0 (Fig. 48) é outra aplicação que gera um tipo de letra modular. Foi implementada em *Processing* em 2010 por Yannick Mathey. Retrata uma grande vantagem da computação no design de tipos, pois automatiza o processo. O uso deste interface estende as alterações feitas numa letra para os restantes glifos em tempo real, ao invés de se ficar horas fazendo ajustes. Esta ferramenta oferece possibilidades de criação e experimentação (Nascimento, 2011; Friends, 2011).

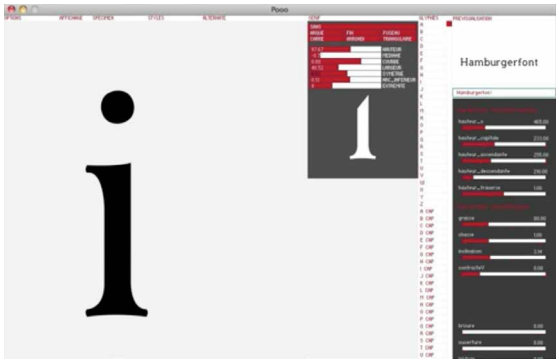


Fig. 48
Interface do *Prototyp-0*, Yannick Mathey, 2010

Em 2002, Jürg Lehni publicou o *Scriptographer*, um *plugin* de *JavaScript*. Um dos principais objetivos do plugin era a geração de complexidade visual. Depois da geração os resultados podem ser modificados através de ferramentas já existentes no Illustrator. Foi através desta ferramenta que Jonathan Puckey criou *Pencil Tool* e *Tile Tool* (Fig. 49). Este último é uma aplicação que à medida que o utilizador desenha um caminho este substitui por blocos previamente definidos (Womack, 2011).

> *A tool doesn't have to be something as simple as a fill bucket (...) It can also be a very personal and precisely defined thing that maybe only a couple of designers will ever be interested in using.*
>
> Jonathan Puckey (Womack, 2011)

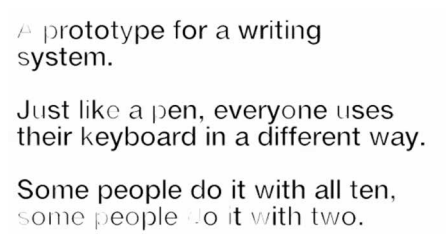


Fig. 49
À esquerda: *Tile tool*, Jonathan Puckey, 2006

Fig. 50
À direita: *Typographic Rhythm*, Jonathan Puckey

Typographic Rhythm (Fig. 50) é outro projeto desenvolvido por Jonathan Puckey. À medida que o utilizador escreve um texto a ferramenta escolhe entre os 140 pesos disponíveis dependendo da velocidade de escrita (Puckey, 2005).

Em 2007 os Catalogue Tree desenvolveram a identidade visual (Fig. 51) para os arquitetos *Monadnock* de Rotterdam em colaboração com Lutz Issler (programador) . O logotipo era gerado a partir de um arquivo *postscript* utilizando dois ou mais caracteres da palavra *Monadnock* (Catalogtree, 2007).

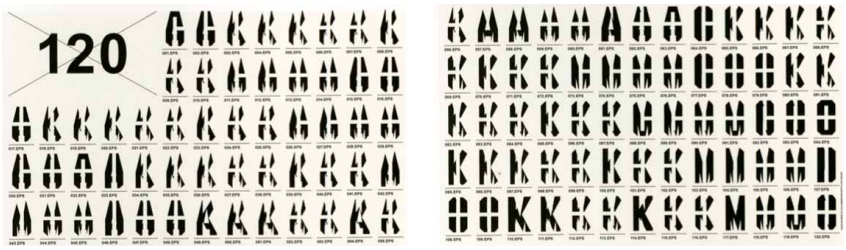


Fig. 51
Logo *Monadnock*, Catalogue Tree e Lutz Issler, 2007

TypeFace (Fig. 52) é outro projeto desenvolvido em *Processing*. A autoria é associada a Mary Huang e consiste num *software* que traduz expressões faciais num tipo generativo. É utilizada a biblioteca *openCV* para a detenção das expressões do utilizador. O projeto foi apresentado no *Siggraph 2010 Student Research Competition* (Huang, 2011).

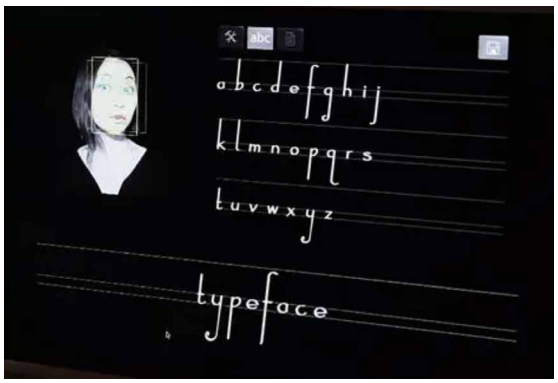


Fig. 52
TypeFace, Mary Huang, 2010

Coloquy (Fig. 53) é um tipo de letra desenvolvido por Joel Baker para o projeto final do curso *Graphic Design Communication na Chelsea School of Art and Design.* Este projeto segue a linhagem de design gráfico que começou com os tipos de letra desenhados através de uma grelha como o *New Alphabet* desenhado por Wim Crouwel. O projeto *ThreeSix* de Hamish Muir e Paul McNeill foi também uma grande referência para Baker no desenvolvimento deste tipo de letra generativo (Baker, n.d.).

Este projeto é caracterizado por uma análise ao conteúdo do texto e alteração, em tempo real, da forma da letra consoante os resultados obtidos. Nesta fonte a extensão, a expansão, a distorção e a abstração são utilizadas para evocar uma resposta (Baker, n.d.).

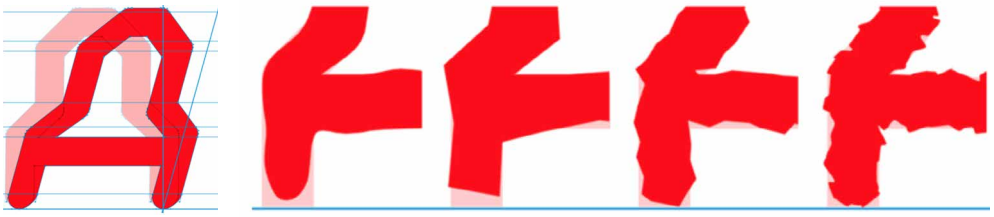


Fig. 53
Coloquy, Joel Baker

Em 1992 nasceu o projeto *Letter Spirit* (Fig. 54) por Gary McGraw que consistia numa tentativa de modelar a percepção e criatividade humana num computador. Basicamente, o que se pretendia era perceber como é que a criatividade funcionava através do uso de tipos de letra. O projeto aborda duas questões importantes na forma das letras: a semelhança categórica possuída por letras pertencentes a uma categoria (exemplo: 'a') e a semelhança estilística possuída por letras pertencentes a um estilo (exemplo: *Helvetica*). Através de um número reduzido de letras representativos do início de um estilo o programa tentará criar o resto do alfabeto de modo que todas as letras partilhem do mesmo aspeto visual (McGraw, n.d.).

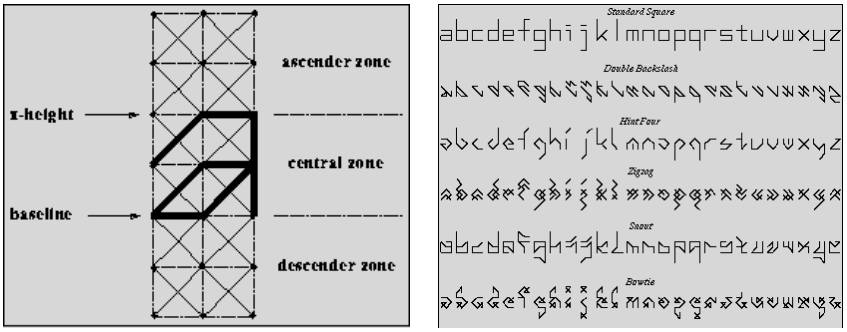


Fig. 54
Letter Spirit, Gary McGraw, 1992

Em 2001, Golan Levin, Jonathan Feinberg e Cassidy Curtis desenvolvem um projeto que permite a criação e desenvolvimento de sistemas de escrita. O projeto, denominado *Alphabet Synthesis Machine* (Fig. 55), é composto por dois sistemas de software: (i) um *applet*[4] interativo do lado do cliente que permite aos utilizadores a criação e desenvolvimento de formas de letras abstratas; (ii) um sistema de arquivo do lado do servidor que armazena as criações de utilizadores em ficheiros *TrueType* que permitem o download.

O algorítmo desenvolvido evolui uma população de glifos candidatos de acordo com um conjunto de métricas de fitness estabelecidas pelo utilizador. Os glifos desenvolvidos evoluem enquanto indivíduos, a fim de melhorar as suas características individuais, e como uma espécie. Este projeto envolve forças como a resposta dos músculos das mãos às taxas de disparo neuronal, a gravidade e o atrito da caneca contra a superfície de escrita virtual. Depois de evoluído os glifos são convertidos em contornos quadráticos de Beziér e guardados no servidor. O projeto foi dado a conhecer através de uma

Fig. 55
Alphabet Synthesis Machine, Golan Levin, Jonathan Feinberg e Cassidy Curtis, 2001

instalação interativa e uma ferramenta online. Desde o ano da criação até 2006 foram produzidos mais de 20.000 sistemas de escrita abstrata no site (Levin, Feinberg, Curtis, 2001).

FF3300 é um estúdio de design criado por Alessandro, Nicolò e Carlotta. Em 2010 desenvolveram a identidade visual para a *Imaginifica* (Fig. 56), um centro de pesquisa e consultoria que aplica a imaginação, narração e a neurociência à política, às instituições e ao mercado. Foi criada uma imagem visual através da aplicação de camadas e posterior sistema para a geração dos caracteres (FF3300, 2011; FF3300, 2012).

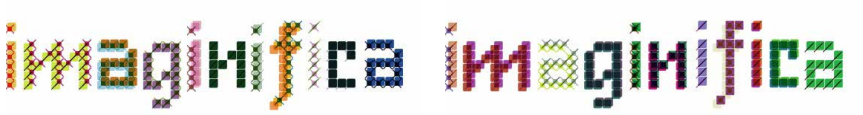


Fig. 56
Versões da Identidade visual para a *Imaginifica* , FF3300, 2014

Em 2014 desenvolveram a identidade gráfica para o *Circuito D'Autore* (Fig. 57 e 58) em colaboração com Silvestro Ferrara, Antonio Vergari, Danilo di Cuia e Pippo Fuina. O projeto foi desenvolvido para a *Apulia Film Commission* e além da identidade foi elaborada uma estratégia de comunicação, publicações editoriais, campanha publicitária, duas publicações especiais, site e aplicação móvel. Neste contexto faz sentido abordar a identidade visual porque é baseada na combinação de caracteres (FF3300, n.d.).

Fig. 57
Processo de geração do logo para a identidade visual do *Circuito D'Autore*, FF3300, 2014

Fig. 58
Identidade visual do *Circuito D'Autore,* FF3300, 2014

Em 2006, Jonathan Keller desenvolveu um script para o *Adobe Illustrator* que gerada glifos a partir dos tipos de letra instalados. O sistema, denominado *Slitscan* (Fig. 59), alinha todas as letras disponíveis e depois corta em fatias. A cada geração é criada uma nova versão e aplicada uma cor aleatória para cada uma das fontes disponíveis (Willen, 2009).

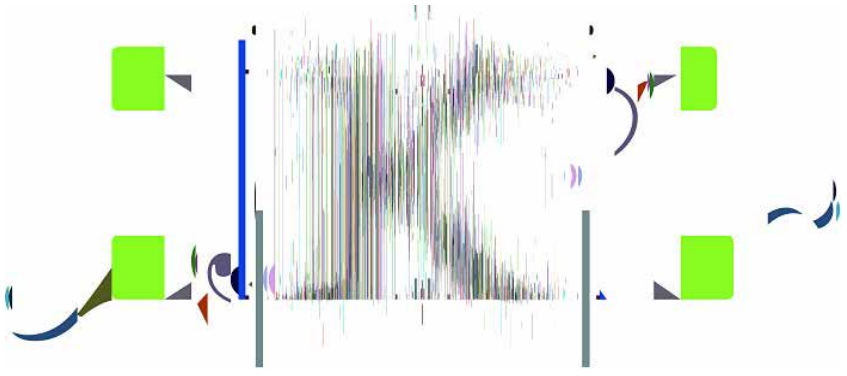


Fig. 59
Slitscan, Jonathan Keller, 2006

Post Bitmap Scripter (Fig. 60) é um script que tenta recriar um tipo letra original através de fontes de bitmap. Consiste no desenvolvimento de versões abstratas e reduzidas de bitmap ou fontes de pixels e converte-as em linhas (Tools, 2007).

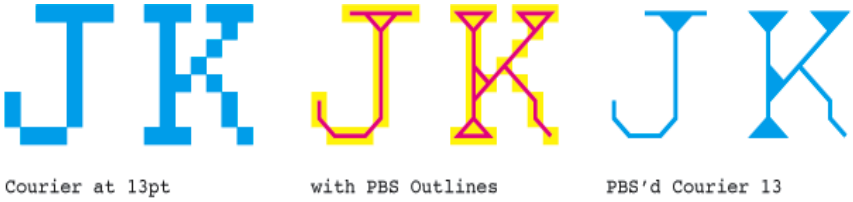


Fig. 60
Post Bitmap Scripter, c71123.com, em construção

III.III Conclusão do estado da arte

No final do capítulo do estado da arte temos a informação necessária para iniciar o projeto prático desta dissertação.

O objetivo desta dissertação, como abordado anteriormente, é a criação de um sistema capaz de desenhar tipos de letra de forma automática. Como tal torna-se necessário algum conhecimento prévio do assunto de modo a criar tipos de letra com qualidade. Para isso foram revistos termos relativos à anatomia da letra e à sua classificação. Foram analisados movimentos artísticos que influenciaram o desenho de tipos e estudados trabalhos e possibilidades já conseguidas no domínio do desenho algorítmico e dinâmico de tipos.

Mais especificamente relativamente ao trabalho neste contexto foram dados a conhecer tipos de letra que além da responderem à sua finalidade possuíam uma camada de conhecimento. Estes projetos abriram também ideias para o desenvolvimento do projeto prático desta dissertação como a combinação de tipos de letra, ou a extração de partes anatómicas, a utilização de camadas e a criação de sistemas dinâmicos que permitem aos tipos a adaptação a diversos contextos.

IV. PROJETO PRÁTICO

Neste capítulo é abordado todo o trabalho prático desenvolvido no contexto desta dissertação. Em primeira instância é conceptualizada a ideia e definidos os objetivos iniciais. Mais tarde, são apresentadas uma série de experimentações iniciais criadas com o objetivo de marcar um ponto de partida para o projeto prático.

Durante o desenvolvimento do projeto surgiu também a ideia da criação de um sistema que respondesse a um dos objetivos desta dissertação, a criação de tipos de letra que além de servirem o seu propósito comum, acrescentassem conhecimento. Neste capítulo é também apresentado esse sistema, juntamente com as suas mais-valias para esta dissertação.

Num fase final, são também apontados os estudos e pormenores técnicos que foram necessários ao desenvolvimento do projeto prático final.

IV.I Concetualização

Ao longo do tempo, e principalmente com o surgimento das novas tecnologias, têm sido criados tipos de letra dinâmicos e generativos que respondem e se adaptam às circunstâncias onde estão inseridos.

Para esta dissertação pretendiamos criar um equilíbrio entre a tipografia tradicional, a tipografia como narrativa e a generatividade para os tipos de letra. Um sistema que além de respeitar as regras de construção tipográfica crie um equilíbrio entre o que o utilizador altera e o que o sistema executa automaticamente. Para isso será criado um sistema computacional que vai separar os diferentes elementos da anatomia das letras em camadas. Os tipos de letra gerados serão resultado da combinação de camadas de diferentes tipos de letra. Os tipos de letra gerados poderão ser modificadas de acordo com *inputs* externos.

O nosso objetivo é permitir a geração de glifos de forma automática e coerente.

A AS CAMADAS E A COMBINAÇÃO ENTRE ELAS

Pretende-se que os tipos de letra desenvolvidos neste sistema sejam criados através da conjugação de diversas camadas — elementos da anatomia da letra — e a sua posterior modificação. O projeto *Bastard* (Fig. 61), de Tobias Tshense, abordado no capítulo anterior, é um exemplo da criação de um tipo através da combinação das diferente partes anatómicas.

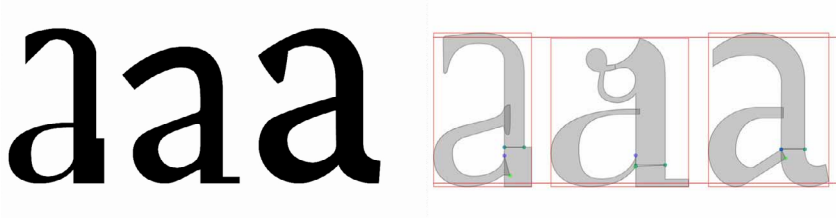


Fig. 61
Bastard (detalhe das camadas combinadas em caracteres gerados), Tobias Tshense, 2008

No nosso sistema, depois de estabelecidas diferentes camadas o objetivo era a combinação entre elas e criação de um sistema para o preenchimento.

B OS TIPOS DE LETRA E A NARRATIVA CRIADA

Um dos nossos objetivos neste projeto era a criação de tipos constituídos por diversas partes anatómicas. Além disso, foi considerado desde o início a forma como os tipos gerados se poderiam adaptar às circunstâncias em que estão inseridos. A narrativa criada pela forma visual dos tipos criados é um ponto relevante na sua construção. Além disso, pretendiamos também que cada tipo gerado fosse personalizado, mas até certo ponto.

Para que os glifos gerados acrescentassem uma camada de conhecimento, um ponto essencial no sistema a desenvolver era o facto de permitir a alteração de parâmetros. Uma das variáveis do sistema poderia ser a forma como cada camada era preenchida. Cada tipo de letra gerado poderia ser alterado se aplicado numa certa circunstância. O sistema poderia receber vários tipos de *inputs*, sendo aqui apresentadas algumas possibilidades:

(i) som, e.g. a altura e/ou frequência da voz de uma pessoa, de uma música ou de som ambiente;
(ii) caligrafia, e.g. altura média das letras e inclinação;
(iii) períodos e circunstâncias históricas, sociais ou culturais, e.g. datas, ou número de pessoas.

IV.II Experimentações inicias

Durante a investigação do Estado da Arte surgiram ideias e possibilidades para o desenvolvimento da parte prática deste projeto. Estas primeiras experimentações aparecem como resposta à necessidade de visualizar essas ideias e ver que caminho este projeto deveria tomar.

À partida já estava estabelecido, em geral, como seria a geração dos tipos de letra. No entanto, existiam ainda muitos aspectos desconhecidos. A primeira preocupação foi a estrutura básica das fontes geradas. Uma vez que o objetivo não era modificar apenas um tipo de letra existente, teria de se arranjar uma forma de criar uma estrutura base.

A A PROCURA DE UM ESQUELETO

Esta primeira abordagem aparece como tentativa de criação de um esqueleto. Pretendiamos estabelecer uma estrutura que, mais tarde, serviría como base para a criação da fonte gerada.

Em suma, o nosso objetivo era gerar um esqueleto de forma automática. No entanto, não tínhamos reunidas informações relativas às características de cada caractere para desenhar. Inicialmente, optamos por comparar diversos tipos de letra — *Adobe Garamond, Baskerville, Scala, Helvetica, Gill Sans e Aktiv Grotesk* — pertencentes a diferentes categorias de classificação tipográfica e encontrar semelhanças para o mesmo caractere. Pretendiamos descobrir o que era comum em todos os 'a's independentemente do tipo de letra, ou seja, o que faz de um 'a' um 'a'.

Mais tarde, criamos um método que nos ajudasse a comparar tipos de letra e que não fosse apenas especulativo. Desenvolvemos um pequeno sistema que, semelhante ao projeto de Stefanie Oppenhäuser, *Fontmixer* (Fig. 62) já abordado anteriormente, combinava diferentes tipos de letra (Mainz, n.d.).

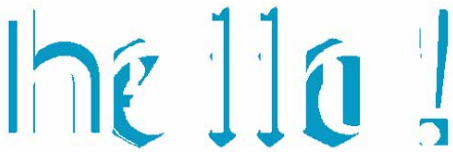


Fig. 62
Fontmixer, Stefanie Oppenhäuser, 2008

O sistema criado foi desenvolvido em *Processing* e baseia-se na sobreposição de diferentes glifos do mesmo caractere, mantendo apenas as partes que se repetem.

Resumidamente, o sistema percorre os *pixels* de cada glifo e guarda aqueles que existirem em todos os tipos de letra. Para isto, foram utilizados seis glifos do 'a' (Fig. 63) correspondentes aos seis tipos de letra referidos anteriormente.

Fig. 63
Os glifos utilizados nesta primeira abordagem: *Adobe Garamond, Baskerville, Scala, Helvetica, Gill Sans* e *Aktiv Grotesk*

Foram testadas várias combinações de glifos e em primeira instância foram comparados elementos com a mesma largura ou com o mesmo tamanho. No entanto, foram obtidos melhores resultados combinando elementos com a mesma altura-x. Os resultados obtidos podem ser observados na Fig. 64. Da esquerda para a direita são combinados cada vez menos glifos, além de existir de cada vez uma maior semelhança entre glifos.

Fig. 64
A procura de um esqueleto — Resultados da combinação de diferentes "a"

Entre os resultados obtidos é possível observar a diferença de mancha existente. Esse aspecto deve-se ao facto de existir, entre experiências, variação no número de glifos sobrepostos. Além disso, é natural que tipos de letra pertencentes a diferentes categorias não tenham tantos aspectos em comum como fontes com a mesma classificação (a letra "a" da *Adobe Garamond* é muito diferente do "a" da *Helvetica* — *Garaldes* vs *Lineáles*). Este aspecto foi minimizado ao longo da experiência com o ajuste da altura-x entre os glifos utilizados.

No âmbito desta primeira experiência foi também desenvolvida uma outra abordagem que derivou desta. Neste caso os mesmos glifos foram combinados e sobrepostos. Alguns dos resultados estão apresentados na Fig. 65.

Fig. 65
Resultados da sobreposição de diferentes "a"

Problemas desta abordagem :

Como referido anteriormente, os resultados obtidos não acompanharam, em parte, o esperado. Pretendia-se chegar a um esqueleto base e apenas se consegue um resultado minimamente aceitável com a combinação de poucos glifos. Grande parte do problema deve-se ao facto dos esqueletos dos tipos de letra utilizados serem muito diferentes.

Possíveis soluções:
Para se chegar a melhores resultados o ideal seria combinar tipos de letra com estruturas semelhantes, ou seja, pertencentes à mesma categoria de fontes. A divisão de tipos de letra serifados e não serifados poderia também produzir melhores resultados.

Possibilidades:
Com esta abordagem surgiu a ideia de criação de tipos de letra através do seu esqueleto. Além disso, originou formas de comparação de tipos de letra. A utilização deste sistema na geração de diversos esqueletos, através da associação de tipos de grupos de classificação tipográfica, abriu novas portas de exploração para este projeto.

B UMA GRELHA PREDEFINIDA

Depois da primeira abordagem já existiam algumas possibilidades em aberto e, apesar da estrutura dos tipos de letra gerados ser de uma importância elevada, o seu aspecto visual ainda não estava definido. Nesta segunda experiência tentamos preencher essa lacuna e utilizar a aleatoriedade como *input* externo para o preenchimento dos caracteres. Esta abordagem foi desenvolvida, também ela, em *Processing* e serviu como tentativa de marcação do aspecto visual.

Como ainda não estava definida uma estrutura para os glifos gerados foi criada uma grelha para o desenho das letras. Com a criação deste segundo sistema apenas pretendíamos focar no preenchimento das letras, portanto criamos uma grelha com o tamanho mínimo necessário para o desenho e diferenciação dos caracteres. Na Fig. 66 pode ser observadada a grelha criada juntamente com três caracteres.

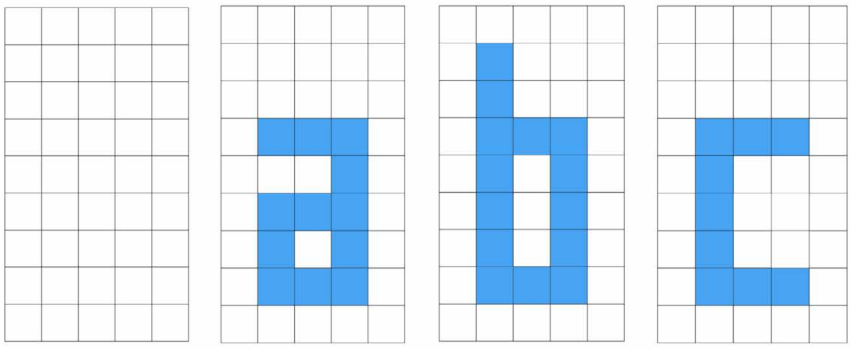


Fig. 66
Grelha e caracteres desenhados na grelha

O sistema é constituído por uma função que recebe que elementos da grelha devem ser "pintados" de acordo com a letra a ser desenhada. Depois, cada módulo é preenchido da mesma forma. A partir desta base foram elaboradas três funções correspondentes a três formas de preencher as letras.

Na primeira solução (Fig. 67) os elementos têm uma cor, dentro da paleta disponível, que é alterada aleatoriamente a cada geração. Nesta abordagem cada módulo é dividido em vários divisões e cada uma delas têm a probabilidade de 90% de ser desenhada. Também em cada módulo é desenhado um círculo que varia a cor aleatoriamente. Nesta abordagem a maior variação em cada geração é a mudança de cor.

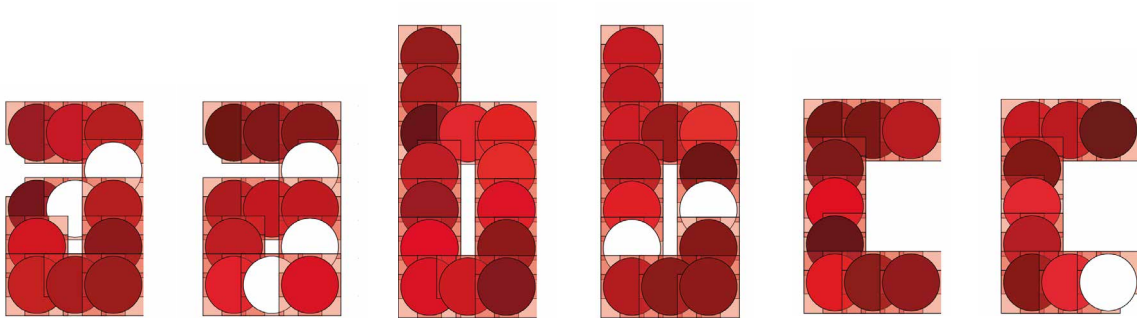


Fig. 67
Glifos gerados no âmbito da segunda experimentação, primeira solução

Mais tarde, como referido anteriormente, foi criada uma segunda função de preenchimento. Nesta, Fig. 68, existe uma maior aleatoriedade, a grelha inicial ganha o dobro das divisões e em cada uma delas são desenhados quadrados brancos. Em cada subdivisão são desenhados vários quadrados pretos com a probabilidade de 60% de serem desenhados. Devido ao facto de a aleatoriedade condicionar o desenho ou não de um elemento é possível observar uma maior alteração de geração para geração.

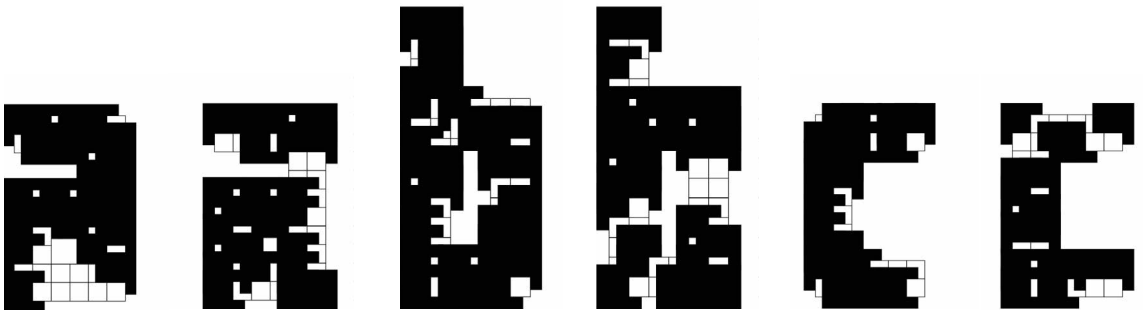


Fig. 68
Glifos gerados no âmbito da segunda experimentação, segunda solução

Depois de duas primeiras tentativas de marcação do aspecto visual decidimos desenvolver uma terceira abordagem. Nesta, Fig. 69, para cada módulo da grelha principal criamos 8 divisões. Para cada um dessas subdivisões existia uma probabilidade de 25% de serem desenhados 2 círculos (um círculo a branco com contorno a preto e um, ligeiramente mais pequeno que o anterior, apenas preto). A posição dos círculos tinha uma aleatoriedade controlada, pois tem os limites do módulo como referência.

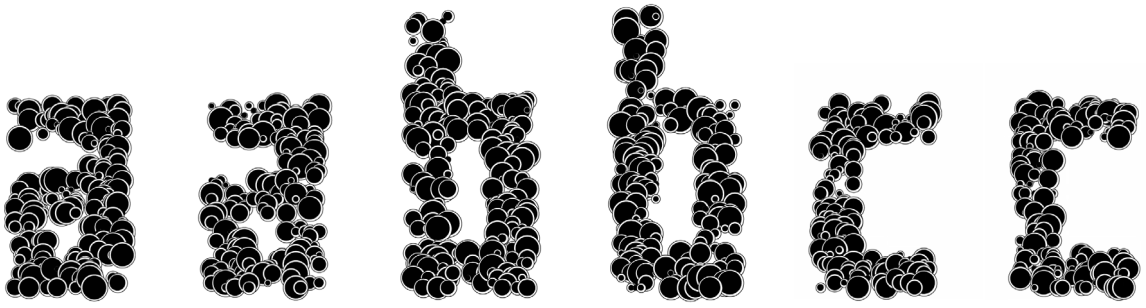


Fig. 69
Glifos gerados no âmbito da segunda experimentação, terceira solução.

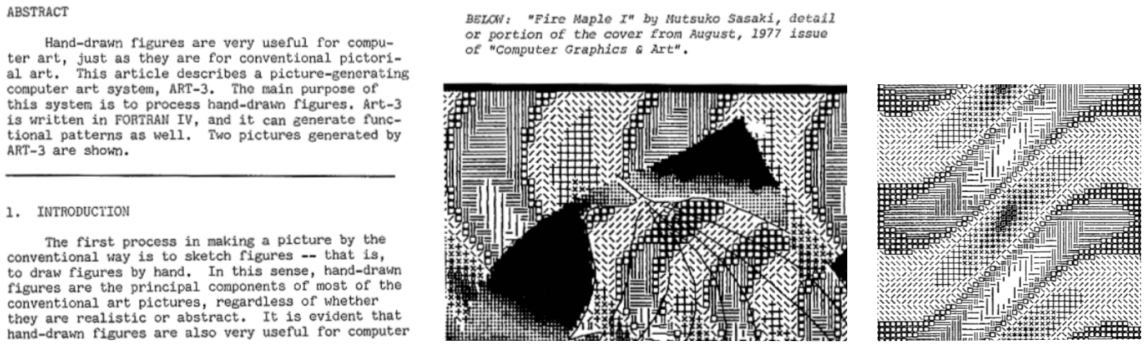
Problemas desta abordagem:

Após três experiências demos por terminada a segunda abordagem. Ao observar os resultados obtidos considerámos que as alterações de geração para geração não são muito significativas devido à grelha previamente definida. Um aumento da aleatoriedade poderia melhorar este aspecto, no entanto a legibilidade estaria condicionada.

Possíveis soluções:
Depois desta segunda tentativa, a hipótese da criação de um esqueleto tornou-se uma solução mais viável, pois poderia permitir uma melhor personalização dos glifos gerados.

C CAMADAS SOBREPOSTAS

Esta abordagem surgiu no âmbito da disciplina de Design Generativo. O que se pretendia era uma nova abordagem de um projeto apresentado na revista *Computer Graphics and Art*. A revista de Agosto de 1978 tinha um artigo denominado *Computer Art System: Art-3* (Fig. 70) que abordava um projeto que criava uma série de padrões. O objetivo desse projeto era a aplicação de texturas em desenhos através de código. A partir de um desenho eram definidas, programaticamente, áreas interiores e aplicadas texturas. Este artigo abriu novas possibilidades para a geração de glifos deste projeto.



ABSTRACT

Hand-drawn figures are very useful for computer art, just as they are for conventional pictorial art. This article describes a picture-generating computer art system, ART-3. The main purpose of this system is to process hand-drawn figures. Art-3 is written in FORTRAN IV, and it can generate functional patterns as well. Two pictures generated by ART-3 are shown.

1. INTRODUCTION

The first process in making a picture by the conventional way is to sketch figures -- that is, to draw figures by hand. In this sense, hand-drawn figures are the principal components of most of the conventional art pictures, regardless of whether they are realistic or abstract. It is evident that hand-drawn figures are also very useful for computer

BELOW: "Fire Maple I" by Nutsuko Sasaki, detail or portion of the cover from August, 1977 issue of "Computer Graphics & Art".

Fig. 70
Computer Art System: Art-3, Agosto de 1978

Tendo este artigo como inspiração, foi desenvolvida a terceira abordagem em *Processing*. Para esta experiência foram desenhadas a priori algumas letras (Fig. 71).

Fig. 71
Letras desenhadas

Mais tarde, foi desenvolvido um sistema que utiliza os vértices das letras desenhadas e gera versões dos glifos. O sistema foi desenvolvido de modo a transformar os vértices num polígono. Depois, são percorridos todos os pixels do ecrã da aplicação; se estes fizerem parte do polígono, é desenhado um carácter. Ao longo do desenvolvimento deste sistema, os caracteres utilizados foram sendo alterados. Em grande parte dos resultados obtidos foram utilizados caracteres aleatórios que variaram entre a letra a ser desenhada e sinais de pontuação. Outra característica desta abordagem é o facto de na zona central da letra os caracteres terem uma probabilidade associada que define se estes são desenhados ou não.

Nesta abordagem existe também o conceito de camadas que faz variar a mancha de cor. Depois da criação do polígono acima definido, é sorteado o número de camadas que a letra gerada terá. Na Fig. 72 são apresentados alguns dos glifos gerados.

