UNIVERSITÉ CADI AYYAD
FACULTÉ DES SCIENCES
SEMLALIA - MARRAKECH

N° d'ordre : 277

****

THÈSE

présentée à la Faculté pour obtenir le grade de :

Docteur

UFR : Mathématiques fondamentales

Spécialité : Mathématiques et Informatique

Représentation formelle pour la réalisation d'une fonte cursive dynamique conforme aux règles de la calligraphie arabe

par :

Abdelouahad BAYAR

(DESS : Informatique)

Soutenue le 16 novembre 2009 devant la commission d'examen :

Président :	Abdelaziz NASROALLAH	PES	FSSM, Université Cadi Ayyad
Examinateurs :	Mostafa BANOUNI	PES	FSA, Université Ibn Zohr
	Mohamed LAGHCHIM LAHLOU	PES	FSSM, Université Cadi Ayyad
	Azzeddine LAZREK	PES	FSSM, Université Cadi Ayyad
Encadrant :	Khalid SAMI	PES	FSSM, Université Cadi Ayyad

Fiche Présentative de la thèse

– Nom et prénom de l'auteur : *BAYAR Abdelouahad*
– Intitulé du travail : *Représentation formelle pour la réalisation d'une fonte cursive dynamique conforme aux règles de la calligraphie arabe*
– Encadrant :
  * Nom, prénom et grade : *SAMI Khalid, Professeur de l'enseignement supérieur*
  * UFR : *Mathématiques Fondamentales*
  * Département : *Mathématiques*
  * Institution : *Université Cadi Ayyad - Faculté des Sciences Semlalia*
– Lieux de réalisation des travaux (laboratoires, institution,. . .) :
  * Département : *Mathématiques*
  * Institution : *Université Cadi Ayyad - Faculté des Sciences Semlalia*
  * UFR : *Mathématiques Fondamentales*
– Période de réalisation du travail de thèse : *2000 à 2009*
– Rapporteurs autres que l'encadrant (nom, prénom, grade, institution) :
  * *Daniel M. Berry, Professor, Cheriton School of Computer Science - Faculty of Mathematics, University of Waterloo, Canada*
  * *Hossam A. H. Fahmy, Assistant Professor, Faculty of Engineering, Cairo University, Egypt.*
  * *Azzeddine LAZREK, PES à la FSSM, Université Cadi Ayyad, Maroc*
– Ce travail a donné lieu aux résultats suivants (communications, publications,. . .) :
  * Abdelouahad Bayar, Khalid Sami, "How a font can respect rules of Arabic calligraphy", *The International Arab Journal of e-Technology*, Vol.1, No.1, pp. 1-18, 2009.
  * Abdelouahad Bayar, Khalid Sami, "Optimization of the curvilinear stretching in a PostScript font with quadratic Bézier curves", *The International Arab Conference of e-Technology, Proceedings of IACe-T'2008*, Arab Open University, Amman - Jordan, pp. 97-104, 2008.
  * Abdelouahad Bayar, Khalid Sami, "An Optimal Way to Encode the Outlines of Variable Sized Arabic Letters in a PostScript Font", *The 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Full Papers Proceedings*, University of West Bohemia, Plzen Czech Republic, pp. 57-64, 2008.

إهداء

- إلى روح أبي الطاهرة تغمده الله بواسع رحمته وأسكنه فسيح جناته.
- إلى من ناءت بحملي في بطنها وابتهجت بمجيئي إلى هذا العالم وتفانت في تربيتي بكل حب و إيثار، إلى الغالية أمي.
- إلى من شاركت وكملت حياتي، إلى أم أسامة ونضال، زوجتي الغالية لطيفة.
- إلى من بدلا طعم حياتي، الحبيبين ولدي أسامة وبنتي نضال.
- إلى أختي وإخواني وأبنائهم و كل أفراد العائلة.
- إلى صهري دحمان احميمسة و أبنائه و كل العائلة.
- إلى أستاذي خالد سامي وكل أسرته مع كامل التقدير و الاحترام.
- إلى أساتذتي الأفاضل، إلى كل من علمني حرفا.

Remerciements

Je tiens à remercier vivement mon directeur de recherche, le professeur Khalid Sami, qui n'a épargné aucun effort pour m'aider à mener à terme ce travail de thèse. Il a toujours accepté mes idées et il y a cru ce qui m'a tout le temps motivé pour avoir de plus en plus confiance en moi et faire de plus en plus mieux pour améliorer la qualité de ce travail de recherche. Il m'a dirigé et assisté techniquement parlant mais aussi il m'a offert un soutien moral en m'encourageant et considérant toutes les circonstances et difficultés que j'ai vécues pendant toutes ces années. Tout cela a été un fort point de motivation pour que mes recherches aboutissent à un travail ayant la qualité d'une thèse de doctorat. Je lui en suis reconnaissant.

Mes forts remerciements s'adressent aussi au professeur Abdelaziz Nasroallah qui m'a honoré d'avoir accepté de juger mon travail de thèse en tant que président de la commission d'examen.

J'ai eu tout l'honneur que les professeurs Mostafa Banouni, Mohamed Laghchim Lahlou et Azzeddine Lazrek soient parmi les membres de jury de cette thèse. Je ne laisserai passer l'occasion sans les en remercier chaleureusement.

Mes sincères remerciements vont également aux professeurs Azzeddine Lazrek, Daniel M. Berry et Hossam A. H. Fahmy qui m'ont fait l'honneur de juger mon travail de recherche en tant que rapporteurs. Leurs remarques et suggestions ont été d'une grande importance pour améliorer le contenu de ma thèse.

Enfin, je voudrais bien profiter de cette occasion pour remercier tous ceux qui ont contribué de près ou de loin pour l'élaboration de ce travail, en particulier ma famille, mes amis et mes collègues.

التمثيل الصوري لإنجاز طقم حروف ممشوقة ديناميكية تحترم قواعد الخط العربي

ملخص : يقدم هذا البحث تطوير وبرمجة نموذج رياضياتي لبناء تمثيل صوري ومن تم إنجاز طقم حروف ممشوقة ديناميكية تحترم قواعد الخط العربي.

قواعد الخط العربي نتاج قرون عدة من التطويرالذؤوب والمتواصل. تجدر الإشارة أن هناك خطوط عربية مختلفة نذكر منها الخطوط الأربعة الذائعة الصيت وهي خط النسخ، خط الثلث، خط الرقعة والخط الكوفي، لكل نوع منها قواعده الخاصة. فللخط الممشوق طرقه الخاصة في دمج بعض الحروف المتصلة كما تتنوع أشكال الحروف تبعا للسياق، والكشيدة (أو التمديد) مثال على تنوع الحالات.

عديد من أدوات تنضيد النصوص الإلكترونية المتعددة اللغات تأخذ قواعد الخط العربي بالحسبان قدر المستطاع. إلا أنه، إلى حد الآن، لا تزال هذه البرمجيات بعيدة عن إتاحة جودة كتابة تضاهي تلك التي تتوفر في النصوص التي ينجزها الخطاطون.

في بداية هذا البحث، كان لزاما تشخيص قواعد الخط العربي التي لا تستطيع نظم معالجة النصوص مراعاتها. بالطبع، قد لا تكون هذه القواعد مصوغة بطريقة دقيقة. فجل كتب الخط العربي تكتفي بتقديم أمثلة ولوحات فقط. وعليه، كان من الواجب دراسة وتجريب نظم معالجة النصوص المكتوبة بالحرف العربي من أجل معرفة قدرتها على احترام قواعد الخط العربي قبل الشروع في البحث عن حلول. وهكذا مثلا، تبين جليا أن تلك النظم لا تتيح بطريقة مثلى وناجعة إنجاز التمديد المنحني الذي تستلزمه الكشيدة، كما تبين أيضا أن بعض أنواع روابط الحروف (Ligature)، والتي تسهم بشكل أساسي في جمالية الخط العربي غير متوفرة. بعد ذلك، كان من المفروض القيام بدراسة للقياسات المتبعة في كتابة الخط العربي خاصة تلك التي تستعمل في تمديد الحروف.

نظرا لأهمية الكشيدة في الخط العربي، خصص جزء مهم من البحث لهذا المفهوم. لذلك قمنا بدراسة بعض نماذج التمديد المتوفرة كنموذج دانيال بري Daniel Berry (الذي يعتمد بدوره فكرة تمديد اقترحها جاك أندريه Jacques André من ذي قبل). لم نجد طريقة لتعميم هاته النماذج لكي تتيح إمكانية التمديد المنحني أفقيا وعموديا في ذات الآن مع الحفاظ على تجانس الكتابة. واستمر البحث إلى أن واتتنا فكرة بناء نموذج تمديد رياضياتي للكشيدة يعتمد على الحفاظ على علاقة مركزية (Barycentric Relation) بين مكونين يدخلان في تركيب الحرف. وبعد إخضاع النموذج للدراسة والتجربة والتأكد من حل الإشكال المطروح بصفة مرضية، أصبح مسار البحث مرسوما وجليا. وبالتالي، درسنا ونمذجنا مختلف حالات الكشيدة، خاصة : تمديد الحرف الواحد وتمديد حرفين مترابطين وحالات خاصة أخرى. ولم يتطلب تطوير طقم حروف عربية لإنجاز الكشيدة إيجاد نموذج تمديد رياضياتي فحسب بل كان من المستلزم أخذ حركة رأس قلم الخطاط بعين الحسبان. فهكذا، يصبح الحرف معرفا بالمنحنيات التي تمثل مسار رأس القلم لا بالمنحنيات التي تحد المساحة الممسوحة به أثناء كتابة الحرف.

في خط النسخ، نوع الخط الذي اعتمدناه في المراحل الأولى من بحثنا، يمكن تمثيل رأس القلم بمستطيل صلب معرف بعرض وسمك، بحيث يساوي سمك رأس القلم سدس عرضه. ويحتفظ رأس القلم، أثناء الكتابة، على زاوية انحناء ثابتة تقريبا ومقدارها $70°$ (انطلاقا من خط الكتابة). ويعود أخذ حركة القلم بعين الاعتبار إلى تحديد غلاف المساحة التي يمسحها رأس القلم وليس المساحة الملونة ذاتها. ويستلزم تحديد غلاف المساحة الممسوحة وضع طرائق لدراسة :

- تغيرات المنحنيات الاستوائية بالنسبة لاتجاه معين.
- مقارنة بين مجموعة من المنحنيات. فكل المنحنيات المكونة للمجموعة عبارة عن إزاحات مختلفة لنفس المنحنى. وتعتمد طريقة المقارنة هذه في بنائها على ما تم وضعه في النقطة السالفة الذكر.

إن طقم حروف ديناميكي بوستسكريبت من النوع الثالث لا يتوفر على مكون التخزين (caching). وهذا يعني أن الطريقة التي ينهجها برنامج بوستسكريبت بخصوص حرف ما سيعاد تنفيذها في كل مرة يظهر فيها هذا الحرف في النص. لذا، يتعين إيجاد طريقة لاستمثال برمجة الصياغة الرياضياتية المطورة في إطار هذا البحث. ويتم الاستمثال بالخصوص من خلال وضع طريقة للتقليص من الوقت اللازم لحساب معاملات تقاطع بعض منحنيات بيزيي المكونة لشفرة الحروف. وفي بعض الحالات، يتم جزء من الاستمثال باستعمال منحنيات تربيعية عوض المنحنيات التكعيبية أو عكس ذلك.

إضافة إلى استعمال النموذج المقترح في تكوين طقم حروف ديناميكي أمثل يراعي قواعد الخط العربي، لهذه الدراسة تطبيقات أخرى كإمكانية اعتمادها في إنشاء أطقم حروف عربية أخرى غير النسخ (الثلث مثلا)، أو في إنجاز أطقم الرموز الرياضياتية القابلة للتمديد، أو برمجة وسائل معلوماتية لدعم تعليم قواعد الخط العربي، وفي التعرف الآلي الضوئي على نصوص الخط العربي.

كلمات المفتاح: الخط العربي، الكتابة الممشوقة، الكشيدة، النسخ، تمديد ديناميكي، طقم حروف بوستسكريبت نوع 3، مقارنة المنحنيات، تقاطع منحنيات بيزيي.

Formal representation for the realization of a cursive dynamic font respecting Arabic calligraphic rules

Abstract: This work gives a development and implementation of a mathematical model for the *formalization and the realization of a cursive dynamic font respecting the rules of Arabic calligraphy.*

The Arabic calligraphy rules are the harvest of several centuries of continuous improvements. There are various calligraphic styles, each one with its own rules. A cursive writing style presents several ligatures, the letters can have a wide variety of shapes according to the context ... the kashida materializing the curvilinear stretching of certain letters, or between two letters, are examples of this differentiation.

Various tools and packages for processing multilingual digital documents take into account the Arabic alphabet and try their best to respect its constraints and observe the rules of the Arabic calligraphy. Unfortunately, up till now, the calligraphers handwritten documents quality is still far beyond what typesetting systems can provide.

In the start of this work, we were to determine precisely what Arabic calligraphy rules the typesetting systems could not observe. Of course, such rules are not always well formulated. Indeed, generally, the calligraphy handbooks supply only some examples and proofs. So, experimenting document processing systems that support Arabic scripts in order to evaluate their aptitude to respect calligraphic rules was a preliminary task. So, one can see that the available systems do not offer an accurate nor optimal way to support the curvilinear stretching the kashida requires. Also, certain types of ligatures that contribute significantly to the aesthetics of the Arabic calligraphy are missing. So, the adopted metrics in Arabic calligraphy, in particular those used in the stretching of characters, were to be studied.

Due to the importance of the kashida in Arabic script based writing, an important part of our work was dedicated to this notion. We had so studied models such as that proposed by D. Berry (based on an idea for stretching letters, formerly proposed by J. André). We had not been able to find a method to extend such models to support the curvilinear stretching in a way to allow both horizontal and vertical stretching and preserving the writing's homogeneity. We experimented other ways and finally appeared the idea of building a mathematical model for stretching the kashida, based on the conservation of a barycentric relation between two components of curves representing parts of the letter. After experimentation, the model provided a quite good solution to the problem. Then, the way was quite drawn. We had then studied and modeled various cases of kashida: kashida supported implying two connected letters, kashida inside a letter as well as other particular cases.

The support of the kashida, with preservation of the calligraphic quality, did not only impose to find a stretching mathematical model but also to take into account the motion of the nib's head (the qalam) the calligrapher uses. In this way, a character becomes defined with curves representing the motion of the qalam and not by the curves delimiting the surface razed with the qalam.

In the Naskh style, the style chosen in the first stage of these works, the nib's head behaves as a rectangle of width l and thickness $l/6$. This rectangle moves with a constant inclination angle of about 70° (with regard to the baseline). Considering the movement of the qalam means determining the envelope of the surface razed with the nib's head not the surface itself. The determination of this envelope requires finding methods for:

- the study of variations of plane curves with respect to a given direction.
- the comparison of a set of curves obtained by translations of different vectors of the same curve. This method is based on what had been developed through the previous method.

In addition, in a dynamic PostScript font Type 3, as there's no caching, the PostScript procedure to draw a dynamic letter will be executed as many times as the letter appears in the document. Therefore, it is necessary to find a way to optimize the implementation of the mathematical formalism. In particular, the optimization will be realized through the development of a method to reduce the time used to compute the intersection coefficients of some Bézier curves in the letters code. In some cases, a part of this optimization will be made through the use of quadratic curves instead of those cubic or vice versa.

Besides its use in the development of an optimized dynamic font respecting Arabic calligraphy rules, our work can be helpful in other applications such as the development of Arabic fonts in styles different from the Naskh, the development of fonts supporting stretchable mathematical symbols, of software tools to assist in learning the Arabic calligraphy, and in optical recognition of Arabic calligraphy scripts ...

Keywords: Arabic calligraphy, Cursive writing, Kashida, Naskh, Dynamic stretching, PostScript Font Type 3, Outlined characters, Comparison of curves, Intersection of Bézier curves.

Représentation formelle pour la réalisation d'une fonte cursive dynamique conforme aux règles de la calligraphie arabe

Résumé : Cette thèse consiste à développer et implanter un modèle mathématique pour la *formalisation et la réalisation d'une fonte dynamique cursive qui respecte les règles de la calligraphie arabe.*

Les règles de la calligraphie arabe sont le fruit de plusieurs siècles de recherche et de raffinement de méthodes. Cet art a donné naissance à un ensemble de styles d'écritures : Naskh, Roquaa, Thuluth... aussi remarquables les uns que les autres. Les règles de la calligraphie arabe diffèrent d'un style à un autre; la cursivité de l'écriture avec ses ligatures, la diversité des formes de la lettre selon le contexte, la kashida qui matérialise l'extension curviligne de certaines lettres ou de deux lettres connectées... en sont quelques exemples.

Un bon nombre d'outils et paquetages de traitement de documents numérique multilingue qui prennent en compte l'alphabet arabe essayent de respecter le plus des contraintes et observer les règles de la calligraphie arabe. Néanmoins, jusqu'à présent, la qualité des documents écrits à la main par les calligraphes est hors de l'atteinte des outils disponibles.

Au début de ce travail, il fallait déterminer ce que les systèmes de traitement de documents ne pouvaient pas observer en matière de règles de la calligraphie arabe. Ces règles ne sont pas toujours bien formulées. En effet, les ouvrages de calligraphie se contentent généralement de fournir des exemples et des planches. L'expérimentation des systèmes de traitement de documents qui supportent le script arabe, dans le but de déterminer leurs performances à respecter la calligraphie, était un préalable avant d'entreprendre les premières investigations. Nous avions ainsi pu mettre en évidence que les systèmes disponibles n'offrent pas de manière efficace ni optimale le support de l'extension curviligne nécessaire pour la kashida. Certains types de ligatures qui contribuent de manière significative à l'esthétique de la calligraphie arabe font également défaut. C'est alors que nous avions entrepris une étude des métriques adoptées dans la calligraphie arabe, en particulier de celles utilisées dans l'extension des caractères.

Vu l'intérêt de la kashida dans une écriture à base du script arabe, une bonne partie du travail a été dédiée à cette notion. Nous avions ainsi étudié des modèles tels celui proposé par D. Berry (qui se base sur une idée d'extension proposée auparavant par J. André). Nous n'avions pas pu trouver un moyen pour étendre ces modèles afin d'offrir le support de l'extension curviligne de manière à permettre à la fois les extensions horizontale et verticale tout en préservant l'homogénéité de l'écriture. Nous

avons expérimenté d'autres voies jusqu'à ce qu'apparaisse l'idée de construire un modèle mathématique d'extension de la kashida basé sur la conservation d'une relation barycentrique entre deux composantes. Lorsque ce modèle a été expérimenté et qu'il ait permis de résoudre le problème de manière satisfaisante, la voie était toute tracée. Nous avions alors étudié et modélisé les différents cas de kashida à savoir la kashida supportée au niveau de deux lettres connectées, la kashida à l'intérieur d'une seule lettre ainsi que d'autres cas particuliers.

Le support de la kashida avec maintien de la qualité calligraphique n'a pas seulement imposé de trouver un modèle mathématique d'extension mais aussi de tenir compte du mouvement de la plume du calligraphe. De cette manière, un caractère est défini par les courbes représentant la trajectoire de la plume et non par les courbes qui délimitent la surface balayée par la tête de la plume.

Concernant le style Naskh, style choisi pour la première phase de ces travaux, la tête de la plume se présente comme un rectangle solide déterminé par deux grandeurs qui sont la largeur de la tête de la plume et son épaisseur. L'épaisseur est le 1/6 de la largeur. La tête de la plume demeure approximativement dans une inclinaison constante d'un angle d'environ 70°. Considérer le mouvement de la plume revient à déterminer l'enveloppe de la surface balayée par la plume et non la surface elle même. La détermination de cette enveloppe requiert les travaux suivants :

- Détermination d'une méthode d'étude des variations de courbes planes par rapport à un vecteur donné.
- Détermination d'une méthode de comparaison d'un ensemble de courbes obtenues par des translations de vecteurs différents d'une même courbe. Cette méthode est basée sur ce qui avait été développé dans le point précédent.

Dans une fonte dynamique PostScript Type 3, la notion de caching n'est pas de mise. Cela veut dire que la procédure PostScript pour une lettre dynamique sera exécutée autant de fois que la lettre apparaît dans le document. Il s'avère donc nécessaire de trouver un moyen pour optimiser l'implantation de ce formalisme mathématique. En particulier, l'optimisation sera réalisée via le développement d'une méthode pour réduire le temps de détermination des coefficients d'intersection de certaines courbes de Bézier dans le code des lettres. Dans certains cas, une part de cette optimisation se fera via le recours à des courbes quadratiques au lieu de celles cubiques ou vice versa.

En plus de l'usage dans le développement d'une fonte dynamique optimisée respectant les règles de la calligraphie arabe, notre travail peut avoir d'autres applications tel dans le développement de fontes arabes dans des styles autre que Naskh, le développement des fontes de symboles mathématiques extensibles, le développement d'outils logiciels d'aide à l'apprentissage de la calligraphie arabe, la reconnaissance optique de

scripts de calligraphie arabe…

Mots-clés : Calligraphie arabe, Écriture cursive, Kashida, Naskh, Extension dynamique, Fonte PostScript Type 3, Caractères en contours, Comparaison de courbes, Intersection de courbes de Bézier.

Table des matières

1 Fontes et calligraphie arabe **1**

1.1 Un vieux problème toujours d'actualité 1

1.2 Motivations 2

1.3 Contexte de la problématique 3

1.4 Points de méthode 6

1.5 Aperçu général 7

2 Respect of basic calligraphy rules **9**

2.1 Introduction 9

2.2 On some Arabic calligraphy constraints 11

2.2.1 The notion of baseline 11

2.2.2 How letters are calligraphed ? 12

2.2.3 Arabic calligraphic metrics 15

2.2.4 Connections and ligatures 16

2.3 Curvilinear stretching in Arabic calligraphy 23

2.3.1 Stretching in Arabic calligraphy 23

2.3.2 Dynamic fonts and the support of stretching 24

2.3.3 Implications for the design of the font 24

2.3.4 Characteristics of the stretching in Arabic calligraphy 25

2.4 Model and support of Kashida 33

2.4.1 Kashida : the mathematical model 33

2.4.2 Kashida : the support in the font 39

2.4.2.1 Inter-letter Kashida 39

2.4.2.2 Intra-letter Kashida 42
2.4.3 Nib's head motion modeling approach versus other approaches . 43
2.5 Conclusions 46

3 Parametric curves variations 49
3.1 Introduction 49
3.2 Problem 50
3.3 Order in an affine plane in $\mathbb{R}^3$ 54
3.4 Functions from $\mathbb{R}$ into an affine plane in $\mathbb{R}^3$ 56
3.5 Bézier curve decomposition 59
3.6 Conclusions 63

4 Arabic characters and optimized encoding 65
4.1 Introduction 65
4.2 On some characteristics of Arabic calligraphy 66
4.2.1 Kashida 66
4.2.2 Kashida and outlined letters 67
4.2.3 Fixed and variable sized letters 68
4.3 Nib's head motion-razing modeling and decomposition 68
4.3.1 Qalam's motion modeling 69
4.3.2 Motion-razing 69
4.4 Stretching model 74
4.4.1 Cubic stretching model 74
4.4.2 Quadratic stretching model 76
4.4.3 Time processing for each model 77
4.5 Implementation of static letters 78
4.6 Optimized implementation of filled in dynamic letters 79
4.6.1 Implementation of the static part 79
4.6.2 Implementation of the stretchable part 79
4.7 Optimized implementation of outlined dynamic letters 82

4.8 Conclusions 90

5 Conclusions et perspectives 91

5.1 Conclusions 91

5.2 Perspectives et problèmes ouverts 92

Bibliographie 95

Table des figures

2.1 Isolated letters in Naskh style with respect to the baseline 12

2.2 Example of letter integrally written in translation motion of the qalam (Standalone NOON) 12

2.3 Example of letters with parts written in translation motion of the qalam and others drawn (TAH, REH and HAH) 13

2.4 Drawing of letters tips in Naskh (REH and TAH) 13

2.5 Three different Arabic letters resulting from the diacritic dot use or position 15

2.6 Metrics of the letter HAH in the calligrapher way 15

2.7 A stretching of 12 diacritic points horizontally and 0.5 vertically 16

2.8 In gray, succeeding connection of the letter KHAH in initial position . 16

2.9 In gray, preceding connection of the letter TAH in final position 16

2.10 Boxes position with horizontal connections 17

2.11 In gray, vertical succeeding connection of the letter NOON in initial position 17

2.12 In gray, vertical preceding connection of the letter HAH in median position 17

2.13 Boxes position with vertical connections 17

2.14 An Arabic letter with succeeding and preceding connections 18

2.15 Double vertical ligature 18

2.16 Triple vertical ligature 19

2.17 a) Compound character with horizontal succeeding connection, b) Compound character with horizontal preceding connection, c) Compound character with horizontal succeeding connection and preceding connection, d) Compound characters as standalone letters. 19
2.18 Contribution of compound characters as simple ones in words 20
2.19 Contribution of compound characters in stretching 20
2.20 a) Oblique preceding connection, b) Oblique succeeding connection . . 21
2.21 a) Horizontal preceding connection *big-strictly above* the baseline, b) Horizontal succeeding connection *big-strictly above* the baseline 21
2.22 a) Horizontal preceding connection strictly above the baseline, b) Horizontal succeeding connection strictly above the baseline 22
2.23 a) Horizontal preceding connection exactly on the baseline, b) Horizontal succeeding connection *exactly on* the baseline 22
2.24 a) Horizontal preceding connection below the baseline, b) Horizontal succeeding connection below the baseline 22
2.25 Two curvilinear stretching in Arabic calligraphy 24
2.26 Nib's head in Naskh style with 12 mm width 25
2.27 The optical scaling characteristics of the Arabic curvilinear stretching . 26
2.28 a) The standalone QAF in Daniel Berry font, b) A stretching of standalone QAF in Daniel Berry font 27
2.29 Surface razed with the edge l_1 (light gray) 28
2.30 Surface razed with the edge l_2 (little dark gray) 29
2.31 Surfaces razed with the two edges e_1 and e_2 30
2.32 Surface razed with the entire nib 30
2.33 The result of filling in the surface between the two curves with the previous technique 31
2.34 Surface razed with the nib 31
2.35 Surface razed over $[0, 0.6184]$ 31
2.36 Surface razed over $[0.6184, 1]$ 32
2.37 Superposed razed surfaces on $[0, 1]$ 32
2.38 Curves of type 1 (in $\mathcal{B}_1$) 33
2.39 Curves of Type 2 (in $\mathcal{B}_2$) 34

2.40 Stretching a curve belonging to the set $\mathcal{B}_1$ with E_{be} $(E_{be}(B_1) = B_2)$. 35

2.41 Stretching a curve belonging to the set $\mathcal{B}_2$ with E_{af} $(E_{af}(B_1) = B_2)$. 36

2.42 initial state of a set of curves $\mathcal{C}$ to be stretched 37

2.43 Stretching the curve B_{21} of $\frac{h}{2}$ and v 37

2.44 Translation of B_{11} towards the right-side of $\frac{h}{2}$ 37

2.45 Stretching the curve B'_{11} of $\frac{h}{2}$ and v 38

2.46 Translation of C_1 towards the right direction in h 38

2.47 The set $\mathcal{C}$ after stretching of h and v 39

2.48 Inter-letter kashida mechanisms 40

2.49 Useful alternatives to stretch 42

2.50 TAH in final position with *oblique* preceding connection 42

2.51 Alternative for stretching the character TAH in final position proceeded by HAH in initial position 42

2.52 Stretching in the level of TAH 42

2.53 Kashida in the letter QAF in final position 43

2.54 Identification of points where slopes are parallel to vectors defining nib's head 44

2.55 The real razed surface 45

2.56 The razed surface with our approach 46

2.57 a) Arabic text lines before justification, in 18 pt, b) Arabic text lines after justification 46

3.1 Surface S1 razed by the side $[M_{10}M_{20}]$ 51

3.2 Trajectory of the point M_{10} 51

3.3 The Surface razed by the segment $[M_{10}, M_{20}]$ 52

3.4 Surface delimited by the outlines associated to the movement 52

3.5 Decomposition of M_{10} trajectory 52

3.6 Surface corresponding to B_1 53

3.7 Surface corresponding to B_2 53

3.8 Superposing surfaces S_1 and S_2 53

3.9 A cubic Bézier curve of a movement 62

3.10 Surface razed by the segment $[M_{10}, M_{20}]$ 62
3.11 Surface delimited by the associated outlines 62
3.12 First decomposition 63
3.13 Second decomposition 63
3.14 Superposition of the surfaces of decomposition 63

4.1 Inter-curvilinear and intro-curvilinear stretching in Arabic calligraphy 67
4.2 Possibilities of Outlines writing in Arabic 67
4.3 Stretchable outlines in Arabic 67
4.4 a) Examples of static letters, b) Examples of variable sized letters . . . 68
4.5 Most popular Arabic writing styles 68
4.6 Example of motion 69
4.7 a) Normal motion, b) The corresponding normal razing 71
4.8 Decomposition of the origin trajectory. 72
4.9 Normal sub-razings 73
4.10 Superposed normal sub-razings 73
4.11 The razing in outline with overlapping 74
4.12 The razing in true outline 74
4.13 a) Cubic curve in $\mathcal{B}_1^c$, b) Cubic curve in $\mathcal{B}_2^c$ 75
4.14 a) Stretching of a curve belonging to the set $\mathcal{B}_1^c$, b) Stretching of a curve belonging to the set $\mathcal{B}_2^c$ 76
4.15 Quadratic curve in $\mathcal{B}_1^q$, b) Quadratic curve in $\mathcal{B}_2^q$ 77
4.16 a) Example of stretching a quadratic curve in $\mathcal{B}_1^q$, b) Example of stretching a quadratic curve in $\mathcal{B}_2^q$ 77
4.17 Qalam's Motion and the corresponding true outlines of the standalone DEL 78
4.18 Non closed outline of the static part of End REH 79
4.19 Motion of the stretchable part of End REH 80
4.20 letters with non closed and non overlapping outlines 82
4.21 Opened overlapping of stretchable part of End REH 82
4.22 Opened non overlapping outline of End REH 88

Liste des tableaux

4.1 Comparison in time of the cubic and quadratic models 78
4.2 Comparison between the complete implementation of the end REH in cubic and quadratic models 81
4.3 Comparison in time of the cubic and quadratic approximations 89
4.4 Filled End REH, overlapping outlines versus non-overlapping outlines . 90

Chapitre 1

Fontes et calligraphie arabe

1.1 Un vieux problème toujours d'actualité

Jusqu'à nos jours, les versions imprimées du Saint Coran sont des reproductions d'épreuves écrites à la main. C'est dire que les exigences de précision et d'esthétique de l'écriture de tels textes ne peuvent toujours pas être satisfaites via les performances des traitements de textes actuels. L'imprimerie traditionnelle, ou plutôt la typographie avait pourtant fait des pas de géant pour produire des textes dont la qualité se rapproche de ce que les scribes pouvaient offrir. La Matabi'e al Amyriyah constituent l'exemple même du génie des artisans typographes en course pour imiter les calligraphes. La calligraphie restait toujours hors d'atteinte.

Au début de nos travaux, nous n'avions pas d'ambition autre que celle de développer un système de composition du texte mathématique arabe qui soit *convivial* sans pour autant nuire à la qualité des documents. Cette qualité devait être conforme aux normes de la publication scientifique telles que celles offertes par le système TEX [16].

Le traitement du document textuel peut être techniquement réparti sur deux volets : les programmes de formatage d'une part et les fontes de caractères de l'autre. Sachant que les programmes de formatage de texte se basent sur des fontes déterminées, il est naturel de s'intéresser en premier lieu aux développement des fontes.

L'étude de ce qui est disponible comme outils de composition de documents utilisant le script arabe[1] montre d'emblée des déficiences tels le manque de moyens de composition de formules mathématiques arabes ou le support de certaines composantes fondamentales des règles de la calligraphie arabe comme la kashida. Si le premier exemple est évident, le second peut être rapidement noté dans un traitement de texte comme

[1]Outre l'arabe, plusieurs langues utilisent le script arabe : le persan, l'ourdou, le pashtu, l'ouzbek... en sont quelques exemples.

MS Word. La kashida est une petite courbure lisse qui intervient au niveau de deux caractères arabes connectés ou à l'intérieur d'un caractère soit pour gérer la justification du texte à gauche (en fin de lignes, puisque l'arabe se déroule de droite à gauche) soit pour produire des effets esthétiques requis par les règles de la calligraphie arabe.

Étant donné qu'un document scientifique ou courant contient du texte avant de contenir des formules mathématiques, il est naturel que nous nous intéressions d'abord au développement des fontes permettant la production de documents arabes dans un tant soit peu de respect de la calligraphie arabe.

Les contraintes de la calligraphie arabe sont de diverses natures. La cursivité avec ses ligatures, la diversité de formes d'écriture d'une même lettre selon le contexte, l'extension curviligne de certains caractères ou entre deux caractères en sont quelques exemples. La prise en compte simultanée de toutes ces contraintes n'est pas simple. Nous avions ainsi été amené à restreindre l'objectif de nos premières investigations à l'identification de spécificités propres au script arabe ainsi qu'au développement d'un formalisme mathématique à même de permettre le support de la kashida [3]. L'aspect *dynamique* de la fonte destinée au script arabe a charrié avec lui la nécessité du développement de méthodes d'optimisation de l'implantation [1, 2].

Les règles de la calligraphie arabe diffèrent d'un style d'écriture à l'autre (Naskh, Roquaa, Thuluth, Dywani...). Nous avions été amené à retenir, dans une première phase, le style Naskh [7, 39, 35] pour l'étude et l'expérimentation des résultats. Ce choix est dicté par le fait que ce style est, de loin, celui qui est le plus utilisé. Tout ce qui va être développé pour ce style pourra être porté à d'autre styles sans gros changements de concepts ou de fondements.

En définitive, le plus gros du contenu de ce mémoire portera sur le développement d'outils pour mettre en œuvre une fonte dynamique qui permet d'obtenir un script qui respecte les règles de base de la calligraphie arabe, plus précisément l'élaboration d'un formalisme mathématique et d'une méthode pour optimiser l'implantation de la fonte en se basant sur ce formalisme.

1.2 Motivations

La langue arabe, ou plus précisément, l'écriture arabe, est un outil de communication utilisé par des millions de personnes depuis plusieurs siècles. Depuis près de quatorze siècles, des scribes doués n'ont pas cessé d'élaborer des méthodes et de dégager des règles pour écrire les textes arabes. Leurs efforts ont abouti durant les trois siècles derniers à l'apparition et au mûrissement d'un ensemble de styles d'écriture : Naskh, Roquaa, Thuluth, ... [52] qui donne à l'écriture arabe des qualités esthétiques indéniables.

Depuis le début des années soixante dix du vingtième siècle, les outils software de traitement de documents ont connu une évolution remarquable. Bien entendu, les documents à base du script latin se sont taillés la part du lion dans les travaux de recherche et de développement des applications de traitement du document. On peut citer à titre d'exemple le fameux système TEX de D. E. Knuth qui est devenu par la suite un standard dans le monde de l'édition scientifique. Plusieurs défis de taille ont été relevés et des documents numériques avec une qualité qui n'a rien à envier à l'imprimerie traditionnelle ont ainsi pu voir le jour.

Qu'en est il du document à base du script arabe ? Bien entendu, il a eu une part d'intérêt dans maints projets tels les versions arabisés de MS Word, les extensions de TEX comme TEX-XET de D. E. Knuth et P. MacKay [18], l'extension ArabTEX de K. Lagally [30] et Ω de Y. Haralambous et J. Plaice [49], les travaux d'Azzeddine Lazrek [11]... Bien entendu, la qualité des documents produits par ces systèmes reste en deçà de celle des documents réalisés à la main par des calligraphes, mais des pas de géants ont été franchis.

Le système RydArab développé par Azzeddine Lazrek est une extension du système TEX/LATEX qui permet la composition de formules mathématiques arabes. Le système CurExt, du même auteur, est dédié à l'aspect dynamique des fontes utilisées en mathématique ainsi qu'à celles utilisées pour la composition du texte en langue naturelle dans le respect de la calligraphie arabe. Il apporte une solution pour le support de l'extension curviligne des caractères, et par suite, de la kashida [11, 12]. La méthode proposée par A. Lazrek retient des variations de caractères qui pourraient être qualifiées de *continues* mais en nombre limité. De cette manière, une partie du document peut subir les extensions curvilignes nécessaires mais certaines limitations s'imposent. Le fait qu'une même lettre puisse survenir dans un nombre important d'états dans le document nous a motivé pour continuer à travailler sur la notion d'extension afin d'offrir un support de la kashida en terme de variation *continue illimitée.*

Aujourd'hui, la “localization” (en anglais) ou l'adaptation de logiciels et de média d'usage courant aux paramètres locaux de pays et régions déterminés, particulièrement le paramètre de langue, est en plein développement. Le document numérique est l'un des médias qui jouissent de plus en plus d'intérêt en la matière.

1.3 Contexte de la problématique

L'une des caractéristiques de la calligraphie arabe est l'extension curviligne des lettres. C'est l'outil de base pour la justification du texte dans un document arabe. Ce concept est appelé la kashida. Le fait que l'extension soit curvilinéaire suggère que le langage de développement de la fonte soit *dynamique*. A ce propos, trois classes de langages de développement de fontes peuvent être considérés : PostScript [5], METAFONT [17] et l'ensemble des instructions TrueType-font [19].

- METAFONT [17] : est le langage mis en œuvre par D. E. Knuth pour développer les fontes utilisées par le système TEX. C'est un langage *compilé*. En effet, la fonte est codée en langage METAFONT. Elle est ensuite compilée par un compilateur METAFONT afin de générer les métriques et les bitmaps des caractères des fontes. Le système TEX fait appel aux métriques relatives aux fontes sans leur apporter de changement lors du formatage du texte. De même, un visualiseur de documents dvi utilise les bitmaps d'une manière statique, sans modification des caractères, pour afficher l'œuvre de formatage de TEX. Le dynamisme manque au moment de l'impression du document.
- PostScript [5] : est un langage de description de documents et des fontes (PostScript). Il a été développé par Adobe. C'est un langage qui permet de spécifier des fontes dynamiques Type 3. Le langage de description du document est le même que celui utilisé pour coder une fonte Type 3. Cette fonte est insérée dans le document en question et les codages de caractères peuvent être interprétés *à la volée*, autrement dit, de manière *dynamique* (au moment de l'impression).
- Instructions TrueType-font [19] : il s'agit de l'ensemble des instructions utilisées pour la description des fontes TrueType de MS Windows. Ces instructions possèdent presque les mêmes caractéristiques que PostScript concernant le *dynamisme* des caractères. En fait, les glyphes dans ce format peuvent être des bitmaps comme elles peuvent être des codages vectoriels. Les codages vectoriels sont interprétés et numérisés au moment de l'impression sur écran ou sur papier et peuvent être alors paramétrés pour supporter le dynamisme.
- OpenType : c'est un autre standard qui s'etait imposé de fait [6]. C'est une combinaison des formats Adobe et des formats TrueType. OpenType est crée par Adobe et Microsoft reprenant la plupart des concepts des fontes TrueType mais en supportant en plus l'unicode [51]. OpenType offre un bon support pour le multilinguisme. Aussi les fontes OpenType ont la caractéristique d'être utilisées sous différents systèmes d'exploitation comme Windows ou Mac Os. Notre modèle d'extension a été défini en premier à base de courbes de Bézier cubique [3] et ensuite il a été spécialisé aux courbes quadratiques [1, 2]. De cette manière, le modèle peut être aussi utilisé dans les fontes TrueType ou OpenType permettant ainsi de couvrir le support d'extension dans d'autres systèmes de traitements de texte. Une partie de notre futur travail va porter sur OpenType.

Quelques systèmes de formatage de texte tels TEX, LATEX, des systèmes WYSIWYM (What You See Is What You Mean) comme LYX [34] et WYSIWYG (What You See Is What You Get) comme TEXMacs [28] fournissent des supports pour composer des documents, à base du script latin, avec une qualité comparable à celle de la bonne imprimerie. Bien que les extensions de TEX et LATEX comme ArabTEX, Ω et RyDArab permettent la composition de documents à base du script arabe, le support le plus adéquat pour produire le *dynamisme* des caractères utilisés dans les documents arabes n'est toujours pas disponible, du moins tant qu'on essaye de résoudre le problème au niveau des fontes. Bien entendu, un certain dynamisme peut être supporté au niveau du formatteur TEX ou LATEX. Tel est l'option offerte par le système CurExt [12]. Dans CurExt, le dynamisme est basé sur le paramétrage du code METAFONT. CurExt a été adapté pour qu'il soit basé sur PostScript [41]. L'approche de baser CurExt sur METAFONT, profondément exploitée, est à la base des travaux de Ameer M. Sherif et Hossam A. H. Fahmy dans [8, 9] pour mettre en oeuvre une fonte en METAFONT pour AlQalam. Avec les méthodes utilisées dans [12, 8, 9], le problème de l'extension curviligne des caractères n'est pas complètement résolu puisque le nombre d'états d'extension ne peut dépasser la limite : 256×256. Ce nombre est imposé par TEX.

La thèse défendue ici est que le problème du *dynamisme* des caractères utilisés dans le script arabe peut être obtenu au niveau des fontes. On fera alors appel aux fontes dynamiques. Ces fontes seront des fonts PostScript Type 3, des fontes TrueType ou encore des fontes OpenType. Nous avons choisi de développer une fonte dynamique PostScript Type 3 pour rester dans le contexte des travaux déjà entrepris dans ce domaine. A ce propos, au moins deux travaux peuvent être cités : celui de J. André avec B. Borghi [22], J. André et I. Vatton [23, 24] et celui de Daniel Berry [15]. J. André et B. Borghi avaient été les premiers à avoir étudié les *fontes dynamiques*. Ils ont même proposé un modèle d'extension pour certains scripts particuliers tels que l'arabe. Dans l'exemple qu'ils ont donné [22], la lettre SEEN, la partie de la lettre qui a à subir l'extension n'était pas la bonne. Toutefois, le modèle a servi de base pour Daniel Berry. Ce dernier a exploité judicieusement l'idée d'extension et a pu mettre le premier système de composition de documents à base du script arabe supportant la kashida [15].

Le système de D. Berry [15] repose sur le développement d'une fonte dynamique PostScript Type 3 qui est utilisée par le système dittroff/fforttid. Le modèle supportant la kashida permet une extension *horizontale*. Néanmoins, la calligraphie arabe recommande aussi une extension verticale, vers le bas, des caractères à côté de, et en relation avec, celle horizontale. Aussi, des extensions de caractères opérées avec le modèle de D. Berry présentent parfois certaines anomalies. Ces anomalies sont dues au fait que le modèle ne tient pas compte du mouvement de la plume. Le modèle développé ici corrige ces défauts [3, 1, 2].

Ainsi donc, c'est dans l'objectif d'apporter des corrections et des extensions à des travaux qui existent que nous avions été amenés à revoir l'entreprise de fond et à élaborer un formalisme mathématique pour coder les lettres de la fonte. Les constructeurs géométriques fournis par le langage PostScript n'offrent pas de support direct pour le développement de la fonte dynamique de manière optimale.

Bien sûr, la fonte dynamique à elle seule ne suffit pas pour composer des documents à base d'un script arabe, dans le respect des règles de la calligraphie arabe. Il faut également s'intéresser aux programmes de formatage. Le problème est vaste et complexe. Nous avons donc été amenés à commencer par le début. Nous avons restreint le problème et même après toutes ces restrictions, nous n'avons réalisé qu'une partie du programme.

1.4 Points de méthode

Au départ, il fallait déterminer ce que les systèmes de traitement de documents ne pouvaient pas produire en matière de règles de la calligraphie arabe. Ces règles ne sont pas toujours bien formulées. Les ouvrages de calligraphie se contentent généralement de fournir des exemples et des planches [39, 35, 7]. L'expérimentation des systèmes de traitement de documents à base du script arabe dans le but de déterminer leurs performances à respecter la calligraphie devait survenir avant d'entreprendre les premières investigations. Nous avions ainsi pu mettre en évidence que les systèmes disponibles n'offrent pas d'une manière efficace ni optimale le support de l'extension curviligne nécessaire pour la kashida. Certains types de ligatures qui contribuent de manière significative à l'esthétique de la calligraphie arabe font également défaut. C'est alors que nous avions entrepris une étude des métriques adoptées dans la calligraphie arabe et en particulier de celles utilisées dans l'extension des caractères.

Vu l'intérêt de la kashida dans la présentation d'un document à base du script arabe, une bonne partie du travail a été dédié à cette notion. Nous avions étudié des modèles tels celui de D. Berry [15] (qui se base sur l'idée d'extension proposée par J. André) et on pouvait alors déterminer les déficiences. En réalité, nous n'avions pas pu trouvé un moyen pour étendre ces modèles afin d'offrir le support de l'extension à même de permettre à la fois les extensions horizontale et verticale tout en préservant l'homogénéité de l'écriture. Nous avons ainsi expérimenté un bon nombre de modèles jusqu'à ce qu'apparaisse l'idée de construire un modèle mathématique d'extension supportant la kashida et se basant sur la conservation d'une relation barycentrique [3]. Lorsque ce modèle a été expérimenté et qu'il ait permis de résoudre le problème, la voie était toute tracée. Nous avons alors étudié et modélisé les différents cas de kashida à savoir la kashida supportée au niveau de de deux lettres connectées, intra-lettres ainsi que d'autres cas particuliers.

Le support de la kashida avec le maintien d'une apparence de qualité n'a pas seulement imposé de trouver un modèle mathématique d'extension mais aussi de ***tenir compte du mouvement de la plume*** du calligraphe. De cette manière, **un caractère est défini par les courbes représentant *la trajectoire du mouvement* de la plume et non les courbes délimitant la *surface balayée par la tête de la plume*.**

Au niveau du style Naskh, style choisi pour les travaux de thèse, la tête de la plume se présente comme étant un rectangle solide disposant de deux grandeurs qui sont la largeur de la tête de la plume et son épaisseur. L'épaisseur est le 1/6 de la largeur. La tête de la plume demeure approximativement dans une inclinaison constante d'un angle d'environ 70° [7, 39, 35]. Considérer le *mouvement* de la plume revient à déterminer l'*enveloppe* de la surface balayée par la plume et non la surface elle même. La détermination de cette enveloppe requiert les travaux suivants :

- Détermination d'une méthode d'*étude des variations de courbes planes par rapport à un vecteur donné.*
- Détermination d'une méthode de *comparaison d'un ensemble de courbes obtenues par des translations de vecteurs différents d'une même courbe.* Cette méthode est basée sur ce qui est développé dans le premier point.