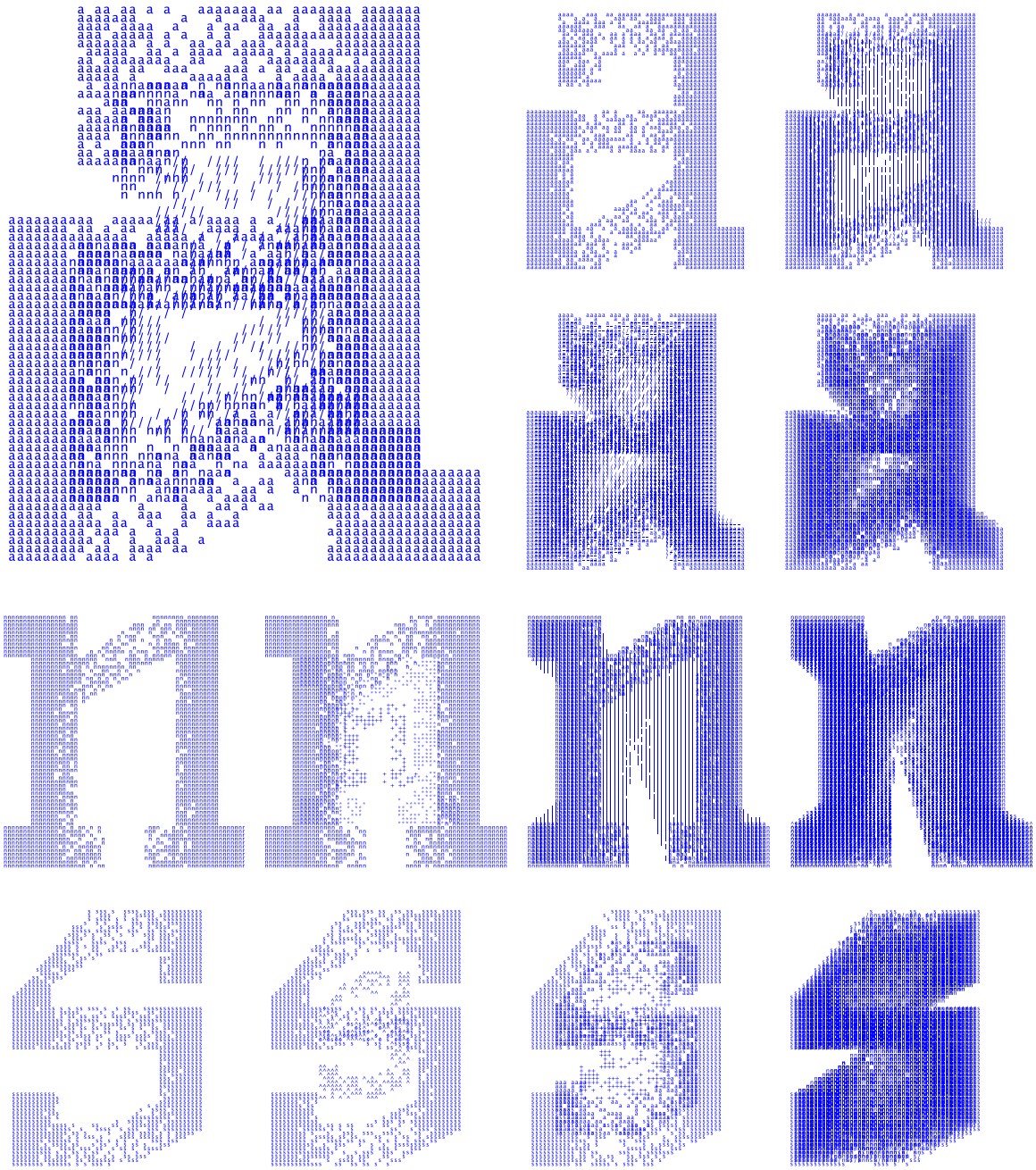


Fig. 72
Glifos gerados no âmbito da terceira experimentação

Problemas desta abordagem:
Relativamente às abordagem anterior esta apresentou uma melhoria no ponto em que existem alterações significativas no aspecto em cada geração. No entanto, a estrutura da letra ainda se mantinha em todas as gerações e esse era um factor a melhorar.

Possíveis soluções:
A partir desta abordagem decidimos que o ideal seria afastarmo-nos de uma grelha predefinida.

Possibilidades:
Esta abordagem abriu também novas portas relativamente à utilização das camadas, pois estas poderiam ajudar à criação de tipos com vários pesos.

D PALETAS DE CORES

Ainda sem o aspecto visual dos tipos de letra definidos decidimos implementar uma versão que fosse uma variação das duas últimas, tirando partido da cor. Para isso foram criadas duas funções que, dado um conjunto de imagens, retiravam a cor média ou as cores dominantes. Depois, estas duas funções foram aplicadas nas abordagens anteriores, a Fig 73 apresenta alguns resultados da aplicação na última.

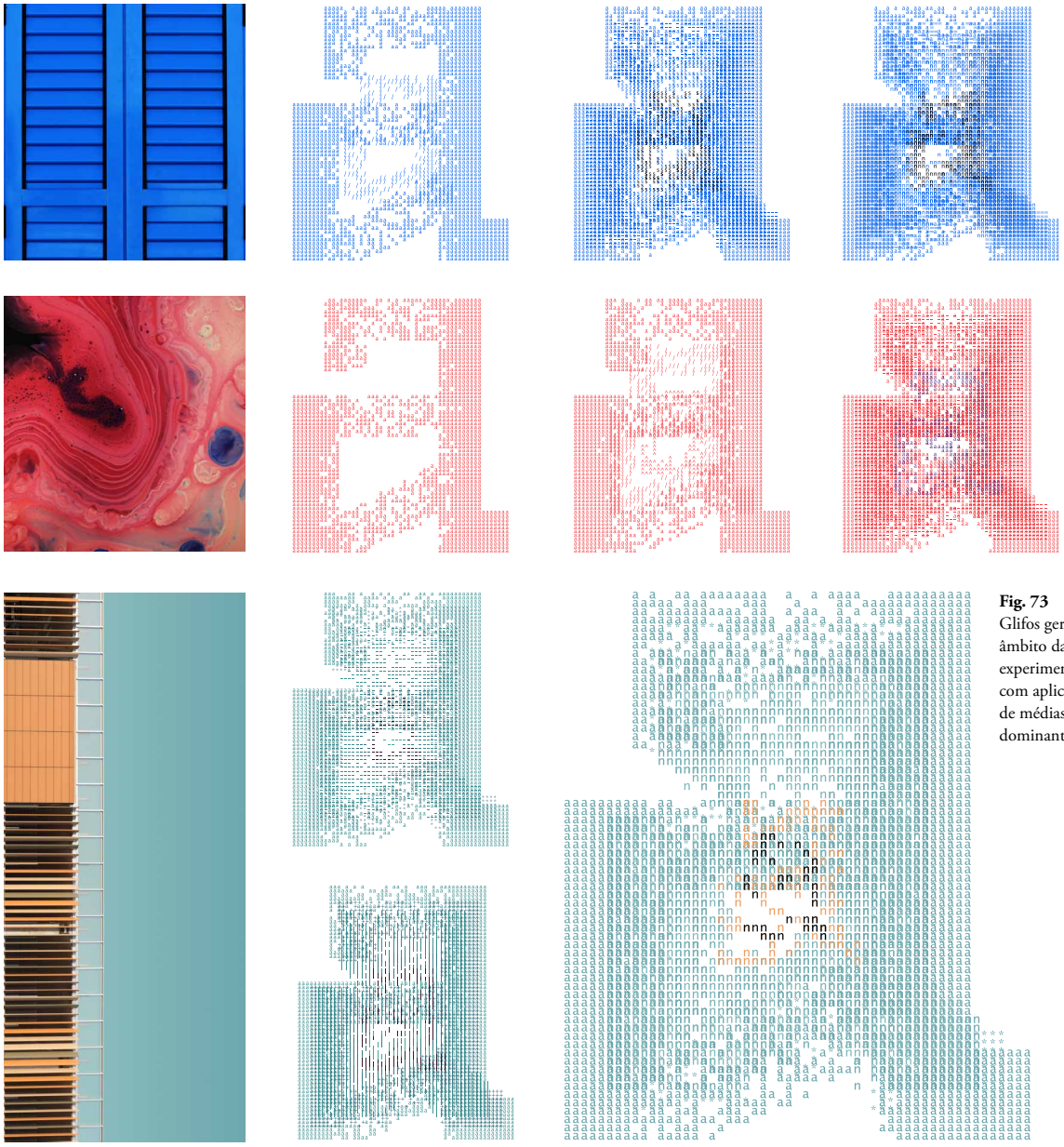


Fig. 73
Glifos gerados no âmbito da quarta experimentação com aplicação de médias e cores dominantes

Problemas desta abordagem:
A imagem que serve como *input* determina que função é que melhor a representa, a média ou as cores dominantes. É então impossível definir uma função que sirva para todas as imagens sem ajustes.

Possíveis soluções:
Depois de observar os resultados gerados constatamos que uma possível solução para o facto das funções por vezes não retirarem as cores mais representativas seria criar uma função que juntasse as duas existentes.

Possibilidades:
Esta abordagem deu novas possibilidades para escolha da cor dos glifos finais. A cor poderia ser também ela retirada de um *input* externo.

E LETRA MODULAR COM CAMADAS SOBREPOSTAS

Todas as experimentações apresentadas até ao momento para este projecto apresentaram aspectos muito variados. Esta abordagem surgiu para explorar o uso de camadas e não tem nenhuma componente prática. O tipo de letra Intersect (Fig. 74 e 75), desenvolvido por MuirMcNeil, serviu como inspiração para esta abordagem devido à forma como as camadas são utilizadas.

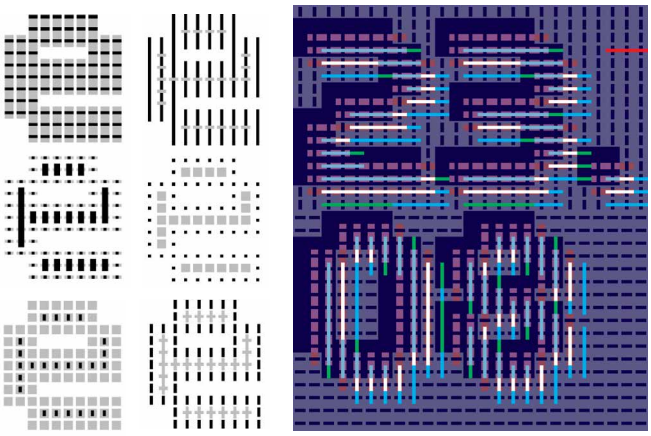


Fig. 74
À esquerda: Tipo de letra *Intersect*, MuirMcNeil

Fig. 75
À direita: Cartaz com o tipo de letra *Intersect* aplicado, MuirMcNeil

Nesta abordagem foram também tidos em conta aspectos relativos à divisão das letras por camadas anatómicas e pela combinação de partes de tipos de letras variados. Além do sistema desenvolvido por MuirMcNeil, o trabalho de David Carson foi também uma fonte de inspiração.

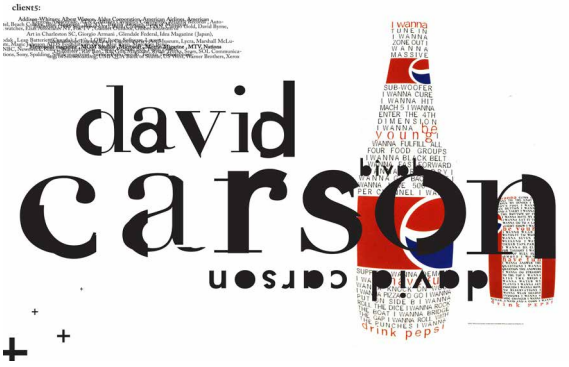


Fig. 76
Publicidade *Pepsi*, David Carson, 1996

Nesta abordagem começamos por desenhar dois glifos do 'a' de forma modular. Posteriormente, procedeu-se à criação de novos elementos através da combinação dos módulos dos glifos criados com mudanças de escala (Fig.77).

Fig. 77
Glifos iniciais e segundos glifos criados

De seguida, os glifos criados foram simplificados e foram retiradas as diagonais para facilitar a utilização de camadas. Foram estabelecidas quatro camadas diferentes (Fig. 78).

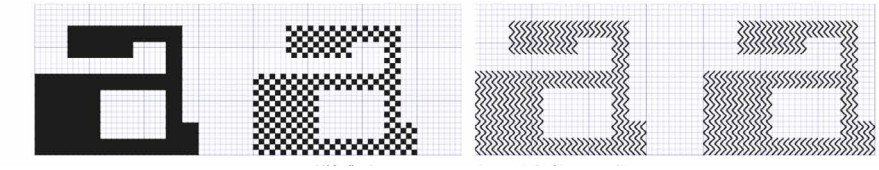


Fig. 78
Glifo final e as quatro camadas estabelecidas na grelha

Foram também feitas algumas combinações (Fig. 79).

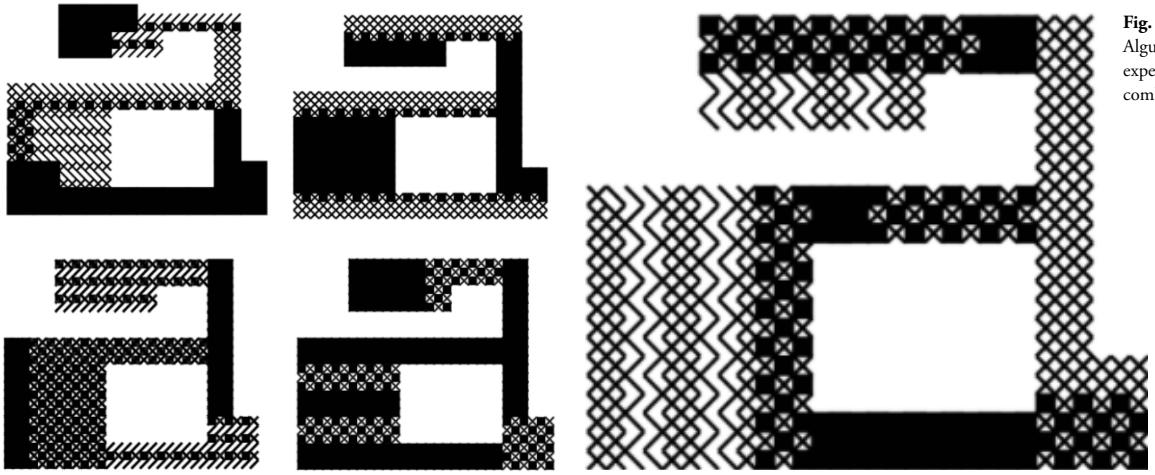


Fig. 79
Algumas experiências combinatórias

Depois da criação de um tipo é de grande importância o teste da sua aplicação (Fig. 80). Portanto, após o desenvolvimento de algumas combinações foram desenhados mais alguns caracteres para perceber até que ponto o tipo de letra funcionava. É de lembrar que os caracteres produzidos foram criados a partir de módulos do 'a' já estabelecido.

Fig. 80
Aplicação do estilo criado numa palavra

Explorando o conceito da modificação das camadas através de *inputs* externos, alteraram-se as posições de algumas camadas (Fig. 81). O objetivo era visualizar de que forma os *inputs* poderiam alterar as camadas da letra.

Fig. 81
Aplicação da alteração da posição de algumas camadas numa palavra

Problemas desta abordagem:
Um dos problemas deste sistema foi o facto de não funcionar em tamanhos de letra reduzidos, pois tem demasiado pormenor. A utilização do mesmo esqueleto para todos os caracteres também não era a opção mais favorável, como vimos anteriormente. No entanto, o sistema foi criado para se poder adaptar a mudanças na sua estrutura, sendo portanto possível a mudança.

Possíveis soluções:
O problema relativo ao pormenor ficaria resolvido se fosse assumido que este tipo de letra só seria utilizado em grandes tamanhos. Uma solução podereria ser a simplificação das camadas criadas ou estabelecimento de pormenores de acordo com o tamanho do tipo.

Possibilidades:
Esta abordagem criou uma série de opções relativas à criação de camadas, como o deslocamento. A existência de pormenores deu ideias para a criação de um tipo de letra que varia o respectivo grau de detalhe em função do tamanho.

F IDENTIFICAR E RETIRAR PARTES ANATÓMICAS

Esta sexta experiência finaliza o conjunto de abordagens iniciais. Surgiu a partir de uma ideia criada no contexto deste projeto — a procura de um esqueleto. Os principais objetivos desta iteração eram: (i) a identificação de partes anatómicas e (ii) a sua posterior utilização na construção de novos glifos. Foi criada também para permitir uma maior elasticidade na estrutura dos glifos gerados, tentando responder aos problemas das abordagens anteriores.

Este sistema é também ele desenvolvido em *Processing* e, como referido acima, foi baseado no primeiro. Estão interligados no método de comparação entre caracteres — a sobreposição. No entanto, neste não são percorridos os pixels de cada glifo comparado. Para este sistema decidimos usar a biblioteca *Geomerative* (Marxer, n.d.) e uma série de métodos disponíveis que facilitam os processos de interseção entre glifos.

Além de facilitar a geometria generativa, a *Geomerative*, desenvolvida por Ricard Marxer, permite o acesso a caminhos e pontos de formas facilmente e por isso é útil no desenvolvimento de tipografia generativa (Marxer, n.d.). No nosso sistema em específico permitiu facilmente o acesso aos pontos de cada glifo e interseção entre formas .

NOTA: A partir daqui utilizaremos a expressão "shapes" para definir os glifos que são combinados. Neste sistema começamos por combinar glifos, mas à medida que

vamos subtraindo, adicionando e intersetando as partes resultantes deixam de ser consideradas glifos.

Para este sistema criamos três formas de combinação entre glifos: (i) diferença, (ii) reunião e (iii) interseção. Para cada delas foi desenvolvida uma função para a sua comparação. Portanto na primeira função e dadas duas formas (*shape Final — shape* que vai sofrendo alterações — e *shape Atual — shape* que está a ser combinada) é mantida a parte comum entre elas. A função de reunião devolve o resultado da adição das duas. A última função devolve as partes comuns.

Fig. 82
Exemplo da subtração ideal (função Diferença) de um ombro a partir de um 'h' e dois 'l'.

Para testar o sistema foram utilizados oito tipos de letra: *Adobe Caslon Pro, Adobe Garamond Pro, Baskerville, Bodoni, Didot, Futura, Helvetica e Times New Roman*. Para desenhar mais facilmente as *shapes* finais foi criada também uma função que recebe que letras serão combinados, a sua posição relativa, o tipo de junção e o tipo de letra escolhido.

Primeira versão do sistema

A. Tentativa de subtração de 'ombro':
Depois de desenvolvidas as funções necessárias aplicámo-las na subtração de um 'ombro'. Para isso utilizamos um 'h' e dois 'l's e subtraímos os dois 'l's ao 'h' (função de diferença), sobrepondo-os à esquerda e à direita da primeira letra. Foram aplicados glifos de todos os tipos de letra referidos anteriormente de forma a validar o sistema. Na Fig. 83 estão apresentados os resultados obtidos.

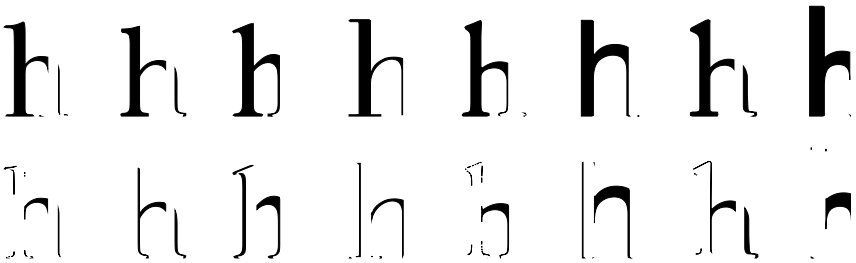


Fig. 83
De cima para baixo respectivamente: Subtração do 'l' e Resultado final — Subtração do 'ombro' (e segundo 'l')

No final, reparamos que os resultados obtidos não eram muito favoráveis, pois a *shape* gerada continha não só o 'ombro', mas também ruído resultante do facto dos glifos subtraídos ('l') não serem exatamente iguais ao ascendente do 'h'. Observamos também que apenas com caracteres da Futura se conseguiam produtos favoráveis. Provavelmente devia-se ao facto de, entre os tipos de letra utilizados, ser o mais geométrico. No entanto nós pretendíamos que o sistema funcionasse para uma grande parte dos tipos de letra existentes, portanto os resultados chegados não correspondiam, de todo, ao nosso objetivo.

Melhoria no sistema – exclusão de pontos em excesso

De modo a minimizar o erro decidimos implementar uma função que retirasse os *pixels* a mais. A função desenvolvida corria a vizinhança horizontal do pixel em questão dentro de uma distância estipulada. Todos os *pixels* que tivessem *pixels* vizinhos que não pertencessem à *shape* gerada eram excluídos. Na Fig. 84 estão representados exemplos da verificação de cada *pixel* feita pela função.

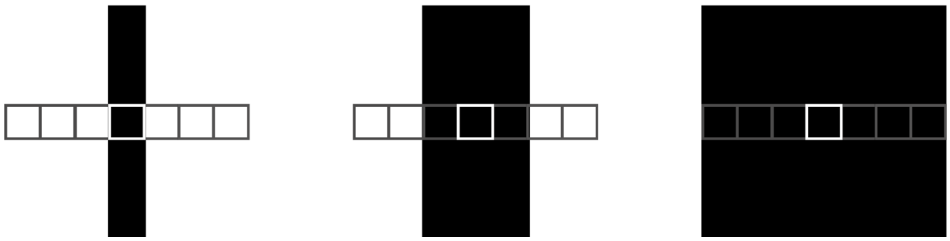
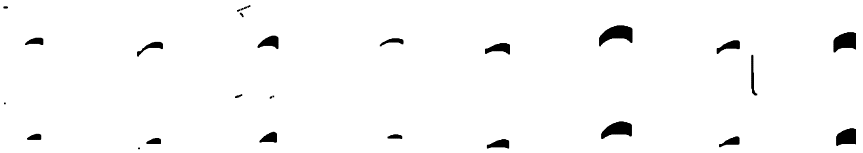


Fig. 84
Demonstração da função de exclusão de pontos em excesso. Neste caso, a distância da vizinhança é igual a 3, pois existem 3 *pixels* à direita e à esquerda daquele que está a ser testado. Apenas são válidos os pixels em que toda a vizinhança pertence à *shape,* portanto só o *pixel* testado no último exemplo é que é válido.

A. Tentativa de subtração de 'ombro':
Depois de criada a nova função testamos novamente a subtração do 'ombro'. Foram seguidos os passos anteriores aplicando apenas no final de cada operação a função de exclusão. Na Fig. 85 estão apresentados alguns dos resultados obtidos com maior e menor distância de vizinhança.

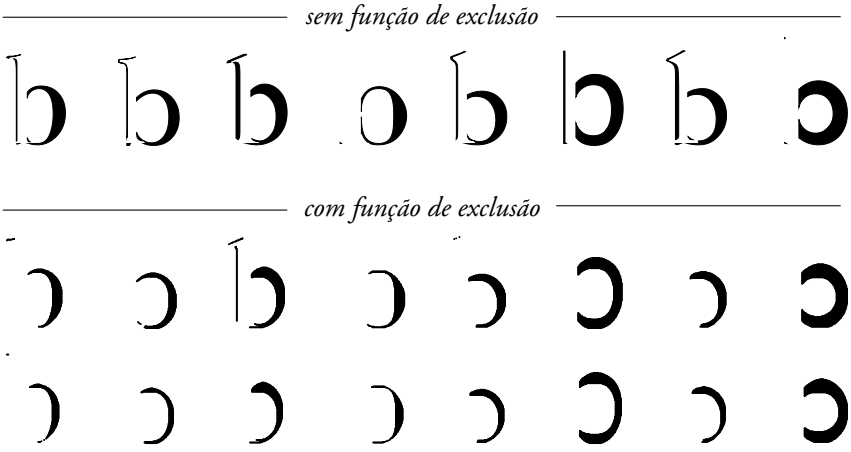
Fig. 85
Subtração do 'ombro', com uma distância da vizinhança de 3 e 5, em cima e em baixo respectivamente.



Após testes com diferentes distâncias concluímos que obtemos melhores resultados com a utilização da função de exclusão, pois os *pixels* em excesso são descartados. Além disso, observamos também que quanto maior a distância à vizinhança maior o erro (retiramos parte relevante da forma), no entanto é menor o ruído à volta.

B. Tentativa de subtração de 'barriga'
Ainda no teste da nova função decidimos gerar uma 'barriga'. Para isso utilizamos um 'l' e um 'b' subtraímos a primeira à última (função de diferença). Também para esta iteração foram aplicados glifos de todos os tipos de letra anteriores de forma a validar o sistema. Na Fig. 86 estão apresentados os resultados obtidos sem a função de exclusão e com ela.

Fig. 86
Subtração de 'barriga', sem função de exclusão e com a função e uma distância da vizinhança de 3 e 5, em cima e em baixo respectivamente.

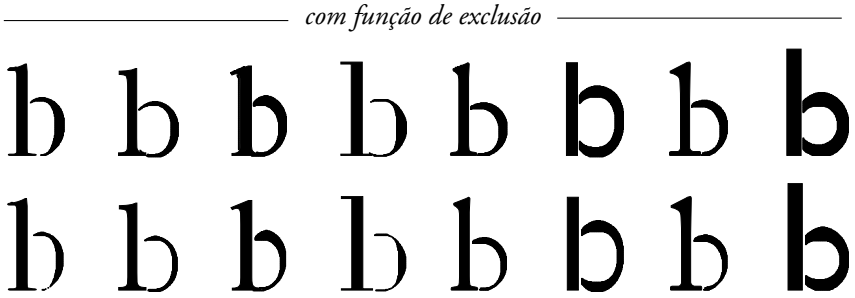


C. Tentativa do desenho de 'b'

O nosso grande objetivo neste projeto era, além da subtração de partes anatómicas, a criação de glifos viáveis. Depois de desenvolvidos os métodos necessários para a extração restava testar a criação de glifos a partir das partes retiradas. Uma vez que já tínhamos extraído 'barrigas' decidimos combinar com 'l's para desenhar 'b's. Observando as Fig. 87 e 88 podemos comparar os glifos originais dos 'b's de cada tipo de letra e aqueles gerados pelo nosso sistema.

Fig. 87
'b's originais dos tipos de letra: *Adobe Caslon Pro, Adobe Garamond Pro, Baskerville, Bodoni, Didot, Futura, Helvetica* e *Times New Roman*

Fig. 88
Geração de 'b', sem função de exclusão e com a função e uma distância da vizinhança de 3 e 5, em cima e em baixo respectivamente.



Depois de várias gerações comparamos os glifos originais e criados . Em geral, as diferenças entre eles devem-se ao facto do ascendente do 'b' não ser exatamente igual ao 'l', exceto nos tipos de letra não serifados utilizados (*Futura* e *Helvetica*). Relativamente à função de exclusão, para este caso temos melhores resultados sem ela. Isto pode dever-se ao facto de estarmos a sobrepor formas, portanto o ruído torna-se irrelevante. Observando também por exemplo o 'b' gerado com a função de exclusão para o sexto tipo de letra, *Helvetica*, verificamos que a 'barriga' é totalmente diferente da original. O 'b' gerado fica totalmente diferente com a utilização da função.

D. Tentativa do desenho do 'r'

Decidimos também testar o desenho do 'r'. Para isso utilizamos um 'ombro' e uma haste de um 'n'. Como descrito anteriormente, para a extração de um 'ombro' são utilizados um 'h' e dois 'l's que são subtraídos à primeira letra. A haste do 'n' é obtida através da interseção de um 'n' e um 'i'.

Para desenhar o 'r' final juntamos as duas formas resultantes. Na Fig. 89 podemos observar os 'r's originais nos tipos de letra utilizados e na Fig. 90 os resultados gerados.

Fig. 89
'r's originais dos tipos de letra: *Adobe Caslon Pro, Adobe Garamond Pro, Baskerville, Bodoni, Didot, Futura, Helvetica* e *Times New Roman*

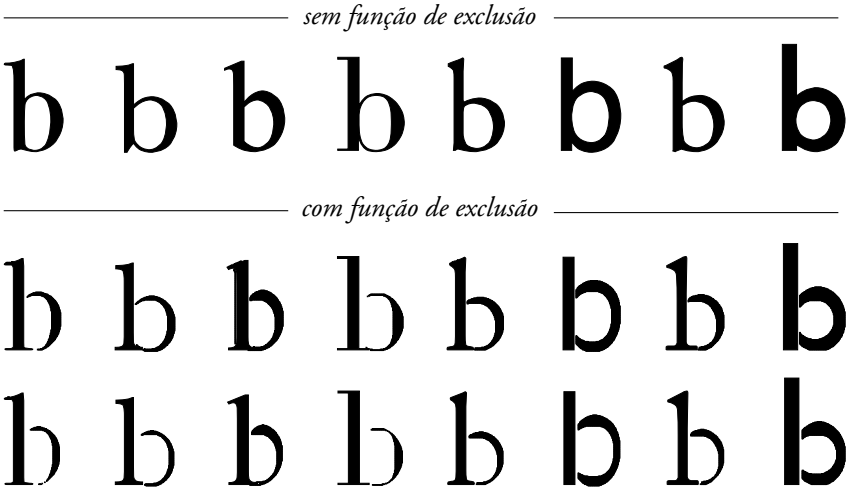


Fig. 90
Geração de 'b', sem função de exclusão e com a função e uma distância da vizinhança de 3 e 5, em cima e em baixo respectivamente.

Depois da geração de vários 'r's comparamos com os originais. As diferenças observadas devem-se ao facto de em alguns tipos de letra o 'ombro' do 'n' não ser igual ao do 'r' e de, por vezes, o 'r' apresentar um terminal em bola. Além disso, observamos que em geral os resultados são mais promissores utilizando a função de exclusão, apesar da percentagem de tinta relevante da forma resultante ser inferior à original.

Por outro lado, também reparámos que quanto maior a distância da vizinhança maior o ruído retirado, no entanto também maior é a percentagem de conteúdo importante excluída. Do mesmo modo quanto maior é a distância mais fácil é a criação de tipos de letra para stencil.

E. Tentativa do extração de 'travessão'

Mais tarde, fizemos também outras experiência ainda na tentativa de testar e melhorar o sistema na criação de glifos viáveis para todos os tipos de letra. Decidimos extrair um 'travessão' utilizando um 'H' e dois 'l's subtraindo estes dois últimos ao primeiro glifo. Na Fig. 91 estão apresentados os resultados obtidos com a utilização da função de exclusão.

Fig. 91
Extração de 'travessão', com a função e uma distância da vizinhança de 5.

F. Tentativa de desenho de 'f'

Um dos últimos casos testados foi o desenho do 'f'. Para isso foi utilizada a 'cauda' do 'j' e a haste do 'n', extraída anteriormente. Inicialmente também era para ser utilizado o 'travessão' gerado na iteração anterior, no entanto devido à serifa do 'i' e do 'n', utilizados na extração da haste do 'n', não foi necessário. O 'gancho' do 'f' foi gerada através da 'cauda' que foi extraída a partir da subtração do 'i' no 'j'. A haste do 'n', como referido anteriormente, foi extraído através da interseção do 'n' com o 'i'. Depois juntamos as duas *shapes* e rodamos 180º, a Fig. 92 explica essas duas operações. No final, concluímos que os resultados (Fig. 93) não eram muito promissores, pois nem todos os 'f's gerados eram viáveis.

Fig. 92
Adição da 'cauda' com o traço da minúscula e posterior rotação da *shape* final

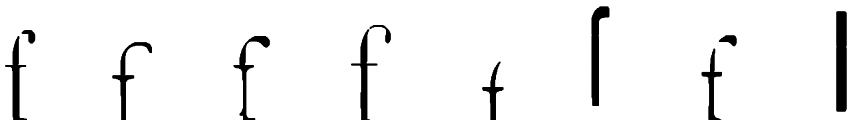


Fig. 93
Geração de 'f', utilização da função de exclusão e com uma distância da vizinhança de 5.

Problemas desta abordagem:
Depois de vários testes ao sistema concluímos que este não era viável. No geral, as partes anatómicas extraídas retiravam partes relevantes dos glifos. Pois a função de extração retirava pixels em excesso, mas também excluía pixels importantes. Existiam também alguns problemas relacionados com o facto de não ser possível extrair partes anatómicas como o 'filete' do 'A', ou a haste do 'v'.

Possíveis soluções:
Depois desta abordagem, a hipótese de extração de um esqueleto tornou-se mais viável. Poderíamos na mesma dividir por partes anatómicas, mas só depois de criar um esqueleto.

G DISCUSSÃO

O final deste subcapítulo marca uma tomada de decisão no desenvolvimento deste projeto. Graças a estas experimentações iniciais optamos por criar tipos de letras a partir de um esqueleto. Mais especificamente pelo desenvolvimento de um sistema que, primeiramente extrai esqueletos e mais tarde, a partir dessa extração, retira partes anatómicas e preenche-as.

Em retrospectiva, foram as primeiras iterações que ajudaram a decidir o próximo passo a tomar no desenvolvimento deste projeto. Aspectos que no início estavam totalmente em aberto, como a criação de tipos a partir de uma grelha ou através de um esqueleto extraído, ficaram decidimos nestas primeiras experiências. Foi através dos resultados chegados que decidimos pôr fim à tentativa de tentar extrair logo à partida as partes anatómicas de um dado tipo de letra.

A partir da primeira abordagem — a procura de um esqueleto — foi desenvolvida uma ideia para a criação de tipos que mais tarde se foi tornando uma solução mais viável com o desenvolvimento das primeiras experiências. Estas abriram também hipóteses relativas à forma como os tipos de letra poderiam ser preenchidos.

IV.III Design de Logotipos baseados em dados

A A IDEIA

Um dos objetivos iniciais desta dissertação era a criação de tipos de letra que além de servir aos propósitos habituais, acrescentassem conhecimento. Tendo esse factor em conta, surgiu-nos a ideia da criação de um sistema de geração de logotipos. Desse modo, poderíamos perceber de que forma esta dissertação poderia ser inserida no mundo real e como é que os *inputs* poderiam modificar os tipos de letra que pretendíamos desenvolver.