Dans une fonte dynamique PostScript Type 3, la notion de *caching* n'est pas de mise. Cela veut dire que la procédure PostScript pour une lettre dynamique sera exécutée autant de fois que la lettre apparaît dans le document. Il s'avère donc nécessaire de **trouver un moyen d'optimiser l'implantation de ce formalisme mathématique**. En particulier, l'optimisation sera réalisée via le développement d'une méthode pour *réduire le temps de détermination des coefficients d'intersection de certaines courbes de Bézier* dans le code des lettres. Dans certains cas, une part de cette optimisation se fera via le recours à des courbes quadratiques au lieu à celles cubiques ou vice versa.

1.5 Aperçu général

Le travail présenté dans ce manuscrit est subdivisé, après ce premier chapitre d'introduction, en quatre chapitres. Quelques notions de la calligraphie arabe pourraient être nécessaires pour bien tirer partie de ce travail. Une certaine initiation aux systèmes de traitement des documents; fontes et logiciels de formatage peut également faciliter la lecture. Enfin, quelques notions de base en mathématiques en analyse de fonctions, analyse numérique et géométrie de courbes et en particulier les courbes de Bézier peuvent également être utiles.

Le chapitre 1 chapitre introductif à certains rappels et généralités.

Le chapitre 2 donne quelques composantes principales de la calligraphie arabe. L'accent y est mis sur la kashida et un premier modèle d'extension de la kashida est donné.

Le chapitre 3 présente une extension de l'étude de la monotonie d'une fonction réelle à une seule variable à celle d'une fonction polaire par rapport à un vecteur. Des opérateurs de comparaison y sont définis. Des propriétés, des lemmes et des théorèmes sont étendus pour supporter le domaine de nos travaux.

Le chapitre 4 décrit une extension de la partie mathématique développée dans le chapitre 3 afin de définir une méthode de comparaison des courbes obtenues par des translations de vecteurs différents d'une même courbe. Cette extension a été à la base du formalisme mathématique permettant de déterminer l'enveloppe des surfaces balayées par la tête de la plume. Le modèle d'extension présenté dans le chapitre 2 est alors amélioré. Une méthode d'approximation est développée pour une implantation optimisée de l'enveloppe de la surface balayée par la tête de la plume. Elle concerne la kashida basée sur le modèle d'extension raffiné. Une partie de ce chapitre est dédiée à l'évaluation des idées et concepts utilisés pour confirmer les choix et les adoptions.

Le chapitre 5 est une conclusion générale. Nous présentons également les problèmes ouverts ainsi que les perspectives pour continuer de futures investigations.

Chapitre 2

How a font can respect basic rules of Arabic calligraphy[1]

2.1 Introduction

The text justification of documents written in an Arabic alphabet based script is based on stretching letters instead of the ordinary insertion of spaces (blanks) among words. This concept called Kashida is almost mandatory in a cursive writing such as Arabic. The stretching of the letters according to the rules of Arabic calligraphy is curvilinear and variable. So the encoding languages are to support what can be called “dynamic font”.

Many typesetting systems such as TEX / LATEX [16, 32], WYSWYM systems such as LYX [34] and WYSIWYG systems such as TEXMacs [28] provide support to typeset documents in Latin alphabet based scripts in a quite good quality. Some extensions of TEX and LATEX like ArabTEX [30] and Ω [49] allow typesetting lines in Arabic alphabet based scripts and so do LYX or other tools based on TEX or LATEX. However, these tools don’t support Kashida.

The best commercial fonts for Arabic, with support for the letters stretching, are probably those provided by Decotype [46, 47]. Decotype fonts support the curvilinear glyphs for some predefined sizes. These fonts will not give all the sizes needed to justify the texts and so, linear static fragments or blanks would have to be inserted for justification.

The curvilinear stretching in Arabic calligraphy, the extension both horizontally

[1]The contents of this chapter are the body of a paper published in the international journal IAJeT [3] with small modifications. The introduction has been changed and also the order and contents of some sections have been revised. Some paragraphs and sub-sections have been added whereas some others have been deleted.

and vertically of letters had been first proposed in the CurExt system [12]. When CurExt is called with RyDArab (an extension of TEX/LATEX) [10], it allows typesetting Arabic documents with stretchable characters. In CurExt, Kashida and the extended mathematical symbols are based on parametrize some macros in some METAFONT fonts. The variable sized characters, in the suitable sizes are produced via a call of METAFONT in TEX. The letters stretching in Arabic documents has gained more interest last years. Some recent works provide improvements of parametrized METAFONT fonts to support the dynamic stretching and so, Kashida [8, 9]. The authors in [8, 9] use a principle similar to the one in [12]. The variable sized letters are generated through a recursive calling of METAFONT in TEX. The stretchability is managed more accurately. The works in [12, 8, 9] are excellent and provide operational solutions to compose Arabic documents but there are some small limitations that we highlight in the following paragraph.

In a document formatted with TEX, every character to appear in the document should have a representation in a font. The parametrization of a METAFONT font to get different extensions of letters can't provide more than 256×256 cases of stretching. This is due to the fact that the TEX environment works with at most 256 fonts. In a big Arabic document (with a large number of pages) the need of different stretching can exceed 65536. The system would be in a more limited situation when it has to cope with multilingual documents since some fonts have to support the other languages. When all fonts are exhausted, blanks must be used to justify texts. One of the main goals of Kashida is to help to perform the justification of Arabic texts without inserting spaces. Then, a golden rule of the Arabic calligraphy would be violated. We can remark that the extension function is not defined on values within the interval of permitted extension amounts when the number of extension cases exceeds 65536. It follows the non-continuity on these values. The extension function presented in [8] can differ from a document to another but in the same document it is a discrete function. So, the stretching function is not continuous in a complete sense.

A suitable dynamic font to support Kashida with a truly continue modeling function should be a PostScript font Type 3 [5]. Actually, a PostScript font Type 3 is interpreted and digitized by the Postscript interpreter which is the same program to process the document based on this font. Then, the stretching amounts can be communicated and computed on the fly by the interpreter when the document is computed.

The dynamic fonts have been studied first by J. André and B. Borghi [22]. Daniel Berry used the concept in [22] and developed a Type 3 PostScript font to be used with the dittroff/ fforttid system to typeset documents in right-to-left alphabets based scripts [15]. This system offers a good support for the characters dynamism even though it contains some flaws. Actually, in [15], the stretching is performed in the horizontal direction only and the stretching model don't take into account the motion of nib's head. This fact leads to some lost of quality when the stretching is big. Solutions to fix such flaws are proposed in this chapter.

The Arabic calligraphy has been developed through long and sustainable efforts. Various calligraphic styles are attested. The popular ones are Farsi, Koufi, Maghrebi, Naskh, Thuluth, Rouqaa, Dywani... The Naskh style is probably the most used in digital typography. Every style in the Arabic calligraphy, especially Naskh style, has some characterizing properties materialized in ligature categories, metrics to control writing of letters, composition of some letters with parts of others, stretching rules, ...

In Arabic calligraphy, some kinds of ligatures are mandatory and others are only for aesthetic. All of the systems cited before use fonts providing some mandatory ligatures such as the horizontal ligatures *exactly on the baseline* and some vertical ligatures (See below for meaning) for aesthetic. However, both mandatory ligatures and aesthetic ones are not always provided. So it's the case of like oblique ligatures, big strictly above the baseline, strictly above the baseline and horizontal below the baseline ligatures. Those ligatures are studied in this chapter.

We'll begin by presenting some general Arabic calligraphy constraints in Section 2. The following section is dedicated to study the characteristics of the curvilinear stretching in Arabic calligraphy. In the fourth section, we will present a mathematical model and the support of Kashida in the font. The chapter will then be finished with some conclusions.

2.2 On some Arabic calligraphy constraints

2.2.1 The notion of baseline

The baseline is one of the basic notions in digital typography [16, 48, 17, 5]. It constitutes the basic reference for positioning the characters, or symbols in a mathematical formula and so on. In Arabic writing normative studies, there is no general agreement on the necessity of this reference. Some calligraphers think that the notion of baseline is meaningless. Others think that its position is not steady [41]. Nevertheless, the reference to the baseline can be very helpful and even necessary for positioning signs. Actually, some parallelism among words and even among letters, with regard to a virtual horizontal line, can be noticed in any written line. We well use the common method related in A. El Husseiny's calligraphy's book [7] to define the baseline used in the development of a Naskh font providing the possibility of characters curvilinear extension. According to this reference, two groups of isolated Arabic letters may be distinguished: those to place above the baseline and those containing pieces above and/or below the baseline (See Figure 2.1).

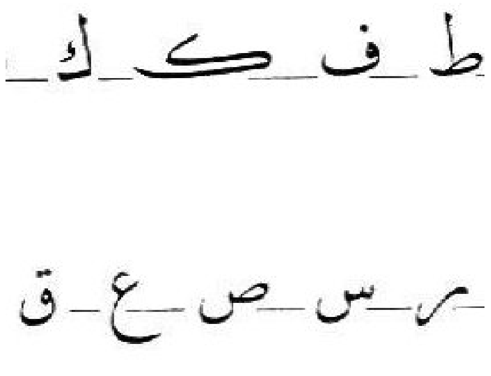

FIG. 2.1 – Isolated letters in Naskh style with respect to the baseline

2.2.2 How letters are calligraphed?

In Naskh style, some letters are integrally written trough a translation of nib's head. It is the case, for example, for the standalone letter NOON (See Figure 2.2). Some rotations can be met in some circumstances due to the loss of hand flexibility or to a bad design of the qalam but they remain imperceptible. In other words, we can't distinguish the difference, visually speaking, between the letter NOON written with or without these light rotations.

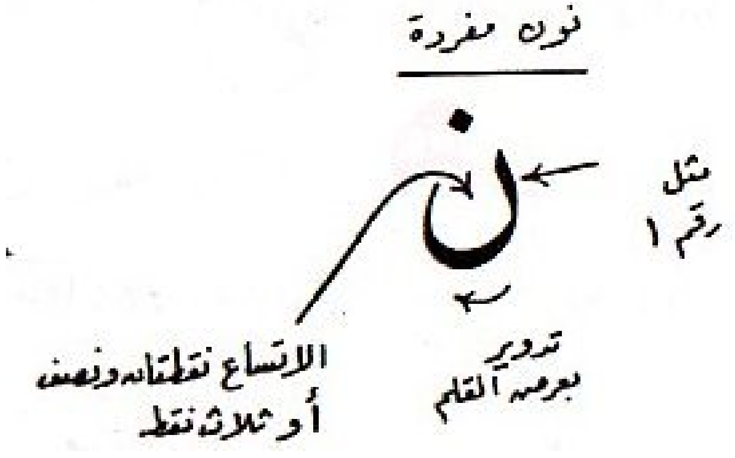

FIG. 2.2 – Example of letter integrally written in translation motion of the qalam (Standalone NOON)

Other letters are compositions of parts simply written by translations of nib's head and others that can not be obtained so. Let us consider, as example, the "Hilya, حلية " or tips of certain letters. A tip in a letter is called, in Arabic, by certain calligraphers

"Shaziah, شظية ". We consider the tips to present the concepts. As example, let us consider the letters: standalone TAH, standalone REH and standalone HAH. The tips of these letters are surrounded with ovals in Figure 2.3.

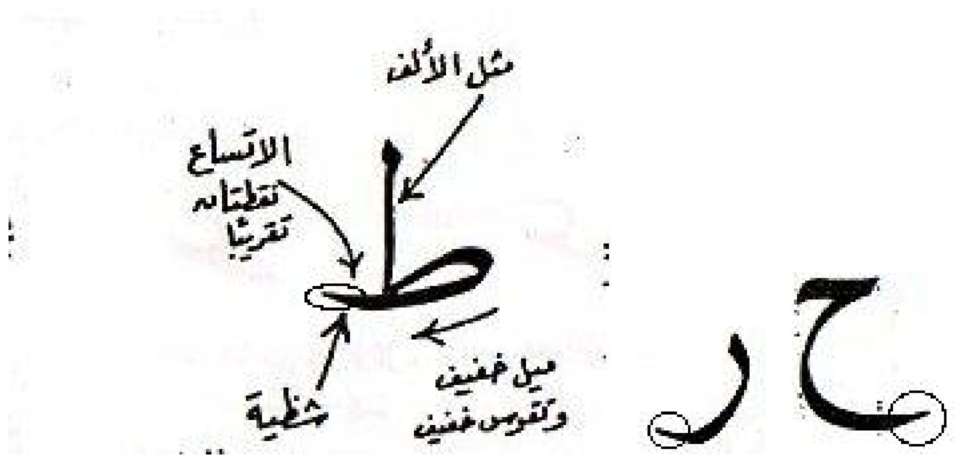

Fig. 2.3 – Example of letters with parts written in translation motion of the qalam and others drawn (TAH, REH and HAH)

Let us consider the letters in Figure 2.3, except for the *Hilyahs* and tips, the parts are written by a simple translation of the qalam. However, parts surrounded with ovals can not be handled in the same way. These parts are produced by a translation of the qalam combined with a varying rotation throughout the motion. In reality, beside the translation and the rotation there is a " Khatf, خطف " of the qalam. It, geometrically, means that the width of the pen decreases along the translation combined to the varying rotation. Proceeding according to this way in Arabic calligraphy is a first variant; a first school.

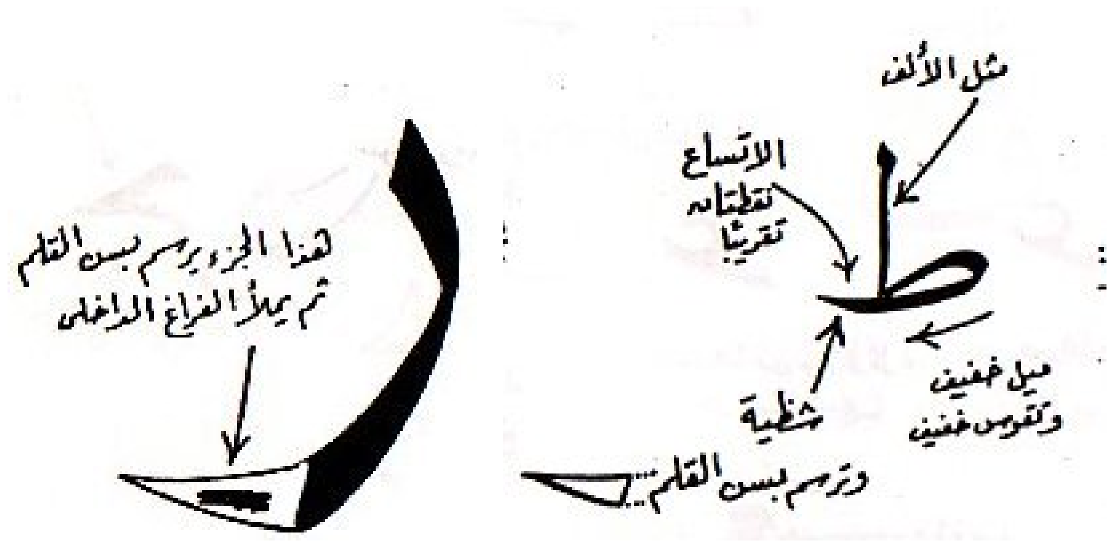

Fig. 2.4 – Drawing of letters tips in Naskh (REH and TAH)

Another way to do consists of decomposing letters into *written* parts and *drawn* parts. In the written parts, the qalam undergoes only translation motions. In the drawn parts, the contour is set first with the right up corner of nib's head (qalam's tooth) and then, the interior of the draw is darkened as with a brush [35]. So, the outlines are first drawn and the interior is filled. An example with the letters: standalone REH and standalone TAH is shown in Figure 2.4. This is the method followed by the calligrapher *Mahdi Essaid Mahmoud*. It is a second school.

For many reasons, we'll adopt the approach of the second school. The tips headed from left to right such as the tip of the standalone letter HAH displayed in Figure 2.3, can be finalized trough a translation combined with a rotation and a Khatf. On the other hand, for tips headed to the left as those of the letters TAH or REH in Figure 2.4, the calligrapher is generally brought to perform retouches to finalize the tips. Bringing retouches means to *draw* and *not to write*. So, the approach of the second school is closer to the reality.

We adopt that the act "to calligraph" (it is not a verb in English but in French the verb "calligraphier" means to do calligraphy) is based on a set of writings and drawings to obtain a letter on a material writing support such as paper, screen ...

Therefore, in the following, a given letter is compound of parts that are *written* in qalam translations only and there are parts that are *drawn*. So, *the motion of the qalam well be supposed to be a translation.*

As language to encode fonts, PostScript is *outlines oriented*. In METAFONT, some particular pens are *predefined* as there is the macro *makepen* that allows to define polygonal convex pens. This provides a support to Naskh's pen. There is no equivalent to manage pens in PostScript except for the circular pen that can be obtained using the operator *setlinewidth*. In PostScript, every letter written with a non-circular pen, should be specified with its outlines curves (to fill in). If letters, to be encoded in a PostScript font, were static, the qalam and its motion, a translation or a translation combined with rotations, would have no interest to be considered. The outlines of letters would be enough. In Naskh style, Kashida can not be modeled only by outlines since these outlines can change contextually. The motion of the qalam must be necessarily taken into account. At the level of the stretchable parts of letters, the motion of the qalam is a translation as we've already seen. We had taken into account this constraint in the font. The static parts of letters are introduced into the font through their outlines. In these parts, we don't give importance to motion. We can say that, in technical terms of encoding, in Naskh style, the qalam operates only translations. This hypothesis remains true even in Naskh in terms of Arabic calligraphy (See the preceding subsections).

2.2.3 Arabic calligraphic metrics

The measure unit in Arabic calligraphy is the diacritic dot used to differentiate some Arabic letters with the same shape. For example, the letters HAH, JEEM and KHAH (from the right to left), in initial position, are presented in Figure 2.5. The diacritic point is the small square filled in light gray. The same dot is the measure unit used to define the letters dimensions. In Naskh style, the diacritic point is a filled in square with sides of a width equal to the width of nib's head and rotated about 60°. In Figure 2.6, some metric indications to write the letter HAH (in the initial form with oblique ligature) are in charge of the diacritic dots. In calligraphy, the space between two consecutive dots in a measure is about 0.45 to 0.5 of a *calligraphic* dot. But, in our work, all the metric values are given in terms of *diacritic points*, without spaces (See Figure 2.7). In Figure 2.7, an example of a stretching of the word ظفر is presented. The first line displays the word ظفر with Kashida in gray in its initial size i.e with a null stretching. Kashida's width is two diacritic points and a fraction. In the following line, the same word is stretched 12 diacritic points horizontally. To the horizontal stretching corresponds a vertical stretching of half a diacritic point downwards with respect to the baseline (See later for the rules of stretching). We can also get information about metrics from [40].

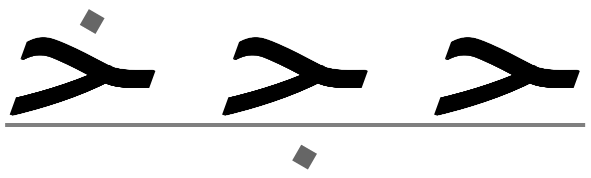

FIG. 2.5 – Three different Arabic letters resulting from the diacritic dot use or position

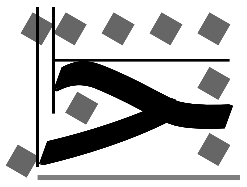

FIG. 2.6 – Metrics of the letter HAH in the calligrapher way

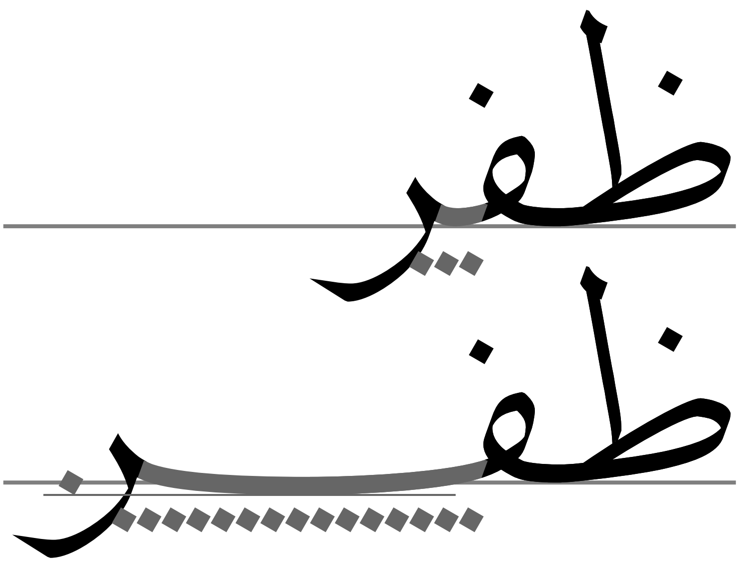

FIG. 2.7 – A stretching of 12 diacritic points horizontally and 0.5 vertically

2.2.4 Connections and ligatures

The Arabic writing is cursive. The notion of ligature is of a particular importance. Handling ligatures in Arabic is of a strong complexity in comparison with latin alphabet based scripts. Now, let us give some details on *connections* of letters since they are the bases of the ligatures. In Arabic calligraphy, a connection is a fine curve following nib's head inclination that connects a letter to another before or after it. So, we can talk about *preceding* and *succeeding connections* (See Figures 2.8 and 2.9). In the sequel, categories and examples of connections are given. Actually, two categories of connections can be distinguished, namely:

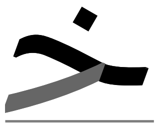

FIG. 2.8 – In gray, succeeding connection of the letter KHAH in initial position

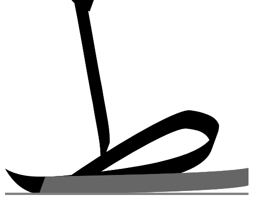

FIG. 2.9 – In gray, preceding connection of the letter TAH in final position

– *Horizontal* connection: horizontal connections can be regarded as tools to tie letters so that the boxes containing these letters are horizontally positioned. In Figures 2.8 and 2.9 are showed the horizontal succeeding connection of the letter KHAH and the preceding connection of the letter TAH respectively. The connections are in light gray. In Figure 2.10, the word خط formed of the letters خ and ط with their digital boxes is presented. So, the definition of the horizontal connection is more clear.

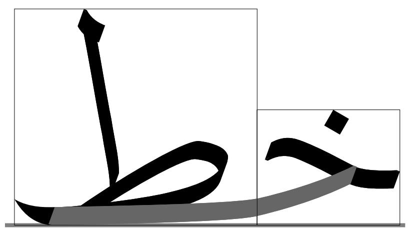

FIG. 2.10 – Boxes position with horizontal connections

– *Vertical* connection: a vertical connection allows connecting two letters so that their boxes can be positioned vertically (See Figure 2.13). The figure 2.11 and 2.12, displays the succeeding and preceding connections of the two letters NOON and HAH.