O sistema criado foi aplicado à nossa instituição, a Universidade de Coimbra, que é composta por diversas faculdades, cada uma representativa de uma área de estudo. Cada faculdade tem várias licenciaturas e mestrados e, consequentemente, um número substancial de alunos. Tendo em mente a diversidade das diferentes faculdades, decidimos criar logotipos que as representassem e distinguissem. Ao mesmo tempo, pretendíamos que os logotipos incorporassem e unificassem as diferentes faculdades de forma coerente, devendo adaptar-se de acordo com o atual espectro de alunos em cada faculdade. Desta forma, o principal objetivo deste projeto era desenvolver logotipos dinâmicos capazes de representar as diferentes faculdades.

Dados

Quando pensámos num determinado curso criamos logo à partida estereótipos para o representar, por exemplo, numa licenciatura de arte pensamos logo que o número de mulheres é maior do que o dos homens. No entanto, às vezes, os estereótipos que criamos não correspondem à realidade, por isso pensamos que poderia ser interessante comparar esses dois. Decidimos compilar informações sobre o número de alunos em cada gênero e nacionalidade. Entre os dados disponíveis, esperávamos que essas duas variáveis fossem representativas dos alunos de cada faculdade.

Os dados que utilizamos referiam-se ao ano de 2015 e foram recolhidos principalmente a partir da página web da Universidade de Coimbra. Decidimos usar apenas faculdades com licenciaturas e mestrados. No final, foram consideradas oito faculdades: Artes e Humanidades (FLUC), Direito (FDUC), Ciência e Tecnologia (FCTUC), Farmácia (FFUC), Economia (FEUC), Psicologia e Ciências da Educação (FPCEUC), Ciências do Desporto e Educação Física (FCDEFUC) e Medicina (FMUC). Na tabela da Fig.94 estão apresentadas as informações coletadas. Em relação à nacionalidade, encontramos dados relativos a quatro grupos correspondentes a: estudantes portugueses (PT), estudantes de países com o português como língua oficial (PL), de países da União Europeia (UE) e de outros países (O).

Faculdade	Género	Nacionalidade				Número	
		PT[a]	PL[b]	UE[c]	O[d]		
FLUC	feminino	1290	135	24	21	1470	2447
	masculino	857	90	16	14	977	
FDUC	feminino	1510	319	10	10	1850	2890
	masculino	849	180	6	6	1040	
FCTUC	feminino	2202	166	19	49	2437	6210
	masculino	3410	258	30	76	3773	
FFUC	feminino	956	17	2	5	980	1272
	masculino	285	5	1	1	292	
FEUC	feminino	860	139	12	21	1033	1984
	masculino	792	128	12	20	951	
FPCEUC	feminino	1211	84	15	4	1314	1524
	masculino	193	14	2	1	210	
FCDEFUC	feminino	228	11	0	1	240	676
	masculino	413	21	0	2	436	
FMUC	feminino	1464	33	6	5	1507	2259
	masculino	730	16	3	2	752	

Fig. 94
Dados recolhidos dos alunos de cada faculdade da Universidade de Coimbra em 2015.

Legenda da nacionalidade:
a. Portugal, b. Países com português como língua oficial, c. Outros Países da União Europeia, d. Outros Países

Formas das Letras

Na Universidade de Coimbra existe uma grande variedade de estudantes. Nessas circunstâncias, torna-se natural escolher vários tipos de letra para representar essa diversidade. Para conseguir isso, escolhemos quatro categorias de classificação tipográfica — *garaldes, reales, didones e lineares* —, abordadas no capítulo "A Forma da Letra", e para cada uma delas adotamos alguns tipos de letra. Depois, decidimos que o ideal seria combinar todos os tipos de letra, criando assim uma fonte que pudesse representar todos os alunos. A Fig. 95 descreve o que pretendiamos.

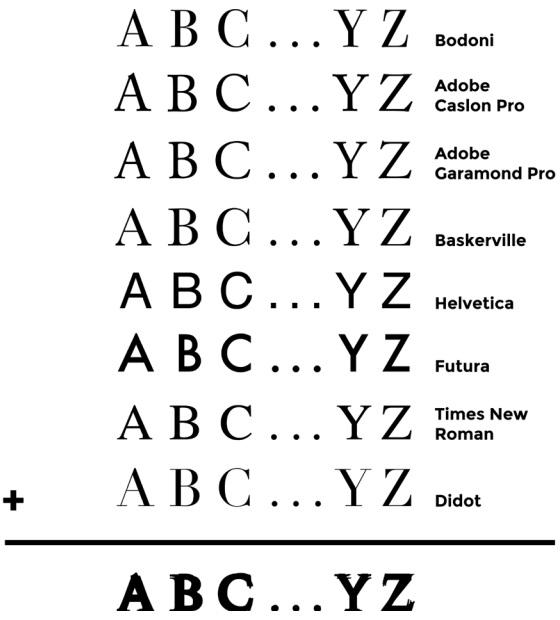


Fig. 95
Combinação/adição dos diferentes dos tipos de letra.

B O SISTEMA

Para a implementação desenvolvemos um sistema em *Processing*. Para cada faculdade foram criados dois objetos do tipo género (feminino e masculino) e para cada género foram criados quatro objetos do tipo nacionalidade (PT, PL, EU e O).

O próximo passo foi combinar as fontes escolhidas, semelhante ao que tínhamos feito na primeira abordagem do capítulo anterior "Experimentações Iniciais". Para cada letra corremos todos os tipos de letra escolhidos e adicionamos as formas. De modo a facilitar o processo utilizamos a biblioteca *Geomerative* novamente (Marxer,n.d.).

O logotipo que criamos para cada faculdade foi desenvolvido a partir de sua sigla. Portanto depois de criadas as *shapes* para todas letras associamos as letras de cada sigla à faculdade correspondente.

Iteração I

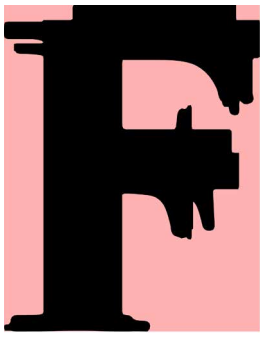
Depois das *shapes* produzidas, precisávamos de uma maneira de as preencher. Como mencionado anteriormente, um dos principais objetivos desta dissertação era o uso da tipografia como uma camada de conhecimento. Portanto, o uso literal de camadas tornou-se óbvio.

O próximo passo foi o desenvolvimento de uma grelha sobre as formas geradas anteriormente. Precisávamos representar três variáveis: (i) a nacionalidade, (ii) o gênero e (iii) o número de alunos.

Evidentemente, relacionamos o número de alunos com a densidade dos elementos. Estabelecemos um sinal de pontuação para cada sexo e uma cor para cada nacionalidade, desse modo poderíamos combinar as duas variáveis. Para distinguir os elementos de cada gênero, decidimos usar camadas. Aplicamos os elementos em duas camadas com a *shape* das letras criadas anteriormente, cada uma para cada gênero.

Para determinar a densidade da grelha para a sigla de cada faculdade calculamos o número máximo de alunos por género e com esse valor mapeamos os valores coletados. Assim sendo, a densidade dos elementos no logotipo está associada ao número de alunos em cada faculdade em relação às restantes faculdades. Os valores de densidade máxima e mínima foram estabelecidos à priori.

Depois de definida a densidade da grelha para cada faculdade foram desenhados os glifos. Para cada letra da sigla era percorrido o "corpo" da *shape* (Fig. 96) de x em x *pixels* de acordo com a densidade. Portanto era percorrida a área ocupada pela *shape*, desde o canto superior esquerdo até ao canto inferior direito, e verificados se os *pixels* pertenciam ou não à forma (Fig. 97).



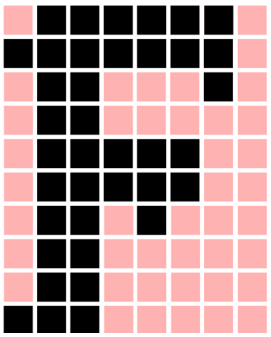


Fig. 96
Esquerda: 'Corpo' da *shape* gerada

Fig. 97
Figura do centro e direita: Aplicação de diferentes densidades à *shape* anteriormente gerada. Da maior para a menor densidade. A vermelho está a área percorrida, se o *pixel* pertence à *shape* é acrescentado o número de módulos totais.

O número de *pixels* pertencentes à forma foi associado ao número máximo de estudantes por género em determinada faculdade. Assim sendo, para o género de maior número foi associado o número de módulos totais e para o género com menor número foi mapeado o número de módulos correspondentes de acordo com o anterior. A estrutura de cada glifo, como referido anteriormente, era composta por duas camadas, correspondentes aos dois géneros, com a mesma densidade de módulos. No entanto apenas a camada representativa do género com maior número de alunos preenchia todos os módulos. A Fig. 98 demonstra a junção das duas camadas e o resultado da adição. Após vários testes optamos por utilizar "|" como o sinal de pontuação representativo do gênero masculino e o "-" do gênero feminino.

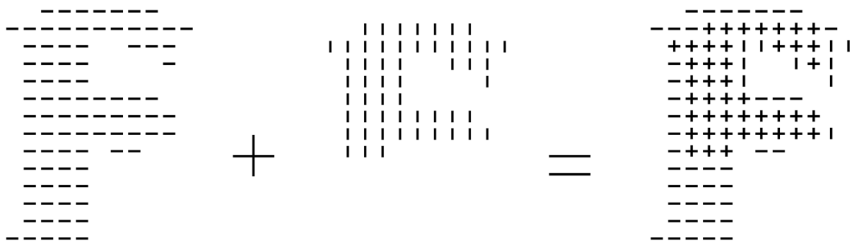


Fig. 98
Junção das duas camadas representativas de cada género. Apesar das duas camadas adicionadas terem a mesma densidade, o nº de módulos está associado à quantidade de alunos em cada género.

Depois de associado o número de módulos correspondentes a cada género foi mapeado a quantidade de módulos correspondentes a cada uma das

nacionalidades dentro do número de módulo disponíveis. Como referido anteriormente, para cada nacionalidade foi associada uma cor. Depois de alguns testes decidimos utilizar o vermelho para representar os estudantes portugueses, o azul para os estudantes de países com o português como língua oficial, o verde para os alunos de países da União Europeia e o amarelo para os estudantes de outros países. As camadas foram sobrepostas, mas não alinhadas e de modo a facilitar a visualização de todos os elementos, aplicamos o efeito *multiply*. Na Fig. 99 é possível observar os logotipos finais.

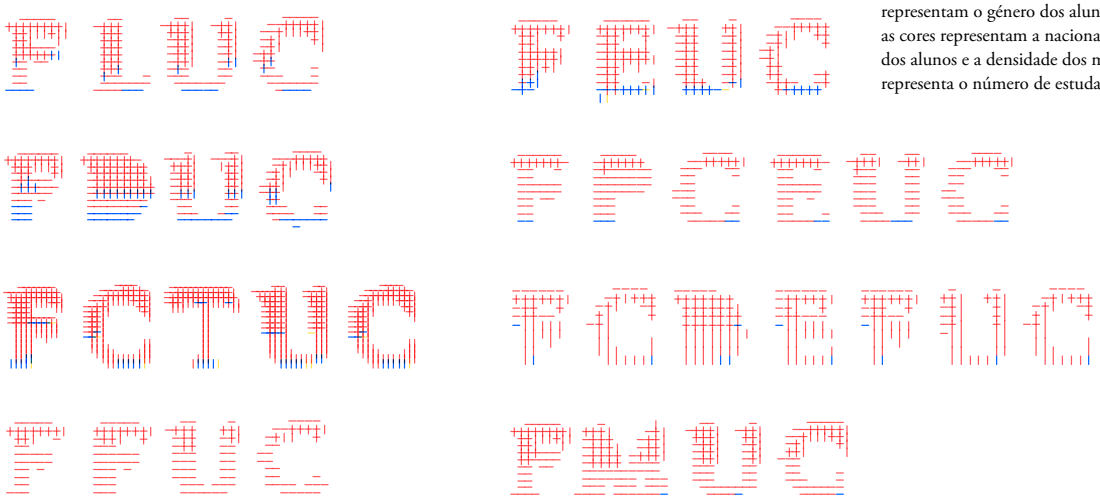


Fig. 99
Logotipos gerados na Iteração 1. Os módulos (sinais de pontuação) representam o género dos alunos; as cores representam a nacionalidade dos alunos e a densidade dos módulos representa o número de estudantes.

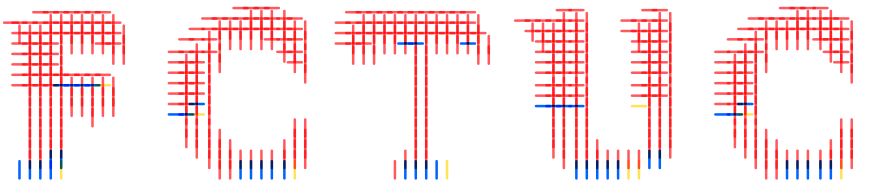


Fig. 100
Logotipo da FCTUC gerado na Iteração 1.

Após vários testes, verificamos que esta abordagem tinha alguns problemas. O efeito *multiply* só era visível através da sobreposição de diferentes cores. Portanto, uma vez que o número de estudantes portugueses era sempre muito superior ao das outras nacionalidades, o efeito não era percetível. Apesar da combinação entre as duas camadas criar uma trama interessante, nesta iteração não estamos a aproveitar ao máximo a sua sobreposição. Por outro lado, os problemas de representação encontrados podem também estar relacionados com os sinais de pontuação escolhidos.

Iteração II

Mais tarde, de modo a ultrapassar os problemas da iteração anterior, decidimos modificar a forma como os logotipos eram gerados e os símbolos e cores utilizados. Optamos por utilizar a cor para a representação de cada gênero e os sinais de pontuação para as nacionalidades. No entanto, decidimos manter as camadas aplicadas a cada género. O principal objetivo desta alteração foi permitir o cruzamento de cores distintas em diferentes camadas. Desta forma, poderíamos identificar as diferentes nacionalidades e também criar uma terceira cor a partir da sua intersecção.

O processo de geração foi semelhante ao da iteração anterior. Calculamos o número máximo de alunos por género, mas também determinamos o mínimo.

Essa cálculo permitiu um maior contraste entre densidades, pois limitou o intervalo de valores para densidade. Depois dessa alteração, a faculdade com menor número de alunos passou a ter o valor mínimo estipulado da densidade. Além disso, optamos por fazer outra mudança de modo a tornar os logotipos mais homogêneos. Para isso, inserimos os caracteres que preenchem cada letra aleatoriamente. Os elementos são incluídos aleatoriamente, mas no espaço reservado para a camada em que são aplicados. Tecnicamente a alteração apenas altera a forma como são posicionados os elementos. Na primeira iteração percorríamos, por ordem, todas as posições disponíveis da grelha, pertencentes à *shape* a ser desenhada, e inseríamos cada elemento o número de vezes correspondentes. O processo de colocação dos elementos utilizado nesta iteração é um pouco mais complexo. Todas as nacionalidades são inseridas num array, repetindo cada uma delas com a quantidade de vezes correspondente ao número de módulos associados. Depois, percorremos todas as posições disponíveis da grelha e vamos sorteando elementos do *array* a serem desenhados. Deste modo, a densidade e a área ocupada por cada camada mantém-se, no entanto os elementos são desenhados aleatoriamente.

Após alguns testes chegamos aos logotipos representados nas Fig. 101 e 102. A cor vermelha e azul representam a quantidade de mulheres e homens respectivamente. Em relação aos sinais de pontuação, "|" representava os alunos portugueses, "<" representava os alunos de países com o português como língua oficial, "-" representava os alunos da União Europeia e "=" representava estudantes de outros países.

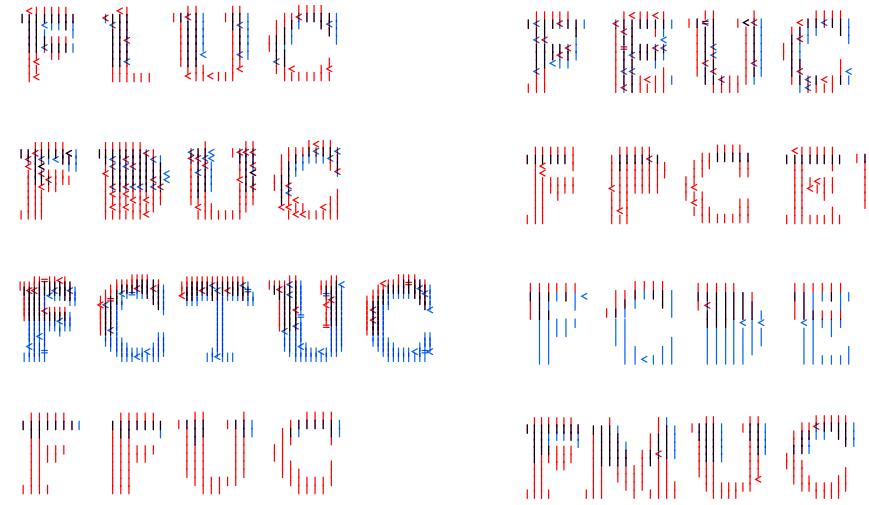


Fig. 101
Logotipos gerados na Iteração II. Os módulos (sinais de pontuação) representam a nacionalidade dos alunos, as cores representam o género dos alunos e a densidade dos módulos representa o número de estudantes.

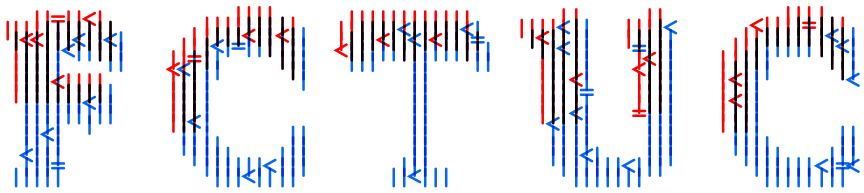


Fig. 102
Logotipo da FCTUC gerado na Iteração II.

A escolha dos elementos foi um aspecto que tivemos de ter em conta, porque o número de alunos portugueses era muito superior ao número de alunos do estrangeiro. O sinal de pontuação representativo dos estudantes portugueses teria de ser mais discreto, devido ao facto de já ganhar ênfase graças à sua quantidade. Na primeira iteração já tínhamos levado esse aspecto em consideração, mas para cor escolhida.

Graças à mudança dos símbolos, as nacionalidades (representadas pelos

sinais de pontuação) tornaram-se mais fáceis de distinguir. Nesta iteração, os sinais de pontuação são mais diversos, no entanto a sobreposição de camadas torna-se menos visível, a terceira cor da interseção das duas primeiras raramente aparece.

Um aumento da área de tinta de cada elemento poderia facilitar a visualização da combinação das camadas. Em retrospectiva, esta iteração apresentou uma melhoria na sobreposição de camadas, no entanto, tornou impossível a distinção as duas camadas, não sendo possível observar onde cada uma delas começa e termina.

Iteração III

A terceira e última iteração marca a finalização do projeto desenvolvido no contexto do artigo. Nesta iteração, em vez de usar sinais de pontuação, optamos por usar módulos previamente desenhados. Dessa forma foram reduzidos os detalhes e tornou-se mais fácil de visualizar a interseção entre as camadas. Os simbologia usada na iteração anterior foi mantida (cada nacionalidade com um símbolo associado e cada gênero é associado a uma camada diferente com a respectiva cor), mas voltamos para a ordenação dos elementos usados na primeira iteração. Isto deveu-se ao facto de algumas nacionalidades terem menos módulos associados e, se estes estiverem separados uns dos outros, podemos não dar conta da sua existência. Assim, nesta iteração, os elementos, em cada camada, foram desenhados da esquerda para a direita e de cima para baixo. Na Fig. 103, 104 e 105 é apresentada a simbologia final usada.

Fig. 103
Representação do género, vermelho para o sexo feminimo e azul para o masculino

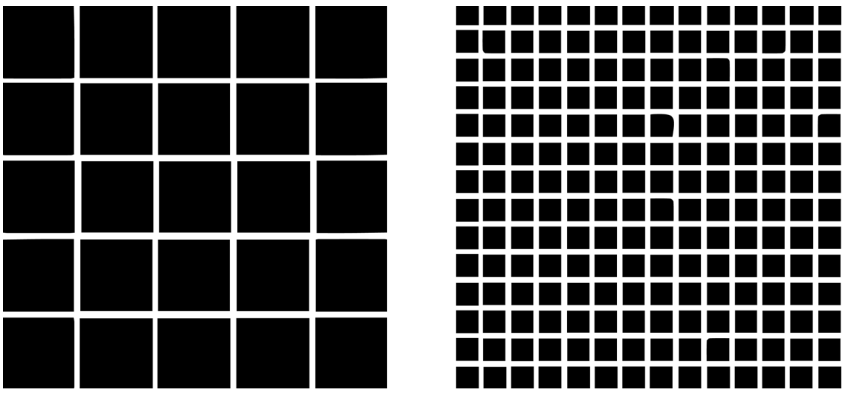


Fig. 104
Representação da densidade, da esquerda para a direita menor e maior densidade respetivamente

Fig. 105
Representação da nacionalidade

Após alguns testes, notamos que devido ao facto das camadas estarem completamente sobrepostas, em algumas faculdades, apenas se destacavam duas cores (uma cor representativa de um gênero e a cor que vinha da interseção das duas camadas). Este problema devia-se ao grande número de estudantes portugueses em comparação com o resto dos alunos. Portanto, a maioria dos elementos das duas camadas era sobreposto (Fig. 105).

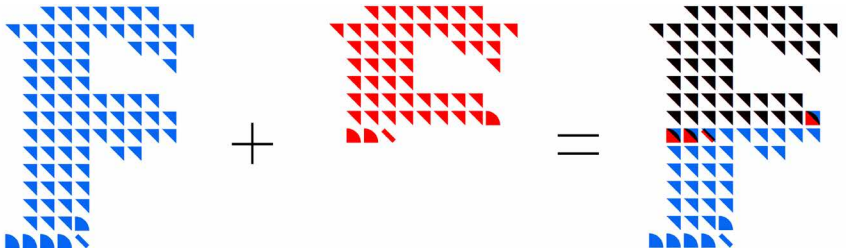


Fig. 105
Cruzamento das duas camadas num 'F'. Os módulos tinham todos a mesma rotação.

De modo a ultrapassar este problema, decidimos aplicar uma rotação aos módulos. A rotação foi aplicada a cada elemento de acordo com a sua camada. Essas modificações permitiram uma diferenciação mais fácil de camadas sobrepostas (destacando os elementos masculinos e femininos). Na Fig. 106 está exemplificada o resultado depois da rotação dos módulos de uma das camadas.

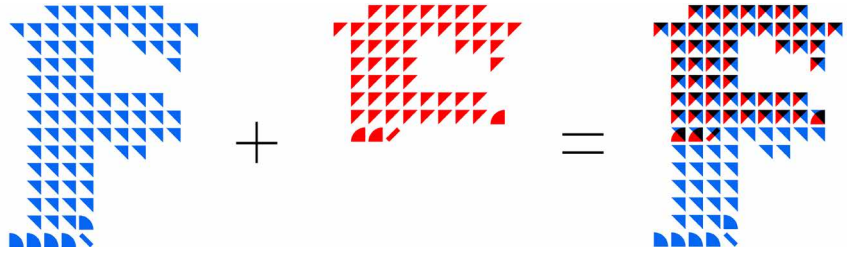


Fig. 106
Cruzamento das duas camadas num 'F' depois da aplicação de diferentes rotações nos módulos em diferentes camadas.

Depois de vários testes, optamos por manter as cores da iteração anterior (vermelho e azul para o género feminino e masculino respectivamente). A grande melhoria desta abordagem foi o uso dos módulos que levou ao aumento da área ocupada por cada elemento. No final chegamos aos resultados apresentados na Fig. 107 e 108, onde o triângulo retângulo representava os estudantes portugueses, o quarto de um círculo representava os estudantes dos países com o português como língua oficial, o outro triângulo representava os estudantes da União Europeia e a linha representava os estudantes de outros países.

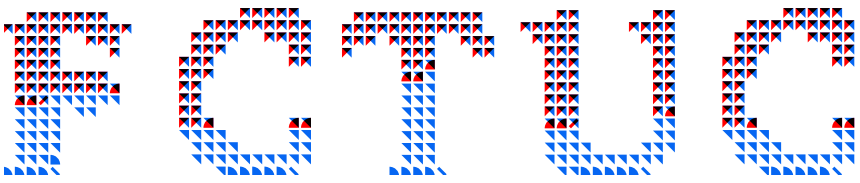


Fig. 107
Logotipo da FCTUC gerado na Iteração III.

Fig. 108
Logotipos gerados na Iteração III. Os módulos representam a nacionalidade dos alunos, as cores representam o género dos alunos e a densidade dos módulos representa o número de estudantes.

Comparando os diferentes logotipos podemos concluir que a faculdade com mais alunos, graças à variação de densidade. Por outro lado, a FCDEFUC e a FPCEU são duas faculdades com menor número. Além disso, podemos também comparar géneros. Por exemplo, podemos observar que a Faculdade de Psicologia e Ciências da Educação (FPCEU) tem mais mulheres do que homens. Como era previsível, a Faculdade de Ciências do Desporto e a Educação Física (FCDEFUC) é aquela com mais estudantes do sexo masculino.

No desenvolvimento deste projeto, sentimos também a necessidade de testar a alteração de valores das densidades mínima e máxima. Consequentemente, concluímos que para utilizar os logotipos em tamanho reduzido se deveriam diminuir os detalhes, factor que é conseguido ao diminuir a densidade mínima e máxima. Na Fig. 109 estão apresentadas diferentes níveis de densidade desenvolvidas no nosso sistema.

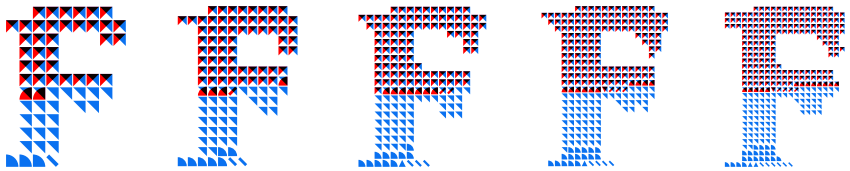


Fig. 109
Variação dos níveis da densidade na iteração III

Além disso, nesta iteração foi também testado a alteração na área que cada módulo ocupava no espaço da grelha onde ele é colocado. Para isso, acrescentamos uma variável para definir a percentagem da área ocupada pelo módulo. Para um melhor entendimento são apresentados alguns testes na Fig. 110.

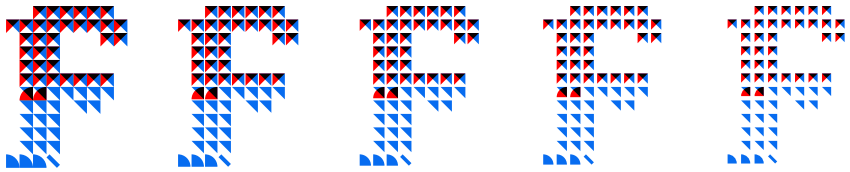


Fig. 110
Variação da percentagem ocupada por cada módulo na iteração III

DISCUSSÃO

Em retrospectiva, consideramos os logotipos gerados interessantes do ponto de vista experimental. Os resultados demonstram como os dados podem influenciar o design dos logotipos e como os logotipos podem transmitir informações. No entanto,

achamos que, para implementar esses logotipos num cenário real, alguns aspectos como a legibilidade e outros detalhes gráficos teriam que ser mais refinados. Por exemplo, identificamos uma limitação dos logotipos gerados com nosso sistema: legibilidade em tamanhos muito pequenos. A partir do momento em que as características de cada faculdade não podem ser visualizadas, o logotipo perde o seu significado.

Este projeto surge no contexto desta dissertação e também nele é explorado o uso de camadas e cores no design de tipos. Além disto, a recente possibilidade tecnológica de projetar fontes compostas por camadas coloridas permite-nos usar um tipo de letra cromático como um logotipo para uma identidade visual. Portanto, este projeto vem também abrir novas possibilidades de exploração na intersecção do design de tipos e identidades visuais dinâmicas baseadas em dados.

Os logotipos propostos fornecem uma camada de conhecimento visualizando dados relacionados aos alunos. O projeto que desenvolvemos permite que os logotipos reajam automaticamente à entrada de dados. Portanto, o uso de dados de entrada em tempo real, juntamente com o processo gerativo que projeta o logotipo, permite que o logotipo esteja vivo e se adapte a diferentes contextos e evolua com o tempo. Como resultado, o logotipo é capaz de mudar ao longo do ano com a entrada e saída de alunos. Além disso, esta abordagem poderia também ser adaptada para permitir a identificação de alunos individuais com base nos seus dados, por exemplo, o número de anos de estudo e a idade. Desta forma, cada aluno pode gerar um tipo de letra único baseado nas suas características individuais.

A criação de logotipos baseados em dados através do uso de camadas vem em grande parte responder aos objetivos anteriormente estabelecidos para esta dissertação.

DISSEMINAÇÃO DO TRABALHO DESENVOLVIDO

Mais tarde, apercebemo-nos da existência de uma conferência de Visualização de Informação que iria acontecer em Julho de 2018, em Itália. A Conferência Internacional de Visualização de Informação contava já com o 22º ano de existência e eram solicitados artigos de diversas áreas. Um dos simpósios da conferência era o *Glyphs – Shapes, Icons, Text and Imagery in Visualization* cujo tema era a codificação e visualização de dados através de glifos e texto e portanto associado ao tema desta dissertação. De modo a disseminar o trabalho desenvolvido decidimos escrever um artigo sobre os logotipos que desenvolvemos para as Faculdades da Universidade de Coimbra, ao qual denominamos *Data Driven Logotype Design.*

Além do artigo foi também escrita uma publicação no Cdv, o laboratório de investigação ao qual pertencemos, pode ser consultado em: *https://cdv.dei.uc.pt/data-logo/.*

Em Maio de 2018 o artigo foi aceite pela organização da 22a Conferência Internacional de Visualização de Informação. Dois meses depois, o mesmo foi apresentado em Salerno, Itália. Em retrospectiva, consideramos que o desenvolvimento e escrita sobre este sistema, além de disseminar o trabalho por nós criado, veio também criar novas possibilidades de exploração no desenvolvimento desta dissertação.

IV.IV Desenvolvimento do sistema

Como referido anteriormente, o objetivo desta dissertação é o desenvolvimento de um sistema capaz de gerar tipos de letra de forma generativa. Até agora todas as experimentações realizadas criaram novos caminhos possíveis para o desenvolvimento deste projeto prático. Aspectos que estavam em aberto inicialmente, como a criação de tipos a partir de uma grelha ou através decum esqueleto extraído, ficaram decididos. Além disso, foram tambémcexplorados de que forma os dados poderão influenciar os tipos de letra gerados neste contexto.

Agora, graças às primeiras experimentações, optamos por produzir um sistema que, antes de mais, extrai esqueletos de tipos de letra.

A EXTRAÇÃO DE ESQUELETOS

O nosso primeiro grande objetivo para este projeto foi a extração de esqueletos. Identificamos como pontos essenciais para este esqueleto: (i) identificação clara do glifo a que se referia, (ii) linha clara que marcasse a estrutura do glifo, (iii) a inexistência ou existência mínima de pontos exteriores ao esqueleto e (iv) a possibilidade de divisão pelas partes anatómicas.

Nesta seção explicamos todo o processo de extração de esqueletos que desenvolvemos. São abordados todos os métodos utilizados até chegar ao final e as opções tomadas no seu desenvolvimento.

Retas aleatórias — *primeira técnica*

O primeiro método que testamos foi a extração de esqueletos através da interseção de retas aleatórias com cada glifo. É desenhado um número pré-determinado de retas com posições aleatórias. Depois, são detetados os pontos de interseção, ou seja, os pontos de entrada e de saída. Utilizando o par de pontos é calculado o ponto médio (Fig. 111) e o processo repete-se para todas as retas aleatórias desenhadas. Para a determinação do esqueleto são unidos todos os pontos médios resultantes (Fig. 112).

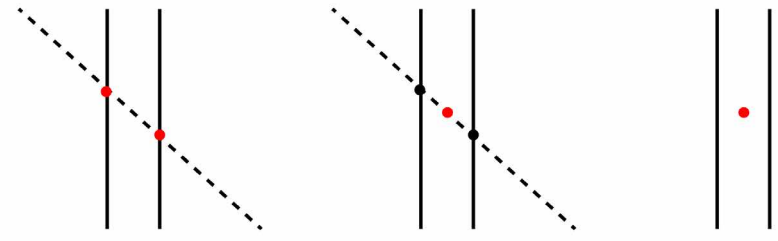


Fig. 111
Demonstração do processo de cálculo dos pontos médios

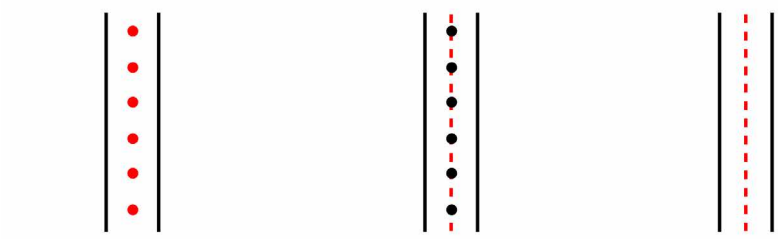


Fig. 112
Demonstração do processo de extração do esqueleto através da ligação dos pontos médios

Extração de pontos

O sistema que criamos para a extração de pontos foi também ele desenvolvido em *Processing*, utilizando a biblioteca *Geomerative* (Marxer,n.d.). O processo iniciou-se com a conversão de um glifo em *shape* e extração dos pontos do contorno da forma. Foram testados vários caracteres de diferentes tipos de letra. Mais tarde, foi desenhada uma série de retas com posições aleatórias dentro de uma variedade de valores. Para cada reta desenhada foram corridos todos os pontos e todos os pontos de contorno da forma e foi verificado se existiam pontos em comum, verificando assim se a reta intersetava a forma. Como referido anteriormente, depois de detetada a interseção era calculado o ponto médio através do ponto de entrada e saída de cada reta na *shape*. A Fig. 113 exemplifica a primeira versão do primeiro método de extração de esqueletos.

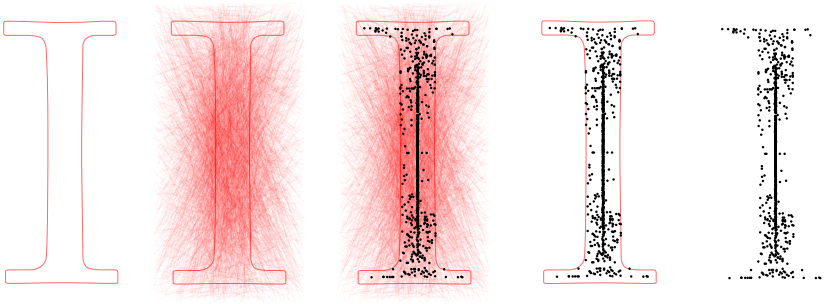


Fig. 113
Processo de extração de pontos do esqueleto (primeira versão). Foram aplicadas 2500 retas.

Apesar de idealmente o nosso método funcionar, os resultados chegados não eram os mais promissores. Mais tarde, testamos também com diferentes números de retas. Quando se aumenta o número de retas é aumentada a definição das linhas do esqueleto, mas também é maior a quantidade de pontos irrelevantes (Fig.114).

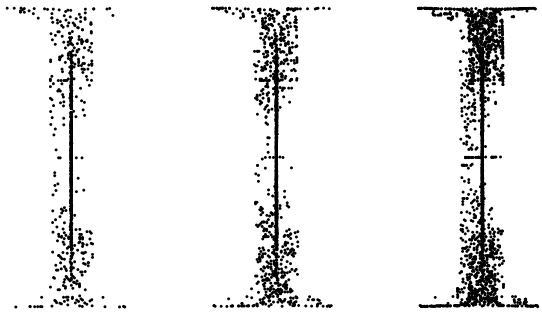


Fig. 114
Pontos do esqueleto (primeira versão) para 2500, 5000 e 8000 retas (da esquerda para a direita, respectivamente)

No final, testamos o método para todas as letras do alfabeto latino (Fig. 115). No entanto resultavam sempre pontos pertencentes ao esqueleto e fora dele.

FUNÇÃO DE REORDENAMENTO DE PONTOS

Através de um método da biblioteca que utilizamos conseguimos obter os pontos do glifo, no entanto estes eram ordenados pelo contorno. Este factor condicionava o cálculo dos pontos de interseção das rectas com a forma. A Fig. 116 apresenta como o sistema determina os pontos do esqueleto através dos pontos retornados pela função e intersetados pela forma.

Fig. 115
Pontos do esqueleto para todas as letras
(primeira versão) para 5000 retas

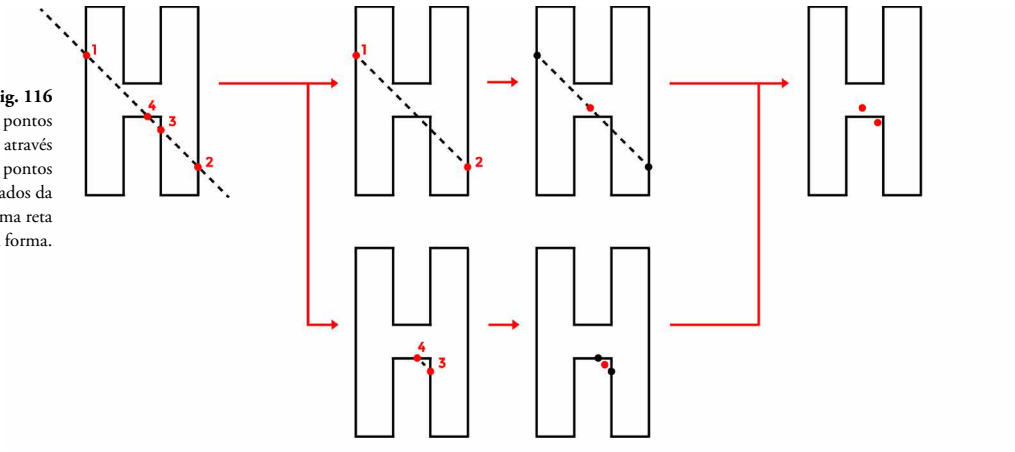


Fig. 116
Extração dos pontos do esqueleto através do cálculo dos pontos médios retirados da interseção de uma reta com a forma.

Devido ao facto dos pontos da forma estarem ordenados pelo sentido dos ponteiros do relógio os pares de pontos de entrada e saída nem sempre eram um par viável. Uma opção para este problema poderia ser associar pontos mais próximos, no entanto também seria problemático, devido às partes ocas (no exemplo da Fig. 116 o ponto 3 e 4 seriam associados, o que era errado).

De modo a ultrapassar o problema decidimos criar uma função para o reordenamento dos pontos, correndo os pontos do ponto superior esquerdo até ao inferior direito. A Fig. 117 apresenta a forma como a função funciona.

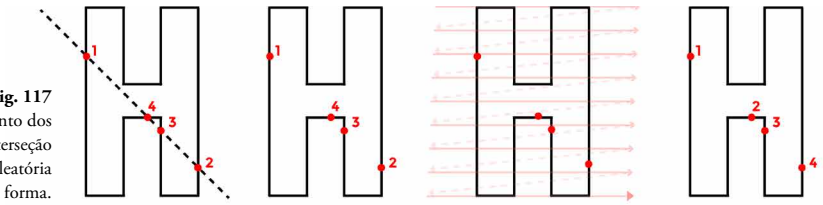


Fig. 117
Reordenamento dos pontos de interseção de uma reta aleatória com a forma.

Depois da reordenação os pontos médios resultantes são mais promissores. A Fig. 118 mostra a extração de dois pontos depois da ordenação dos pontos.

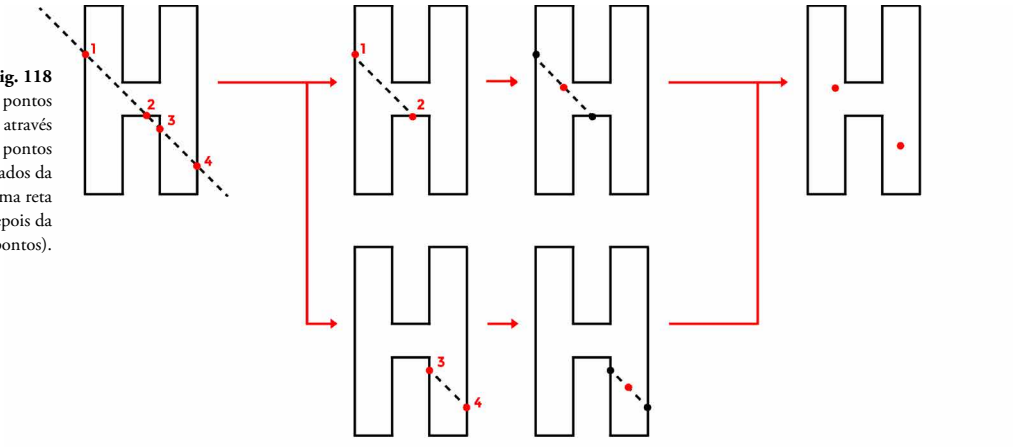


Fig. 118
Extração dos pontos do esqueleto através do cálculo dos pontos médios retirados da interseção de uma reta com a forma (depois da ordenação dos pontos).

Posteriormente, testamos a função de reordenação no nosso sistema, geramos esqueletos para todas as letras do alfabeto latino (Fig. 119). Comparando com a geração anterior é notável a melhoria nos pontos resultantes com a aplicação da função. Na primeira existe uma mancha de pontos próxima do contorno da forma que diminui com o reordenamento dos pontos.

Apercebemo-nos, no entanto, que ainda existiam alguns pontos de ruído, ou seja, pontos que idealmente não fariam parte do esqueleto de cada glifo. No entanto, a ligação dos pontos poderia diminuir esse problema.

Fig. 119
Pontos do esqueleto para todas as letras (primeira versão) com a utilização de reordenação de pontos para 5000 retas

Ligação entre pontos

O nosso objetivo era extrair esqueletos, portanto depois de detetados os pontos mais favoráveis tínhamos de uni-los para formar a linha do esqueleto. Em primeira instância, ligamos os pontos médios resultantes da interseção das retas aleatórias na forma. Na Fig. 120 está apresentado um dos resultados gerados, a tentativa de extração do esqueleto de um "H".

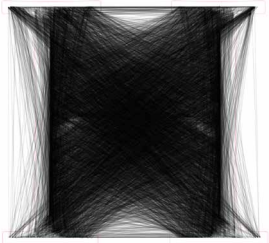


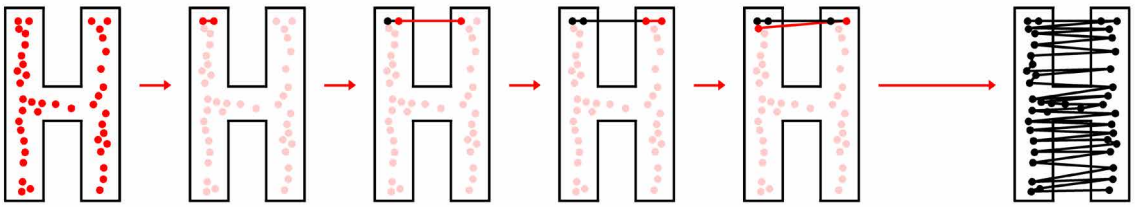
Fig. 120
Retas resultantes da ligação entre os pontos resultantes

No entanto os resultados gerados não eram os mais favoráveis. O motivo deste problema era o uso das retas aleatórias e portanto os pontos médios resultantes não estavam por ordem. Dois pontos de contorno retornados por ordem tinham uma probabilidade muito baixa de ser serem consecutivos.

LIGAÇÃO DE PONTOS — DO CANTO SUPERIOR ESQUERDO AO INFERIOR DIREITO

De modo a ultrapassar o problema decidimos criar uma função que recebesse os pontos médios resultantes da utilização das retas aleatórias e devolvesse os pontos ordenados da menor abcissa e ordenada para a maior. No entanto, como os pontos resultantes da aplicação da função eram ordenados da esquerda para a direita e de cima para baixo o resultado apresentava uma série de linhas horizontais que não pertenciam ao esqueleto, nem à forma (Fig. 121).

Fig. 121
Demonstração da utilização da função de ordenamento de pontos, primeira versão.



Para eliminar as ligações inconvenientes acrescentamos, à função, a verificação da distância entre os pares de pontos que seriam ser ligados (Fig.122).

Mais tarde, geramos esqueletos para todas as letras do alfabeto latino (Fig. 123). Apesar de os resultados gerados apresentarem uma estética interessante e de ser possível identificar cada um dos caracteres, optamos por trocar o método de ligação de pontos. Tomamos esta decisão pois o esqueleto gerado era, em grande parte, constituído por linhas horizontais, que na sua grande maioria não deveriam existir.

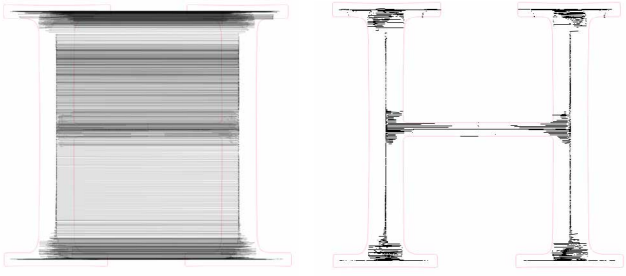


Fig. 122
Resultado gerado depois da aplicação da função de ordenamento de pontos num 'H'. Da esquerda para a direita primeira e segunda versão da função respectivamente. Na segunda geração foi aplicada a distância máxima de 10% da altura do glifo.

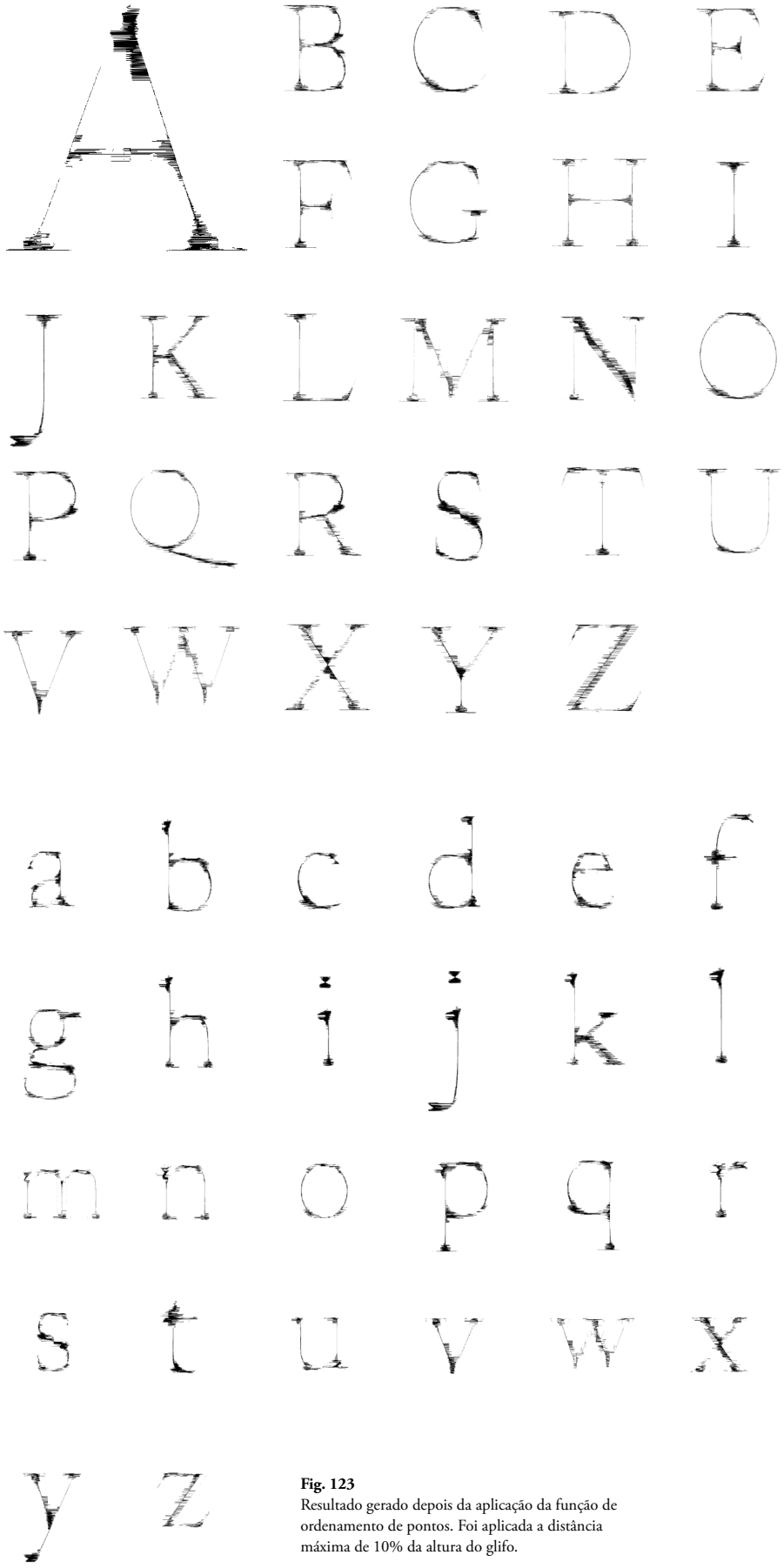


Fig. 123
Resultado gerado depois da aplicação da função de ordenamento de pontos. Foi aplicada a distância máxima de 10% da altura do glifo.

LICAÇÃO DE PONTOS PELOS PRÓXIMOS

Depois da função anterior não responder da melhor forma ao nosso objetivo decidimos implementar outro método para a ligação entre pontos. Para a nossa função decidimos ordenar os pontos unindo sempre o mais próximo. Para isso calculamos a distância entre o primeiro ponto e os pontos restantes calculando a distância entre entre eles. Depois de encontrado o ponto mais próximo do primeiro era desenhada uma linha e assim sucessivamente até passar por todos os pontos. Após algumas gerações apercebemo-nos que, tal como o que acontecia na função anterior, por vezes eram unidos pontos muito distantes. Isto acontecia quando o ponto que estávamos a unir situava-se uma posição sem saída e todos os pontos antes dele já estavam unidos. Então, a função ligava ao ponto mais próximo que poderia estar na extremidade oposta do glifo, mas entre os disponíveis era o mais próximo. Por esse motivo foi também implementada uma verificação da distância entre os dois pontos a ser ligados. Na Fig. 124 estão apresentados os resultados gerados com a aplicação da função antes e depois da verificação da distância.

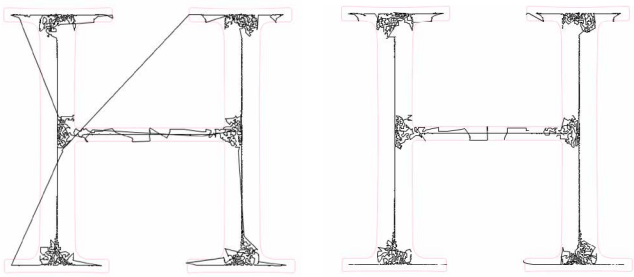


Fig. 124
Resultado gerado depois da aplicação da função de ordenamento pelos pontos mais próximos num 'H'. Da esquerda para a direita primeira e segunda versão da função respectivamente. Na segunda geração foi aplicada a distância máxima de 10% da altura do glifo.

Depois da melhoria na função geramos esqueletos para todas as letras do alfabeto latino (Fig. 125). No entanto, os resultados apresentamos continuavam a não ser os mais promissores. Existiam demasiados pontos que não faziam parte do esqueleto ideal de cada glifo.

LIGAÇÕES ATRAVÉS DE REGRESSÕES LINEARES

Depois de duas funções desenvolvidas para ligar os pontos do esqueleto os resultados ainda não eram os mais favoráveis. Por esse motivo, optamos por arranjar outra forma de unir os pontos, tentando excluir os pontos irrelevantes. Decidimos então aplicar várias regressões lineares aos pontos médios resultantes da interseção das retas aleatórias com a forma.

Para contextualizar, uma regressão linear é uma equação que normalmente serve para descobrir um valor que não se consegue estimar inicialmente. No nosso caso, tornava-se útil, pois gerava uma reta com as posições estimadas dos pontos. Os pontos de fora são excluídos, já que a reta se aproxima mais do local onde exista uma maior quantidade de pontos. A Fig. 126 demonstra a aplicação de uma regressão linear.

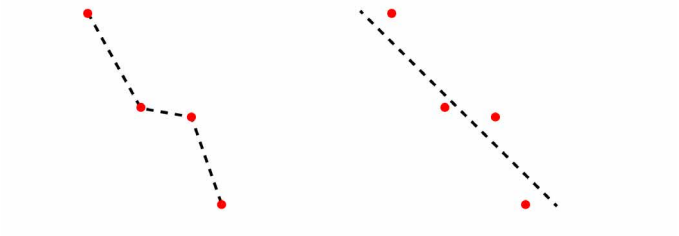


Fig. 125
Da esquerda para a direita respectivamente: Ligação dos pontos aplicando a função anterior e aplicando uma Regressão Linear.

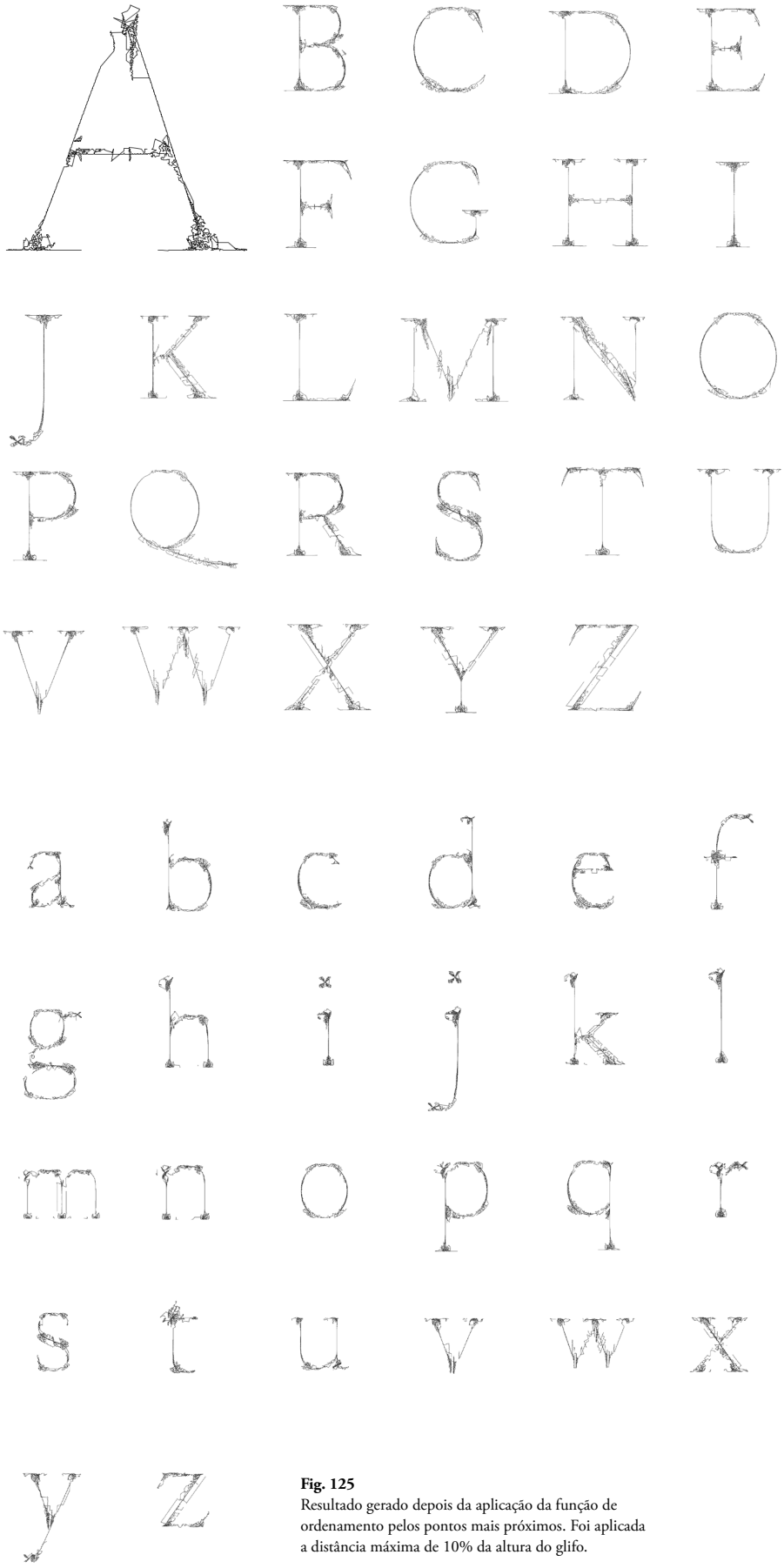


Fig. 125
Resultado gerado depois da aplicação da função de ordenamento pelos pontos mais próximos. Foi aplicada a distância máxima de 10% da altura do glifo.

Para a ligação dos pontos do esqueleto necessitavamos então de dividir o conjunto de pontos em pequenos grupos, pois uma regressão linear retornava apenas uma reta. Com a aplicação da função anterior os pontos já estavam ordenados pelos mais próximos, portanto tivemos apenas de arranjar uma forma de saber quando acabava e começava o novo grupo de pontos. Definimos uma distância máxima do primeiro ponto aos restantes do grupo e a partir do primeiro ponto fomos percorrendo os seguintes verificando a distância. Quando o ponto a ser verificado encontrava-se a uma distância superior da máxima era criado um novo grupo de pontos para calcular a regressão linear. Na Fig. 127 estão apresentados três gerações de um 'H' com diferentes distâncias máximas.

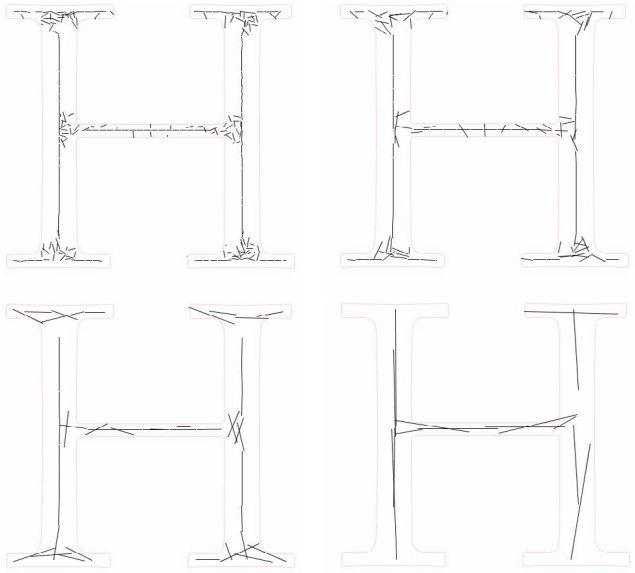


Fig. 127
Resultado gerado depois da aplicação de regressões lineares nos pontos do esqueleto de um 'H'. Da esquerda para a direita e de cima para baixo, a aplicação de uma distância máxima de 2.5%, 5%, 10% e 30% da altura do glifo respectivamente.

Depois de várias gerações de esqueletos através da aplicação de diferentes distâncias reparamos que quanto maior é a distância máxima aplicada menos são as retas desinteressantes para a estrutura do esqueleto. No entanto, ao observar algumas retas reparamos que os pontos menos favoráveis para o esqueleto estavam a afetar o declive. Mas só tínhamos observado resultados para a geração de uma letra. Decidimos então aplicar a nova função a todas as letras e números do alfabeto latino (Fig. 128). No entanto, os resultados mantinham-se.

Até ao momento, tínhamos desenvolvido três métodos para a ligação dos pontos e criação de esqueletos. No entanto, nenhum resultado parecia responder aos nossos objetivos. Possivelmente o problema não se devia à forma como estávamos a unir, mas ao facto de existirem demasiados pontos que não faziam parte do esqueleto ideal de cada glifo.

O método de extração de pontos retornava uma conjunto de pontos que não fazia parte do esqueleto ideal. O grande problema era o acomular de pontos nas ligações entre as diferentes partes do esqueleto. A Fig. 129 apresenta um 'pé' de um 'h' gerado através dos três métodos de ligação de pontos desenvolvidos.

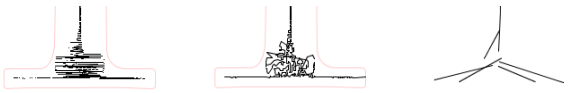


Fig. 129
Resultado gerado através da aplicação das três formas de ligação de pontos do método de extração de esqueletos através de retas aleatórias .

Depois de uma análise detalhada de todos os esqueletos gerados até ao momento decidimos optar pelo desenvolvimento de outro método para a extração dos pontos do esqueleto.

Fig. 128
Resultado gerado depois da aplicação de regressões lineares. Aplicação de uma distância máxima de 10% da altura do glifo respectivamente.

Normais aos pontos da forma — *segunda técnica*

O novo método que desenvolvemos consistia na determinação da reta normal para a curva num determinado ponto do contorno do esqueleto. Depois, era calculado o ponto médio resultante dos pontos de interseção da reta com a forma.

Para contextualizar, a reta normal à curva num ponto é aquela que passa pelo ponto e é perpendicular à reta tangente da curva no ponto. Por outro lado, a reta tangente à curva num ponto é a aquela que toca em apenas um ponto. Na Fig. 130 é demonstrado o cálculo dos pontos médios.

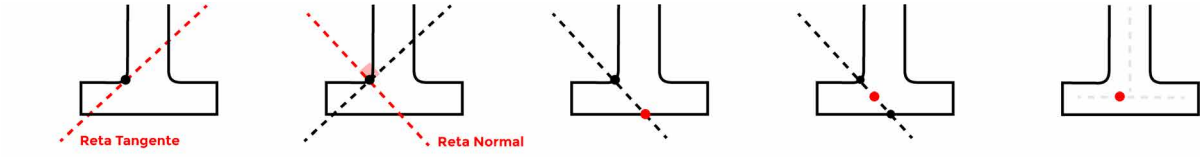


Fig. 130
Determinação da reta normal à curva num ponto e posterior cálculo do ponto médio, ponto que pertence ao esqueleto.

Na sua essência, o sistema era semelhante ao método anterior de extração de pontos. Depois de calculada a reta normal, que passava pelo ponto em questão, era determinado o outro ponto de interseção da reta com a forma. Imediatamente a seguir, era calculado o ponto médio a partir desse par de pontos. Depois de calculadas todas as normais aos pontos do glifo, os pontos eram ligados. Na Fig. 131 é demonstrado o processo de extração do esqueleto pelo método anterior e pelo atual.

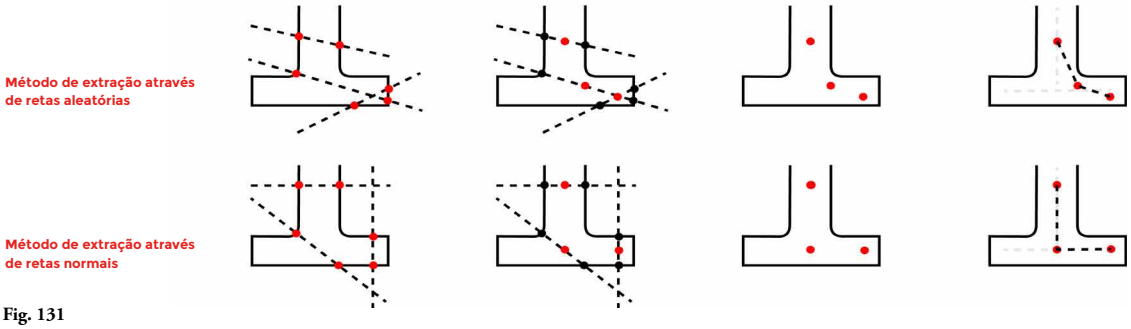


Fig. 131
Exemplificação do processo de cálculo do esqueleto resultante do método anterior e do atual, em cima e embaixo, respetivamente.

Com a utilização do método anterior para a extração dos esqueletos surgem pontos que fazem parte do esqueleto. No entanto, como as retas são aleatórias existe também a possibilidade do surgimento de pontos que vão funcionar como ruído para a criação do esqueleto. Por outro lado, no método atual temos a vantagem de utilizarmos apenas o conjunto de retas mais promissoras, portanto possivelmente teremos melhores resultados.

Extração de pontos

De modo a testar a nova ideia foi desenvolvido outro sistema em *Processing*, utilizando a biblioteca *Geomerative* (Marxer,n.d.). O processo iniciou-se, semelhante ao método anterior, com a extração dos pontos do contorno da forma e aplicação da função de reordenação de pontos.

Depois, foram percorridos todos os pontos e para cada um deles foi calculado o declive da reta tangente. A seguir, calculamos o declive da reta normal que, como referido anteriormente, é perpendicular à reta tangente. Através desse declive descobrimos se a reta normal era horizontal, vertical ou oblíqua.

Após a determinação de todas as retas normais tivemos de percorrer, para cada uma delas, todos os pontos pertencentes ao contorno da forma e verificar se estes

pertenciam à reta. O processo de verificação do ponto variou consoante o tipo de reta. De modo a não associar pontos demasiado afastados foi também verificada a distância entre os pares de pontos.

No final, cada reta normal tinha dois pontos associados, o ponto que originou a determinação da reta e o ponto extraído. Ambos pertenciam ao contorno da *shape* usada. A partir desses dois foi calculado o ponto médio. O processo repetiu-se para todas as retas normais, a Fig. 132 apresenta as retas normais utilizadas e os pontos médios que constituem o esqueleto de um 'H'.

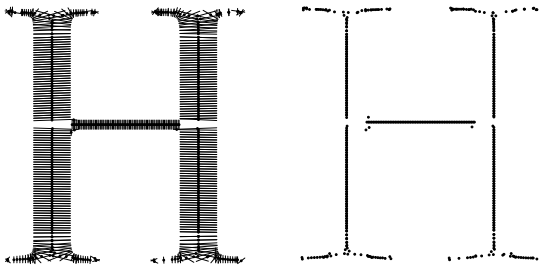


Fig. 132
Normais e pontos do esqueleto de um 'H' gerados depois da aplicação do método da extração através do uso de retas normais.

Como previsto, este novo método apresentava resultados muito mais promissores que o anterior. No entanto, existiam algumas falhas de pontos no esqueleto. Estas aconteciam devido à verificação da distância que o sistema fazia para o cálculo dos pontos médios. Por outro lado, essa verificação era necessária, porque, no caso do 'H', o ponto médio não poderia ser calculado tendo em conta pontos de 'hastes' diferentes.

De modo a validar o sistema geramos resultados para letras diferentes. No entanto, durante esta validação apareceram alguns problemas. A Fig. 133 apresenta alguns dos resultados gerados.

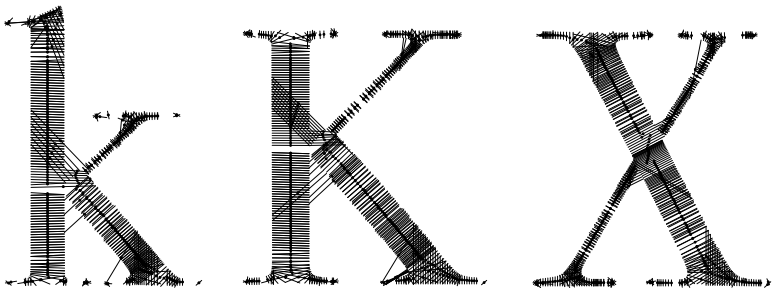


Fig. 133
Normais e pontos do esqueleto gerados de um 'k', 'K' e 'X' depois da aplicação do método da extração através do uso de retas normais.

Em alguns casos a verificação da distância entre os pontos não era suficiente tínhamos de acrescentar outro método de validação. Para isso decidimos verificar se o ponto médio pertencia à *shape*, se sim, o ponto era válido. Depois, melhoramos a verificação percorrendo todos os pontos da reta, desde o ponto que a originou até ao ponto extraído, e analisamos se cada um deles pertencia à reta. Este acrescento na validação resolveu o problema (Fig. 134).

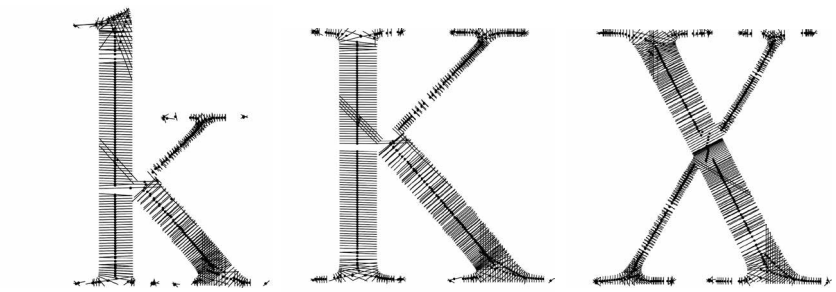


Fig. 134
Normais e pontos do esqueleto gerados de um 'k', 'K' e 'X' depois da aplicação do método da extração através do uso de retas normais.

Decidimos então gerar pontos para todas as letras do alfabeto latino. A Fig.135 apresenta os resultados.