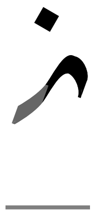

FIG. 2.11 – In gray, vertical succeeding connection of the letter NOON in initial position

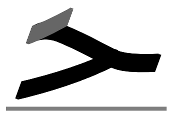

FIG. 2.12 – In gray, vertical preceding connection of the letter HAH in median position

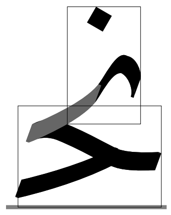

FIG. 2.13 – Boxes position with vertical connections

When they are in the middle of the word, some Arabic letters can have both a succeeding connection and a preceding connection at the same time. For instance, consider the letter ـفـ in the middle of the word ظفر (See Figure 2.14).

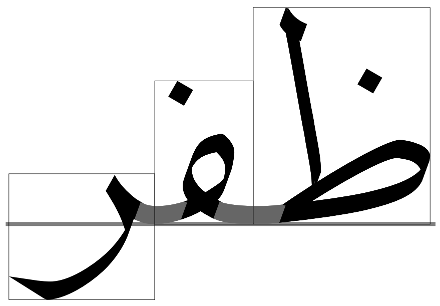

FIG. 2.14 – An Arabic letter with succeeding and preceding connections

We define a ligature as a set of one or more pair of connections. A pair of connections is composed of a succeeding connection and a preceding one. There are two categories of ligatures namely:

- *Horizontal* ligature: it is a combination of an horizontal succeeding connection and an horizontal preceding one. In Figure 2.10, the ligature between the letters KHAH in initial position and TAH in final one of the word خط is an example of such ligature.
- *Vertical* ligature: it is a set of pairs of vertical connections. Every pair is formed of a vertical succeeding connection and a vertical preceding connection. In Figure 2.13, a vertical ligature appears between the letters NOON and HAH. It contains one pair of vertical connections. It is then a simple vertical ligature. In Figure 2.15, the ligature is composed of two pairs of connections connecting the letter LAM to MEEM and MEEM to JEEM. The ligature is then double. Whereas, in Figure 2.16, the ligature connects four letters and thus is formed of three pairs of vertical connections. It is a triple ligature.

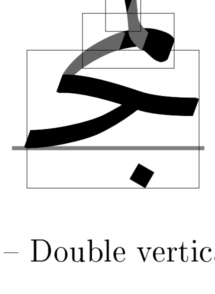

FIG. 2.15 – Double vertical ligature

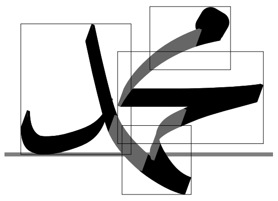

FIG. 2.16 – Triple vertical ligature

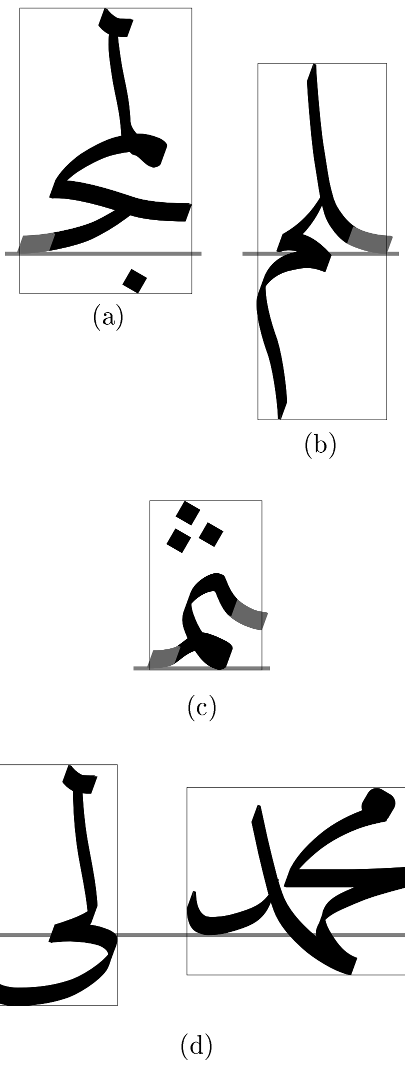

FIG. 2.17 – a) Compound character with horizontal succeeding connection, b) Compound character with horizontal preceding connection, c) Compound character with horizontal succeeding connection and preceding connection, d) Compound characters as standalone letters.

Some difficulties occur in handling vertical ligatures. Solutions can be found through considering the connected letters as a single character [48]. We'll adopt the same strategy concerning this kind of ligature. We will refer to those characters as compound characters. In this way, the compound characters are considered as single characters with horizontal preceding connections, horizontal succeeding connection or both of the two connections. They can also be considered as standalone characters. All of these cases are presented in Figure 2.17. So, the compound characters can accept preceding or succeeding connections or contribute simply as standalone characters in words (See Figure 2.18). They can also contribute in stretching (See Figure 2.19).

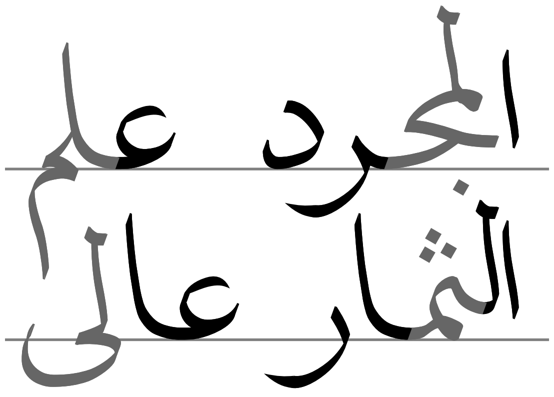

FIG. 2.18 – Contribution of compound characters as simple ones in words

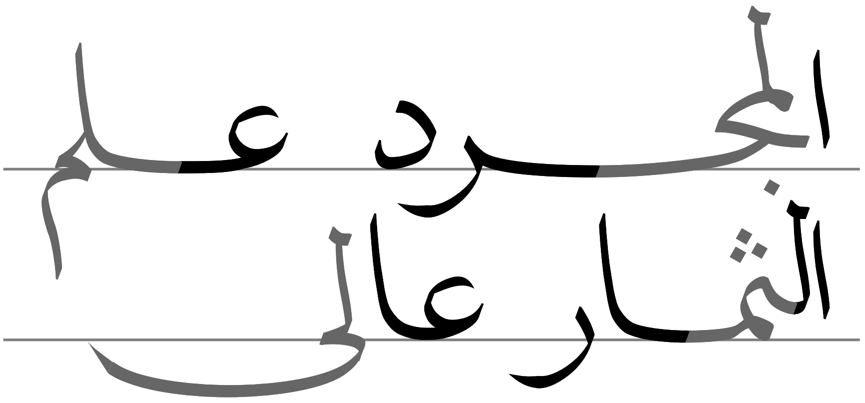

FIG. 2.19 – Contribution of compound characters in stretching

We can remark that the handling of vertical ligatures can be brought to the study of the horizontal ligatures. For that reason, we will give in the following all kinds of horizontal connections and ligatures. We have seen in Subsection 2.2.2 that the movement of nib's head is a translation. As qalam's head is a rigid body, the motion of the qalam can be characterized by one of its corners. We present then the kinds of connections and ligatures in their geometrical forms through considering the trajectory of the left bottom corner of nib's head. In designing the letters of our font, the connections are formed of one cubic Bézier curve. In the following, we suppose that the preceding connections have L_0, L_1, L_2 and L_3 as control points whereas, the succeeding connections have R_0, R_1, R_2 and R_3 as control points. There are various kinds of horizontal

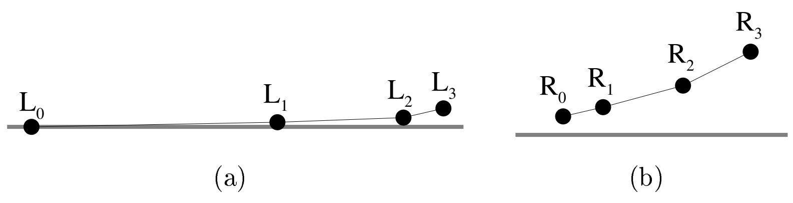

FIG. 2.20 – a) Oblique preceding connection, b) Oblique succeeding connection

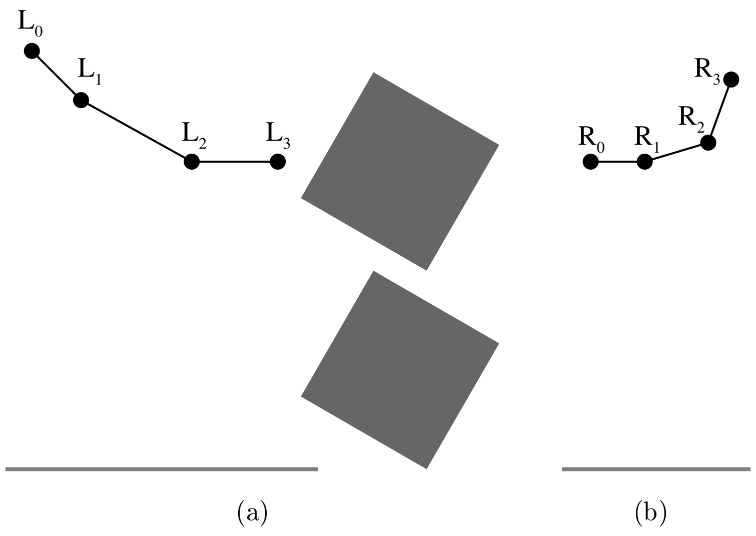

FIG. 2.21 – a) Horizontal preceding connection *big-strictly above* the baseline, b) Horizontal succeeding connection *big-strictly above* the baseline

connections. A lot of them are, in general, not respected for reasons of simplification when that is not due to ignorance. Horizontal connections can be one of the five kinds:

- *Oblique* connection: in Figure 2.20 are presented in the left a preceding connection and in the right a succeeding one. These connections are oblique because the segments $[L_2, L_3]$ and $[R_0, R_1]$ are not parallel to the baseline.
- *Horizontal* connection *big-strictly above* the baseline: a connection is horizontal *big-strictly above* the baseline when the segment $[L_2, L_3]$ for the preceding connection and segment $[R_0, R_1]$ for the succeeding connection are parallel to the baseline and over it about one and 0.54 diacritic Point (See Figure 2.21).
- *Horizontal* connection *strictly above* the baseline: the connections presented in Figure 2.22 are horizontal *strictly above* the baseline because the segments $[L_2, L_3]$ and $[R_0, R_1]$ are parallel to the baseline and over it about a half of a diacritic point (0.55 precisely).
- *Horizontal* connection *exactly on* the baseline: in this case $[L_2, L_3]$ and $[R_0, R_1]$ are exactly on the baseline as in Figure 2.23.

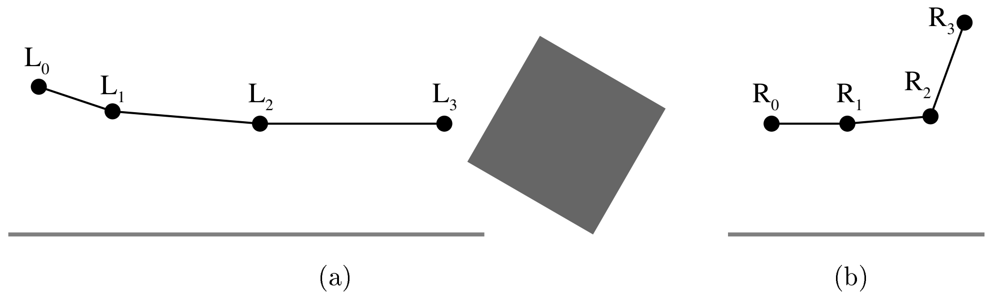

FIG. 2.22 – a) Horizontal preceding connection strictly above the baseline, b) Horizontal succeeding connection strictly above the baseline

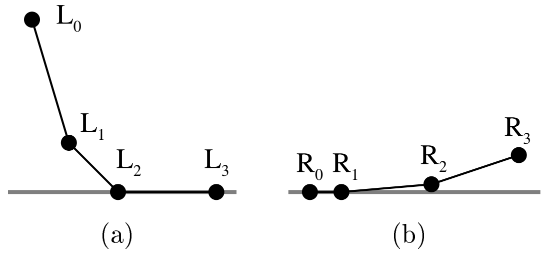

FIG. 2.23 – a) Horizontal preceding connection exactly on the baseline, b) Horizontal succeeding connection *exactly on* the baseline

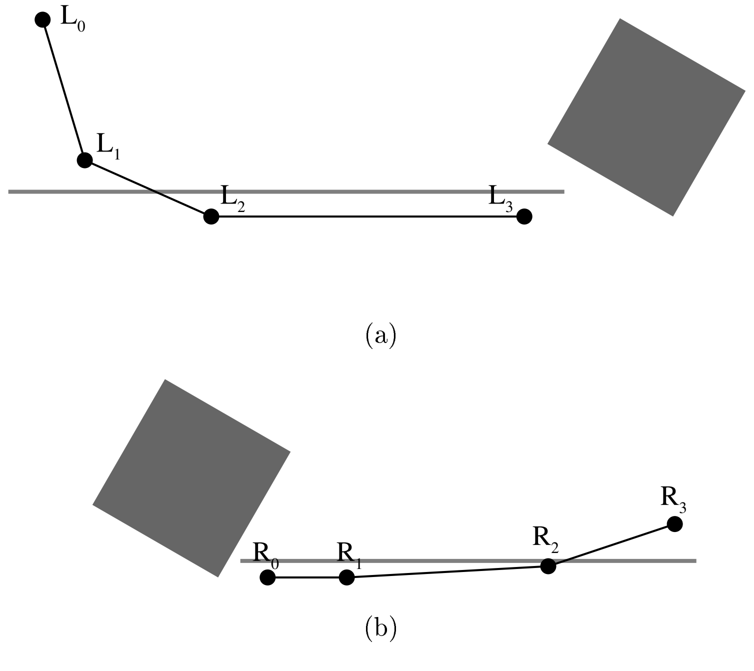

FIG. 2.24 – a) Horizontal preceding connection below the baseline, b) Horizontal succeeding connection below the baseline

– *Horizontal* connection *below* the baseline: $[L_2, L_3]$ and $[R_0, R_1]$ are parallel to the baseline and below it, about an amount that goes from zero to a half of diacritic point. In general, the depth of this connection depends on its width.

These connections are used in certain cases of Kashida. An example is given in Figure 2.24.

Now, we can define what does *horizontal ligature* means geometrically. An horizontal ligature is a combination of two horizontal connections C_1 and C_2 such that:

- C_1 and C_2 are of the same kind (oblique, big strictly above the baseline, ... etc.),
- C_1 is a preceding connection,
- C_2 is a succeeding connection and
- if L_0, L_1, L_2 and L_3 are the control points of C_1, and R_0, R_1, R_2 and R_3 are control points of C_2 then L_3 and R_0 coincide and, L_2, L_3 and R_1 are aligned.

Note that the kind of the ligature is entirely determined by the kind of its connections. So, there are five kinds of horizontal ligatures. A font respecting a minimum of Naskh calligraphic rules may support at least the *horizontal connections (ligatures) exactly on the base line* and *below the baseline*. The connections below the baseline are used to stretch. We can state that these two kinds of horizontal ligatures are mandatory whereas other horizontal and vertical ligatures are rather aesthetic. However, Arabic handbooks calligraphy show more aesthetic ligatures than mandatory ones. That's one of the motivations behind the design of a font that allow producing documents closer to handwritten texts. The horizontal *oblique* connections *big strictly above the baseline*, *strictly above the baseline* and *strictly below the baseline* are not supported in [30, 15, 10, 49].

2.3 Curvilinear stretching in Arabic calligraphy

2.3.1 Stretching in Arabic calligraphy

As it has been mentioned before, Kashida is a small flowing curve that stretches characters or junctions between characters in Arabic-alphabet based scripts. It is used, for instance for justification. So, instead of inserting spaces (blanks) between words, characters and ligatures are stretched according to a structured set of calligraphic rules. More rules of justification in Arabic documents can be found in [40]. In that paper, the authors present most of the existing methods of justifications especially Kashida. In our work, we give a method to implement Kashida in a PostScript font. This stretching goes in both the horizontal and vertical directions. Horizontal curvilinear stretching can go to a maximal value of 12 diacritic points. To this horizontal maximal value corresponds a maximal vertical stretching of a half diacritic point downwards. Moreover, we can distinguish two kinds of stretching:

- Stretching the ligature between two connected letters and

– Stretching inside a letter.

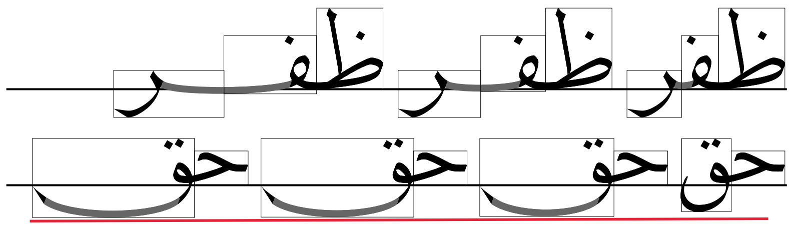

FIG. 2.25 – Two curvilinear stretching in Arabic calligraphy

For instance, consider the words presented in Figure 2.25. The first line displays the word ظفر with three extensions, zero, 6 and 12 diacritic points (from right-to left). Kashida takes place between the letters ظ and ر. In the following line, the word حق is stretched with zero, 9, 11 and 12 diacritic points. Then, the stretching is performed inside the last letter QAF ق. In the calligraphy books [7, 35, 39], the letter QAF without stretching is different from the stretched one. That is also true for all letters that undergo stretching inside them. This property hadn't been respected in [15] (See Figure 2.28 and 2.25 to compare). In Figure 2.25, Kashida is darken in gray and the boxes are reproduced for more clarity. Most of Arabic letters, are composed in two parts. One is static and an other that is dynamic. The dynamic part is the set of curves that undergo really the stretching. As example, we have to look to the letter ظ in Figure 2.25. The static part is in black whereas the dynamic one is in gray.

2.3.2 Dynamic fonts and the support of stretching

The *stretching amount* is the difference between the length of the current line and the sum of the lengths of the words added to the sum of the normal blanks (a normal blank is the minimal blank used to separate normally two words). This stretching amount depends on the line context. A good font to support the stretching may allow continuous variations in stretching. We have here the same need in type of font like in [15, 23, 24], a dynamic font.

2.3.3 Implications for the design of the font

A font with a good support for the stretchability may be based on a language that provides the possibility to use variables in character procedures that can be computed on the fly (dynamically). Postscript dynamic fonts [22] and TrueType fonts [19] are examples of such kind of fonts. As we are here extending and improving what has been done in [15], we will rather opt for PostScript. The PostScript fonts Type 3, as proposed in [22] use the full PostScript language and violates the assumption for Type 1 fonts [4], namely the printing in a fast way thanks to the concept of caching the character bitmaps for later usage, the ability to provide hints to improve low resolution or small point size rasterization and also the ability to be handled by the Adobe Type Manager (ATM). Here I agree with the opinion of the author of [15] that the beauty of what can be done with dynamic fonts, and so write an Arabic text with Kashida outweighs the disadvantages. Also, nowadays, the computers inside printers are so fast and so large that the time to move the paper inside the printer dominates the printing speed, and the *efficiency* benefits of Type 1 fonts are gone.

2.3.4 Characteristics of the stretching in Arabic calligraphy

In Naskh style, nib's head behaves as a rectangle of width l and thickness $e = \frac{l}{6}$. This rectangle moves with a constant inclination angle of about 70° with regard to the baseline. A nib's head with $l = 12$ mm (and $e = 2$ mm) is presented in Figure 2.26.

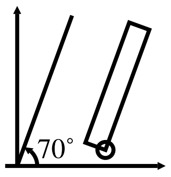

FIG. 2.26 – Nib's head in Naskh style with 12 mm width

As mentioned before, the stretching goes in both the horizontal and vertical directions. In addition, the stretching ought to respect the following constraints:

- The horizontal stretching can go from 0 to 12 diacritic points (the dot . in the letters ظ and ـغـ is a diacritic point and it can be used as a metric unit).
- The vertical stretching depends on the horizontal one and varies from 0 to half of a diacritic point. The stretching model provided in [15] considers stretching only in the horizontal direction.
- In the connection of two stretched zones, the state of continuity of degrees 0 and 1 is preserved. This will be well explained through considering the Bézier geometrical representations. Prematurely, the reader can see Figures 2.42 and 2.47. The set of curves in Figure 2.47 is obtained after a stretching in the set of curves in Figure 2.42. We can remark that the angle on the point L_{10} is conserved on L_{20} and the alignment on R_{13} is conserved on R_{23}

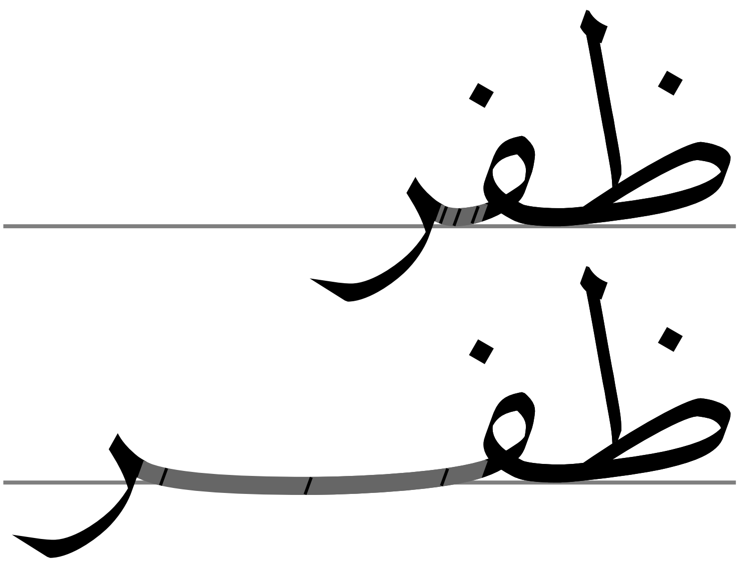

FIG. 2.27 – The optical scaling characteristics of the Arabic curvilinear stretching

In Figure 2.27, the word ظفر is presented with a null stretching in the first line, whereas, in the second line the same word is displayed with an horizontal stretching of 12 diacritic points and the corresponding vertical stretching that is a half of a diacritic point. Nib's head (in black) appears in the gray of Kashida. We can remark that the thickness (in the direction of nib's head inclination) doesn't change and consequently, the stretching model is an optical scaling in opposition to the linear scaling (See [24] about optical scaling). In the following, the stretching model is presented in the same way.

Let us look at the relationship between the external curves delimiting the surface razed with nib's head. We have to darken the surface taking into account this relationship. That has not been considered in [15]. In [15], particular curves for representing stretchable pieces of the characters were to find out. The approach adopted in [15] to stretch letters can lead sometimes, to write some parts of the letter, in a width exceeding, or exceeded by, nib's head width. See, for instance, the stretched QAF in [15, p.1433] (See Figure 2.28). The figure 2.28 presents the letter QAF with a null stretching and a stretched QAF from [15] (a slightly different variant from the same letter). We tried to show the same states (for the same parameters of the Bézier curves representing the contours) of nib's head in the two cases of the letter QAF. We remark that in 2.28(a), the width of nib's head is approximately equal to the surface. The letter QAF with a null stretching is a genuine *written* QAF. In the stretched QAF (See Figure 2.28(b)), on the right, nib's head width exceeds largely the surface darkened of the letter and conversely on the left of the same letter. So, some parts that are written in the null stretching case can not be obtained when the letter is stretched unless they are *drawn*. This is due to the curves chosen to stretch. Actually, these curves should verify a dependency property since they represent trajectories of two corners of nib's head following the same movement with respect to a translation (taking into account the motion in shading the surface razed with nib's head). And now, we'll consider that:

The character is defined by the curves representing nib's movement, not by those representing the limits of the surface razed with the nib.