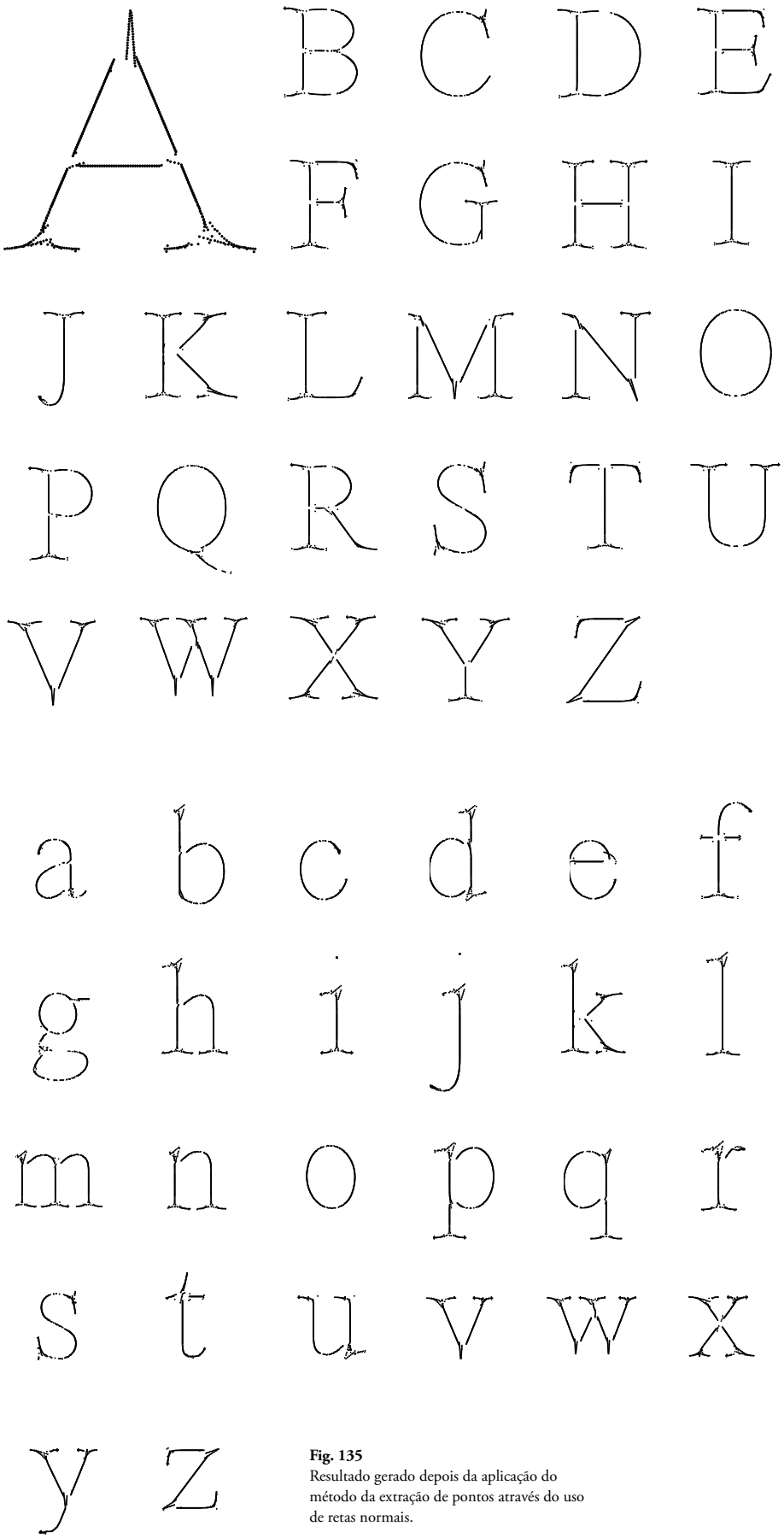


Fig. 135
Resultado gerado depois da aplicação do método da extração de pontos através do uso de retas normais.

Ligação entre pontos

Uma vez mais, e agora com pontos mais promissores, tínhamos de ligar os pontos para formar o esqueleto.

LICAÇÃO DE PONTOS PELOS PRÓXIMOS

No método de extração anterior tinham sido criadas uma série de funções para a ligação de pontos. No entanto, como abordado anteriormente, não tiveram os resultados mais promissores devido ao conjunto de pontos recebido. Com a implementação desta nova forma de extração tínhamos o necessário para construir esqueletos sem ruído.

Portanto para este método de ligação de pontos foi usada a função anteriormente criada. A Fig. 136 (página seguinte) apresenta os esqueletos gerados para todas as letras do alfabeto latino. Após uma análise aos resultados gerados reconhecemos que este poderia ser o caminho desta dissertação. No entanto, decidimos ainda testar a outra função que tínhamos desenvolvido anteriormente através do uso de regressões lineares.

LIGAÇÕES ATRAVÉS DE REGRESSÕES LINEARES

Este método de ligação foi criado para o método de extração de pontos anterior como tentativa de exclusão dos pontos irrelevantes. Além disso, pretendíamos também encontrar uma linha média que representasse os pontos recebidos da melhor forma. Por esse motivo, esta seção não vai ser muito extensa, porque iremos apenas apresentar as gerações resultantes da aplicação da função de regressão linear (Fig. 137, duas páginas à frente) e analisar resultados.

Com a utilização deste método para a ligação dos pontos os resultados gerados são mais simplificados. No entanto não parece ser formado um esqueleto completo, pois as várias linhas desenhadas não se ligam. Foi testado também a ligação dessas linhas, mas os resultados não eram promissores portanto não estão aqui exemplificados. Em geral, o método de ligação anterior responde melhor aos nossos objetivos.

Fig. 136
Esqueletos gerados depois da aplicação do método de extração de pontos através de retas normais e o método de ligação de pontos mais próximos. Foi aplicada a distância máxima de 10% da altura do glifo.

Fig. 137
Esqueletos gerados depois da aplicação do método de extração de pontos através de retas normais e o método de ligação de pontos através de regressões lineares. Foi aplicada a distância máxima de 10% da altura do glifo.

Técnica final

Até ao momento tínhamos um sistema de extração de pontos mais ou menos satisfatório. No entanto, os esqueletos gerados ainda tinham alguns problemas por resolver. O sistema de ligação de pontos mais promissor era através da ligação por pontos mais próximos, mas mesmo esse, devido à grande quantidade de pontos, era constituído por linhas muito irregulares. Esse factor condicionava a validade dos esqueletos até agora gerados, pois não existiam linhas claras a marcar a estrutura do esqueleto.

Além disso, um dos nossos objetivos iniciais era dividir o esqueleto em partes anatómicas para posteriormente combinação de diferentes tipos de letra e ainda não estava feito. Estávamos num impasse. A conceção dos esqueletos era uma parte importante desta dissertação, mas não era a única. Ainda havia muito a fazer e tínhamos já gasto bastante tempo na tentativa de extração de esqueletos viáveis. Refletimos sobre qual seria o próximo passo e encontramos um algoritmo que poderia ser adaptado para facilitar o processo de extração de esqueletos. Mais tarde, apercebemo-nos também que, além dele, existiam uma série de métodos de extração de esqueletos, técnica bastante utilizada em domínios como a animação 3D (Schaefer, Yuksel, 2007; Tierny, Vandeborre, Daoudi, 2008; Tierny, Vandeborre, Daoudi, 2006).

O algoritmo que encontramos, denominado *Zhang-Suen Thinning Algorithm* (Wu, Marquez, 2003), tem como objetivo extrair as linhas estruturais de uma imagem binária. Por outras palavras, o algoritmo recebe uma imagem constituída apenas por pixels de duas cores diferentes, por exemplo, preto e branco e devolve-a modificada (Fig. 137). Todos os pixels imprescindíveis para a compreensão da mesma são retirados, portanto é ideal para o nosso caso, a extração de esqueletos.

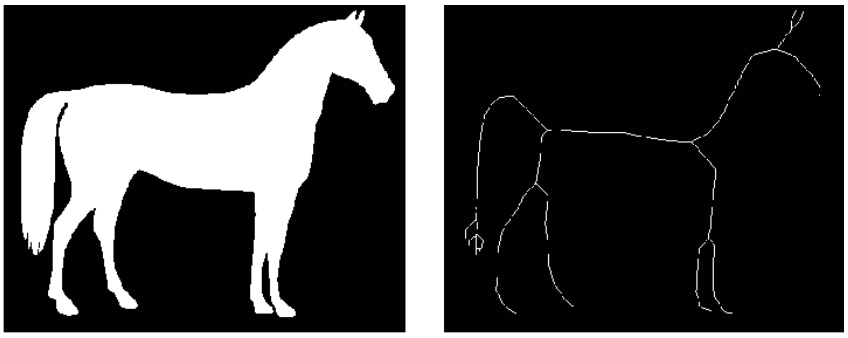


Fig. 137
Demonstração do uso do *Zhang-Suen Thinning Algorithm*. Da esquerda para a direita, a imagem original e depois de aplicado o algoritmo (Scikit-image, n.d.)

Encontrada a possível solução para a extração dos esqueletos, faltava validar para o nosso problema. O mecanismo desenvolvido para o cálculo do esqueleto de glifos foi implementado em conjunto com os orientadores deste trabalho.

Extração do esqueleto

Este último método difere muito dos anteriores pelo facto de não separar a extração de pontos com a ligação entre eles. Através da implementação do *Zhang-Suen Thinning Algorithm* é extraído logo a estrutura do glifo.

O processo de geração do esqueleto de cada glifo inicia-se com o varrimento de todos os *pixels* com oito vizinhos. Além deles, os *pixels* pertencentes ao contorno também são percorridos. A Fig. 138 demonstra um *pixel* avaliado pelo sistema.

P9 P2 P3
P8 **P1** P4
P7 P6 P5

Fig. 138
Pixel propício a ser avaliado pelo sistema (P1) e a sua vizinhança. P1 tem de ser um *pixel* preto.

Para cada *pixel* avaliado é calculado o número de transições de branco para preto. A ordem pela qual são percorridos inicia-se no *pixel* acima do atual e vai pelo sentido horário até chegar ao primeiro pixel (P1, P2, P3, P4, P5, P6, P7, P8, P9, P2). Para cada *pixel* é também apurado o número de *pixels* vizinhos pretos.

Depois, se o *pixel* encaixar numa série de condições: (i) ter apenas uma transição de branco para preto, (ii) ter entre dois a seis vizinhos inclusive, (iii) ter o pixel acima (P2), à direita (P4) ou em baixo (P6) branco e (iv) ter pixel à direita (P4), em baixo (P6) ou à esquerda (P8) branco; é definido como branco.

Após as alterações, é novamente testado o pixel e se ele: (i) tiver apenas uma transição de branco para preto, (ii) tiver entre dois a seis vizinhos inclusive, (iii) ter o pixel acima (P2), à direita (P4) ou à esquerda (P8) branco e (iv) tiver o pixel acima (P2), em baixo (P6) ou à esquerda (P8) branco; é definido como branco.

No final, se pelo menos um *pixel* sofreu alteração são repetidas novamente as duas verificações até que nenhum *pixel* seja alterado. Na Fig. 139 é mostrado o esqueleto gerado para um 'h'.

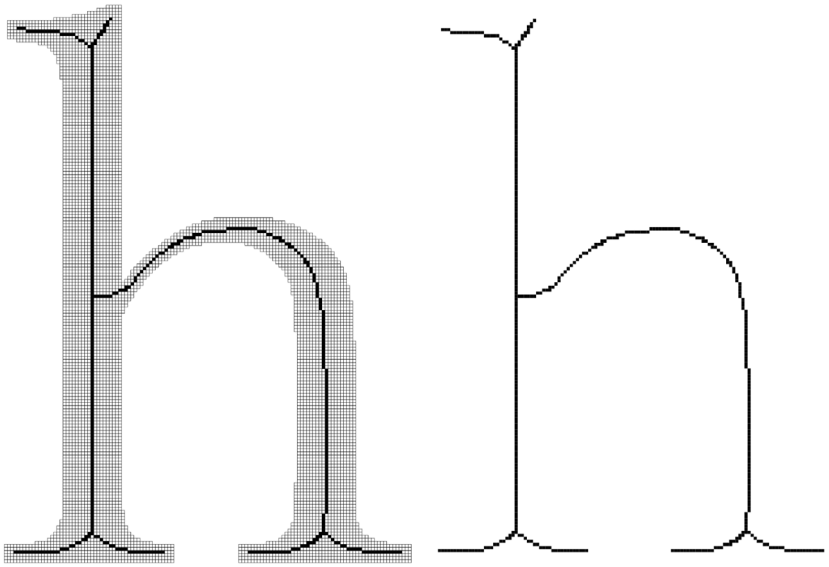


Fig. 139
Pixels percorridos e esqueleto final gerado pela Técnica Final.

Depois, como o objetivo de finalizar o esqueleto, convertemos os *pixels* desenhados em linhas. Para validar o novo método geramos esqueletos para todas as letras e números do alfabeto latino para diferentes tipos de letra. Na Fig.140 está apresentado um dos resultados gerados. Como previsto, este método de extração de esqueletos apresentou resultados melhores que os métodos anteriores.

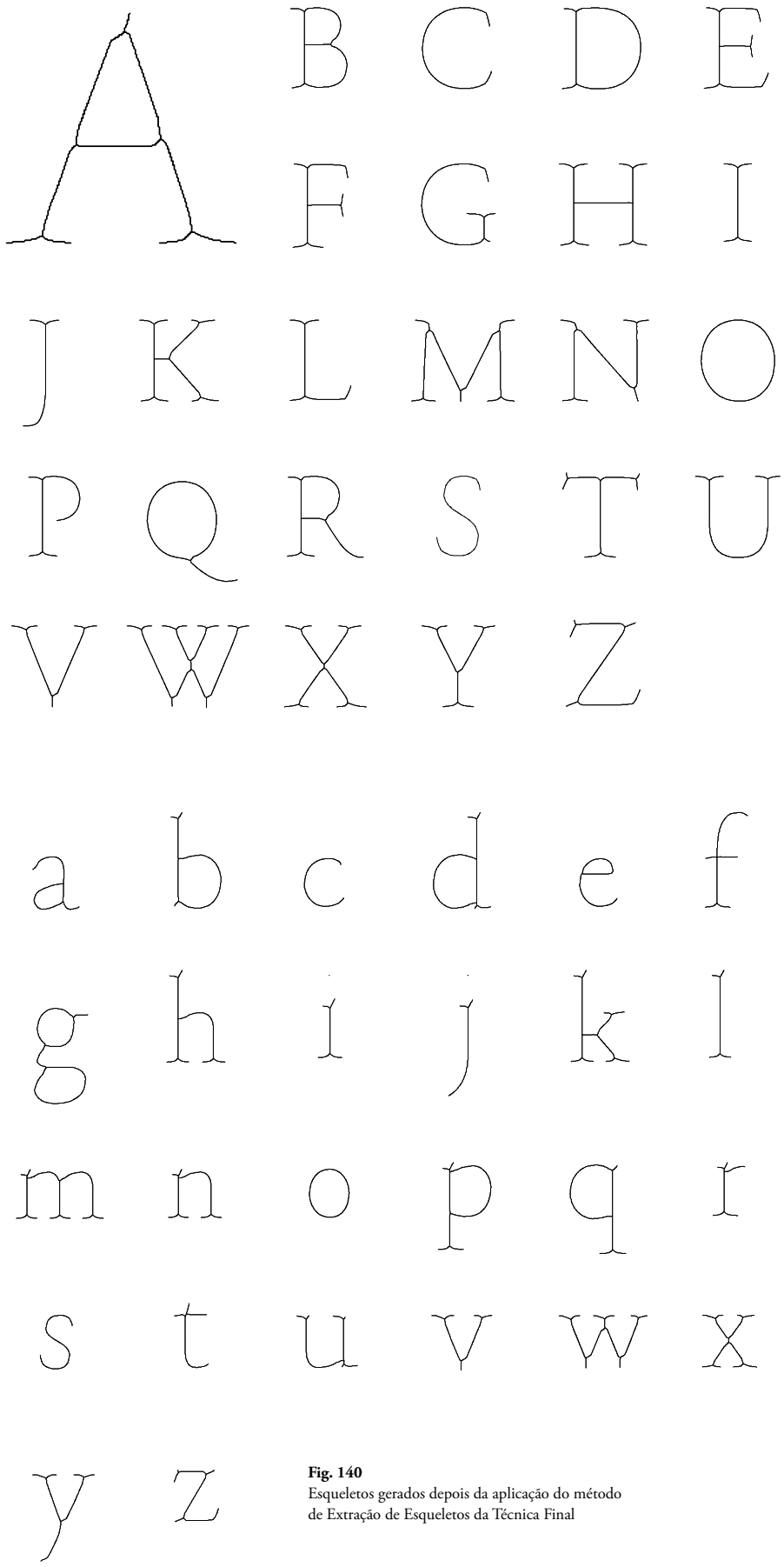


Fig. 140
Esqueletos gerados depois da aplicação do método de Extração de Esqueletos da Técnica Final

Divisão do esqueleto em diferentes partes anatómicas

Depois do sucesso do último método da extração de esqueletos, com a implementação do algoritmo de *Zhang-Suen*, tínhamos finalmente todas as condições para a divisão do esqueleto em diferentes partes anatómicas. Anteriormente, já o tínhamos tentado fazer através de interseções, diferenças ou reuniões com diferentes glifos, no entanto sem grande sucesso. Com os esqueletos extraídos e simplificados precisávamos apenas de arranjar um método que identificasse as diferentes partes.

Após uma análise dos esqueletos gerados reparamos que quando um ponto fazia parte de três segmentos ele dividia diferentes partes do esqueleto. De forma a testar essa teoria percorremos todos os pontos do esqueleto e quando era encontrado um ponto de fronteira era criado uma nova linha. Na Fig. 141 apresentamos um esqueleto de um 'h' gerado com a divisão em partes.

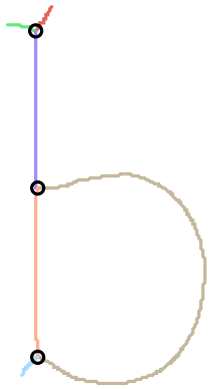


Fig. 141
Esqueleto de um 'h' gerado depois da aplicação do método de Extração de esqueletos da Técnica Final e divisão em partes.

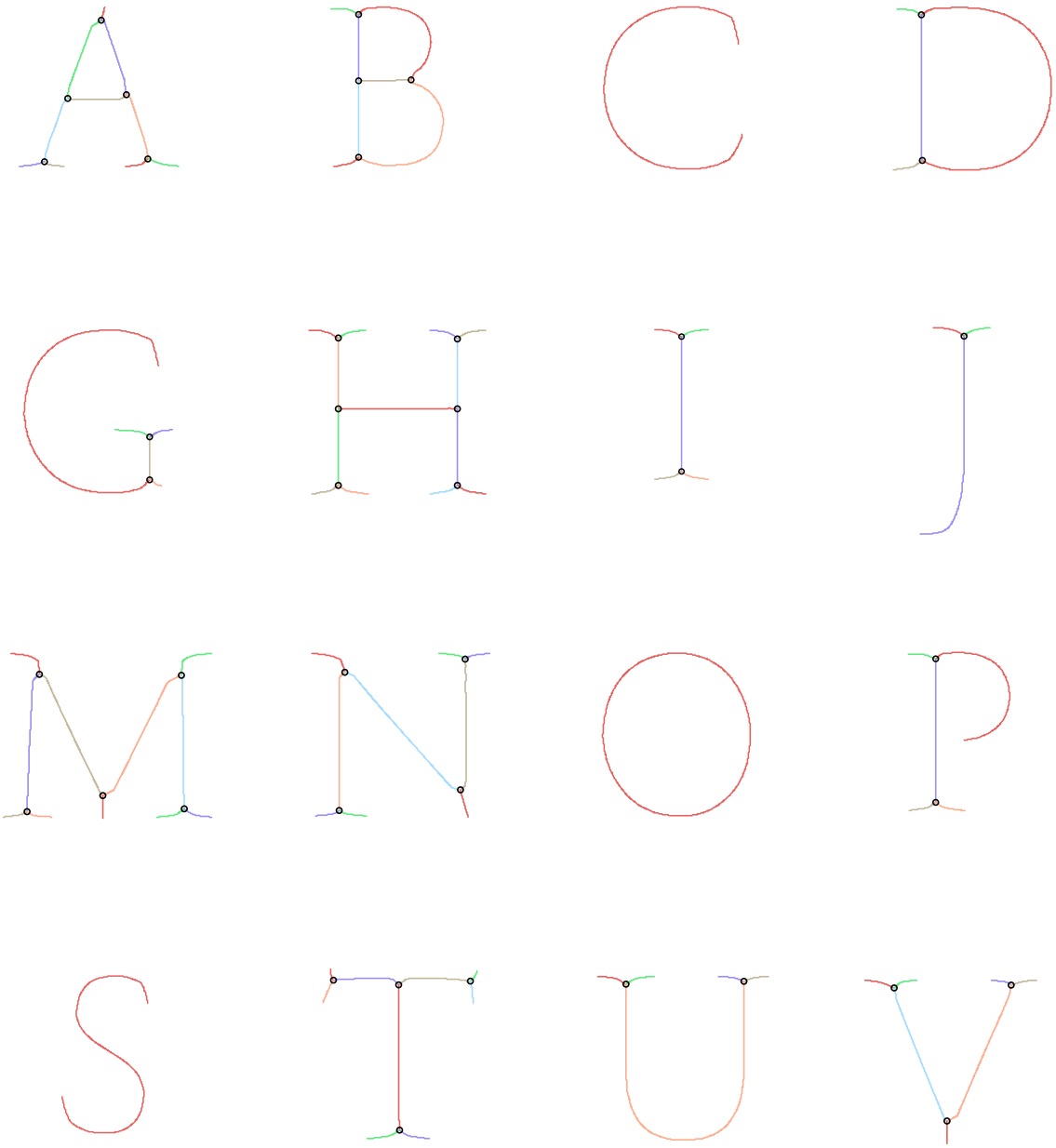
Os círculos destacam os pontos de fronteira e foi aplicada uma cor diferente para diferenciar as diferentes partes detetadas. Com a utilização deste método o esqueleto era dividido em vários segmentos, no entanto, por vezes, eram divididas partes incorretas. No caso do 'h' da Fig. 141 as serifas são geradas separadas, mas o problema poderia ser ultrapassado. No entanto, apenas o vamos abordar na seção seguinte.

Uma vez mais, a fim de testar o sucesso do método geramos esqueletos para todas as letras do alfabeto latino, identificando as diferentes partes (Fig. 142). No geral, os resultados são bastante satisfatórios. O método através dos pontos de fronteira identificava as partes anatómicas no geral.

Simplificação de pontos

Ao longo do processo, fomo-nos apercebendo que por vezes os esqueletos gerados tinham algumas irregularidades. A ligação dos pontos era, por vezes, em zig-zag e portanto causava algum ruído. De modo a diminuir os pontos em excesso e tornar a estrutura dos esqueletos mais claros decidimos fazer a implementação de um outro algoritmo.

O algoritmo que optamos por utilizar, designado *Ramer Douglas Peucker* (Wu, Marquez,2003), é um algoritmo iterativo de ajuste de pontos. Dada uma curva composta por segmentos de reta ele devolve uma curva semelhante com menos pontos. O algoritmo verifica se a curva é semelhante através do cálculo da distância máxima entre a curva original e a simplificada. O resultado do uso do algoritmo é um subconjunto de pontos que definiram a curva original (Fig. 143).



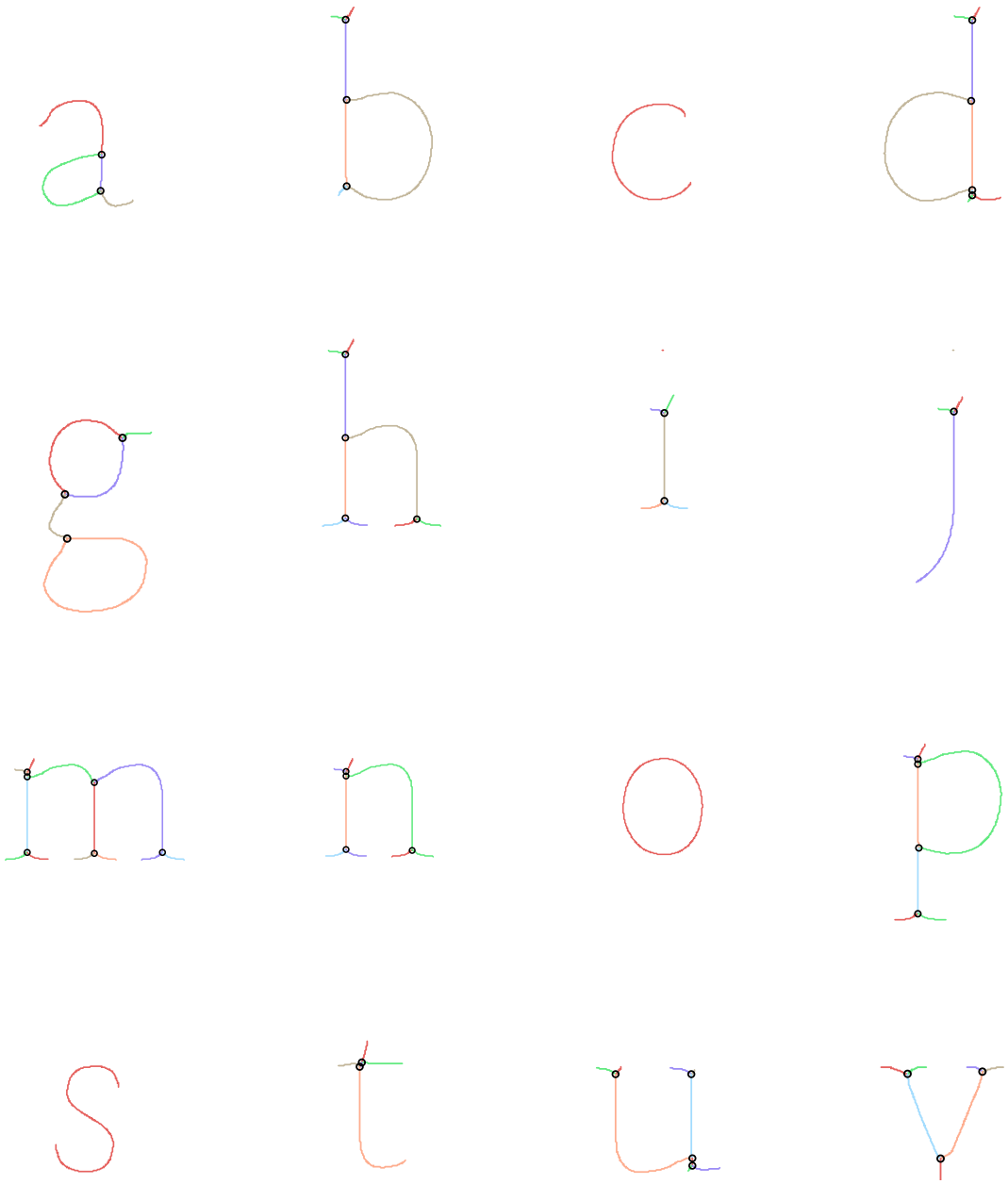


Fig. 142
Exemplos de squeletos gerados com a divisão das diferentes partes, depois da aplicação do método de Extração de Esqueletos da Técnica Final

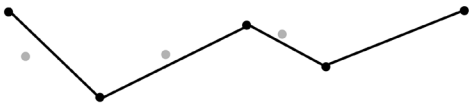


Fig. 143
Exemplo dos pontos iniciais e a curva final gerada

O método de simplificação foi então aplicado a cada um dos segmentos dos esqueletos gerados. Definimos um grau de simplificação para ditar a diferença entre a curva original e a final e calculamos a distância desde o primeiro ponto ao último (Fig. 144). Também inicialmente, marcamos logo o primeiro e último ponto para serem mantidos.

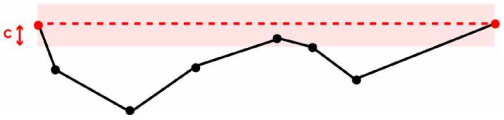


Fig. 144
Pontos e curva iniciais e demonstração da distância do primeiro ao último ponto (c).

Depois, o processo traduz-se na divisão recursiva de cada segmento. Seguidamente, fazemos um varrimento de todos os pontos e encontramos o ponto que é perpendicularmente mais afastado do segmento que une o primeiro ao último ponto.

Se o ponto encontrado tiver uma distância ao segmento menor que o grau da simplificação, todos os pontos não marcados para serem mantidos são descartados. Se, por outro lado, o ponto tiver uma distância ao segmento maior que o grau da simplificação erro, esse ponto deve ser mantido. Depois é chamado o algoritmo recursivamente do primeiro ao ponto com maior distância e do ponto mais distante, inclusivo, até ao último ponto. Quando a recursividade é terminada é retornado o novo segmento, constituído apenas pelos pontos marcados para serem mantidos. Na Fig. 145 é demonstrado o resultado da aplicação de diferentes graus de simplificação.

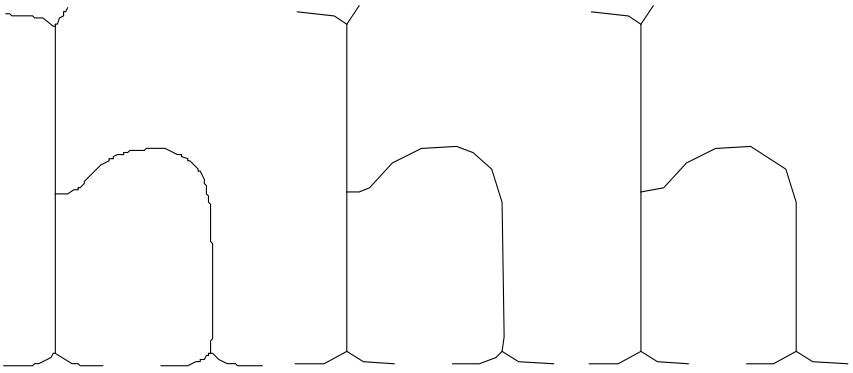


Fig. 145
Esqueletos gerados depois do uso de diferentes graus de simplificação (0.5,1,2)

Os resultados gerados com o uso do algoritmo tornam-se interessantes, pois abrem caminhos para o preenchimento dos esqueletos. O uso de um *input* que possa modificar o grau de simplificação dos esqueletos poderia gerar glifos bastante diversificados.

Combinação de diferentes tipos de letra

EXTRAÇÃO DE SERIFAS E JUNÇÃO DE SEGMENTOS PROMISSORES

Um dos nossos objetivos era a combinação de tipos de letra. Uma vez que já tínhamos um sistema que gerava esqueletos ramificados o próximo passo era a combinação de partes. Inicialmente, geramos esqueletos para vários tipos de letra, já utilizados anteriormente, e analisamos as diferenças (Fig. 146).

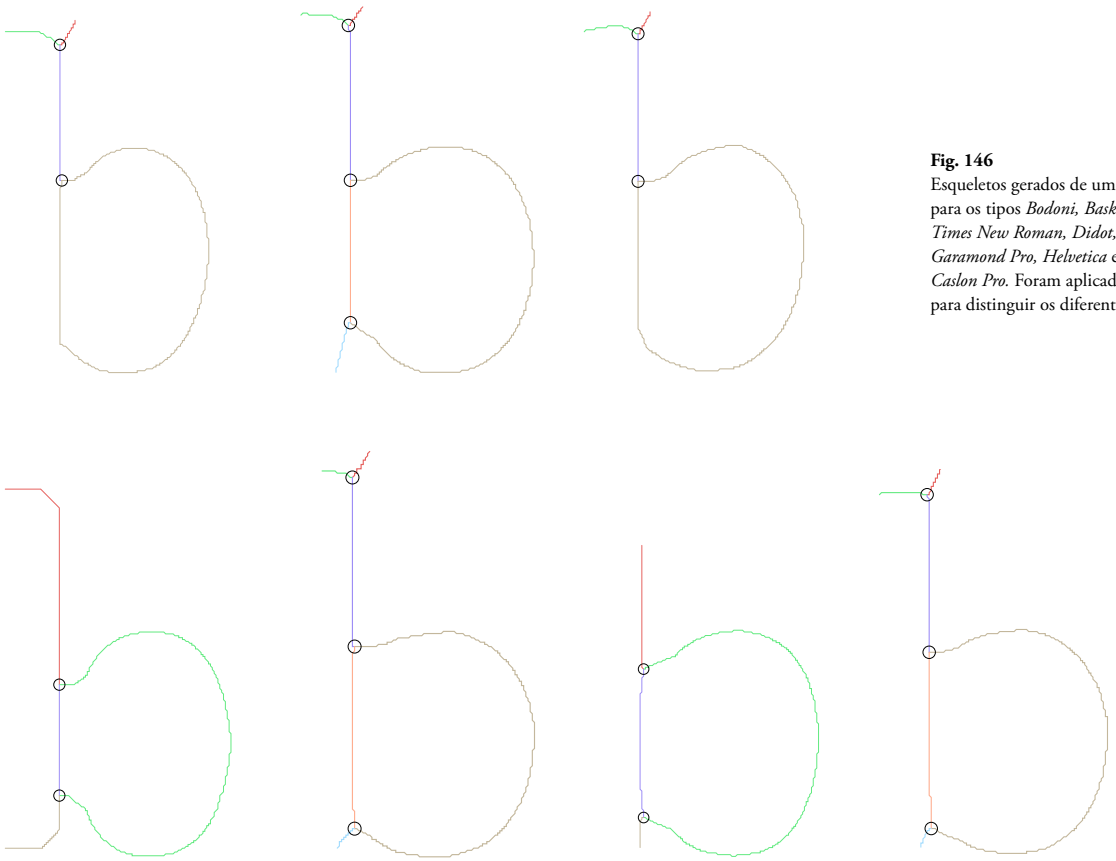


Fig. 146
Esqueletos gerados de um 'b' para os tipos *Bodoni, Baskerville, Times New Roman, Didot, Adobe Garamond Pro, Helvetica* e *Adobe Caslon Pro.* Foram aplicadas cores para distinguir os diferentes traços.

Analisando os diferentes 'b's gerados apercebemo-nos que os esqueletos eram ramificados de formas diferentes para os vários tipos de letra. Parte do problema era devido à existência de serifas. Em alguns glifos, estas criavam pontos de fronteira e portanto acrescentavam um novo segmento. De forma a ultrapassar esse problema decidimos, para cada caracter, definir um número de segmentos ideal e todos os esqueletos gerados deveriam ter esse número de linhas. No entanto, reparamos que o problema não ficava resolvido apenas com essa verificação.

Mesmo com o mesmo número de segmentos, por vezes estes não era equiparáveis. Por exemplo, na Fig. 146, o esqueleto gerado para o 'b' da Bodoni e da Helvetica (primeiro e sexto glifo) tem quatro segmentos, no entanto não são correspondentes. Grande parte deste problema devia-se, novamente, ao facto da existência das serifas. Portanto decidimos, antes de verificar a quantidade de segmentos de cada glifo, distinguir traços que fossem serifas. Para isso determinamos um número máximo de pontos e uma distância máxima, pois as serifas são, normalmente, os segmentos mais pequenos entre as partes anatómicas de um glifo. Para validar o nosso método de distinção de serifas geramos, uma vez mais, esqueletos de 'b's para diferentes tipos de letra, destacando as serifas (Fig. 147).

Os resultados gerados pareciam promissores e através deste método podiamos facilmente desenhar tipos com ou sem serifas. Também agora, com a deteção das serifas, já podíamos associar partes anatómicas semelhantes entre glifos. Mas antes de mais, tivemos de aperfeiçoar o método de validação de cada esqueleto e definir um novo número de partes ideais para cada caractere, excluindo as serifas. Agora, a validação do esqueleto gerado era feita apenas aos traços do esqueleto que não eram serifas.

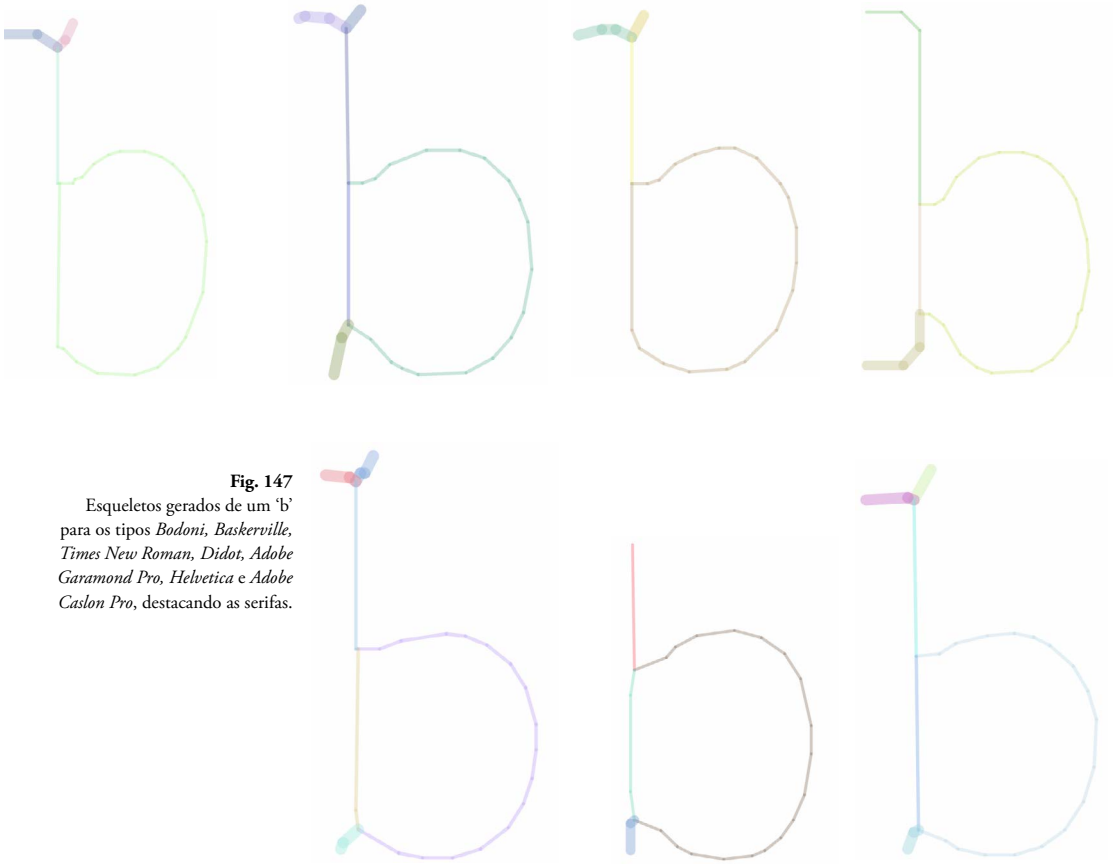


Fig. 147
Esqueletos gerados de um 'b' para os tipos *Bodoni, Baskerville, Times New Roman, Didot, Adobe Garamond Pro, Helvetica* e *Adobe Caslon Pro*, destacando as serifas.

Se o esqueleto tivesse mais ramificações do que as definidas para o caractere eram unidos o número de traços necessários para ficar com o número ideal. No entanto, só eram unidos pares de segmentos que respeitassem uma série de condições: (i) tivessem pelo menos um ponto em comum e (ii) tivessem um número de pontos menor ou igual ao máximo por nós definido. O número de pontos máximo estava sempre associado ao grau de simplificação usado. Para cada par de segmentos que passassem a estas condições era guardado o ângulo entre eles e o seu ponto em comum. Depois, os pares de segmentos eram ordenados do melhor ao pior ângulo. O melhor ângulo seria aquele que mais se aproximava do 180º (Fig. 148). Depois, eram apenas unidos os segmentos necessários.

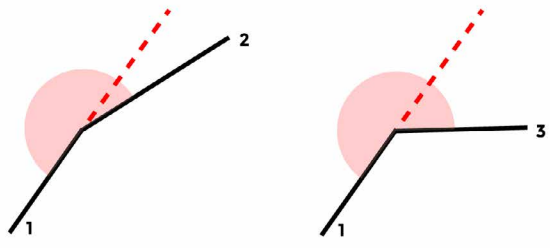


Fig. 148
Demonstração do calculo do ângulo entre segmentos (ângulo entre 1 e 2 e entre 1 e 3). A linha vermelha tracejada representa o par ideal para o segmento 1, portanto o segmento 2 é melhor par do 1 do que o 3.

Depois de definidos os pares de segmentos era necessário unir. Para isso, foram ordenados todos os pontos do primeiro segmento desde o mais afastado do ponto comum até ao mais próximo. Depois, ao conjunto de pontos anterior eram acrescentados os pontos do segundo segmento desde o ponto comum até ao mais afastado. No final, apenas era necessário substituir o conjunto de pontos finais pelos dois conjuntos iniciais e, deste modo, eliminávamos uma ramificação ao esqueleto final. Na Fig. 149 é demonstrado o método de união de traços.

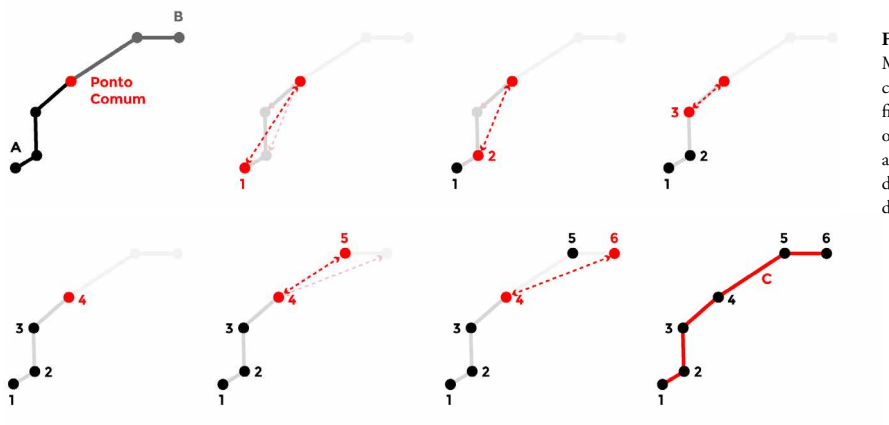


Fig. 149
Método de união de dois traços (A e B) com um ponto comum. O segmento final C é constituído pelos pontos ordenados do A, do mais afastado até ao mais próximo do ponto comum, e dos pontos ordenados do segmento B, do mais próximo até ao mais afastado.

Existiam, no entanto, outros problemas nas ramificações geradas. Por vezes, o problema não era o esqueleto gerado estar demasiado dividido, mas sim estar pouco ramificado. Para alguns casos a solução era diminuir o número de divisões para o dado caractere. Era preferível ter um esqueleto menos dividido, mas com as partes associadas corretamente para diferentes tipos de letra. A Fig. 150 apresenta os esqueletos gerados para um 'b' com duas ramificações e excluindo as serifas.

Depois de uma análise aprofundada dos resultados dos diferentes 'b's reparamos que, apesar do nosso sistema gerar sempre 'b's com duas ramificações, estas por vezes não eram idênticas. O esqueleto do 'b' gerado para a *Bodoni* e *Baskerville* (primeiro e segundo esqueleto da Fig. x) eram uma prova disso. Este problema devia-se, como referido anteriormente, às serifas de cada tipo. Quando um tipo de letra não tinha serifas era menos ramificado e o método que tínhamos criado para a união de traços não estava a ultrapassar este problema, pois não associava os traços corretos. Testamos ainda com outros caracteres e os resultados continuavam a ser pouco satisfatórios.

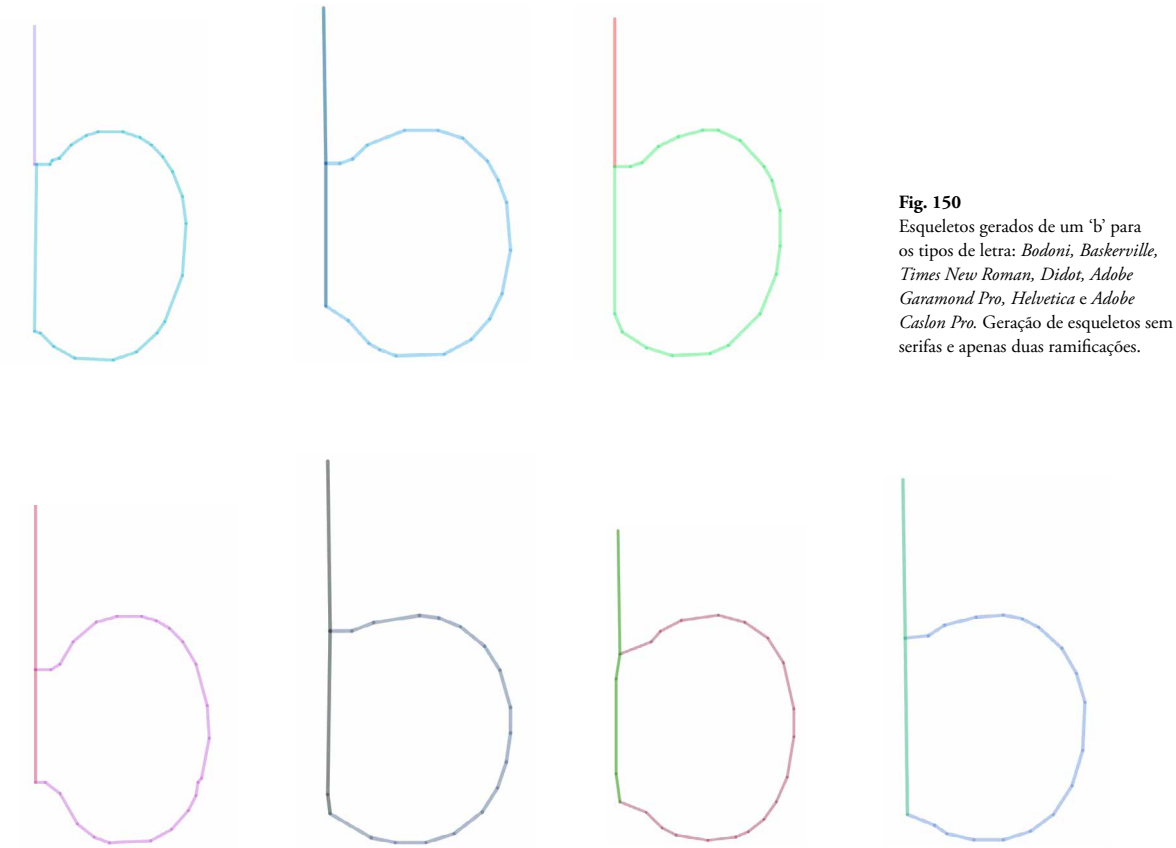


Fig. 150
Esqueletos gerados de um 'b' para os tipos de letra: *Bodoni, Baskerville, Times New Roman, Didot, Adobe Garamond Pro, Helvetica* e *Adobe Caslon Pro.* Geração de esqueletos sem serifas e apenas duas ramificações.

Até ao momento, tínhamos um método que distinguia serifas das restantes partes anatómicas e um método de união de traços que por vezes não associava as ramificações mais indicadas. Estavamos a gastar demasiado tempo na tentativa de tornar as partes geradas o mais idênticas possível e ainda não sabíamos como combinar diferentes tipos de letra. Precisávamos de uma forma eficaz de combinar as partes anatómicas entre diferentes tipos de letra sem necessitar dessas partes serem completamente semelhantes. Portanto, decidimo-nos focar no método de combinação de tipos.

COMBINAÇÃO DE CAMADAS

O objetivo do nosso projeto era a geração de tipos de letra viáveis. Além disso, pretendiamos também que a estrutura dos tipos gerados fosse totalmente automática. Um outro ponto relevante para esta dissertação era a possibilidade dos tipos gerados se poderem adaptar a diferentes circunstâncias. Portanto, a combinação de partes anatómicas oriundas de um conjunto diversificado de tipos de letra traria um vasto conjunto de possibilidades para a estrutura das fontes geradas.

Na seção anterior, geramos esqueletos para um dado caractere a partir dos diferentes tipos de letra. Depois, tentamos aproximar as partes deles o máximo possível, no entanto não conseguimos completamente. Com alguns caracteres não funcionava de todo e o facto de termos de inserir um número de partes para cada caractere não era muito vantajoso. Tínhamo-nos de focar num dos pontos fulcrais deste projeto: a combinação de tipos de letra.

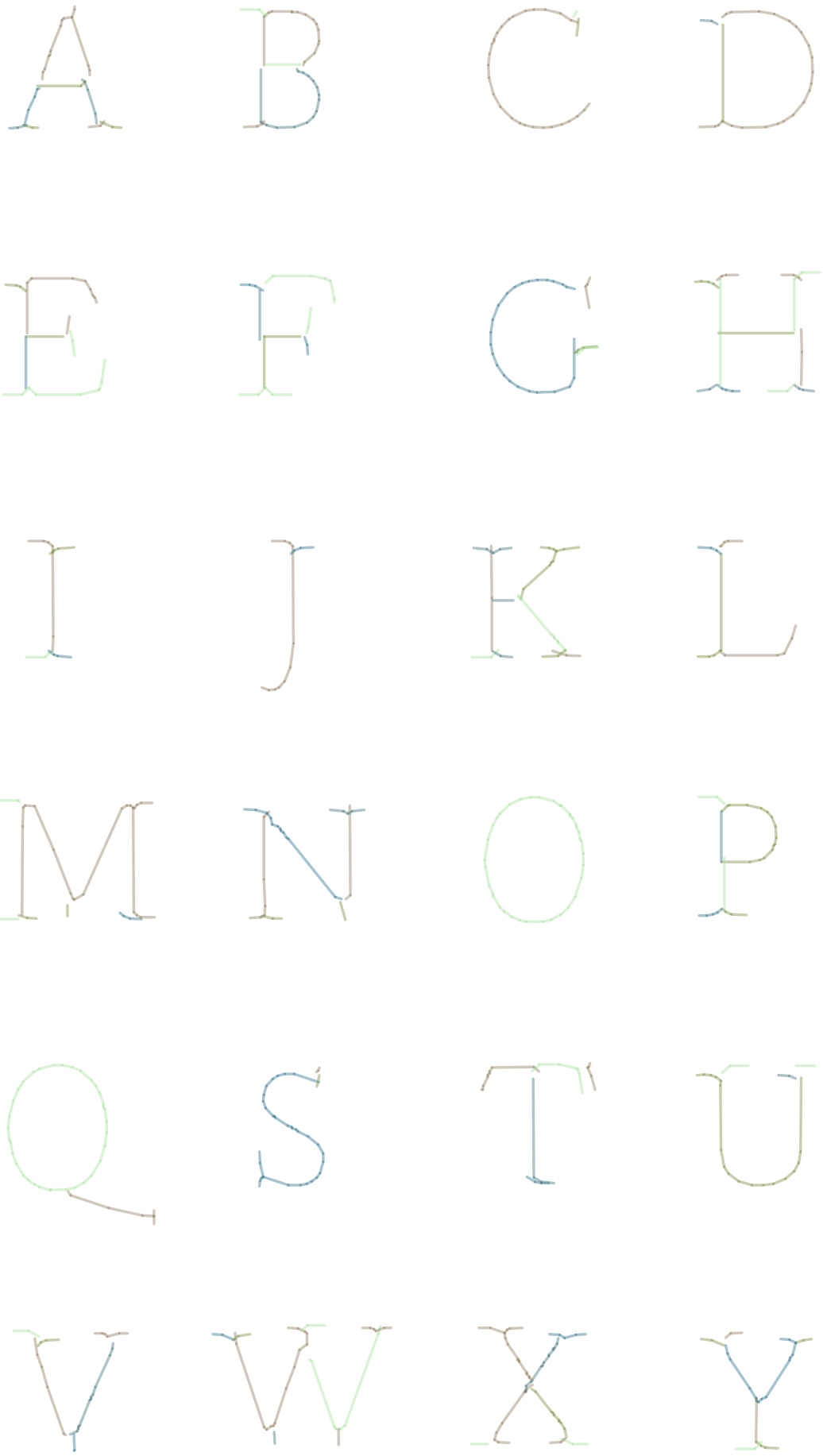
Até agora o nosso sistema gerava esqueletos para qualquer tipo de letra e dividia-os em diversas partes. Nesta seção, apresentamos o método criado para a combinação entre essas partes. Para avaliar se as partes entre diferentes esqueletos eram propícias a serem associadas determinamos a velocidade angular e o ponto central para cada uma delas. Desse modo, cada uma das partes não precisava de ser totalmente igual a outra para ser associada.

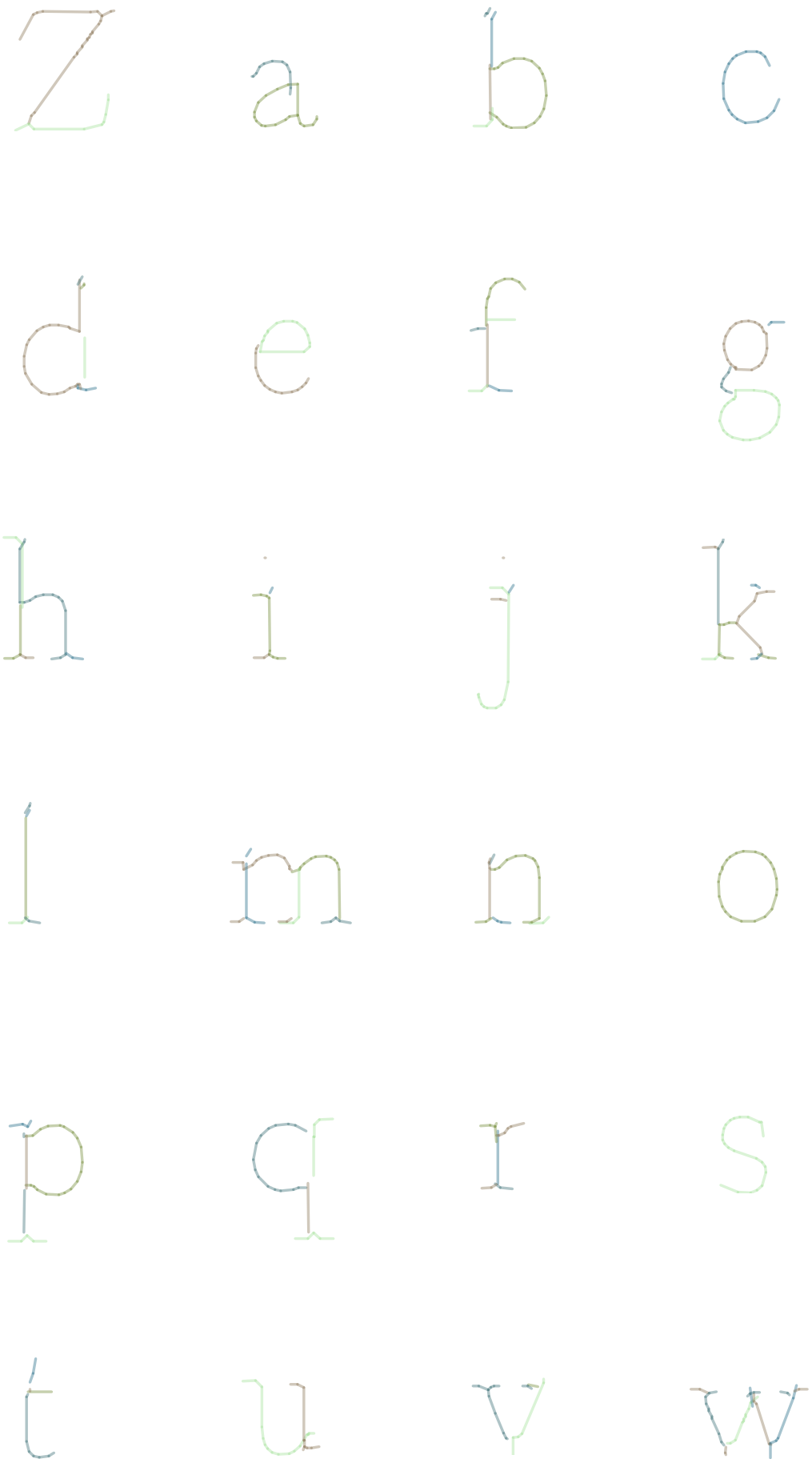
Para contextualizar, a velocidade angular é um vetor que representa o processo da mudança de orientação de uma dada linha. Para uma reta o valor será igual a 0, este vai aumentando consoante esta se torna mais curva. Então, para cada parte foi calculada a velocidade angular e o ponto central. O ponto central é uma média de todos os pontos da parte em questão. Com o cálculo destas duas variáveis podemos identificar uma parte de um caractere.

O processo de associação das diversas partes iniciou-se com a ordenação das partes do primeiro esqueleto, do maior para o menor comprimento. Em princípio, o erro que poderia surgir da combinação seria minimizado.

Depois, cada uma dessas partes foi comparada com todas as partes constituintes do segundo esqueleto e assim sucessivamente. Quando as partes comparadas tinham uma velocidade angular e um ponto central semelhante eram consideradas correspondentes e passávamos para o próximo esqueleto. No final do ciclo tínhamos, para um dado caractere, várias partes anatómicas e cada uma delas tinha várias versões.

De forma a validar a viabilidade das combinações geramos esqueletos combinados para todas as letras e números do alfabeto latino (Fig. 151). Nesta fase as combinações foram geradas de forma aleatória. Ao analisar os resultados reparamos que nenhuma das combinações criadas torna o esqueleto ilegível, apesar de por vezes as linhas não se unirem completamente. Esse fator é devido ao uso de vários tipos de letra com medidas diferentes. No geral, ficamos satisfeitos com o resultado pois respondia aos nossos objetivos para o esqueleto.


A B C D
E F G H
I J K L
M N O P
Q S T U
V W X Y


Z a b c
d e f g
h i j k
l m n o
p q r s
t u v w

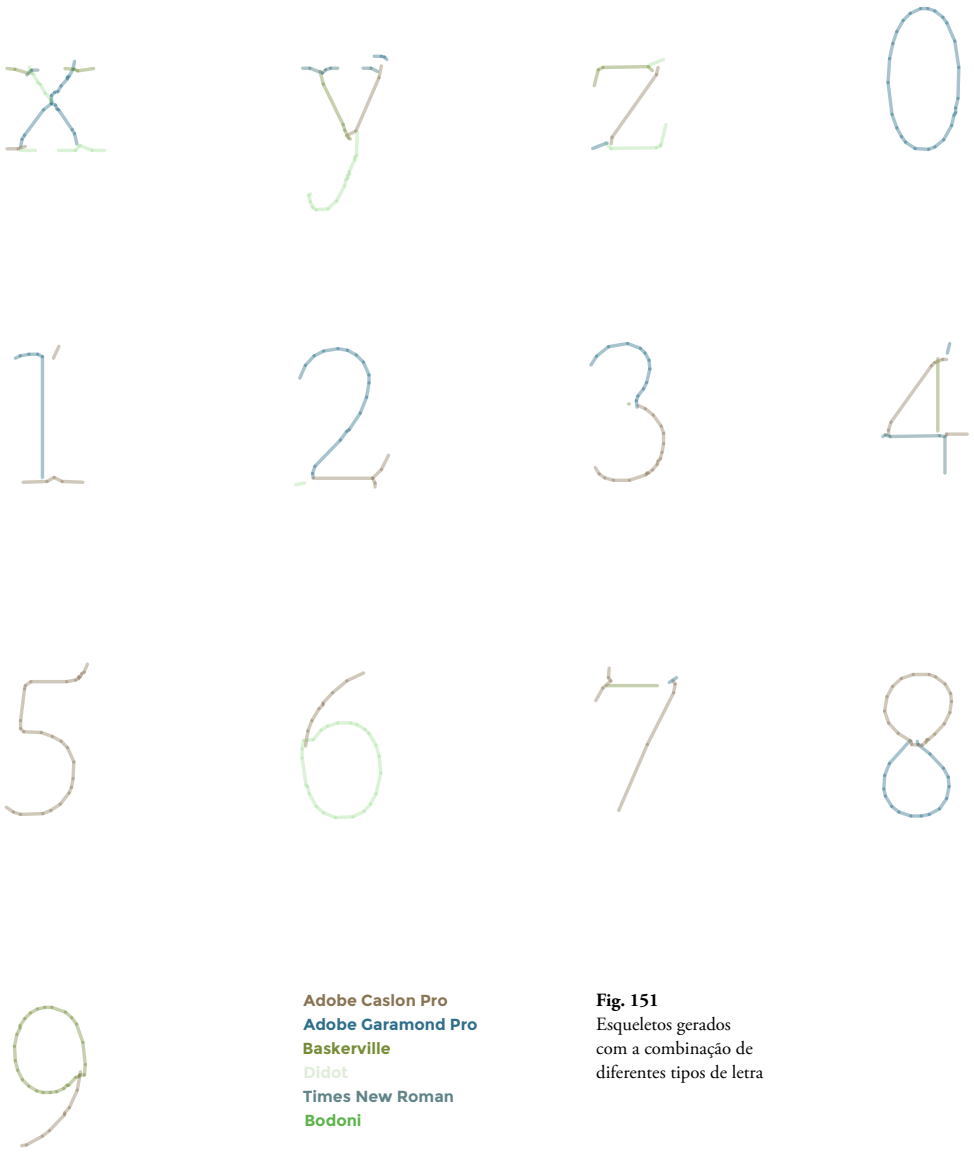


Fig. 151
Esqueletos gerados com a combinação de diferentes tipos de letra

Preenchimento dos glifos

Até agora, tinham sido estabelecidos esqueletos para diferentes tipos de letra. Tais, tinham sido combinados e, como consequência, tínhamos criado estruturas para os tipos de letra que o nosso sistema iria gerar. Determinadas todas estas variáveis faltava criar o preenchimento dos tipos. Anteriormente, nas experimentações iniciais, tentamos extrair partes anatómicas. Mas agora, com o sucesso na extração de esqueletos, já era possível.

Na seção relativa ao último método de extração de esqueletos, tínhamos referido que o processo de extração iniciava-se com a exclusão dos pixels mais afastados do centro da forma. Portanto, ao calcular a distância do *pixel* do background mais próximo até ao *pixel* do esqueleto final determinamos a largura do tipo de letra original (Fig. 152). Com essa medida podemos replicar o tipo, ou aumentar ou diminuir o peso de forma proporcional.

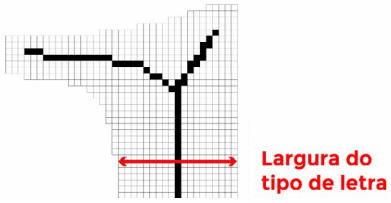


Fig. 152
Determinação da largura de um tipo de letra num determinado ponto através do método de extração do esqueleto.

Para testar a ideia de replicação dos tipos iniciais guardamos a largura de cada tipo de letra, em cada ponto. Fizemos este processo para três tipos de letra e no final combinamos as partes de forma aleatória. Utilizando círculos percorremos as partes do esqueleto final utilizando a largura guardada. Para cada parte, corremos ponto a ponto e aumentamos ou diminuímos gradualmente o raio do círculo desenhado. A Fig. 153 mostra esse preenchimento, e a Fig. 154 apresenta o resultado gerado para alguns caracteres.

Os resultados obtidos eram interessantes, pois, além das irregularidades, era possível observar características dos tipos de letra utilizados. No entanto, como definido anteriormente o nosso objetivo era construir tipos de letra que fossem constituídos por camadas.

Ainda utilizando os círculos, geramos vários caracteres utilizando cores diversificadas e a transparência em nosso proveito. De seguida são apresentados uma série de resultados.

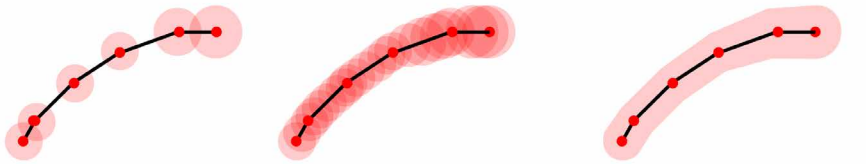


Fig. 153
Método de preenchimento dos pontos intermédios

Fig. 153
Caracteres gerados da combinação de três tipos de letra diferentes. As camadas usadas foram escolhidas aleatoriamente.

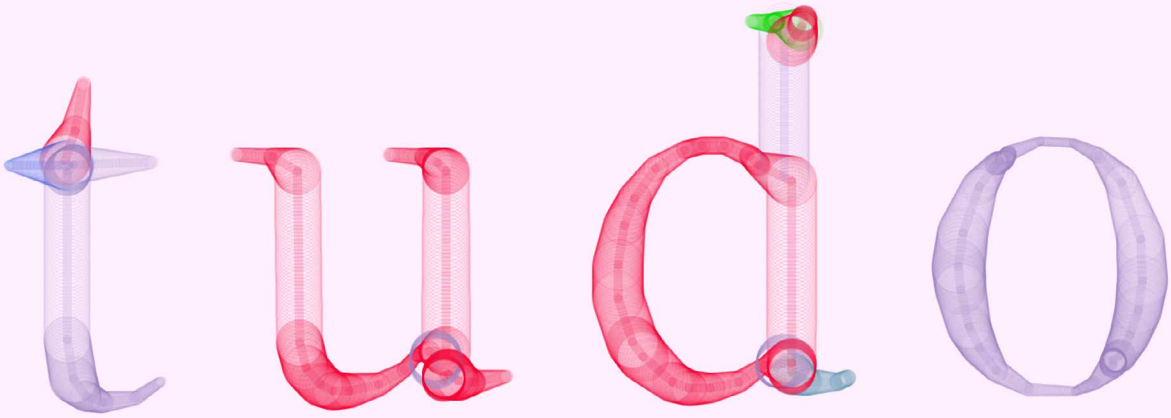
SISTEMA INICIAL

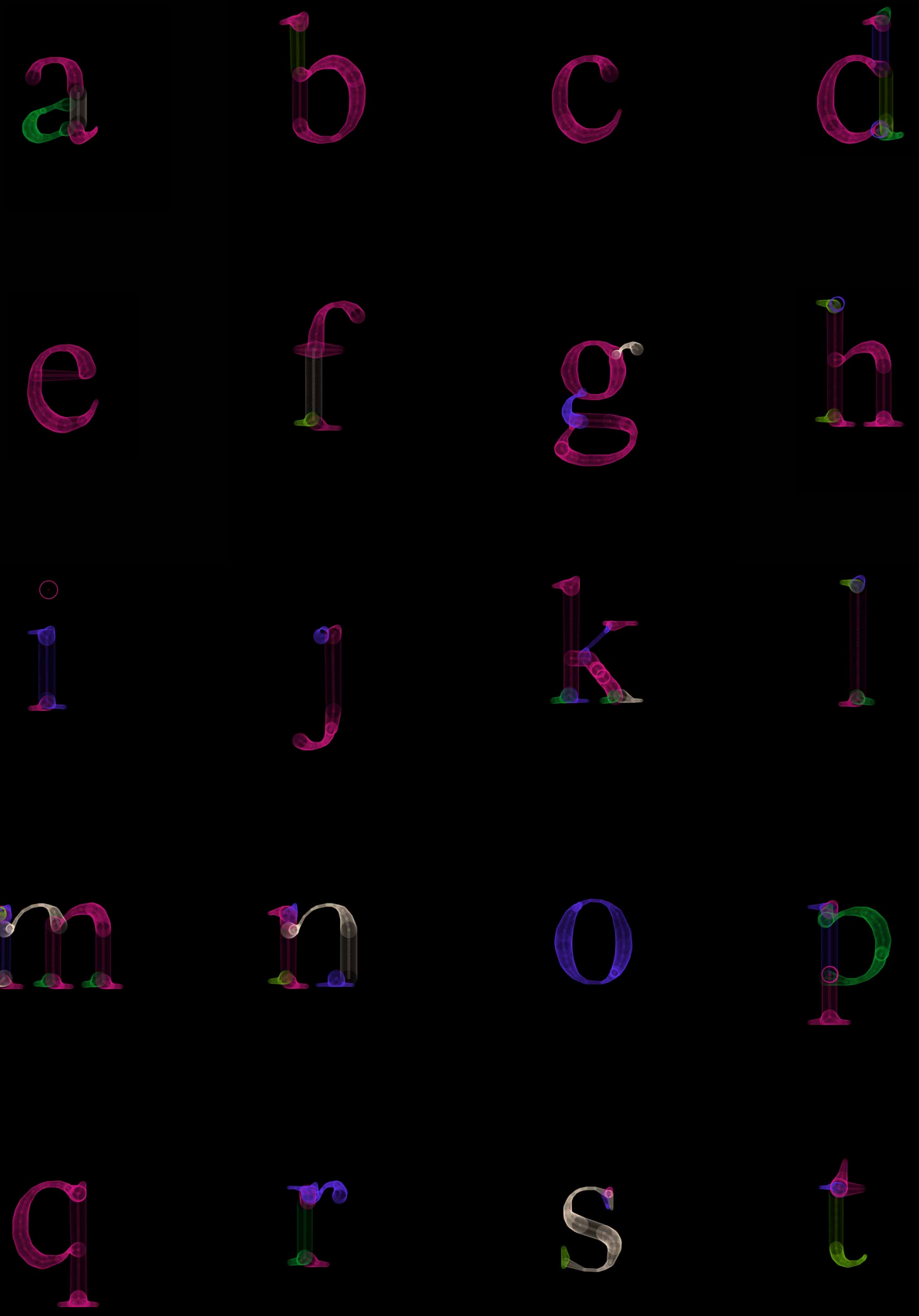
1º Gerações

Neste conjunto de gerações iniciais são utilizados quatro tipos de letra. Em todas as gerações o preenchimento da forma é acompanhado pelo esqueleto, mas nas duas primeiras (Fig. 154 e 155), este último tem alguma transparência. As cores utilizadas foram geradas de forma aleatória, associando uma cor a cada tipo de letra. A utilização de transparências permite uma melhor combinação entre as diferentes camadas.