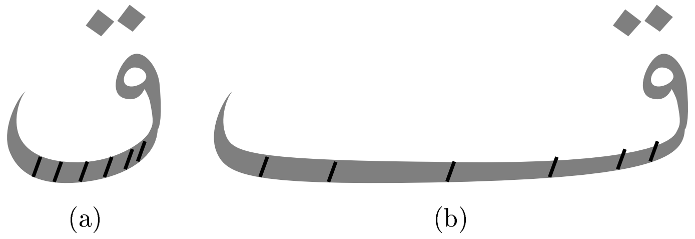

FIG. 2.28 – a) The standalone QAF in Daniel Berry font, b) A stretching of standalone QAF in Daniel Berry font

This means that a character results from blacking the surface razed with the rectangle representing nib's head. As the four corners of nib's head follow the same movement, the movement can be represented by the curve corresponding to the way followed by one of the four corners. Say for instance, the extremity surrounded with a small circle in Figure 2.26. Of course, the movement of this corner when writing a character is determined by a set of Bézier curves. So, we'll consider a unique Bézier curve.

Let B_1 be a Bézier curve [42] with the four control points M_{10}, M_{11}, M_{12} and M_{13} (See Figure 2.29), the curve B_1 represents the movement of the right bottom corner of the nib. Width, thickness and inclination angle of the nib are respectively denoted by l, e and α.

Consider B_2, the Bézier curve with the four control points M_{23}, M_{22}, M_{21} and M_{20} such that:
$M_{2i} = t_{\overrightarrow{u}}\left(M_{1i}\right)$, $i \in \{0,1,2,3\}$ where $t_{\overrightarrow{u}}$ is the translation of vector $\overrightarrow{u}$ such that $\overrightarrow{u} = \overrightarrow{\left(l\cos\left(\alpha\right), l\sin\left(\alpha\right)\right)}$.

The surface S_{l1} delimited by the curve B_1, the segment $[M_{13}, M_{23}]$, the curve B_2 and the segment $[M_{20}, M_{10}]$ coincides exactly with the surface razed by the edge l_1. The surface is filled in light gray in Figure 2.29.

The surface S_{l2} razed by the edge l_2 can be obtained through translating the surface S_{l1} with vector $\overrightarrow{v} = \left(l\cos\left(\alpha + \frac{\pi}{2}\right), l\sin\left(\alpha + \frac{\pi}{2}\right)\right)$ (See Figure 2.30).

Let $S_{l2} = t_{\overrightarrow{v}}\left(S_{l1}\right)$ be the result of this translation.

to get the whole surface razed with all nib's head, the surfaces S_{e1} and S_{e2} are also to

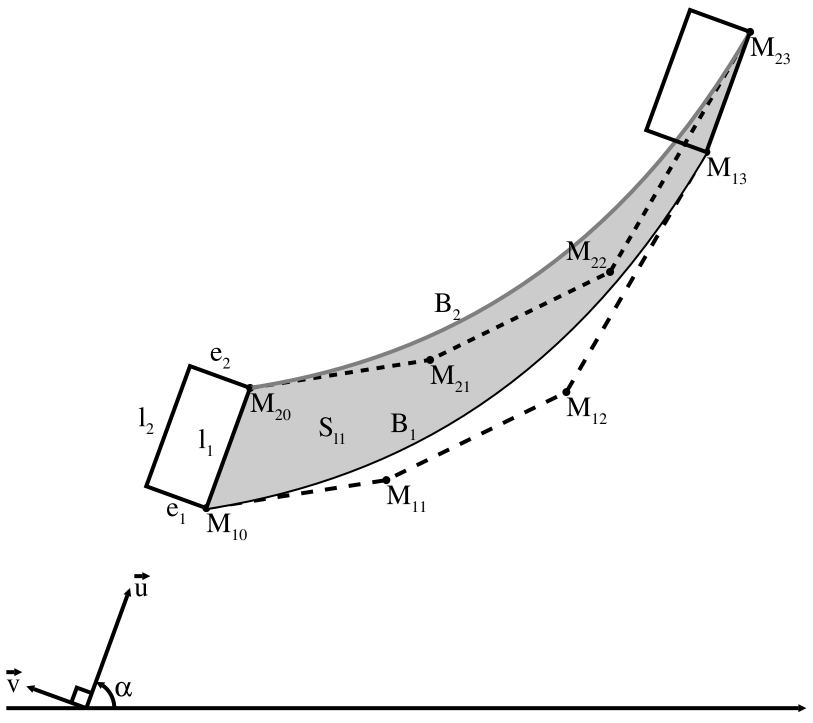

FIG. 2.29 – Surface razed with the edge l_1 (light gray)

be taken into account (See Figure 2.31) so that we get:
S_{e1}: surface bounded by B_1 , $[M_{13}, M_{33}]$, B_3 and $[M_{30}, M_{10}]$, that is the surface razed by edge e_1.
B_3: a Bézier curve with control points M_{33}, M_{32}, M_{31} and M_{30} so that:
$M_{3i} = t_{\overrightarrow{v}}\left(M_{1i}\right)$, $i \in \{0, 1, 2, 3\}$, and
$S_{e2} = t_{\overrightarrow{u}}\left(S_{e1}\right)$.

These four surfaces allow the reconstitution of the surface razed by nib's head. Figure 2.32 shows an illustration of this situation. It gives the control points of the Bézier curve modeling the movement of the left bottom corner.

This way of blacking the surface razed by the nib does not give always good result. Actually, consider the example in Figure 2.33. In order to simplify, consider a nib with a very thin head (with negligible thickness). The surface obtained through applying the technique used previously (See Figure 2.33) does not coincide exactly with the surface razed with that nib (See Figure 2.34). The geometric reason why it is so is the existence of an element $t_0 \in]0, 1[$ such that $B_1'(t_0)$ ($B_1'(t_0)$ is a vector) and the vector $\overrightarrow{M_{10}M_{20}}$ are collinear (such condition was not satisfied in the case of the previous examples). More formally, consider the function R_1 defined as follows:

$$\begin{array}{rcl} R_1 : [0,1] & \rightarrow & \mathbb{R} \\ t & \mapsto & \left(\overrightarrow{M_{10}M_{20}} \wedge \overrightarrow{OB_1(t)}\right) \cdot \overrightarrow{k} \end{array}$$

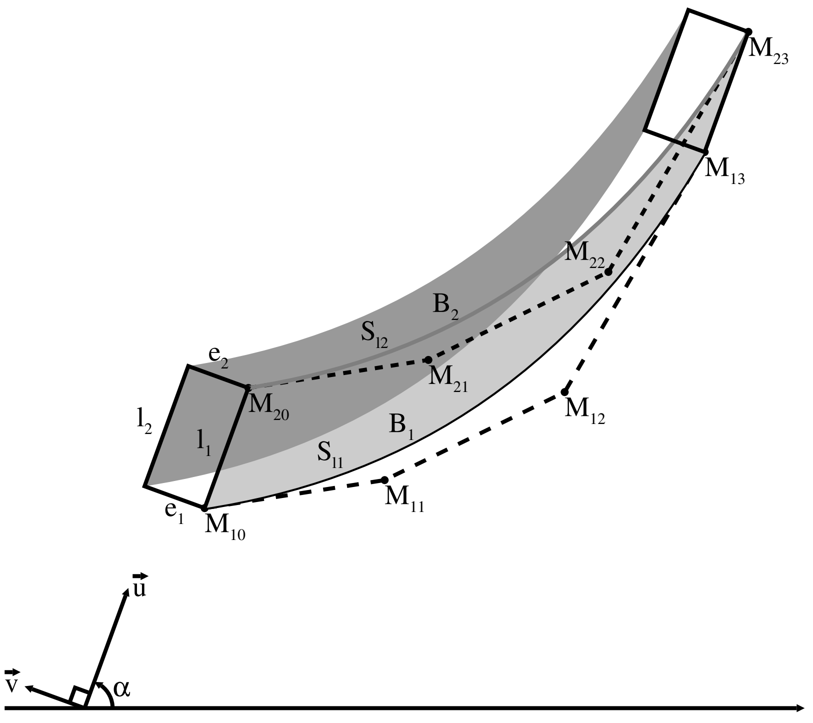

FIG. 2.30 – Surface razed with the edge l_2 (little dark gray)

when the considered Bézier's curves are defined on the affine plane $\mathbb{R}^2$ reported to the orthonormal basis $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}\right)$ containing the origin O and that is a sub-space of the affine space $\mathbb{R}^3$ with the orthonormal basis $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$ and the operator $\wedge$ stands for the vectorial (cross) product.
$R_1(t)$ is the direct orthogonal range of the Bézier curve B_1 according to the vector $\overrightarrow{k}$. if $R_1(t)$ vanishes at $t_0 \in]0,1[$, that means that $B_1'(t_0)$ and $\overrightarrow{M_{10}M_{20}}$ are superposed.

As that $R_1'(t) = \left(\left(\overrightarrow{M_{10}M_{20}} \wedge \overrightarrow{OB_1(t)}\right) \cdot \overrightarrow{k}\right)'(t) = \left(\overrightarrow{M_{10}M_{20}} \wedge B_1'(t)\right) \cdot \overrightarrow{k}$, we get the interpretation:

- if R_1 is monotone on an interval $[0,1]$ then the razed surface can be obtained through applying the method mentioned before,
- the method fails if monotony's sense of R_1 changes.

We can remark that the study of the function R_1 can help to blacken the surface razed in Figure 2.34. The curve B_1 is a perfectly determined Bézier curve. The study of R_1 over $[0,1]$ shows that R_1 admits an extrema at $t_0 = 0.6184$ and monotony's sense doesn't change over the sub-intervals $[0, 0.6184]$ and $[0.6184, 1]$.

Now, let us decompose B_1 with respect to the generalized Bézier algorithm of refinement (case of no median decomposition) [14, 25, 36, 45] with respect to the coefficient $t_0 = 0.6184$. So we get two curves B_{11} and B_{12} whose corresponding razing can be obtained through proceeding as in the example in Figure 2.29. The two razings are in Figures 2.35 and 2.36.

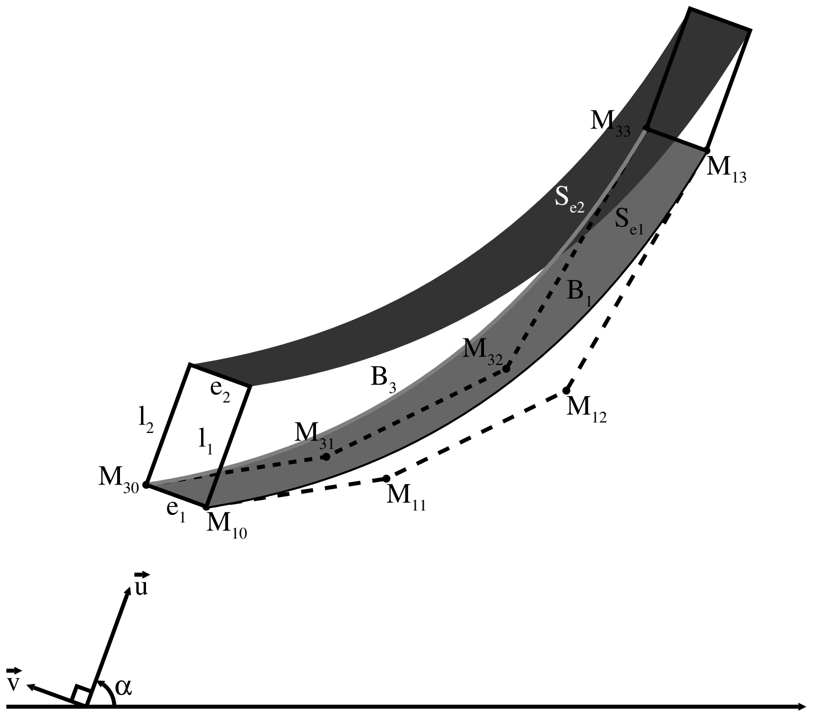

FIG. 2.31 – Surfaces razed with the two edges e_1 and e_2

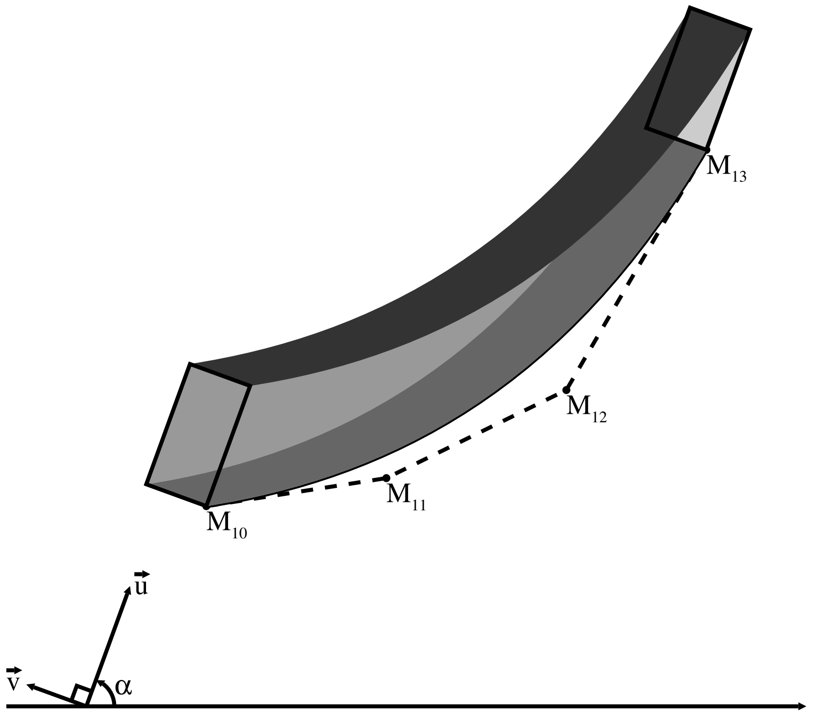

FIG. 2.32 – Surface razed with the entire nib

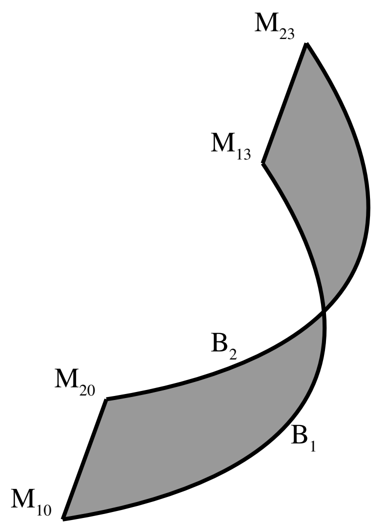

FIG. 2.33 – The result of filling in the surface between the two curves with the previous technique

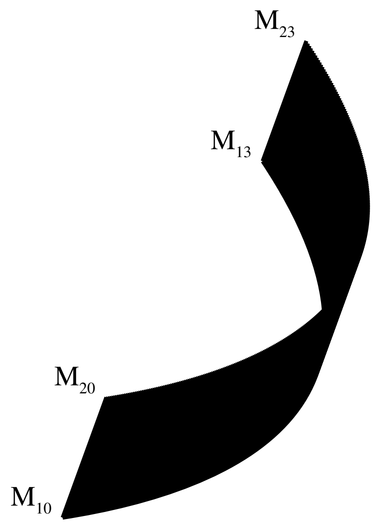

FIG. 2.34 – Surface razed with the nib

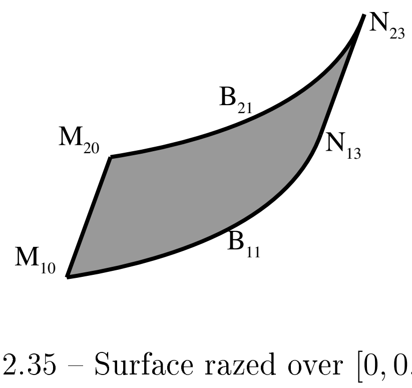

FIG. 2.35 – Surface razed over $[0, 0.6184]$

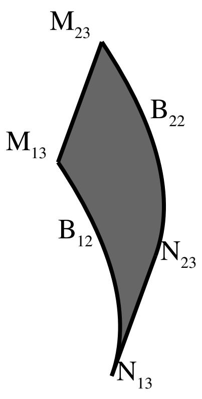

FIG. 2.36 – Surface razed over $[0.6184, 1]$

The two surfaces overlap and reconstitute a surface that fit exactly with the surface razed with the segment $[M_{10}M_{20}]$ (See Figure 2.37). This way of proceeding can be very useful for the design of characters shapes in PostScript or METAFONT so that the trajectories of the vertices nib overlap.

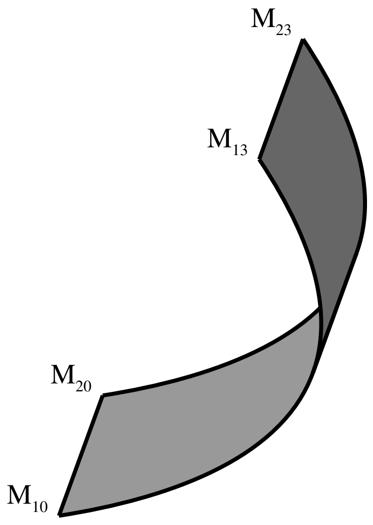

FIG. 2.37 – Superposed razed surfaces on $[0, 1]$

The difficulties to find out a way to represent and to *coat* nib's head movement reminds those met in Knuth's approach in METAFONT [17] or those found in Kinch's approach in MetaFog [43]. The comparison will be considered in Subsection 2.4.3.

2.4 Model and support of kashida in Arabic calligraphy

2.4.1 Kashida: the mathematical model

In this subsection, we give the geometrical representation and processing of the curvilinear stretching. In order to present how to handle geometrically Kashida, we'll first define the way of stretching some particular curves that constitute basic elements of kashida. Consider the following notations:

- The notation $[M_0, M_1, M_2, M_3]$: is the Bézier curve with control points M_0, M_1, M_2 and M_3,
- The set $\mathcal{B}_1$: is the set of Bézier curves $[M_0, M_1, M_2, M_3]$ with an invariant concavity verifying:
  $\overrightarrow{M_2M_3} = \lambda \overrightarrow{i}$, $\lambda \in \mathbb{R}_+^*$ and $0 \leq \left(\widehat{\overrightarrow{M_0M_1}, \overrightarrow{i}}\right) \leq \frac{\pi}{2}$, (See Figure 2.38), where $\overrightarrow{i}$ is the axis X director vector and $(\widehat{\overrightarrow{u}, \overrightarrow{v}})$ stands for the angle between the two vectors $\overrightarrow{u}$ and $\overrightarrow{v}$.

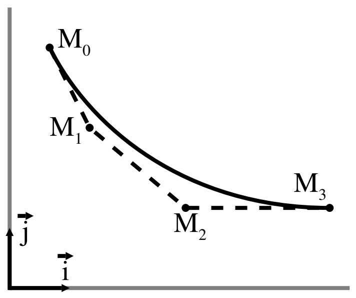

FIG. 2.38 – Curves of type 1 (in $\mathcal{B}_1$)

- The set $\mathcal{B}_2$: is the set of Bézier curves $[M_0, M_1, M_2, M_3]$ with an invariant concavity verifying:
  $\overrightarrow{M_0M_1} = \lambda \overrightarrow{i}$, $\lambda \in \mathbb{R}_+^*$ and $0 \leq \left(\widehat{-\overrightarrow{i}, \overrightarrow{M_3M_2}}\right) \leq \frac{\pi}{2}$ (See Figure 2.39).

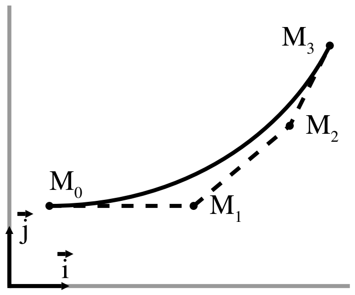

FIG. 2.39 – Curves of Type 2 (in $\mathcal{B}_2$)

The kashida is a juxtaposition of two Bézier curves, B_1 from the set $\mathcal{B}_1$ and B_2 from the set $\mathcal{B}_2$. If L_0, L_1, L_2 and L_3 are the control points of B_1 and R_0, R_1, R_2 and R_3 are the control points of B_2 then L_3 and R_0 are equal. Kashida's stretching results from the definition of the functions that stretch curves in $\mathcal{B}_1$ and $\mathcal{B}_2$. So, we give the definition of two stretching functions E_{be} that stretches curves in $\mathcal{B}_1$ and E_{af} that stretches curves in $\mathcal{B}_2$ as follows. Let E_{be} be the stretching function defined by:

$$\begin{array}{rrcl} E_{be} : & \mathcal{B}_1 \times [0, h_m] \times [0, v_m] & \rightarrow & \mathcal{B}_1 \\ & (B, h, v) & \mapsto & E_{be}(B, h, v) \end{array}$$

The transformation E_{be} stretches curves in $\mathcal{B}_1$. The details of its definition are given below:
Let $B_1 = [M_{10}, M_{11}, M_{12}, M_{13}]$ and $B_2 = [M_{20}, M_{21}, M_{22}, M_{23}]$ be two curves in $\mathcal{B}_1$.
Let $(h, v) \in [0, h_m] \times [0, v_m]$. h stands for the horizontal stretching amount whereas v is the vertical one.

Definition 2.4.1 (E_{be} transformation)**.**
$B_2 = E_{be}(B_1, h, v)$ if and only if the control points of B_2 are:

$M_{20} = M_{10} - (h, 0)$
$M_{23} = M_{13} - (0, v)$
$M_{21} = (1 - c_1) M_{20} + c_1 J$
$M_{22} = (1 - c_2) J + c_2 M_{23}$

With $c_1, c_2 \in [0, 1]$ satisfying:

$M_{11} = (1 - c_1) M_{10} + c_1 I$ *and*
$M_{12} = (1 - c_2) I + c_2 M_{13}$

Where

$\{I\} = (M_{10}M_{11}) \cap (M_{12}M_{13})$

and

$\{J\} = \Delta_1 \cap \Delta_2$ *with*

Δ_1*: the parallel to* $(M_{10}M_{11})$ *passing through the point* M_{20}*,*
Δ_2*: the parallel to* $(M_{12}M_{13})$ *passing through the point* M_{23}*.*

An example of a stretching using the function E_{be} is presented in Figure 2.40.

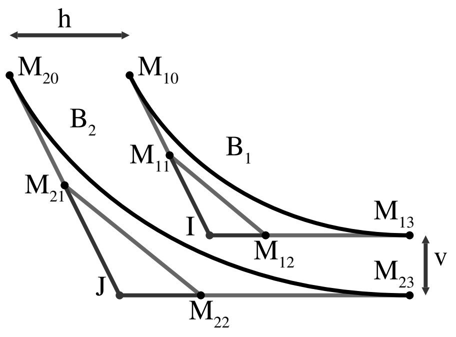

FIG. 2.40 – Stretching a curve belonging to the set $\mathcal{B}_1$ with E_{be} $(E_{be}(B_1) = B_2\ )$

Now, we have a function that allows stretching a curve belonging to the set $\mathcal{B}_1$, what about curves in $\mathcal{B}_2$?