Fig. 154
Esqueletos gerados com a combinação de quatro tipos de letra diferentes.

Aplicação do alfabeto numa palavra




a b c d
e f g h
i j k l
m n o p
q r s t

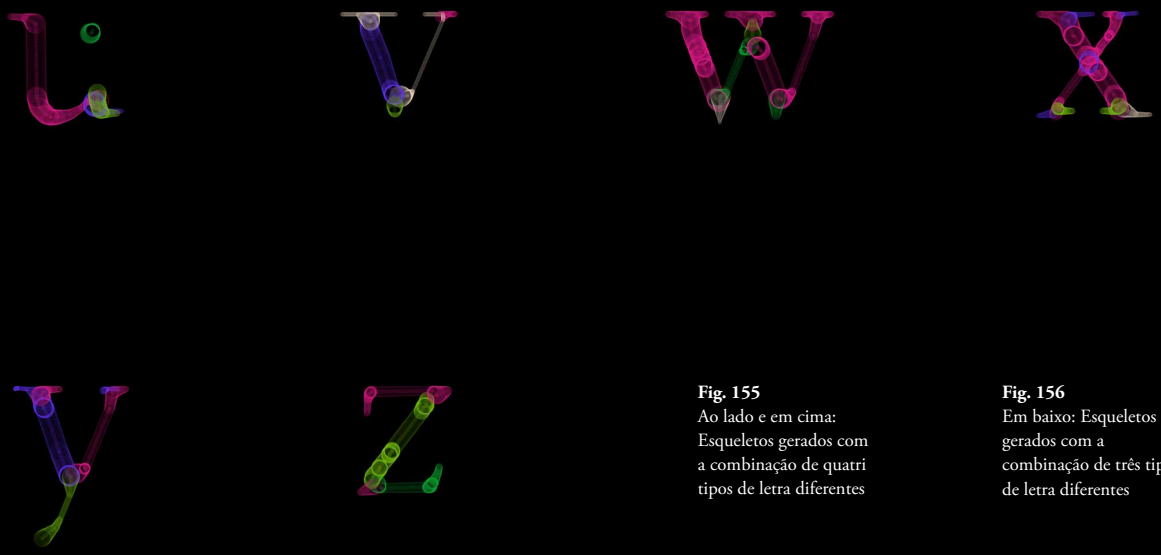


Fig. 155
Ao lado e em cima: Esqueletos gerados com a combinação de quatri tipos de letra diferentes

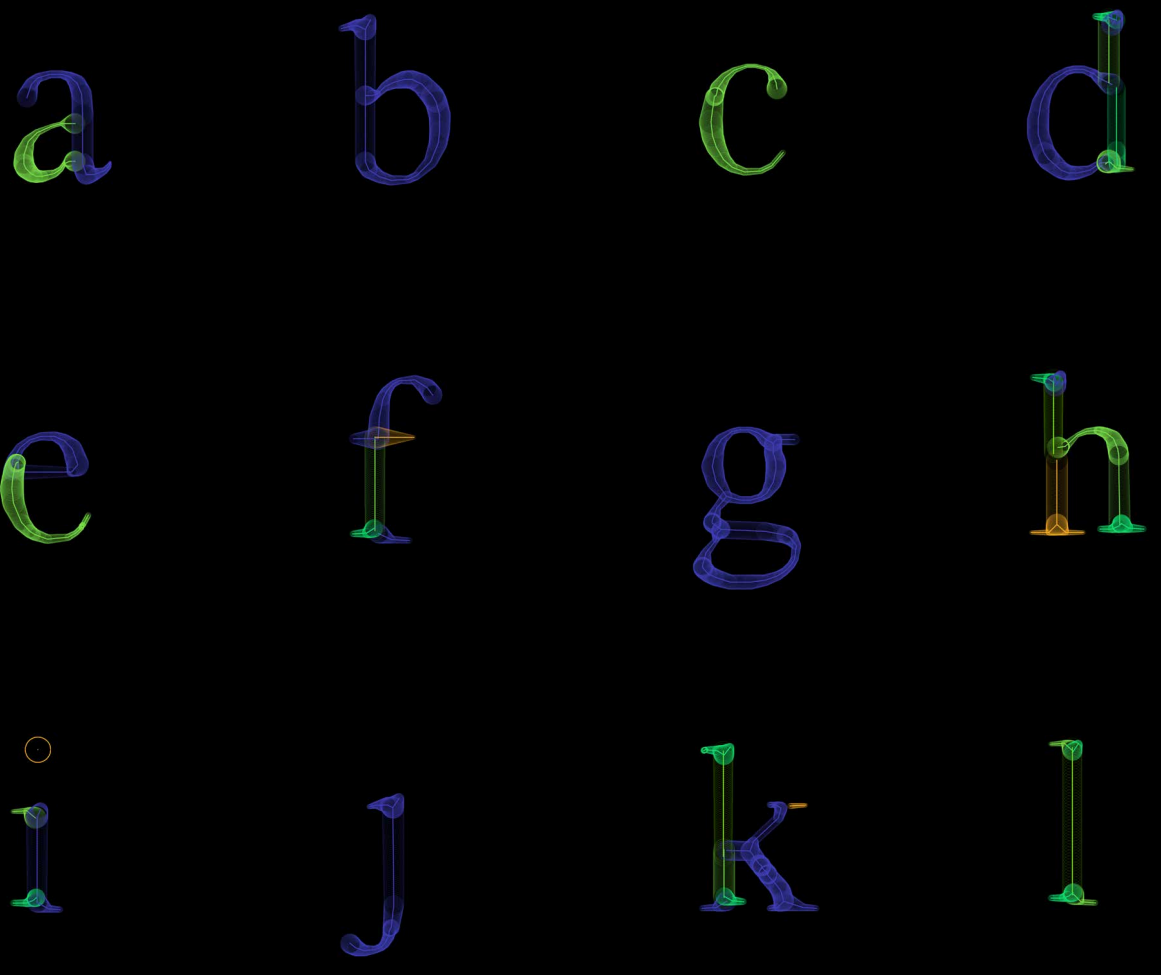


Fig. 156
Em baixo: Esqueletos gerados com a combinação de três tipos de letra diferentes

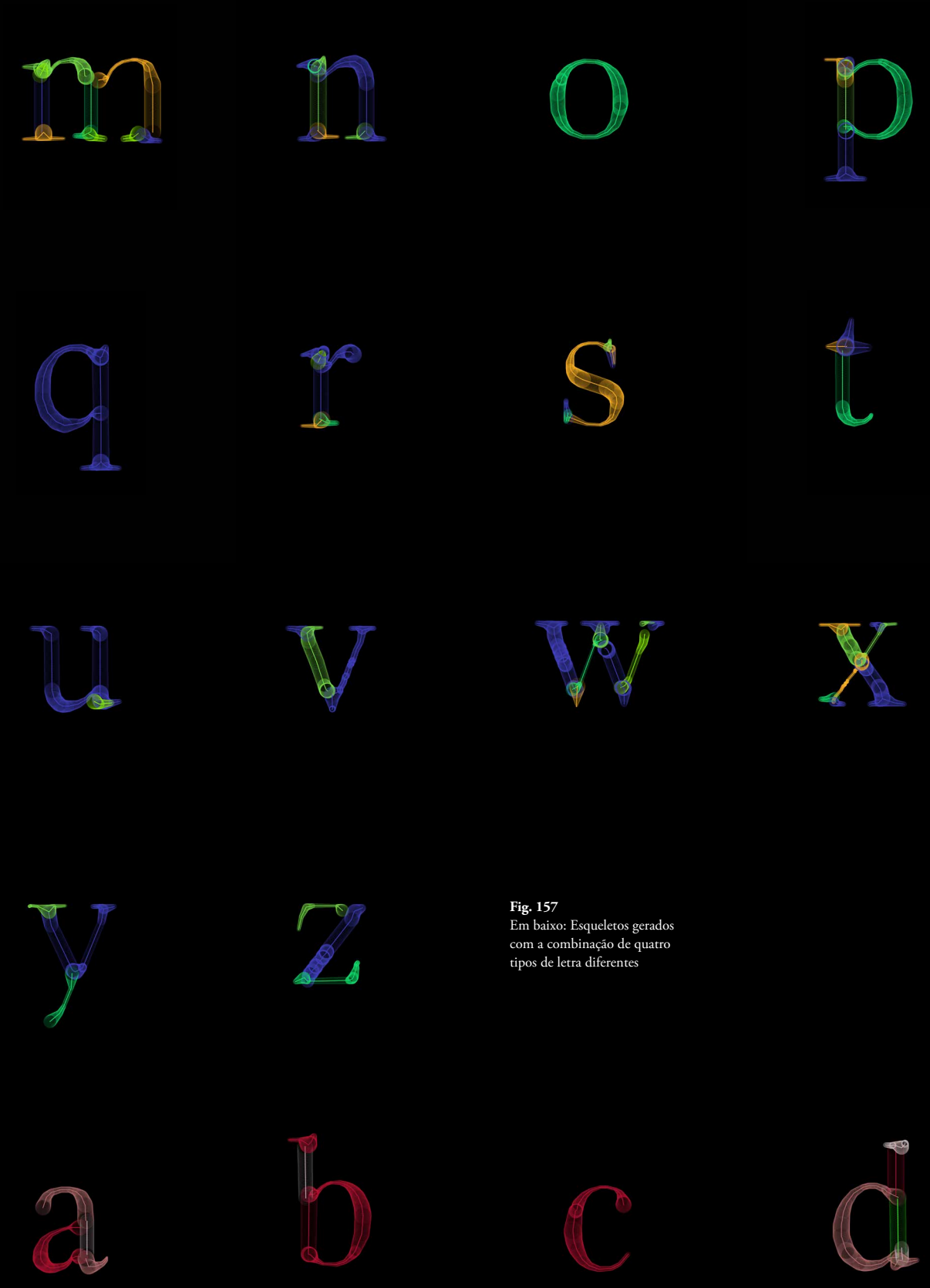
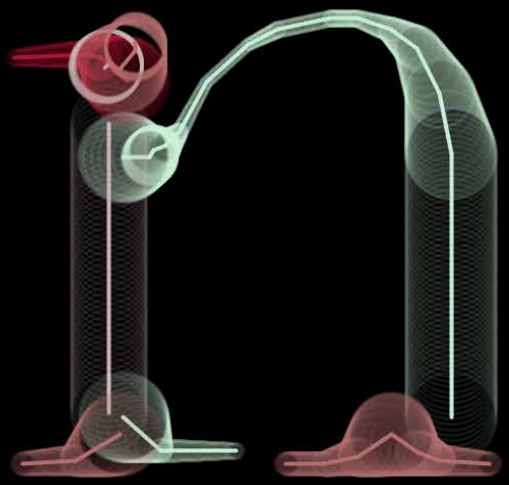


Fig. 157
Em baixo: Esqueletos gerados com a combinação de quatro tipos de letra diferentes


e
f
g
h
i
j
k
l
m
n
p
q
r
s
t
u
v
w
x
y



2º Gerações

Mais tarde, geramos também uma série de glifos excluindo o esqueleto. Uma vez mais, foram utilizados quatro tipos de letra e cores aleatórias. Foi mantida a utilização dos círculos como módulo e a transparência (Fig. 158)

Fig. 158
Esqueletos gerados com a combinação de quatro tipos de letra diferentes, sem esqueleto.

UTILIZAÇÃO DE SÍMBOLOS

1º Geração — mudança nos módulos

No seguimento das experimentações, decidimos utilizar outros símbolos para a geração de glifos. Para estas geração (Fig. 159) optamos por utilizar o quadrado rodado em vez do círculo. Decidimos também alterar o número de módulos desenhados em cada parte anatómica.

Com a diminuição do número de módulos a serem desenhados a variação da densidade entre as partes do glifo tornou-se mais visível. Isto acontecia porque todas as partes eram desenhadas com o mesmo número. Portanto, nas partes mais curtas existia uma área de maior acomulação de cor. Apesar de não ter sido previsto à partida essa variação da densidade tornava-se interessante no ponto que tornava cada parte mais única. Nestas duas gerações, além da troca do módulo, colocamos de parte as transparências e utilizamos só o contorno do elemento replicado. Dessa forma, poderíamos na mesma obter combinações entre partes, como se verificou.















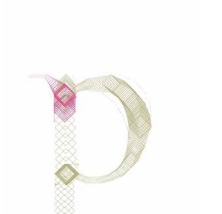





Fig. 159
Esqueletos gerados com a combinação de quatro tipos de letra diferentes, sem esqueleto. Utilização do quadrado rodado como módulo.

2º Gerações — vários módulos

Mais tarde, geramos também glifos com a utilização de diferentes módulos para preenchimento das camadas. Portanto, nas gerações desta seção, atribuímos um módulo para cada tipo de letra (quadrado rodado, triângulo retângulo, círculo e cruz).

Para a primeira geração atribuímos uma cor aleatória, além do módulo para cada tipo de letra (Fig. 160). A utilização de duas variáveis como representação do tipo de letra ajudam à sua diferenciação. Além disso, a combinação dos diferentes módulos apenas com contorno acrescenta detalhes aos glifos obtidos.

Com o intuito de testar o uso de diferentes módulos como distinção dos diferentes tipos de letra geramos a versão presente na Fig. 161. Esta era uma versão da anterior, com os módulos, mas a preto e branco. Apesar de, na nossa opinião, a versão a cores ser mais interessante visualmente, os resultados apresentados na segunda respondem também positivamente aos objetivos desta dissertação.

Fig. 160
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos e diferentes cores para cada tipo de letra.

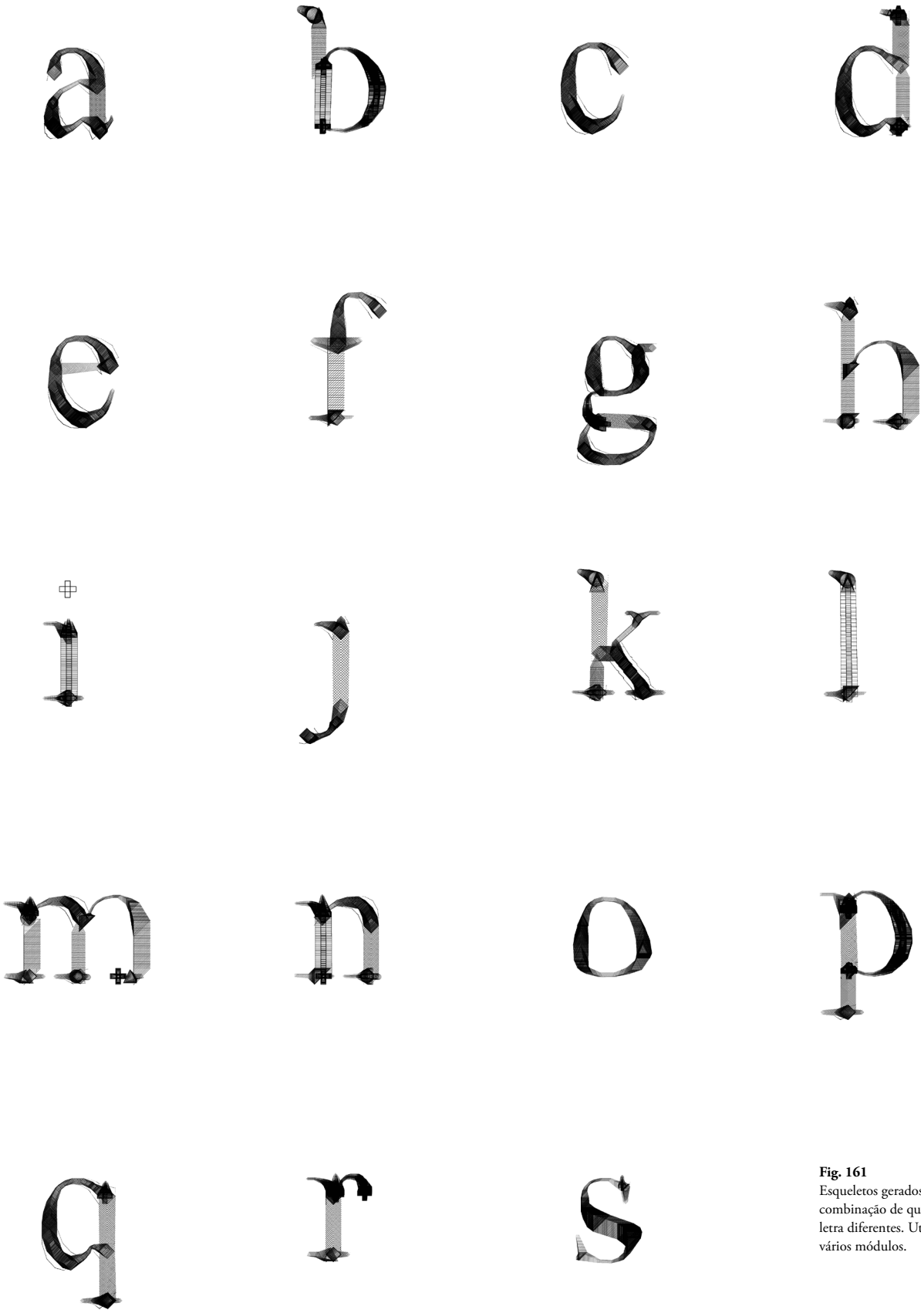


Fig. 161
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos.

UTILIZAÇÃO DE RODAÇÕES

Mais tarde, optamos por uma opção inspirada em técnicas de caligrafia. Para isso decimos aplicar uma rotação aos módulos desenhados. Deste modo, o desenho de cada elemento era rodado consoante a inclinação da linha do esqueleto. As gerações desta seção apresentam vários ângulos de rotação. As Figs. 162, 163 e 164 apresentam gerações com uma alteração mais brusca do ângulo de rotação. Nas Figs. 165, 166, 167 e 168 a curva desenhada é mais suave.

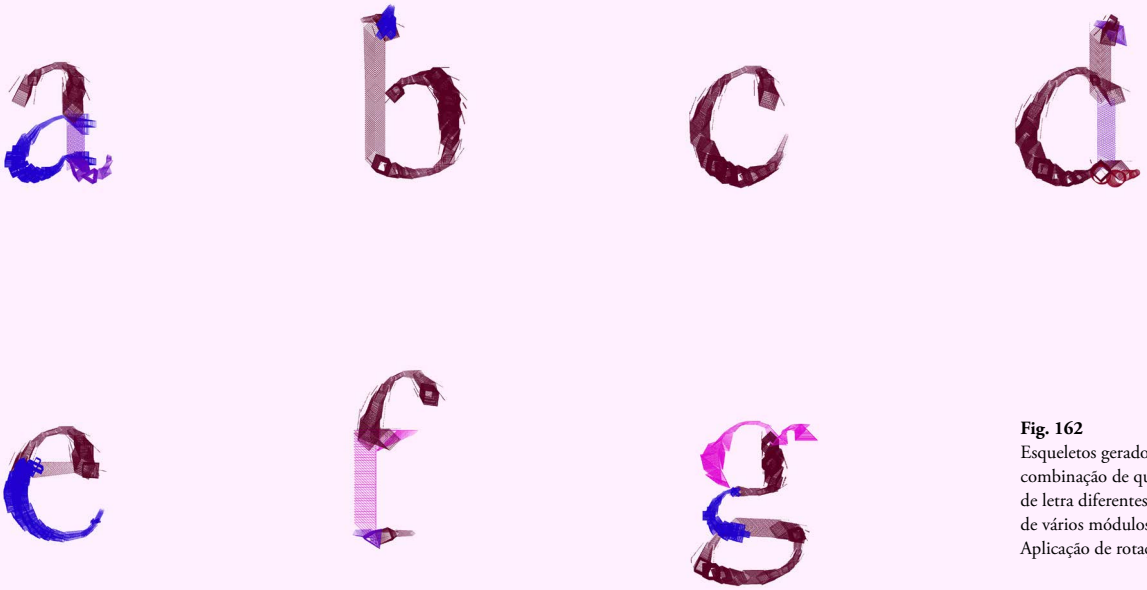


Fig. 162
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos, cores. Aplicação de rotação.

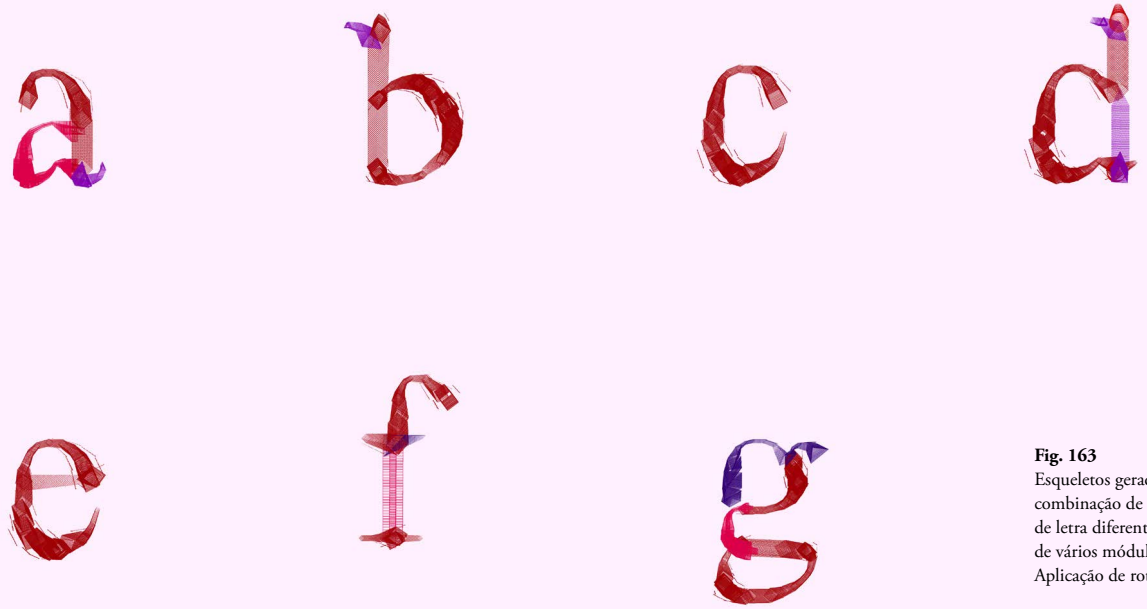


Fig. 163
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos, cores. Aplicação de rotação.

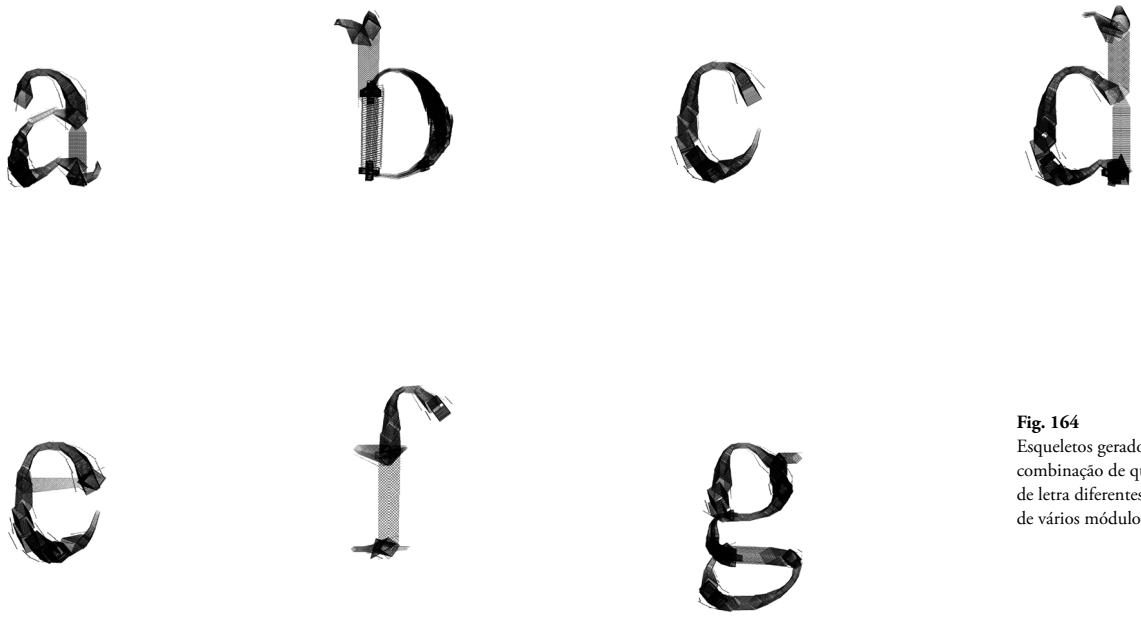


Fig. 164
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos e rotação.

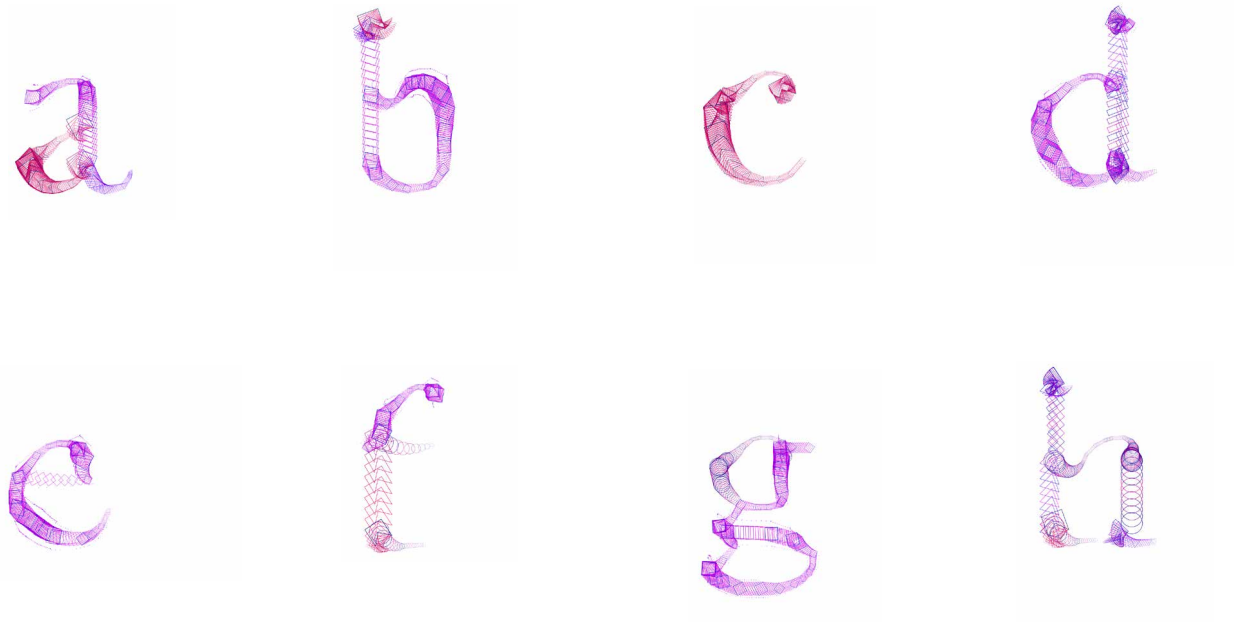


Fig. 165
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos, cores. Aplicação de rotação.

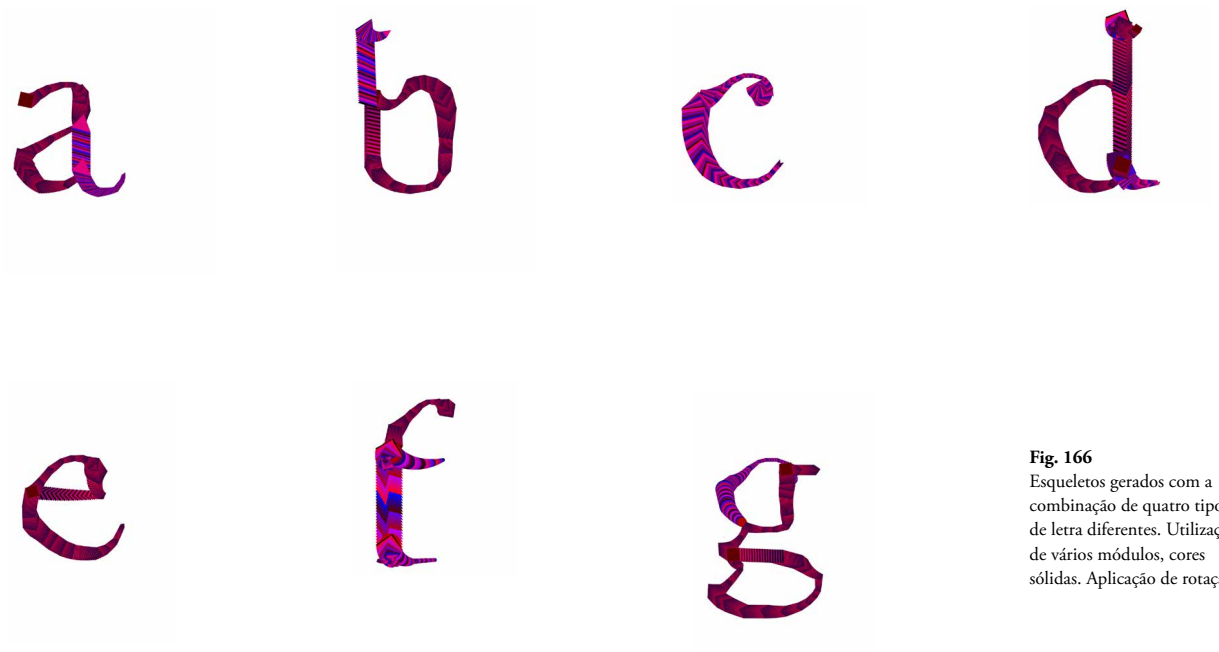


Fig. 166
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos, cores sólidas. Aplicação de rotação.

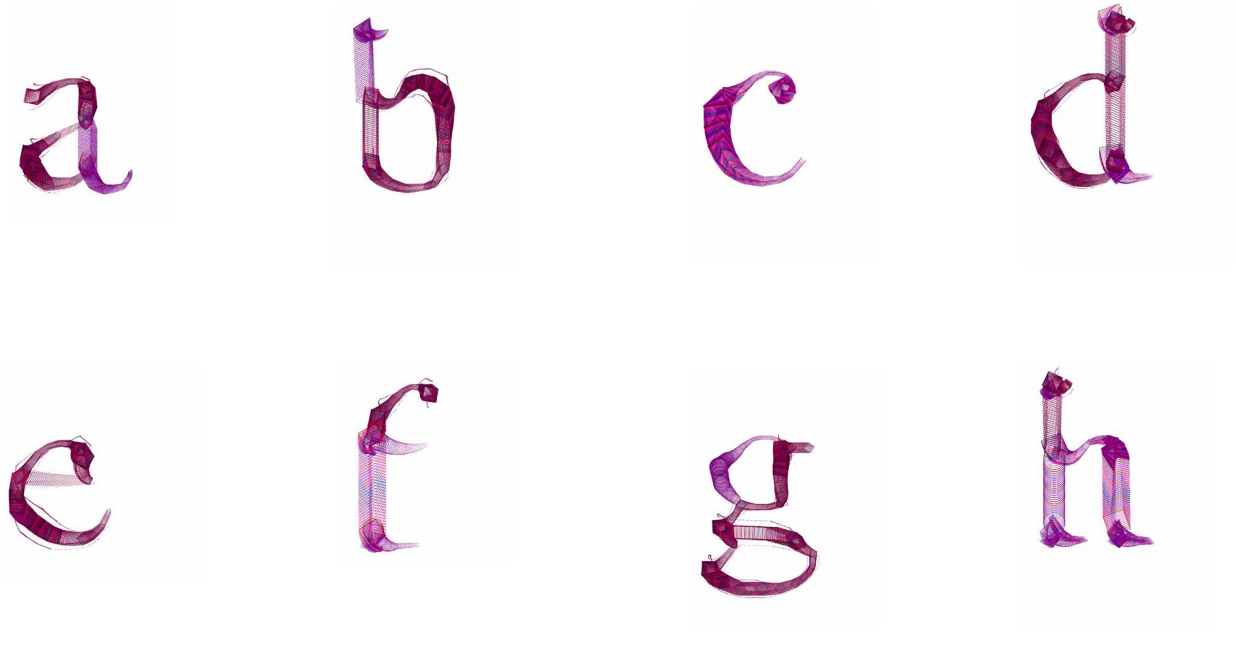


Fig. 167
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos. Aplicação de rotação.