Let E_{af} be the stretching function defined as follows:

$$\begin{array}{rcrcl} E_{af} & : & \mathcal{B}_2 \times [0, h_m] \times [0, v_m] & \rightarrow & \mathcal{B}_2 \\ & & (B, h, v) & \mapsto & E_{af}(B, h, v) \end{array}$$

The details of the definition of E_{af} are:
Let $B_1 = [M_{10}, M_{11}, M_{12}, M_{13}]$ and $B_2 = [M_{20}, M_{21}, M_{22}, M_{23}]$ be in $\mathcal{B}_2$.
Let $(h, v) \in [0, h_m] \times [0, v_m]$.

Definition 2.4.2 (E_{af} transformation)**.**
$B_2 = E_{af}(B_1, h, v)$ *if and only if the control points of* B_2 *are:*
$M_{20} = M_{10} - (h, v)$
$M_{23} = M_{13}$
$M_{21} = (1 - c_1) M_{20} + c_1 J$
$M_{22} = (1 - c_2) J + c_2 M_{23}$

With $c_1, c_2 \in [0,1]$ *satisfying:*

$$M_{11} = (1 - c_1)\, M_{10} + c_1 I$$
$$M_{12} = (1 - c_2)\, I + c_2 M_{13}$$

Where

$\{I\} = (M_{10}M_{11}) \cap (M_{12}M_{13})$ *and*
$\{J\} = \Delta_1 \cap \Delta_2$

with

Δ_1*: is the parallel to* $(M_{10}M_{11})$ *passing through the point* M_{20}*,*
Δ_2*: is* $(M_{12}M_{13})$*.*

As for the function E_{be}, an example of stretching with the function E_{af} is given in Figure 2.41.

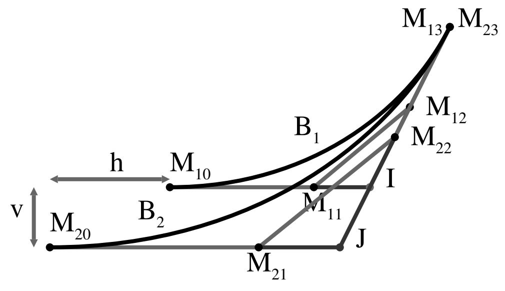

FIG. 2.41 – Stretching a curve belonging to the set $\mathcal{B}_2$ with E_{af} ($E_{af}(B_1) = B_2$)

Now, consider a set of curves $\mathcal{C}$ (See Figure 2.42) to be stretched. More exactly, the stretching will be done with a kashida modeled with the curves B_{11} and B_{21}. Consider the following curves:

C_1: a curve bound to the left of B_{11} (See Figure 2.42),

C_2: a curve bound to the right of B_{21}.

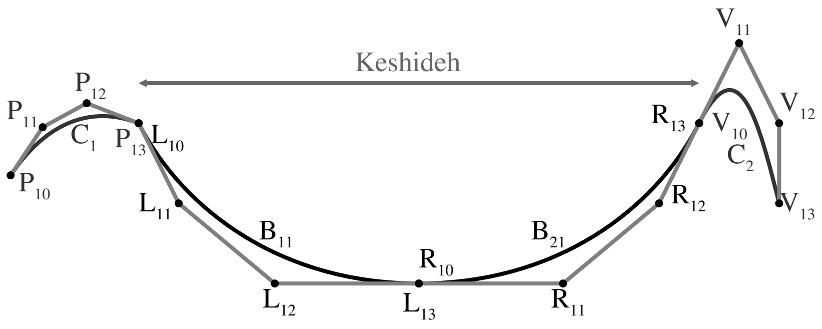

FIG. 2.42 – initial state of a set of curves $\mathcal{C}$ to be stretched

The stretching or Kashida with regard to h horizontally and v vertically results from the following transformations:

- $E_{af}\left(B_{21}, \frac{h}{2}, v\right)$ to get B_{22} (See Figure 2.43),

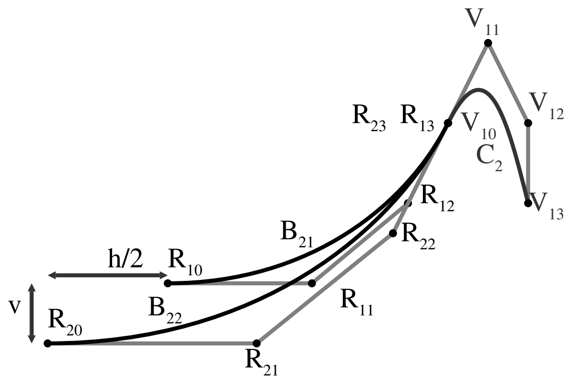

FIG. 2.43 – Stretching the curve B_{21} of $\frac{h}{2}$ and v

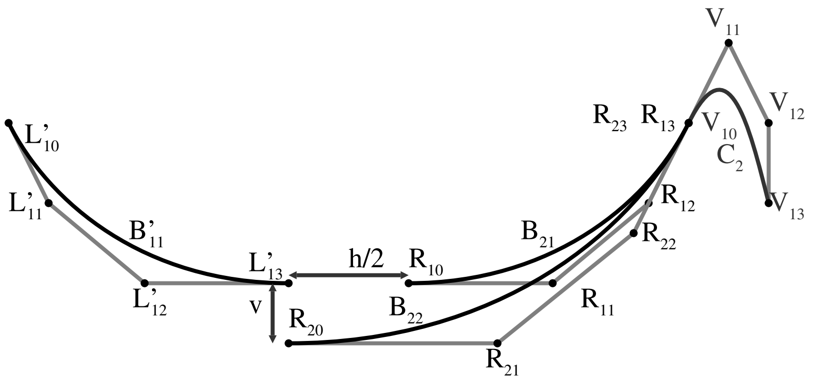

FIG. 2.44 – Translation of B_{11} towards the right-side of $\frac{h}{2}$

- $t_{\overrightarrow{u}}\left(B_{11}\right)$ to get B'_{11}, where $t_{\overrightarrow{u}}$ is a translation of vector $\overrightarrow{u}=\overrightarrow{\left(-\frac{h}{2},0\right)}$ (See Figure 2.44),
- $E_{be}\left(B'_{11},\frac{h}{2},v\right)$ to have B_{12} (See Figure 2.45)

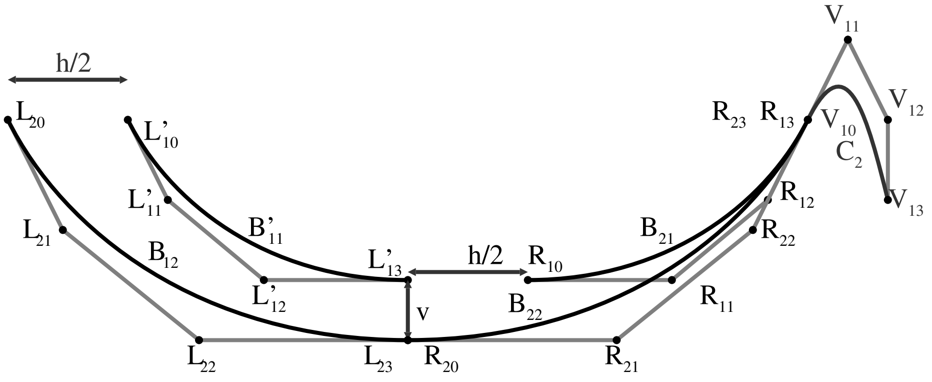

FIG. 2.45 – Stretching the curve B'_{11} of $\frac{h}{2}$ and v

- $t_{\overrightarrow{v}}\left(C_{1}\right)$ to have C'_{1}, where $t_{\overrightarrow{v}}$ is a translation of vector $\overrightarrow{v}=\overrightarrow{\left(-h,0\right)}$ (See Figure 2.46).

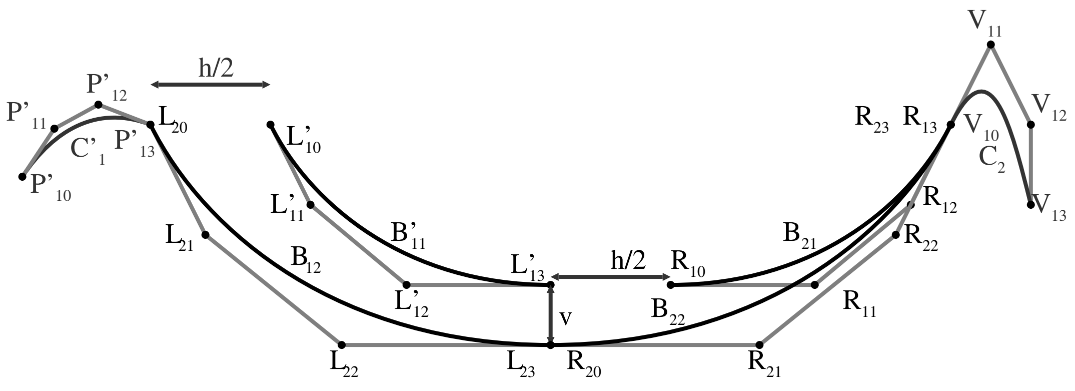

FIG. 2.46 – Translation of C_1 towards the right direction in h

After all these transformations, the set of curves $\mathcal{C}$ gives a stretching as in Figure 2.47.

In the PostScript encoding, not all these transformations are to be handled explicitly. In some cases, some of them (some translations) are automatically processed with the PostScript interpreter. We remark that before stretching, R_{12}, R_{13} and V_{11} are aligned. After the stretching R_{22}, R_{23} and V_{11} are aligned too. The angles $(\widehat{L_{10}P_{12}, L_{10}}L_{11})$ and $(\widehat{L_{20}P'_{12}, L_{20}}L_{21})$ are equal. The derivatives of degree 0 and 1 are preserved in the points where Kashida is connected to other curve sets.

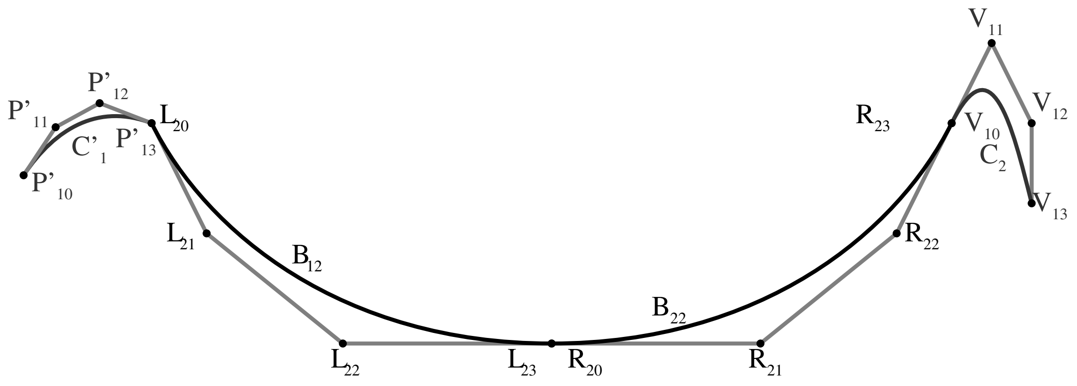

FIG. 2.47 – The set $\mathcal{C}$ after stretching of h and v

2.4.2 Kashida: the support in the font

As it has been said before, Kashida is a mean that stretches and/or connects Arabic letters. Even though it is not a character it is supported as parts of the stretched letters. And, as it has also been said before, the stretching can appear inside a letter or into two connected letters (See Figure 2.25). When a word, in a text, is to be stretched, if the last letter of this word is a stretchable one, the stretching occurs in this letter otherwise the stretching will be in the ligature between the two last connected letters of the word. If the kashida is inside a letter we will call it “intra-letter kashida”. When it implies two connected letters, we will call it “inter-letters kashida”. Here, we have to understand that “inter-letter kashida” dœsn’t mean that the kashida is a curve or a character that we add between the two connected letters but it implies directly the two connected letters. The two cases are presented separately in the following.

2.4.2.1 Inter-letter Kashida

Kashida is a juxtaposition of two curves of Type 1 and Type 2. When Kashida appears between two connected letters, the first curve is the horizontal preceding connection below the baseline of the letter in the left and the second part of Kashida is the horizontal succeeding connection below the baseline of the letter in the right. When we designed the font, we parametrized the horizontal connections exactly on the baseline in a way such that they can be stretched horizontally to the right and vertically downwards. So, when the stretching is null, the connections are simply those exactly on the baseline, otherwise they are horizontal strictly below the baseline. From now, we mean by parametrized horizontal connection exactly on the baseline a connection that support the connection exactly on the baseline and below the baseline. For example, when the word ظفر is to be stretched, the stretching occurs between the letters ﻔ and

ﺮ. The letters ﻔ and ﺮ have horizontal connections exactly on the baseline that are parametrized to support stretchability. Then, the stretching consists of keeping the word as it is and communicating horizontal and vertical amounts of stretching as parameters to the procedures for writing these two letters. Suppose that we want to stretch the word ظفر of an amount h horizontally and v vertically downwards, then, we have to perform the two actions:

- to stretch the horizontal succeeding connection of the letter ﻔ of $\frac{h}{2}$ horizontally and v vertically and
- to stretch the horizontal preceding connection of the letter ﺮ of $\frac{h}{2}$ horizontally and v vertically.

In Figure 2.48, an illustration of a stretching in a font size equal to 64 points is given. In the first line, the word ظفر with a null stretching is presented. In the last line, the same word is stretched 12 dp (**d**iacritic **p**oint) horizontally and 0.5 dp vertically. In the second line the letters ﻔ and ﺮ in their initial shapes are showed whereas in the third line, both of the two letters have a stretching of 6 dp horizontally and 0.5 dp vertically.

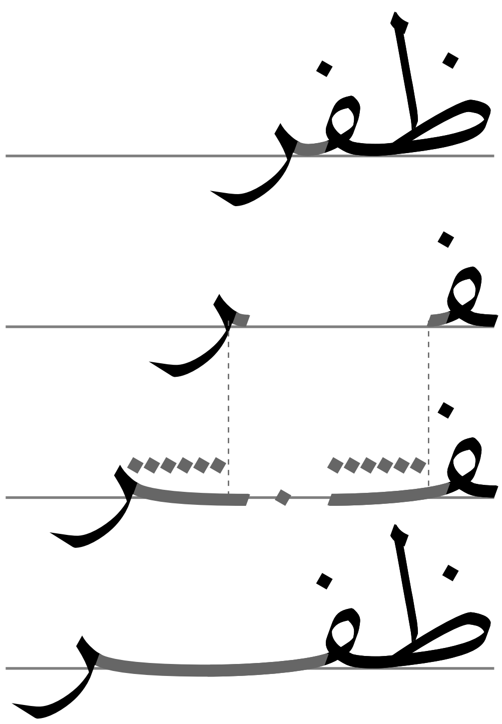

FIG. 2.48 – Inter-letter kashida mechanisms

Most of the handbooks of calligraphy [7, 35, 39], teach that: in the beginning of a word in Arabic letters, the commonly used horizontal connections are oblique, big strictly above the baseline and strictly above the baseline. That's an aesthetic need. When we have to write the word formed of the letters BEH in initial position and AIN in the end without stretching, a letter BEH and AIN with horizontal succeeding connection and preceding connection respectively strictly above the baseline are used. Let بع be such a word (the word to the right in Figure 2.49), when a stretching is needed, the alternatives of these two letters with horizontal connections below the baseline (they are horizontal connections exactly on the baseline when the amount stretching is null) are used. In this way, the word بع would be used. The two alternatives of the word are in clear in Figure 2.49. We can introduce these alternatives in the font to solve the problem of stretchability needs so that the texts processed with the font satisfies the calligraphic rules. Of course, these alternatives are allowed by the calligraphic rules and exist. Consider the word بد, the letter بـ admits an horizontal connection after *exactly on* the baseline because ـد is a strongly rising letter from the baseline. It is the same for the letter ـع that admits a preceding connection exactly on the baseline. It is mandatory in a situation such as after the letter ط in the word اطع or after فـ inside the word نفع. We can apply the alternatives of letters also when the ligature is oblique, in the case of a null stretching. But it doesn't work in some particular situations such that concerning the stretching of the word خط. Stretching the ligature between the two letters of the word خط has been handled in a particular way. Indeed, when some characters such as خـ (or its variants) are in initial position and that they are followed by TAH in final position ـط (or its variants), the connections are *oblique* and then, stretching through using the alternative of the letter KHAH and TAH with parametrized horizontal connections exactly on the base line gives a bad result in comparison with the handwritten proof. It took a lot of time to find a solution to this case. It reminds the case where D.E. Knuth spent a lot of time modeling mathematically the letter S in METAFONT. The cases are not analogous, technically speaking, but in difficulties to find a solution they are. A method to find a solution to this case consists on including the stretching inside the character TAH. We developed an alternative of the TAH in final position with a connection before consisting of two Bézier curves B_l in $\mathcal{B}_1$ with control points L_0, L_1, L_2 and L_3 on the left, followed by another Bézier curve B_r in $\mathcal{B}_2$ with control points R_0, R_1, R_2 and R_3 on the right such L_3 coincides with R_0. Let ـط be that alternative (See Figure 2.51), Kashida will then be the combination of these curves and there is precisely the place where stretching will take place (See Figure 2.52, the first occurrence of the word in the right is with oblique connections). We conclude that to stretch the word خط, we may keep the letter خـ without changes and replace the letter ـط (See Figure 2.50) by ـط (See Figure 2.51) and then, the stretching can be performed. With this alternative the stretching of the word خط has been evaluated by a calligrapher as good in comparison to the handwritten proof.

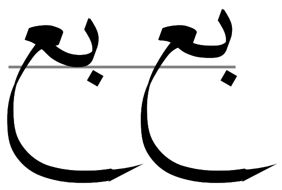

FIG. 2.49 – Useful alternatives to stretch

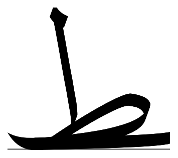

FIG. 2.50 – TAH in final position with *oblique* preceding connection

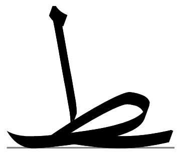

FIG. 2.51 – Alternative for stretching the character TAH in final position proceeded by HAH in initial position

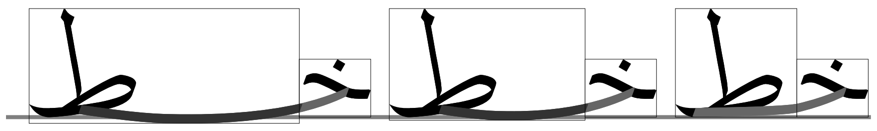

FIG. 2.52 – Stretching in the level of TAH

2.4.2.2 Intra-letter Kashida

When Kashida is inside a character, say for instance, into the letter QAF, in final position of the word حق, according to the Arabic calligraphy rules, the letter QAF admits an horizontal stretching between 9 and 12 diacritic points. The shape of the letter QAF ق without stretching is quite different from the stretched one, the character ـق. ـق is only a variant of ق. So, it has the width of that of ق increased by 9 diacritic points. It also has the depth of that of ق increased by the vertical downward stretching corresponding to the horizontal stretching of 9 diacritic points. To compute the stretching, the letter ـق is used and can be stretched horizontally between zero and three diacritic points. The vertical stretching varies in the range corresponding to the horizontal extension from zero to three diacritic points. Kashida is performed inside ـق. The part modeling Kashida is displayed in gray in Figure 2.53. The letter QAF in final position is in font size 128 points. The control points of the left bottom corner of nib's head are given to make it more understandable.

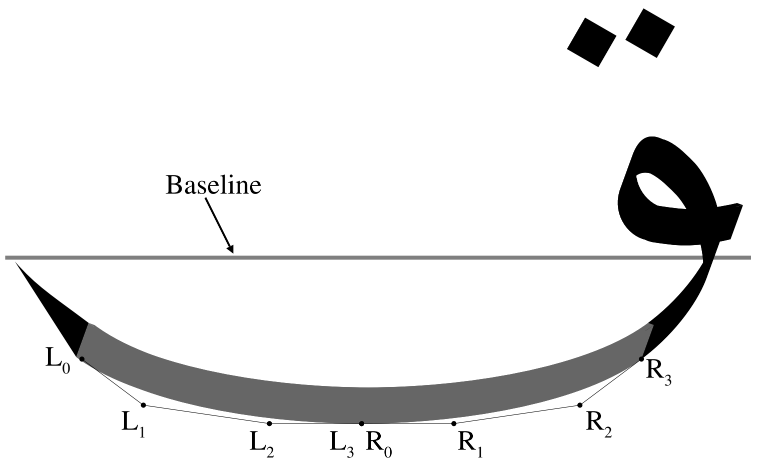

FIG. 2.53 – Kashida in the letter QAF in final position

In the development of our PostScript font, the horizontal stretching is a value communicated to character procedures as a global variable h. As in [15], at the end of each call of procedure, the value of h is put back to zero. The value h is determined by means external to the font, such as the text formatting programs. Then it is communicated to the character procedures in the font. The vertical stretching v is determined from the horizontal stretching in the font through the simple function φ defined in the following:

- h: value of horizontal stretching (towards the right direction),
- v: value of vertical stretching corresponding to h (downward), v depends on h, so we can write $v = \varphi(h)$ where φ is defined by:

$$\begin{array}{rcccl} \varphi & : & [0, h_m] & \to & [0, v_m] \\ & & h & \mapsto & \frac{v_m}{h_m} h \end{array}$$

- h_m: maximal value of horizontal stretching,
- v_m: maximal value of vertical stretching.

With regard to the definition of the function φ, the curvilinear stretching functions E_{af} and E_{be} defined above can be regarded as linear scaling, but they are not so.

2.4.3 Nib's head motion modeling approach versus other approaches

In this subsection, the approach for modeling of nib's motion is compared in one hand to Knuth's approach in METAFONT [17] and to Kinch's one in MetaFog [43] in the other hand. These two approaches seek the envelope of an ellipse stroking (a nib with an elliptic head) in two different ways. Knuth's approach follows the Hobby method [26] to represent the envelope in terms of the raster instead of scalable curves. Kinch solve the problem through representing the envelope in an algebraic and topological way. In our approach, the envelope of the characters static parts is determined using tools outside the PostScript interpreter. A specific program (based on a mathematical idea presented in Subsection 2.3.4) helps to determine this envelope. Of course, we also can use Metapost [27], to determine the envelope and cope with other programs to eliminate the fact that some zones are painted twice. The characters static parts have *true* outlines (there are no overlapping curves), the envelope is generated as with MetaFog. Concerning the characters dynamic parts, such as the parts of Kashida, the envelope is determined in a way different from Knuth's and Kinch's approaches. In order to well explain and justify our approach, let us consider a left component of a kashida (See Figure 2.54).

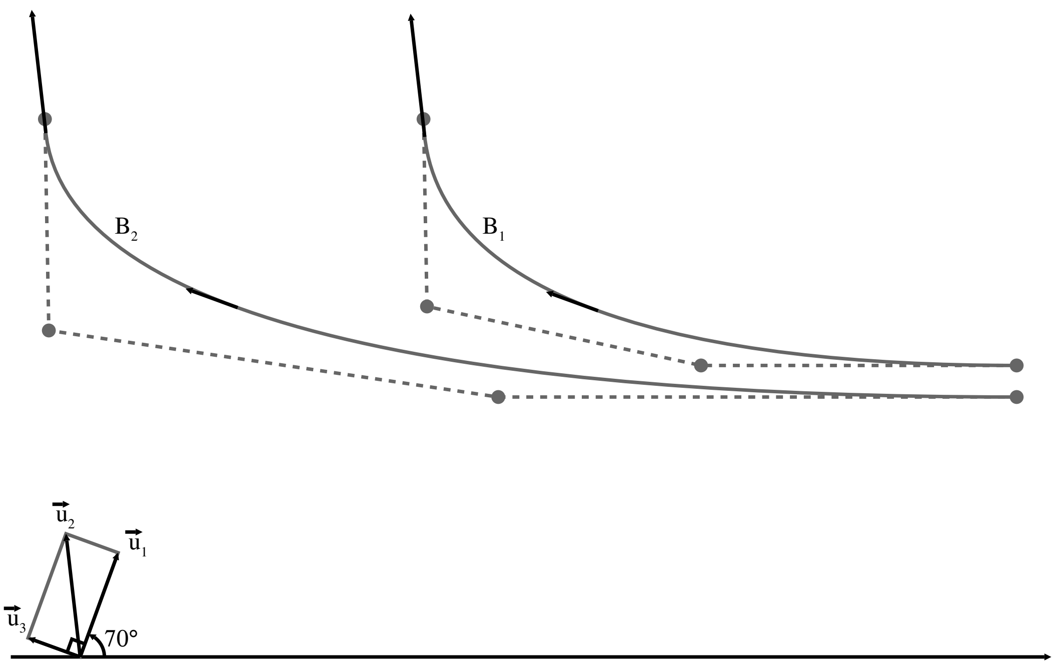

FIG. 2.54 – Identification of points where slopes are parallel to vectors defining nib's head

The curve B_1 of Type 1 in Figure 2.54 has the control points (169,205), (170,151), (249,134) and (340,134). Nib's head width and thickness are 31.9643 and 15.9821 points respectively. Here, we have considered a thickness equal to half of the width for clarity. The points where the slopes are parallel to the vectors defining the nib when the thickness is a sixth of the width are the same. The vectors $\overrightarrow{u}_1$, $\overrightarrow{u}_2$ and $\overrightarrow{u}_3$ are respectively (10.9177,30.042), (-4.10678,35.4913) and (-15.0244,5.44933). $B_1'(t)$ is parallel to $\overrightarrow{u}_2$ at 0.0323 and to $\overrightarrow{u}_3$ at 0.4923. After applying a stretching with the E_{be} to B_1 with an horizontal stretching equal to 109.1601 points and a corresponding vertical one 9.0966 points, we obtain the curve B_2 with the control points (59.8398972,205), (60.9680176,144.081482), (190.549225,124.903397) and (340,124.903397). Then $B_2'(t)$ is parallel to $\overrightarrow{u}_2$ and $\overrightarrow{u}_3$ at 0.0224 and 0.3980 respectively. The coefficients where the slopes are parallel to the vectors of nib's head change from a stretching state to another. The reason for this is simply that the functions of stretching used to compute Kashida are not a *linear scaling.* To determine these coefficients and therefore, to be able to determine the envelope as Kinch's approach seek through computing these coefficients in the PostScript character procedure. Then, printing the character will be very slow. To avoid such problem, we opt for painting the surface razed with every edge of nib's head. So, some parts are painted twice or more causing a larger CPU time consuming but it is still less than the time used to compute the outlines of the curve. As we have seen before, some parts of the surface razed with nib's head can be lost in the neighborhood of the points where the slopes are parallel to the vectors $\overrightarrow{u}_2$ and $\overrightarrow{u}_3$. In Figure 2.55, the real surface razed with nib's head according to the curve B_1 is showed. Whereas, in Figure 2.56, we have drawn the surface that is in Figure 2.55 and then drawn the envelope (in gray) with our adopted technique on it. We remark that we can not distinguish the lost zones (that must be in black). The reason is that Kashida curves have curvature vectors [13] with small magnitudes (Or the radius of the osculating circles are big) especially on the points where the derivative is collinear to $\overrightarrow{u}_2$ and $\overrightarrow{u}_3$. We can accept then this inaccuracy because the results are considered more in their visual aspects. Reeves [50] call this phenomenon "visual accuracy". When we consider a nib's head that respect the ratio existing between the width and the thickness of nib's head (one sixth) the result would be more satisfying visually.

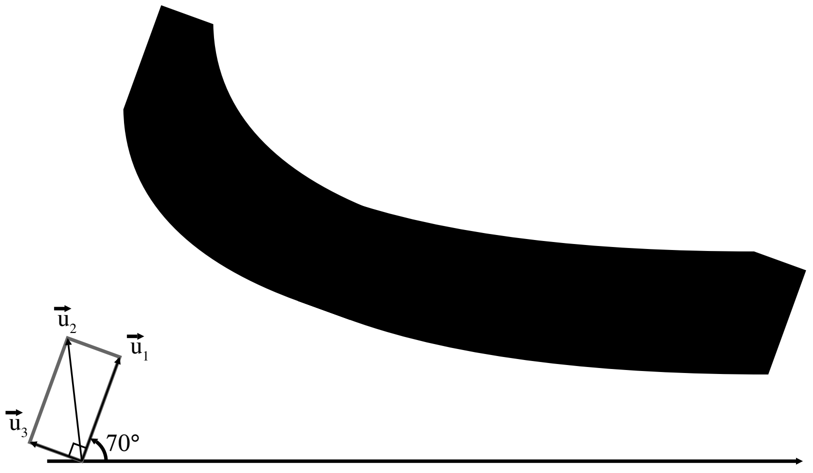

FIG. 2.55 – The real razed surface

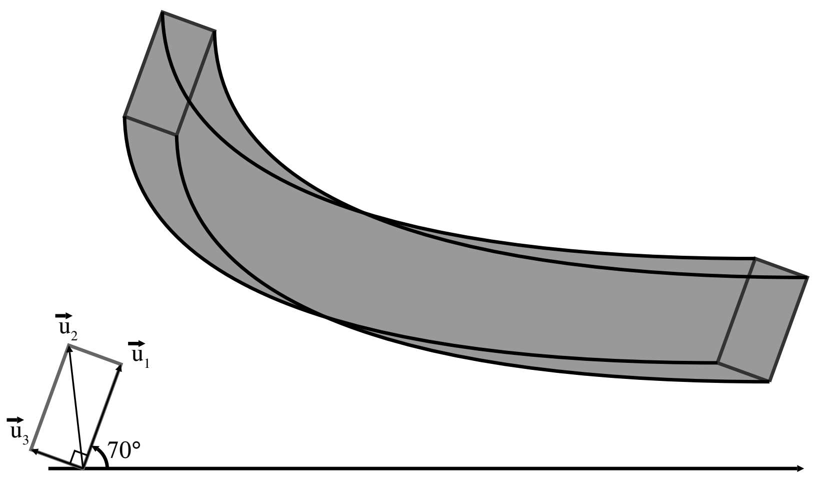

FIG. 2.56 – The razed surface with our approach

This method to reconstitute the envelope of the razed surface by the qalam along Kashida would be more improved in Chapter 4. The method used in this chapter has been the first approach we have adopted in the beginning of works on this subject. It gives excellent results but it will be limited in very big sizes and it doesn't allow writing texts in outlined letters. It is also slower than the approach presented in chapter 4.

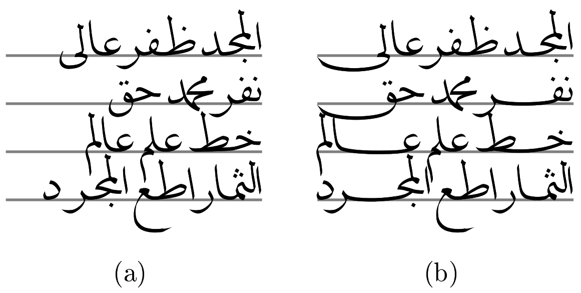

FIG. 2.57 – a) Arabic text lines before justification, in 18 pt, b) Arabic text lines after justification

2.5 Conclusions

Our main goal consisted on designing a font that helps in producing Arabic documents looking like handwritten proof. So, it was necessary to identify all kinds of ligatures and to introduce kashida in an adequate way. We were brought to scan the handwritten proofs and then to work on these proofs with simple graphic tools like Kontour [31] under Linux [37], in order to get the encoding of the letters in terms of Postscript Bézier's curves. The existing font tools were not enough to taking into account the motion of nib's head, the metrics in diacritic point, the possibility to correct some errors allowed for the calligrapher in handwritten proofs etc. We worked almost manually to design the PostScript encoding of some characters that represent most of the cases and we got so a *mini-font*. This font has been used to give the example in Figure 2.57. This is a set of lines (with no meaning in Arabic) to assess justification through kashida. To develop an optimized Arabic PostScript font as rapidly as possible, some mathematical formalization and study are necessary. This would be useful in implementing the font and in developing the graphical tools to help in the Arabic fonts design. This is the goal of the two following chapters.

Chapitre 3

Parametric curves variations with respect to a given direction[1]

3.1 Introduction

A font respecting the rules of the Arabic calligraphy may have specific characteristics. Such characteristics are mentioned in chapter 2. Of course, graphical tools are more used in geometric design, particularly in fonts design. The specificities of the fonts respecting the rules of Arabic calligraphy, especially the use of Kashida, the qalam shape, taking into account the motion of the qalam to encode letters... all that make that the available font graphic tools like FontCreator [21], FontForge [20] and so don't provide an adequate support to help in designing Arabic fonts.

In order to automatize and speed up as possible the job of designing a font respecting the Arabic calligraphy rules, we have to identify and develop mathematical tools to support the corespondent needs. The main goal of this formalization may consist on finding out a mathematical method that allows determining the optimized encoding of the letters in their filled in or outlined formats and that supports Kashida. An important part of the formalization consists on developing a way to study parametric plane curves with respect to a given direction as a real function of one variable is usually studied .

After showing the need of a method to study the variations of parametric curves according to a given direction in Section 2, we define and discuss an order relation among points in an affine plane in $\mathbb{R}^3$ in Section 3. In Section 4, the properties, lemmas and theorems supporting the study of plane curves are developed. In Section 5, an

[1]The contents of this chapter has been accepted as a full paper in WSCG 2006 conference but not published because we have withdrawn it since we could not go to Czech Republic to present it. The introduction and the conclusion has been modified after insertion in the thesis.

application of the method to decompose a Bézier curve into monotone Bézier sub-curves according to a given direction is presented. The chapter will end by some conclusions.

3.2 Problem

One of the ways to build a program that can help to develop Arabic fonts can be described as follows: After displaying a scanned image of the character, the font developer will seek the draw with it's mouse pointer as if it were a nib's head. Therefore, the program will generate the PostScript encoding corresponding to the character in the font. The program will have to take into account the curvilinear stretching, or *Kashida,* of the characters.

In Arabic writing, nib's head can be represented as a shaded rectangle whose width is one/a sixth of the length. The character is materialized by the surface razed by this rectangle. According to the writing style, Naskh, Rouqaa, Dywani etc..., the angle between rectangle's length and the baseline can vary a little or remain unchanged while writing. For instance, in Naskh style this angle remains at about 70° with the baseline [35] (See Figure 2.26).

In Arabic writing, many characters are dynamic, the shape and size of the character are context dependent. The stretching or change of size of the character in a given context is not a linear scaling nor a simple enlargement. Therefore, no suitable vectorial modeling can be found of a character through focusing on the outlines of the surface razed by nib's head. The optimal solution consists on the curves modeling nib's head *motion* instead of those representing the razed surface outlines. Then, the draw modeling the character is to be shaded taking into account nib's head movement.

The rectangle modeling nib's head can be considered as a rigid body (the length, width and angles between sides are invariant). The movement of one of the rectangle vertices determine entirely the movement of the other vertices through simple translations. The rectangle represented in Figure 3.1 has a length $M_{10}M_{20}$, a width $M_{10}M_{40}$. It makes an angle α with the baseline. The movement of the vertex M_{10} is represented by the Bézier curve with four control points M_{10}, M_{11}, M_{12} and M_{13}. From now, we make the assumption that the angle α remains invariant while the rectangle is in motion (all movements are reduced to simple translations, no rotations are performed). Taking into account rotations will need supplementary efforts on approximation. Now, it is out of the scope of this thesis. The three vertices will seek the same movement through translations of vectors $\overrightarrow{u}$, $\overrightarrow{u}+\overrightarrow{v}$ and $\overrightarrow{v}$ respectively ($\overrightarrow{M_{10}M_{20}}$, $\overrightarrow{M_{10}M_{30}}$ and $\overrightarrow{M_{10}M_{40}}$). How to shade the surface razed by the rectangle? A way to do so consists on shading the surface razed by each of the sides of the rectangle. So for the segment $[M_{10}, M_{20}]$, we will have to shade the surface delimited by the Bézier curve B_1 with control points M_{10}, M_{11}, M_{12} and M_{13}, the segment $[M_{13}, M_{23}]$, the Bézier curve B_2 with control points M_{23} , M_{22}, M_{21} and M_{20}, and finally the segment $[M_{20}, M_{10}]$. The points M_{20}, M_{21}, M_{22} and M_{23} are derived from M_{10}, M_{11}, M_{12} and M_{13} trough the translation of vector $\overrightarrow{u}$. Consider the movement of the segment $[M_{10}, M_{20}]$, then we define the Bézier outlines associated to this movement to be the *multi-curve* $(B_1, [M_{13}, M_{23}], B_2, [M_{20}, M_{10}])$. This way is meaningful. Actually, the PostScript language supports operators for shading outlines as with graphic development languages such as Java [44].

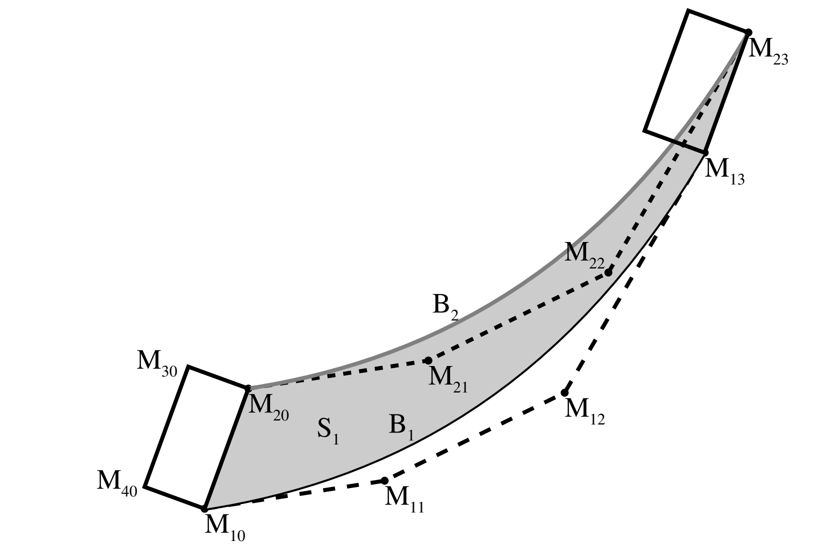

FIG. 3.1 – Surface S1 razed by the side $[M_{10}M_{20}]$

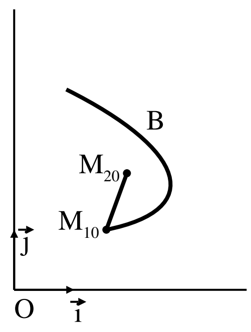

FIG. 3.2 – Trajectory of the point M_{10}

This method of shading surfaces doesn't provide always the desired results. Actually, consider a very smart nib (a nib with a negligible thickness (width)) or simply a segment in motion. As in Figure 3.2 where the segment $[M_{10}, M_{20}]$ is in movement, the extremity

M_{10} follows cubic Bézier's curve B. The surface razed by the segment is presented in Figure 3.3 and the surface delimited with the outlines associated to the movement doesn't fit exactly with the surface razed by the segment $[M_{10}, M_{20}]$ (See Figure 3.4).

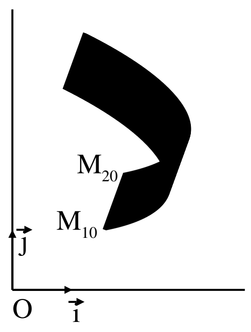

FIG. 3.3 – The Surface razed by the segment $[M_{10}, M_{20}]$

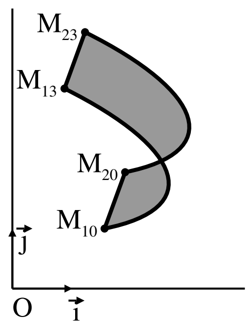

FIG. 3.4 – Surface delimited by the outlines associated to the movement

Now, let us decompose the curve B with the generalized algorithm of refinement [14, 25, 36, 45] (case of non median decomposition) with respect to the coefficient $T = 0.3613981051$. We get two Bézier curves B_1 and B_2 that can represent vertex's M_{10} movement (See Figure 3.5). Let S_1 and S_2 be the surfaces delimited by the outlines associated to the curves B_1 and B_2 (See Figures 3.6 and 3.7). Superposing S_1 and S_2 will give a shaded surface that fits exactly with the surface razed by the segment (See Figure 3.8).

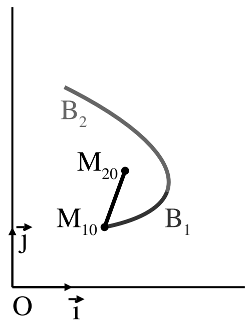

FIG. 3.5 – Decomposition of M_{10} trajectory

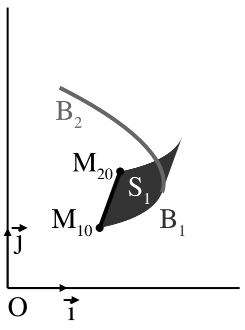

FIG. 3.6 – Surface corresponding to B_1

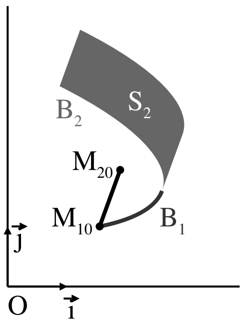

FIG. 3.7 – Surface corresponding to B_2

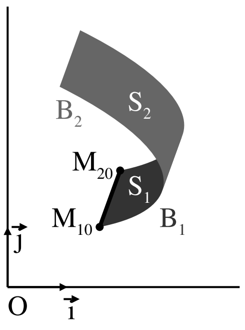

FIG. 3.8 – Superposing surfaces S_1 and S_2

The coefficient of decomposition T is an extrema of the Bézier parametric curve. Of course, there are no order relation in an affine plane. Next, we'll define comparison operators that will make the study of parametric curves as easy as the study of real functions with a single variable.

Consider the curve B_1 in Figure 3.1, the razing of the segment $[M_{10}M_{20}]$ is obtained directly with the multi-curve $(B_1, [M_{13}, M_{23}], t_{\overrightarrow{u}}(B_1), [M_{20}, M_{10}])$. This happens because the curve B_1 satisfies a particular condition that is the monotony on the interval $[0,1]$ with respect to the vector $\overrightarrow{u}$ (See below for more explanation about the *curve's monotony*). Whereas, the monotony of the curve B in Figure 3.2 changes on the interval $[0,1]$ according to the vector $\overrightarrow{u}$. One extrema of B equal to 0.3613981051 on $[0,1]$ with respect to $\overrightarrow{u}$ is determined and used to decompose B into two sub-curves B_1 and B_2 whose monotony doesn't change on $[0, 0.3613981051]$ and $[0.3613981051, 1]$ respectively. For a given parametric plane curve f on $[0,1]$ (or on $[a,b]$) and a vector $\overrightarrow{u}$, the study of the monotony of f requires a definition of a total order relation between points in a plane ($f(t), t \in [0,1]$ is a point in the plane). The total order will be a basic material to define the notion of extrema of parametric plane functions with respect to a given direction. That's the goal of Section 3.3. After and based on Section 3.3, useful tools and methods to study plane curves may be to develop. This will be done in Section 3.4.

3.3 Order in an affine plane in $\mathbb{R}^3$

Consider $\mathcal{P}$ an affine plane in $\mathbb{R}^3$ containing the origin O. Let $\overrightarrow{u}$ be a given vector in $\overrightarrow{\mathcal{P}}$, the director vectorial plane associated to $\mathcal{P}$. We will define in $\mathcal{P}$ an *equality operator* $\underset{\overrightarrow{u}}{=}$ and an *order operator* $\underset{\overrightarrow{u}}{\leq}$ with respect to the vector $\overrightarrow{u}$.

In the sequel, we consider:

- The affine space $\mathbb{R}^3$ with the orthonormal basis $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$.
- An affine plane $\mathcal{P}$ in $\mathbb{R}^3$ containing the origin O. Then, $\overrightarrow{\mathcal{P}}$ stands for the vectorial plane associated to $\mathcal{P}$. A direct orthonormal basis of $\overrightarrow{\mathcal{P}}$ will be denoted by $p = \left(\overrightarrow{P_1}, \overrightarrow{P_2}\right)$.
- Let $\overrightarrow{u} \in \overrightarrow{\mathcal{P}}$.

Definition 3.3.1 (Equality with respect to a vector in an affine plane)**.** *We define an equality relation* $\underset{\overrightarrow{u}}{=}$ *, among points in $\mathcal{P}$, with respect to the vector $\overrightarrow{u}$ by:*
$\forall M_1, M_2 \in \mathcal{P}$:

$$M_1 \underset{\overrightarrow{u}}{=} M_2 \Leftrightarrow \left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) = \left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right)$$

Definition 3.3.2 (Direct orthogonal range of a point with respect to a vector and a basis)**.** *Let M in $\mathcal{P}$, the value $\left(\overrightarrow{u} \wedge \overrightarrow{OM}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$, denoted by $\mathcal{R}_{(\overrightarrow{u},p)}(M)$, is called the direct orthogonal range of M with respect to a given vector $\overrightarrow{u}$ and a normed direct basis p.*

Remark 3.3.1.

The vectorial direct range is the component of the surface vector with respect to $\overrightarrow{P_1} \wedge \overrightarrow{P_2}$.

Property 3.3.1. *Let $M_1, M_2 \in \mathcal{P}$, then:*

$$M_1 \underset{\overrightarrow{u}}{=} M_2 \Leftrightarrow \mathcal{R}_{(\overrightarrow{u},p)}(M_1) = \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$$

Proof (3.3.1)

The implication from left to right is obvious.
Conversely,

suppose that $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) = \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$
it means $\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right) = \left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$
we have $\overrightarrow{u} \in \overrightarrow{P}$, $\overrightarrow{OM_1} \in \overrightarrow{P}$ and $\overrightarrow{OM_2} \in \overrightarrow{P}$.
Therefore $\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) = \lambda\left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$ and $\left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right) = \beta\left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$.
It follows that
$\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right) = \lambda \cdot \left\|\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right\|^2$ and
$\left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right) = \beta \cdot \left\|\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right\|^2$
But $\left\|\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right\| \neq 0$.
Consequently $\lambda = \beta$.
We then get $\left(\overrightarrow{u} \wedge \overrightarrow{OM_1}\right) = \left(\overrightarrow{u} \wedge \overrightarrow{OM_2}\right)$
and finally $M_1 \underset{\overrightarrow{u}}{=} M_2$. ■

Definition 3.3.3 (Order with respect to a vector in an affine plane). *The relation $\underset{\overrightarrow{u}}{\leq}$ defined in the set $\mathcal{P}$, with respect to the direction $\overrightarrow{u}$, by:*
$\forall M_1, M_2 \in \mathcal{P}$, we have:

$$M_1 \underset{\overrightarrow{u}}{\leq} M_2 \Leftrightarrow \mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$$

is a total order.