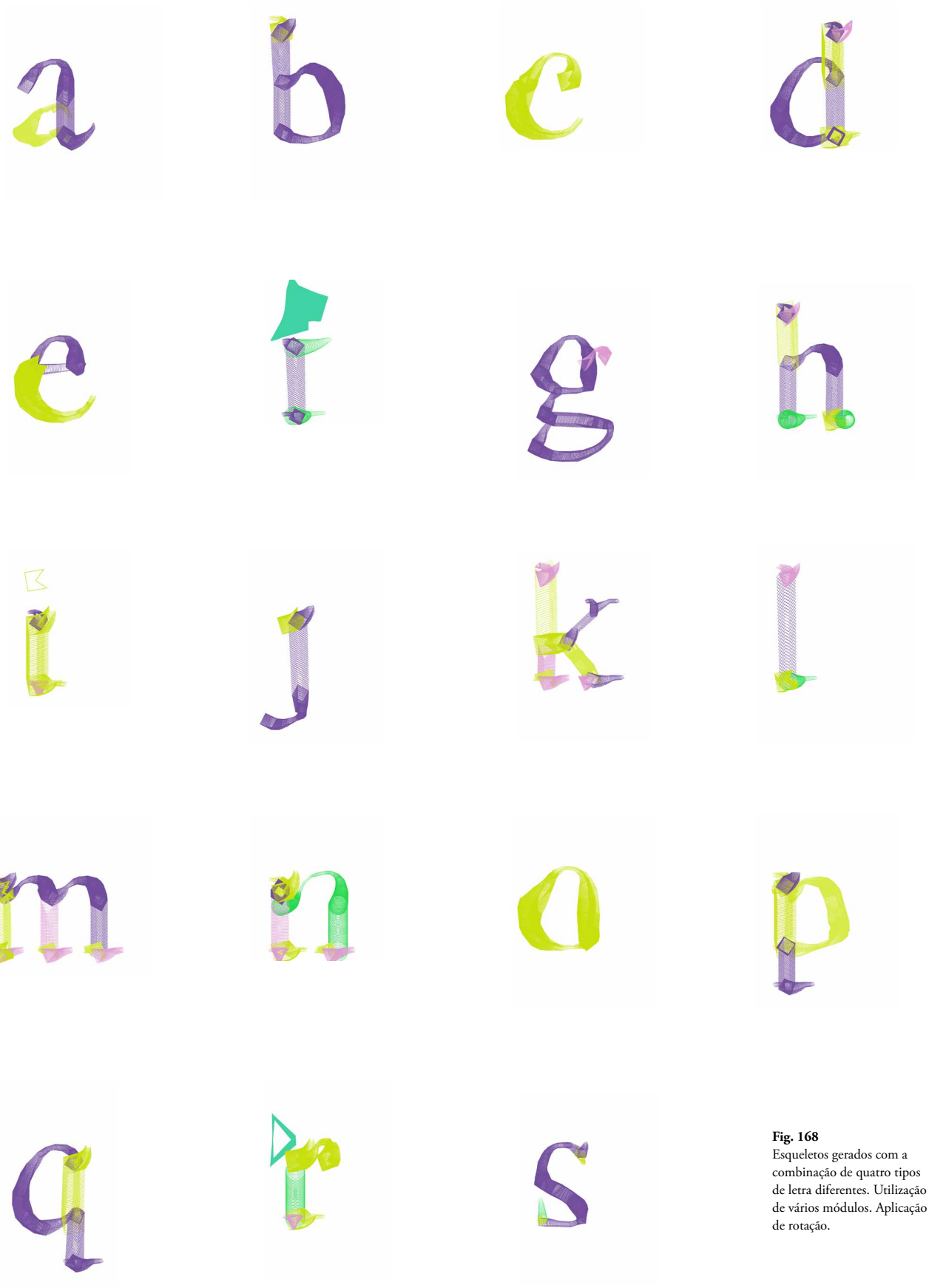


Fig. 168
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos. Aplicação de rotação.

Alteração da largura dos glifos

Mais tarde, decidimos fazer algumas experiências modificando a largura dos glifos. Para a geração da Fig. 168 descartamos a largura do glifo original. Por outras palavras, determinamos uma largura fixa para todos os módulos (partes anatómicas).

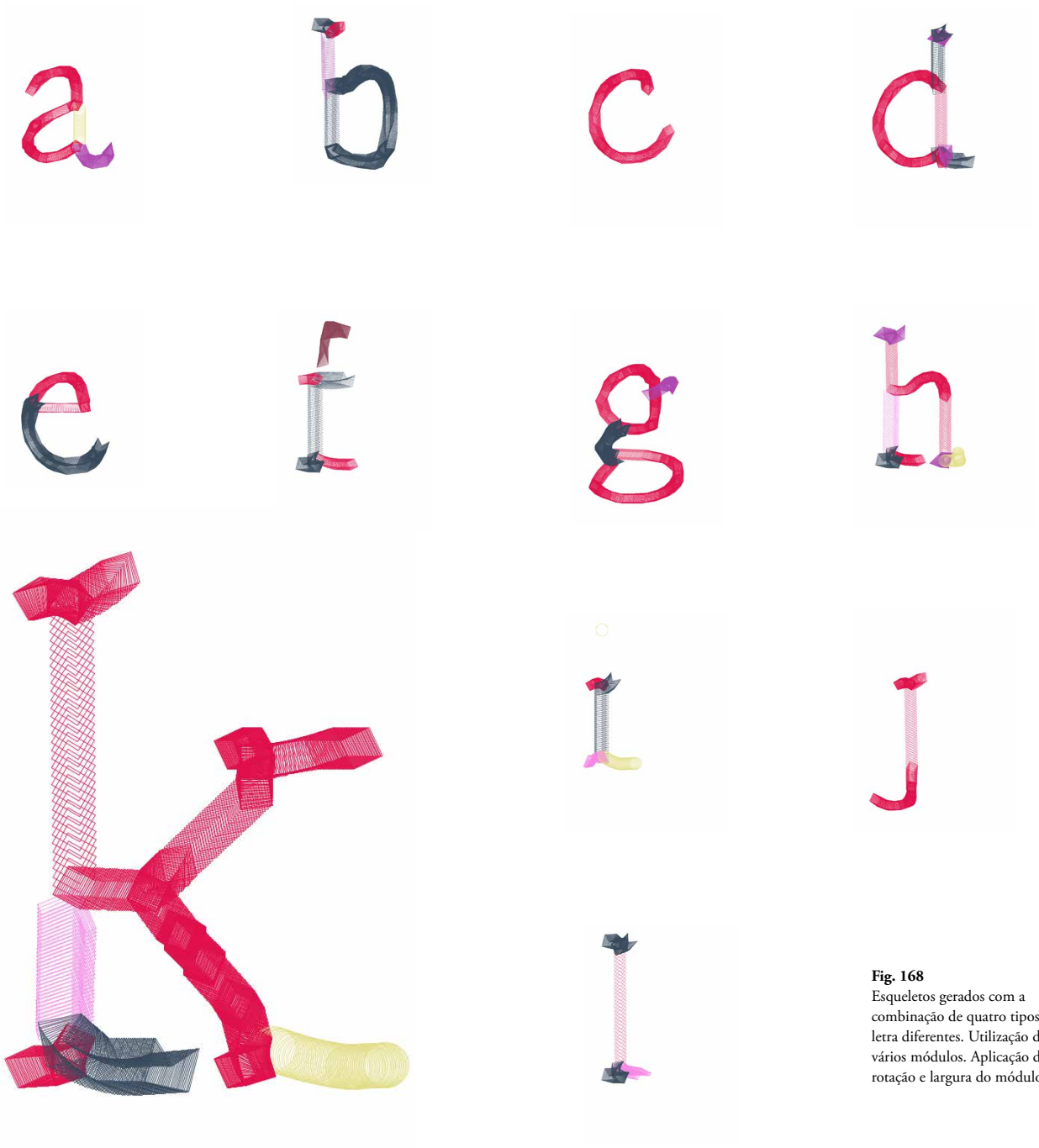


Fig. 168
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos. Aplicação de rotação e largura do módulo fixa.

Por outro lado, testamos também o uso da largura dos tipos de letra iniciais para alteração do peso dos tipos gerados. Nas Fig. 169 é apresentada uma geração de glifos com a diminuição da largura relativamente aos originais, são utilizados módulos com apenas contorno e preenchidos. Testamos também a geração de glifos com um peso superior ao original, a Fig. 170 apresenta os resultados.

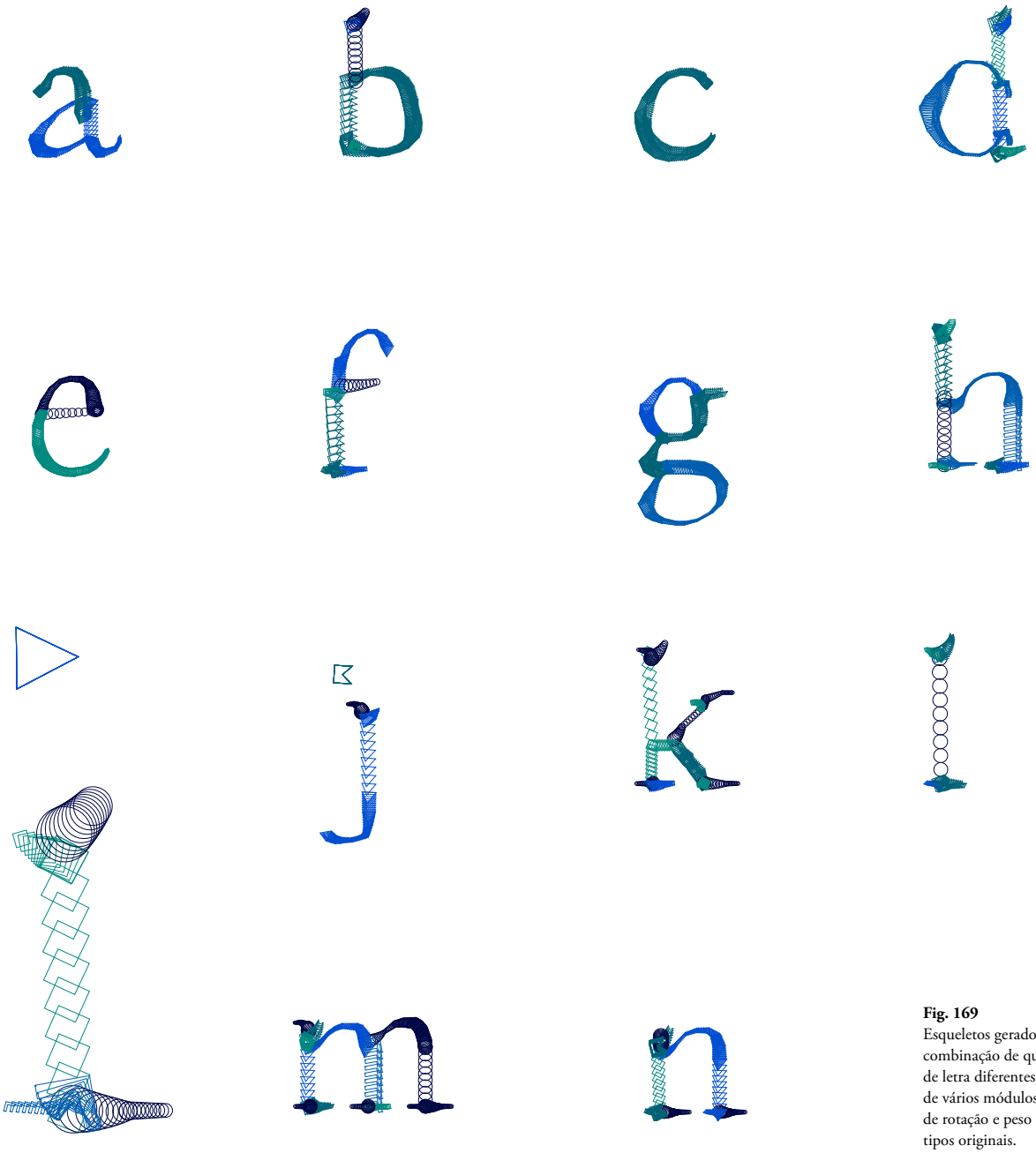
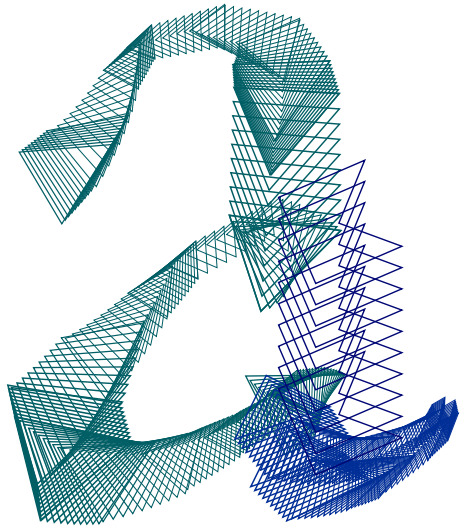
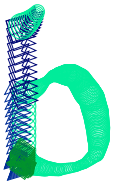
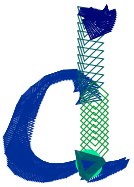
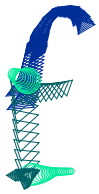
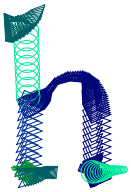
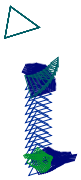
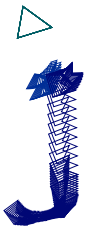


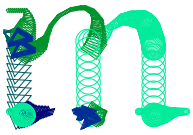
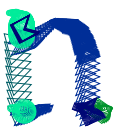
Fig. 169
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos. Aplicação de rotação e peso inferior aos tipos originais.

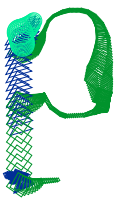
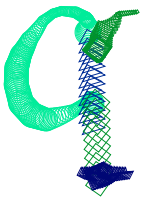







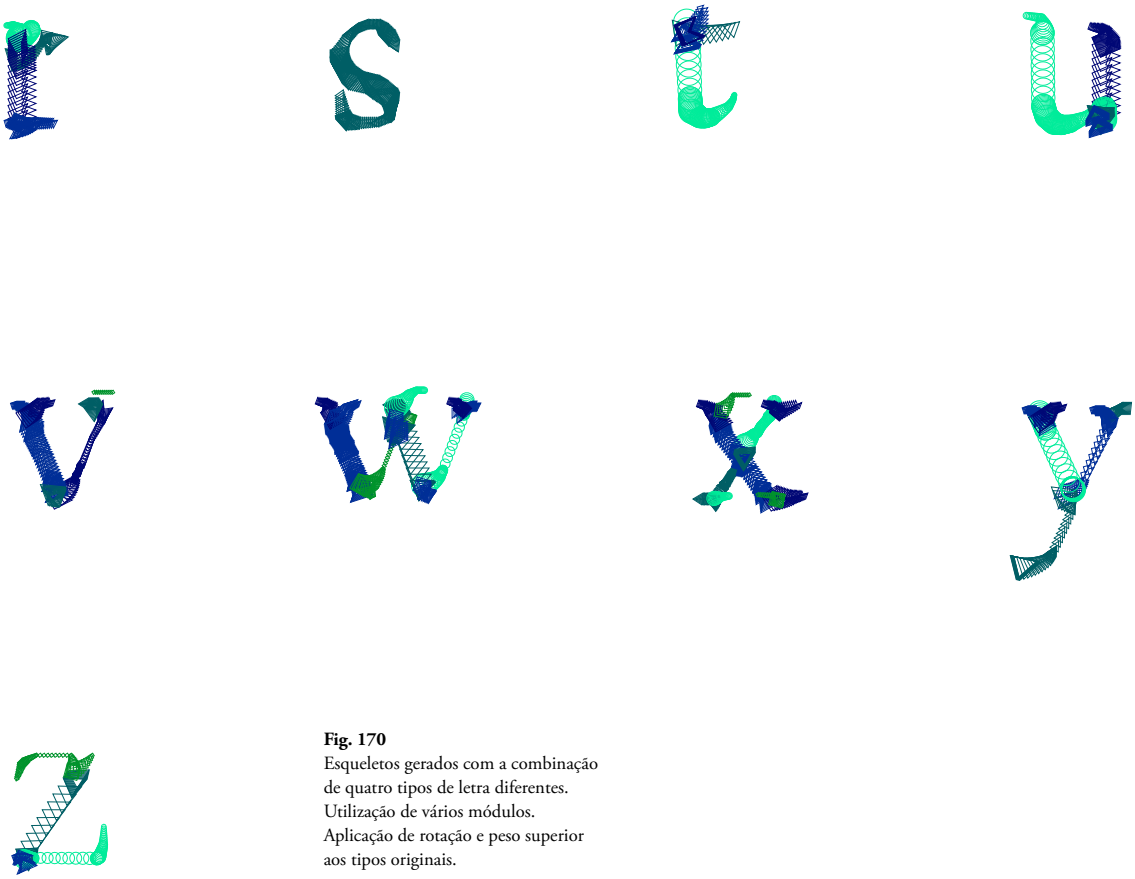


Fig. 170
Esqueletos gerados com a combinação de quatro tipos de letra diferentes. Utilização de vários módulos. Aplicação de rotação e peso superior aos tipos originais.

Depois destas gerações, concluímos que a variação da largura dos tipos poderá ser uma variável para estar à escuta na aplicação deste sistema no mundo real. No geral, todas estas gerações e toda aquelas que podem ser geradas no nosso sistema são possibilidades. Não existe uma versão mais correta do que a outra.

V. CONCLUSÃO

Enquanto designers lidamos continuamente com tipografia. O seu correto uso pode ser utilizado como um acréscimo ao conteúdo que se pretende transmitir. É neste contexto que surge esta dissertação com o objetivo de cruzar a tipografia com as possibilidades que surgiram com a revolução tecnológica.

O sistema que criámos possibilita o desenho de tipos de letra de forma generativa. De modo a criar tipos com qualidade foram revistos os termos relativos à anatomia da letra e à sua classificação, assim como as perspectivas dos especialistas da área. Foram também analisados os movimentos artísticos que influenciaram o desenho de tipos. Relativamente a pormenores técnicos, foram estudados trabalhos e possibilidades já conseguidas no domínio do desenho algorítmico e dinâmico de tipos.

Depois de toda a revisão elaborada no âmbito do desenho generativo de tipos de letra, tinhamos a informação necessária para iniciar o projeto prático desta dissertação.

Primeiramente, foi desenvolvida uma série de abordagens que marcaram o caminho deste projeto. Aspectos que estavam totalmente em aberto, como a criação de tipos a partir de uma grelha ou através de um esqueleto extraído, ficaram decididos nestas primeiras experiências. Graças a elas optamos por criar tipos de letra a partir de um esqueleto. Além disso, abriram também possibilidades em relação à forma como os tipos de letra poderiam ser preenchidos.

Mais tarde, foi também criada uma abordagem com o intuito de visualizar dados num tipo de letra. Uma vez que a estrutura dos esqueletos finais ainda não estava finalizada foi utilizado um sistema mais simples para a criação da forma dos tipos de letra que foi, mais tarde, modificado com dados reais. Foi neste contexto que criámos, como exercício experimental, logotipos para as Faculdades da Universidade de Coimbra. Os resultados gerados apresentavam possibilidades não só no contexto do projeto desta dissertação, mas também na forma como os dados podem influenciar o design de logotipos e de que forma os logotipos podem transmitir informações. Além disso, nesta abordagem intermédia exploramos também o uso de camadas e cores no design de tipos que surgiu recentemente devido a novas possibilidades tecnológicas.

Todas estas abordagens conduziram a novas possibilidades para o desenvolvimento do projeto prático desta dissertação. O caminho que foi tomadofoi, também ele, resultado das pequenas experimentações realizadas ao longo do percurso. Portanto, esta dissertação mais do que o sistema gerado no final, é o caminho percorrido que nos fez chegar a ele e às decisões tomadas para isso.

Graças a essas abordagens optamos pela criação de tipos de letra gerados a partir da sua estrutura e codificação dos diferentes elementos da anatomia da letra em diversas camadas. Posteriormente, os glifos criados foram resultado da combinação de camadas de diferentes tipos de letra e aplicação de um método de preenchimento das mesmas.

O percurso de extração de esqueletos não foi muito fácil e para isso foram desenvolvidos vários métodos como a aplicação de linhas aleatórias e retas normais para o cálculo de pontos médios e sua posterior junção.

O sucesso no sistema de extração de pontos só o teríamos mais tarde com a exploração de algoritmos existentes para a extração de esqueletos. Um deles era o Zhang-Suen Thinning Algorithm que implementamos para a criação da estrutura dos caracteres. Gerados os esqueletos dividimos por partes anatómicas e combinámos diversas partes de tipos de letra diferentes. Os tipos

de letra gerados resultaram dessa combinação e da aplicação de métodos de preenchimento dos glifos.

Neste momento o sistema está preparado para responder a *inputs* que lhe forem associados, pois os glifos por ele gerados têm essa possibilidade. Existe uma série de variáveis como: (i) a largura dos glifos para alterar o peso, (ii) os módulos utilizados (iii) e o número de tipos de letra combinados e as cores a eles associadas, que podem ser adaptadas para modificação dos tipos da mesma forma como os logotipos que geramos para as faculdades da Universidade de Coimbra.

Em retrospectiva, o sistema final responde aos objetivos propostos inicialmente. Os glifos gerados são percetíveis, mas obviamente não são a melhor opção para texto longo devido ao detalhe. No entanto, os tipos de letra gerados podem facilmente ser adaptamos para uma logotipo ou uma identidade dinâmica. De facto existe uma procura emergente por identidades visuais baseadas em formas de letras projetadas especificamente para elas e portanto o nosso sistema vem responder a essas aspirações. Desta forma, somos capazes de criar inúmeras variações para os logotipos do mesmo conceito.

Apesar de o sistema responder aos nossos objetivos iniciais seria interessante melhorar alguns aspetos. Para que os tipos de letra gerados estejam de acordo com as regras tradicionais de desenho de tipos as partes comuns entre caracteres do mesmo tipo teriam de ser iguais (exemplo: o 'p' e o 'q' terem sempre o mesmo descendente). No entanto, no nosso sistema tal ainda não acontece, mas pensamos não ser um problema muito difícil de resolver como trabalho futuro. Além disso, poderíamos também melhorar um dos métodos descartados que era a deteção de serifas e aplicá-lo ao novo sistema. Esta alteração tornaria o sistema mais completo, permitindo ao utilizador escolher a geração de um tipo serifado ou não serifado.

Relativamente ao futuro, seria interessante a projeção do sistema em entidades reais, como a identidade de um museu ou de uma conferência e observar de que forma a sua componente dinâmica se poderia adaptar aos dados recebidos.

De modo o disseminar o trabalho realizado nesta dissertação, foram escritos dois artigos. O primeiro artigo *Data-Driven Logotype Design* (Parente, Martins, Bicker, 2018a) abordava o desenvolvimento de logotipos para as Faculdades da Universidade de Coimbra. Os logotipos tinham a particularidade de se adaptarem às informações relativas aos alunos de cada faculdade. De ano para ano os logotipos modificariam. O artigo foi publicado na 22º Conferência Internacional de Visualização de Informação.

O segundo artigo denominado "Desenho generativo de Tipos de Letra" (Parente, Martins, Bicker, 2018b) foi escrito no contexto do 9º Encontro Nacional de Tipografia que será apresentado em Outubro de 2018 e que tem como tema esta dissertação.

VI. BIBLIOGRAFIA

Amado, P., & Silva, C. (2011). Anatomia Tipográfica in Veloso, A.; Dias, N.; Martins, O.; Amado, P. "II Encontro de Tipografia: Livro de Atas". Aveiro: Edição eletrónica do II Encontro de Tipografia, Departamento de Comunicação e Arte da Universidade de Aveiro.

Analogue76. (2014). Dutch typographies. Retrieved January 18, 2018, from http://analogue76.com/blog/entry/dutch_typographies

Baker, J. (n.d.). Colloquy. Retrieved January 7, 2018, from http://colloquytype.com/typeface/

Bargues, C. (2016). Dada optophonetic. Retrieved January 1, 2018, from http://www.diptyqueparis-memento.com/en/dada-optophonetic/

Borteh, L. (2018). Bauhaus Movement, Artists and Major Works. Retrieved January 18, 2018, from http://www.theartstory.org/movement-bauhaus.htm

Catalogtree. (2007). Monadnock logo. Retrieved January 7, 2018, from http://www.catalogtree.net/projects/monadnock_logo

Chen, A., & Woo, D. (n.d.a). Type Galapagos. Retrieved January 10, 2018, from http://www.typegalapagos.com/

Chen, A., & Woo, D. (n.d.b). Galapagos Evolutionary Type Design. Retrieved January 10, 2018, from http://annhchen.com/following/annhchen.com/Galapagos

Cheng, K. (2006). Designing type. Laurence King Publishing.

Costa, C. (2013). Organizador de tipos de letra. Dissertação de Mestrado. Coimbra: Universidade de Coimbra.

Deck, B. (2011). Template Gothic. 1990. Retrieved December 28, 2017, from https://www.moma.org/collection/works/139319

Devantay, A. (n.d.). No Title. Retrieved December 28, 2017, from http://www.audreydevantay.ch/en/works/regina/

FF3300. (n.d.). Circuito D'Autore. Retrieved January 3, 2018, from http://www.ff3300.com/en/progetti/circuito-d-autore

FF3300. (2012). Imaginifica. Retrieved January 3, 2018, from https://www.slideshare.net/ff3300/imaginifica

FF3300. (2011). Imaginifica. Retrieved January 3, 2018, from http://www.ff3300.com/en/progetti/imaginifica

Flask, D. (n.d.a). Emigre. Retrieved January 18, 2018, from http://www.designishistory.com/1980/emigre/

Flask, D. (n.d.b). David Carson. Retrieved January 1, 2018, from http://www.designishistory.com/1980/david-carson/

Flask, D. (n.d.c). Ed Fella. Retrieved January 1, 2018, from http://www.designishistory.com/1980/ed-fella/

Flask, D. (n.d.d). Design After Modernism. Retrieved January 1, 2018, from http://www.designishistory.com/1980/new-ideas-/

Flask, D. (n.d.e). Wim Crouwel. Retrieved January 1, 2018, from http://www.designishistory.com/1960/wim-crouwel/

Flask, D. (n.d.f). Dada. Retrieved January 1, 2018, from http://www.designishistory.com/1850/dada/

Flueckiger, M., & Kunz, N. (2009). LAIKA. Retrieved January 3, 2018, from https://vimeo.com/6993808

Flueckiger, M., & Kunz, N. (n.d.). LAIKA – a dynamic typeface. Retrieved January 3, 2018, from http://laikafont.ch/

FontFont. (n.d.a). FF ThreeSix. Retrieved January 18, 2018, from https://www.myfonts.com/fonts/fontfont/threesix/

FontFont. (n.d.b). FF Beowolf. Retrieved December 30, 2017, from https://www.fontshop.com/families/ff-beowolf

FontFont. (n.d.c). FF Fudoni. Retrieved January 1, 2018, from https://www.fontfont.com/fonts/fudoni

Friends, L. A. M. (2011). Modular Type from Yannick Mathey. Retrieved December 30, 2017, from http://www.lettersaremyfriends.com/modular-type-from-yannick-mathey/

Gosling, E. (2015). Muir McNeil's typography-led LCC Summer Shows identity. Retrieved January 18, 2018, from https://www.itsnicethat.com/articles/muir-mcneil-lcc

Heitlinger, P. (n.d.). No Title. Retrieved January 18, 2018, from http://www.tipografos.net/designers/johnston.html

Henestrosa, C., Meseguer, L., & Scaglione, J. (2015). Cómo crear tipografías: del boceto a la pantalla. Tipo e Editorial.

Hewitt, C. (n.d.). Objects that Edward Fella and Detroit Focus Gallery both worked on. Retrieved January 18, 2018, from https://collection.cooperhewitt.org/people/18041625/collaborators/18054865/

Huang, M. (2011). Bold, Italic, Emphatic—Possibilities for Interactive Type.

Hustwit, G. (2007). Helvetica. USA.

Jung, I.-H. (n.d.). PDE_Lubanoise. Retrieved January 1, 2018, from http://www.il-ho.com/#/design/PDELUBANOISE

Klein, D. (n.d.). Denis Klein - communication design. Retrieved January 2, 2018, from http://denis-klein.de/
Knuth, D. E. (1986). The METAFONTbook, Computers & Typesetting-C. MA: Addison-Wesley.

Kupferschmid, I. (n.d.). Type classifications are useful, but the common ones are not. Retrieved January 2, 2018, from http://kupferschrift.de/cms/2012/03/on-classifications/

Lehni, J. (2011). Typeface As Programme. Retrieved November 15, 2017, from https://www.typotheque.com/articles/typeface_as_programme

Levin, G., Feinberg, J., & Curtis, C. (2001). Alphabet Synthesis Machine. Retrieved January 13, 2018, from http://flong.com/projects/alphabet/

Lupton, E. (2006). Pensar com tipos: guia para designers, escritores, editores e estudantes.

Mainz, G. M. (n.d.a). Gestalten mit Code, FH Mainz. Retrieved January 2, 2018, from http://generative-typografie.de/generativetypografie/bastard/

Mainz, G. M. (n.d.b). Gestalten mit Code, FH Mainz. Retrieved January 2, 2018, from http://generative-typografie.de/generativetypografie/fontmix/

Mainz, G. M. (n.d.c). Gestalten mit Code, FH Mainz. Retrieved January 2, 2018, from http://generative-typografie.de/generativetypografie/elien/

Mainz, G. M. (n.d.d). Gestalten mit Code, FH Mainz. Retrieved January 3, 2018, from http://generative-typografie.de/generativetypografie/pong/

Mainz, G. M. (n.d.e). Gestalten mit Code, FH Mainz. Retrieved January 3, 2018, from http://generative-typografie.de/generativetypografie/zwirn/

Mainz, G. M. (n.d.f). Gestalten mit Code, FH Mainz. Retrieved January 3, 2018, from http://generative-typografie.de/generativetypografie/broken-grid/

Mainz, G. M. (n.d.g). Gestalten mit Code, FH Mainz. Retrieved January 3, 2018, from http://generative-typografie.de/generativetypografie/irratio/

Mainz, G. M. (n.d.h). Gestalten mit Code, FH Mainz. Retrieved January 3, 2018, from http://generative-typografie.de/generativetypografie/blast/

Mainz, G. M. (n.d.i). Gestalten mit Code, FH Mainz. Retrieved January 3, 2018, from http://generative-typografie.de/generativetypografie/pde-lubanoise/

Makela, P. S. (2011). Dead History. 1990. Retrieved December 28, 2017, from https://www.moma.org/collection/works/139317

Marxer, R. (n.d.). Geomerative. Retrieved January 4, 2018, from http://www.ricardmarxer.com/geomerative/

McGraw, G., & Rehling, J. (n.d.). The Letter Spirit project. Retrieved January 1, 2018, from http://goosie.cogsci.indiana.edu/farg/mcgrawg/lspirit.html

MuirMcNeil. (n.d.a). About TwoPlus . Retrieved December 31, 2017, from M. (n.d.). About TwoPlus . Retrieved December 31, 2017, from http://www.muirmcneil. com/project/threesix-3/

MuirMcNeil. (n.d.b). Intersect. Retrieved January 2, 2018, from http://www.muirmcneil.com/project/intersect-2/?section=about

MuirMcNeil. (n.d.c). ThreeSix. Retrieved January 3, 2018, from http://www.muirmcneil.com/project/threesix-3/

Nascimento, G. (2011). Prototype-0. Retrieved December 30, 2017, from https://labvis.eba.ufrj.br/prototype-o/

Parente, J., Martins, T., & Bicker, J. (2018a). Data-driven Logotype Design. In 22 International Conference Information Visualisation (IV2018). IEEE.

Parente, J., Martins, T., & Bicker, J. (2018b). Generative Type Design. In 9 Encontro Tipografia (9ET). Instituto Politécnico de Tomar. (por publicar)

Puckey, J. (2005). Typographic Rhythm. Retrieved January 7, 2018, from https://jonathanpuckey.com/projects/typographic-rhythm/

R. I. T. Rochester Institute of Technology. (2010). No Title. Retrieved December 30, 2017, from http://library.rit.edu/cary/les-mots-en-liberté-futuristes-futurist- words-freedom

Reigel, A., & Müller, M. (2012). Metaflop. Retrieved January 3, 2018, from https://www.metaflop.com/modulator

Saibadesign. (2010). O design rock n' roll da revista Ray Gun. Retrieved January 18, 2018, from https://saibadesign.wordpress.com/2010/08/22/o-design-rock-n'-roll-da-revista-ray-gun/

Samara, T. (2004). Typography workbook: A real-world guide to using type in graphic design. Rockport Publishers.

Scikit-image. (n.d.). Skeletonize. Retrieved from http://scikit-image.org/docs/dev/auto_examples/edges/plot_skeleton.html

Schaefer, S., & Yuksel, C. (2007). Example-based Skeleton Extraction. In Proceedings of the Fifth Eurographics Symposium on Geometry Processing (pp. 153–162). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. Retrieved from http://dl.acm.org/citation.cfm?id=1281991.1282013

Shaughnessy, A., & Bierut, M. (2009). Graphic design: a user's manual. Laurence King.

Silanteva, D. (2011). Typographic Music. Retrieved December 31, 2017, from http://www.ddina.com/index.php?/2011/typographic-music/2/

Sullivan, M. (2011). Dina Silanteva — Typographic Music. Retrieved December 31, 2018, from http://www.typetoken.net/publication/dina-silanteva-—-typographic-music/

Tatssachen. (2010). Elien – generative typeface. Retrieved January 2, 2018, from http://www.tatssachen.de/%0A68%0Aportfolio/elien/

Tentrotterdam. (2008). An Evening with Monadnock & Air. Retrieved January 10, 2018, from http://www.tentrotterdam.nl/en/event/luna_piena_monadnock_air-2/

Tierny, J., Vandeborre, J.-P., & Daoudi, M. (2006). 3D Mesh Skeleton Extraction Using Topological and Geometrical Analyses. In 14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006) (p. s1poster). Tapei, Taiwan. Retrieved from https://hal.archives-ouvertes.fr/hal-00725576

Tierny, J., Vandeborre, J., & Daoudi, M. (2008). Fast and precise kinematic skeleton extraction of 3D dynamic meshes. In 2008 19th International Conference on Pattern Recognition (pp. 1–4). https://doi.org/10.1109/ICPR.2008.4761011

Tools, D. (2007). Post Bitmap Scripter. Retrieved January 5, 2018, from http://www.digital-tools-blog.com/blog/44-post-bitmap-scripter

Tracy, W. (n.d.). Type Design Classification. Retrieved December 18, 2017, from http://visiblelanguagejournal.com/issue/17/article/98

Tracy, W. (2003). Letters of credit: a view of type design. David R. Godine Publisher.

Westgate, A. (2007). An Interview with David Carson. Retrieved January 1, 2018, from https://layersmagazine.com/an-interview-with-david-carson.html

Willen, B., & Strals, N. (2009). Lettering & type: creating letters and designing typefaces. Princeton Architectural Press.

Womack, D. (2011). Historical digital. Retrieved January 7, 2018, from http://www.eyemagazine.com/blog/post/historical-digital1

Wu, S.-., & Marquez, M. R. G. (2003). A non-self-intersection Douglas-Peucker algorithm. In 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003) (pp. 60–66). https://doi.org/10.1109/SIBGRA.2003.1240992

GenoTyp. (n.d.). Retrieved January 5, 2018, from https://interaktivegestaltung.net/genotyp/

Digital Applications Change Editorial Design. (n.d.). Retrieved January 18, 2018, from http://www.csun.edu/~pjd77408/DrD/Art461/LecturesAll/Lectures/PublicationDesign/DigitalTimes/David-Carson.html

The Foundry. (n.d.). Retrieved January 18, 2018, from http://www.foundrytypes.co.uk/the-foundry-typefaces/experimental/params/new-alphabet/opentype/level-1/new-alphabet-1

Typefaces. (n.d.). Retrieved January 17, 2018, from http://luc.devroye.org/fonts-58232.html

Collection. (n.d.). Retrieved January 17, 2018, from https://www.artgallery.nsw.gov.au/collection/works/268.1984/