Proof

- Reflexivity: obvious
- Antisymmetry:
  Let $M_1, M_2 \in \mathcal{P}$ such that $M_1 \underset{\overrightarrow{u}}{\leq} M_2$ and $M_2 \underset{\overrightarrow{u}}{\leq} M_1$.
  then $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$ and $\mathcal{R}_{(\overrightarrow{u},p)}(M_2) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_1)$.
  So $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) = \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$
  according to the property 3.3.1, it follows
  $M_1 \underset{\overrightarrow{u}}{=} M_2$
- Transitivity:

Let $M_1, M_2, M_3 \in \mathcal{P}$ such that $M_1 \underset{\overrightarrow{u}}{\leq} M_2$ and $M_2 \underset{\overrightarrow{u}}{\leq} M_3$.

Then $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$ and $\mathcal{R}_{(\overrightarrow{u},p)}(M_2) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_3)$.

It follows $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_3)$.

So $M_1 \underset{\overrightarrow{u}}{\leq} M_3$

- Total order:

  Let $M_1, M_2 \in \mathcal{P}$, then we have

  $\mathcal{R}_{(\overrightarrow{u},p)}(M_1) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_2)$ or $\mathcal{R}_{(\overrightarrow{u},p)}(M_2) \leq \mathcal{R}_{(\overrightarrow{u},p)}(M_1)$

  and therefore, $M_1 \underset{\overrightarrow{u}}{\leq} M_2$ or $M_2 \underset{\overrightarrow{u}}{\leq} M_1$

Thus $\underset{\overrightarrow{u}}{\leq}$ is a total order relation. ■

3.4 Functions from $\mathbb{R}$ into an affine plane in $\mathbb{R}^3$

The equality and order relations defined before will allow studying parametric functions defined from an interval in $\mathbb{R}$ into an affine plane in the space $\mathbb{R}^3$ reported to an orthonormal basis $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$ in terms of extrema and variations.

In the following, let $\mathcal{P}$ be an affine plane contained in $\mathbb{R}^3$, containing the origin O, with director vectorial plane associated $\overrightarrow{\mathcal{P}}$ and a normed direct basis $\left(\overrightarrow{P_1}, \overrightarrow{P_2}\right)$.

Consider a given vector $\overrightarrow{u} \in \overrightarrow{\mathcal{P}}$ and an interval $[a, b] \subset \mathbb{R}$.

Let f be a parametric function (parametric curve) such that:

$$\begin{array}{rcccl} f & : & [a,b] & \longrightarrow & \mathcal{P} \\ & & t & \longmapsto & (f_1(t), f_2(t)) \end{array}$$

where f_1 and f_2 are two real functions defined on $[a, b]$.

In the following, we will establish some lemmas and generalize theorems about real functions in order to take into account parametric functions with respect to a given vector. Before that, let us give an obvious property that will next be used.

Property 3.4.1. *Let f be a continuous function defined on the interval $[a, b]$. Suppose that f is differentiable on $]a, b[$. Then $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$ is continuous on $[a, b]$, differentiable on $]a, b[$ and $\left(\mathcal{R}_{(\overrightarrow{u},p)} \circ f\right)' = \mathcal{R}_{(\overrightarrow{u},p)} \circ f'$.*

Theorem 3.4.1 (Rolle's Theorem)**.** *If f is a continuous function on $[a, b]$ such that:*

- *$f(a) \underset{\overrightarrow{u}}{=} f(b)$ and*

– *f is differentiable on $]a,b[$*

then $\exists\, c \in\,]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} O$ (O is the point with null coordinates in $\mathbb{R}^3$).

<u>Proof</u>

The function $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$ is continuous on $[a,b]$, and differentiable on $]a,b[$ (from the property 3.4.1) and

$\left(\mathcal{R}_{(\overrightarrow{u},p)} \circ f\right)(a) = \left(\mathcal{R}_{(\overrightarrow{u},p)} \circ f\right)(b)$.

from Rolle's theorem for real functions of one variable we get

$\exists\, c \in\,]a,b[$ such that $\left(\mathcal{R}_{(\overrightarrow{u},p)} \circ f\right)'(c) = 0$.

Consequently

$\exists\, c \in\,]a,b[$ such that $\mathcal{R}_{(\overrightarrow{u},p)}\left(f'(c)\right) = 0$.

Thus, $\exists\, c \in\,]a,b[$ such that

$\left(\overrightarrow{u} \wedge \overrightarrow{Of'(c)}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right) = 0$.

It follows $\exists\, c \in\,]a,b[$ such that

$\left(\overrightarrow{u} \wedge \overrightarrow{Of'(c)}\right) = \overrightarrow{(0,0,0)}$,

because $\overrightarrow{u} \wedge \overrightarrow{Of'(c)}$ and $\overrightarrow{P_1} \wedge \overrightarrow{P_2}$ are collinear and $\overrightarrow{P_1} \wedge \overrightarrow{P_2}$ is not null.

Then, we will have

$\exists\, c \in\,]a,b[$ such that $\overrightarrow{u} \wedge \overrightarrow{Of'(c)} = \overrightarrow{u} \wedge \overrightarrow{OO}$.

finally $\exists\, c \in\,]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} O$. ■

Theorem 3.4.2 (Mean value theorem)**.** *If f is a continuous function on $[a,b]$, differentiable on $]a,b[$, then $\exists\, c \in\,]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} \frac{f(b)-f(a)}{b-a}$.*

<u>Proof</u>

As in the previous proof of Theorem 3.4.1, consider the real function $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$.

By the Mean value theorem for real functions of one variable, we get:

$\exists\, c \in\,]a,b[$ such that $f'(c) \underset{\overrightarrow{u}}{=} \frac{f(b)-f(a)}{b-a}$. ■

Definition 3.4.1 (Monotony)**.** *A function f is a monotone increasing (respectively decreasing) on $[a,b]$ with respect to the direction $\overrightarrow{u}$ if and only if:*

$\forall t_1, t_2 \in [a,b]\,,\ t_1 \leq t_2 \Rightarrow f(t_1) \underset{\overrightarrow{u}}{\leq} f(t_2)$ *(respectively* $\forall t_1, t_2 \in [a,b]\,,\ t_1 \leq t_2 \Rightarrow$ $f(t_2) \underset{\overrightarrow{u}}{\leq} f(t_1)$*).*

Remark 3.4.1.

The monotony is *strict* whenever the order relation $\underset{\overrightarrow{u}}{<}$ is used.

Lemma 3.4.1. *The functions f and $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$ have the same monotony on $[a, b]$.*

Theorem 3.4.3 (varaiations sens)**.** *If f is continuous on $[a, b]$ and differentiable on $]a, b[$ then:*

- $\forall t \in]a, b[\,,\ f'(t) \underset{\overrightarrow{u}}{\geq} O \Leftrightarrow f$ *is monotone increasing on $[a, b]$ with respect to the direction $\overrightarrow{u}$.*
- $\forall t \in]a, b[\,,\ f'(t) \underset{\overrightarrow{u}}{\leq} O \Leftrightarrow f$ *is monotone decreasing on $[a, b]$ with respect to the direction $\overrightarrow{u}$.*

Proof
That's an obvious corollary of the lemma 3.4.1. ■

Definition 3.4.2 (Extrema)**.** *The function f admits a maximum (resp a minimum) at the point $t_0 \in [a, b]$ with respect to the direction $\overrightarrow{u}$ if and only if there exist $t_1, t_2 \in [a, b]$, $t_1 \neq t_2$ such that:*

$t_0 \in]t_1, t_2[$ and $\forall t \in [t_1, t_2]$ $f(t) \underset{\overrightarrow{u}}{\leq} f(t_0)$ (resp $f(t_0) \underset{\overrightarrow{u}}{\leq} f(t)$).

Lemma 3.4.2. *The functions f and $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$ have the same extrema on $[a, b]$.*

Theorem 3.4.4 (Extrema and derivative)**.** *Suppose that f is continuous on $[a, b]$ and differentiable on $]a, b[$. Let $t_0 \in]a, b[$. Then if $\overrightarrow{u} \wedge \overrightarrow{Of'(t)}$ vanishes at t_0 and changes its direction, f admits an extrema in t_0 with respect to the direction $\overrightarrow{u}$.*

Proof
As previously, consider the real function $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$.
The vector $\overrightarrow{u} \wedge \overrightarrow{Of'(t)}$ is collinear with $\overrightarrow{P_1} \wedge \overrightarrow{P_2}$, and
$\overrightarrow{u} \wedge \overrightarrow{Of'(t)}$ vanishes at t_0 and changes in direction.
This will imply that $\mathcal{R}_{(\overrightarrow{u},p)} \circ f'$ vanishes at t_0 and changes in sign,
it follows that t_0 is an extrema of $\mathcal{R}_{(\overrightarrow{u},p)} \circ f$ on $[a, b]$
We will have from the lemma 3.4.2, that t_0 is an extrema of f on $[a, b]$. ■

Remark 3.4.2.

At $t_0 \in]a, b[$ which is an extrema according to the theorem 3.4.4, the directional tangent $\overrightarrow{Of'(t_0)}$ is collinear with the vector $\overrightarrow{u}$.

In the Arabic document processing area, there is no system that typeset Arabic documents with a quality that is like the one in the calligrapher handwritten texts.

The simple cause is that the development of fonts respecting the Arabic calligraphy rules isn't a simple continuation of the works done concerning the Latin documents. The problem is not simple as the majority had thought in the beginning. The first step and mandatory in this area is to offer to the community a good formalization of the problem. So the goal of the chapter is not to *invent a new theory but to give an adequate and direct mathematical formalism.* The theory studied here would be extended in the next chapter and in some coming works out of this thesis, to offer mathematical ways to develop dynamic fonts of Arabic letters in stretchable outlines when nib's head is translated without and with rotations.

We have developed a way to study and decompose a plane curve parametrized on $[a, b]$ into monotone restrictions to sub-intervals of $[a, b]$ as we can do with a real function of one variable. The work here is based on Bézier curves. Restrictions of Bézier curves are obtained by generalized Bézier refinement and the restrictions are to be re-parametrized on $[0, 1]$ and not on sub-intervals of $[0, 1]$ as we have seen in the general case. In Section 3.5, the preceding results are applied and restricted to Bézier curves.

3.5 Bézier curve decomposition

Up till now, a way to decompose parametric curves characterizing a segment motion has been developed. So, the surface razed by the segment can be obtained. It is the same for any rigid body. Now, we will focus on the polar Bézier curves. In the following, we'll give a method for decomposing Bézier curves of any degree. An example with a cubic Bézier curve will be given as illustration.

- Let $\mathbb{R}^3$ be the affine space with the orthonormal basis $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$.
- Let $\mathcal{P}$ denotes the $\mathbb{R}^2$ affine plane with direct normed basis $p = (\overrightarrow{\imath}, \overrightarrow{\jmath})$ passing through the origin O.
- Let $[M_{10}, M_{20}]$ be a segment in translation, in $\mathcal{P}$.
- If $[a, b] = [0, 1]$.
- If the Bézier curve B describes the movement of M_{10}. Suppose that their control points M_{10}, M_{11}, $\ldots$, M_{1n} are such that $M_{1i} = (x_i, y_i, 0)$.

The curve B can be defined in $\mathbb{R}^3$ through considering a null component with respect to the vector $\overrightarrow{k}$:

$$\begin{array}{rccl} B & : & [0,1] & \longrightarrow & \mathcal{P} \\ & & t & \longmapsto & \displaystyle\sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} M_{1i} \end{array}$$

In order to decompose a Bézier's curve into monotone curves, with respect to the direction $\overrightarrow{M_{10}M_{20}}$, we should determine the set of extrema of the direct orthogonal range $\mathcal{R}_{(\overrightarrow{M_{10}M_{20}},p)} \circ B$ on $[0,1]$. Then, these extrema will be classified by increasing order. Let $T_0 = \{t_{01}, t_{02}, \ldots, t_{0m}\}$ be this ordered set, with $m \leq n-1$. Thus, we decompose B with respect to t_{01}, we get two Bézier curves B_0 and B_1 such that B_0 has a constant monotony with respect to $\overrightarrow{M_{10}M_{20}}$ on $[0,1]$ and B_1 is a curve to decompose. We won't determine the extrema of B_1 with respect to $\overrightarrow{M_{10}M_{20}}$ because the set of these extrema is $T_1 = \left\{t_{11}, t_{12}, \ldots, t_{1(m-1)}\right\}$ where $t_{1i} = \frac{t_{0(i+1)}-t_{01}}{1-t_{01}}$ (the formula can be established by a an easy calculation). We continue the process until obtaining the last monotone curve. This method is presented as a mechanic algorithm in the following.

Suppose that we have already defined the data structures and algorithms:

`Point` a data structure modeling a point in $\mathbb{R}^3$,

`Vector` a data structure modeling a vector in $\overrightarrow{\mathbb{R}}^3$, (the Point data structure can be used instead of this one),

`Bezier` a data structure modeling a Bézier's curve,

`PtVect(M10:Point,M20:Point):Vector` a function that results in the vector defined by two points.

`ExtremaNbr(B:Bezier,U:Vector):INTEGER` a function that results in the number of extrema of B with respect to U,

`Extremas(B:Bezier,U:Vector): array[1..NMax]of REAL` a function that results in an array of extrema of B with respect to U,

`LDecp(B:Bezier,t:REAL):Bezier` a function that results in the left Bézier curve by decomposing B with respect to t,

`RDecp(B:Bezier,t:REAL):Bezier` a function that results in the right Bézier curve by decomposing B with respect to t.

Now, we will give the meaning of some variables and algorithms.

`M10, M20` are the points which determine the segment,

`U` is the decomposition direction,

`B` is the curve to decompose,

`Nx` is the B extrema number according to U,

`AExt` is the array displaying all the extrema of B according to U, *DBC* is the array displaying the Bézier curves obtained by decomposition of B according to U,

Ctrd and *Ctre*: are indexes for acceding DBC and AExt arrays

```
⋮
M10,M20 : Point;
U : Vector;
B : Bezier;
Nx : INTEGER;
AExt : array[1..Nmax] fo REAL;
DBC : array[1..Nmax+1] of Bezier;
Ctrd, Ctre : INTEGER;
BEGIN
 /* B, M10, M20 initialising
 ⋮
 U ← PointVector(M10,M20);
 Nx ← ExtremaNumber(B,U);
 AExt ← Extremas(B,U);
 FOR Ctrd=1, Nx, +1
 BEGIN
  DBC[Ctrd]←LDecp(B,AExt[Ctrd]);
  B←RDecp(B,AExt[Ctrd]);
  FOR Ctre=Ctrd+1, Nx, +1
  AExt[Ctre]←
   (AExt[Ctre]-AExt[Ctrd])/(1-AExt[Ctrd]);
 END
 DBC[Nx+1]←B;
END
```

We can remark that there is no need for two separate functions LDecp and RDecp since both parts of the subdivided curve can be computed by one execution of the algorithm of de Casteljau. We use two functions for better clarity and also when we use the algorithm of de Casteljau we must extract the decomposition parts separately.

The Method will be illustrated through an example modeling the movement of a segment $[M_{10}, M_{20}]$ with a cubic Bézier curve B with two extrema with respect to the vector $\overrightarrow{M_{10}M_{20}}$ (See Figure 3.9).

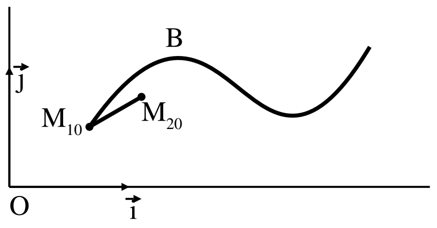

FIG. 3.9 – A cubic Bézier curve of a movement

The surface razed by the segment $[M_{10}, M_{20}]$ (See Figure 3.10) is different from the surface delimited by the outlines associated to the movement (See Figure 3.11). The reason behind that is that B is not monotone with respect to $\overrightarrow{M_{10}M_{20}}$.

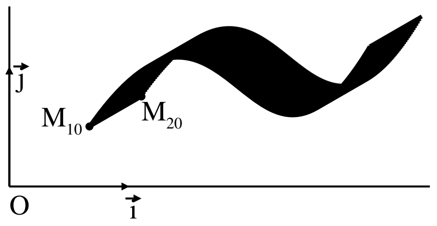

FIG. 3.10 – Surface razed by the segment $[M_{10}, M_{20}]$

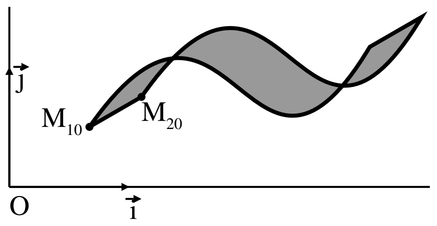

FIG. 3.11 – Surface delimited by the associated outlines

We will have to decompose the curve B into monotone sub-curves with respect to the direction $\overrightarrow{M_{10}M_{20}}$. The B control points are $M_{10} = (95, 50)$, $M_{11} = (130, 100)$, $M_{12} = (135, 20)$ and $M_{13} = (165, 70)$. We determine the direct orthogonal range of B. In order to find the extrema on $]0, 1[$, we derive this range. Then, we find that the direct orthogonal range has two extrema $T_{01} = 0.1571942250$ and $T_{02} = 0.8301512030$. We decompose B with respect to T_{01} to obtain two curves; B_1 monotone, and a curve B_2 to decompose again (See Figure 3.12).

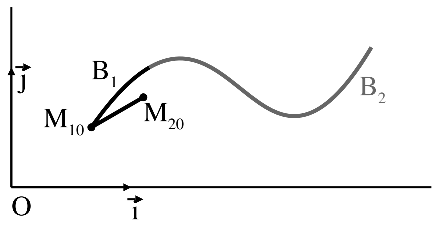

FIG. 3.12 – First decomposition

Then, the curve B_2 is decomposed with respect to $T_{11} = \frac{T_{02}-T_{01}}{1-T_{01}}$, say for instance, $T_{11} = 0.7984721960$. The decomposition gives two curves B_{21} and B_{22}. Both of them are monotone with respect to $\overrightarrow{M_{10}M_{20}}$ (See Figure 3.13).

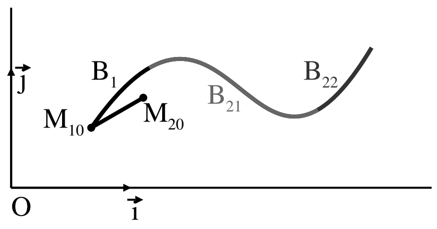

FIG. 3.13 – Second decomposition

If we superpose all the surfaces delimited by the outlines associated to the three curves B_1, B_{21} and B_{22} (See Figure 3.14), we get exactly the surface in Figure 3.10.

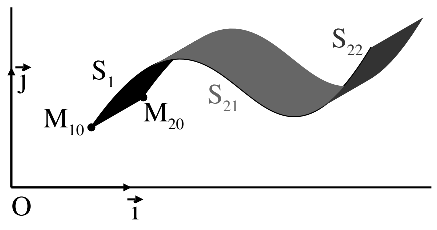

FIG. 3.14 – Superposition of the surfaces of decomposition

3.6 Conclusions

A method to study parametric curves and determine their extrema according to a given direction has been set up. It provides a good base to understand and develop a way to make comparisons among curves obtained with shifting transforms of the same curve with a set of vectors. This is a part of the mathematical formalization done in the next chapter. More mathematical tools, other than curves comparison, are useful to develop. That's the case of the support of the mathematical representation of the qalam, an optimized way to determine some Bézier curve intersections coefficients... This is presented in the fourth chapter.

Chapitre 4

Mathematical ways to optimize and encode the static and variable sized Arabic letters in a PostScript font[1]

4.1 Introduction

In Chapter 2, we have shown that a Naskh font respecting as possible the rules of Arabic calligraphy, and therefore Kashida, may be dynamic. PostScript has been recommended as an adequate language to develop the font.

Kashida is an optical scaling, not a linear one (See 2.3.4). In order to take this fact into account, the motion of the qalam should be considered in the font encoding of the letters stretchable parts. A way to give a support to the motion of the pen in a writing system consists on representing the pen by its head's shape and *to calligraph* with this representation along a path. The concept is supported by METAFONT directly or via the makepen macro (the definition of pen is also supported by Metapost) but the supported dynamism is *limited*. That has been shown in chapter 2. PostScript supports the dynamism but it doesn't support mechanisms to manage pens as METAFONT do. Postscript is *outlines oriented* and doesn't provide a ready to use support to encode our font. Both the static and dynamic parts of the letters may be represented by their outlines taking into account the motion of the qalam. This will be based on the fact that the external curves are trajectories of vertices of the qalam or of some of its edges. To do so, a mathematical way to model and study the motion of the qalam is developed in this chapter.

[1]This chapter is based on two papers published in WSCG-2008 proceedings [1] and IACe-T-2008 proceedings [2].

Dynamic fonts in PostScript are Type 3. In this type of fonts there is no mechanism of caching. That means that a procedure encoding a letter has to be executed whenever the letter is to appear in the document. Thus, taking into account the optimization in encoding the dynamic letters will be helpful. One of the first steps to achieve consists on using a quadratic stretching model instead of the cubic one presented in Chapter 2.

As for other writing systems (Latin script based for example), we want to give some concepts to develop a font to write Arabic in genuine outlines. As qalam's head has a rectangular shape, some intersection between the stretchable parts is to be determined dynamically. All the known methods to determine intersection coefficients in our case would not be adequate since an iterative processing is necessary and therefore would require more time in processing. A method to approximate and optimize the determination of the intersection coefficients will be developed in the chapter.

All along the mathematical concepts presentation in this chapter, some processing time evaluation to confirm the optimization techniques is necessary.

The chapter follows the plan: the second section highlights Kashida notion, the requirements of writing in outlined letters and the classification of characters in terms of *dynamism*. The third section presents the geometrical representation and therefore the notion of qalam's motion. In the fourth section, the stretching models are presented and compared for optimization. In the fifth section, we show how static letters are implemented in the font. The way to implement the dynamic letters, in filled in or outlined version, in an optimized way are given in the sixth and seventh sections. The chapter ends with some conclusions and perspectives.

4.2 On some characteristics of Arabic calligraphy

4.2.1 Kashida

In Arabic alphabet based scripts writing, the calligraphic rules can be considered as almost mandatory as grammar. In such writing, stretching letters is used instead of the ordinary insertion of spaces (blanks) between words, to justify the lines. There are two kinds of kashida stretching: on two connected letters and inside the same letter. For clarity, the two kinds of stretching are presented with *bounding boxes* of characters in Figure 4.1. In the first line, there are stretching in different sizes between the two connected letters ـفـ and ـر in the word ظفر. In the following line, the last letter ـق of the word حق is stretched in four different sizes.

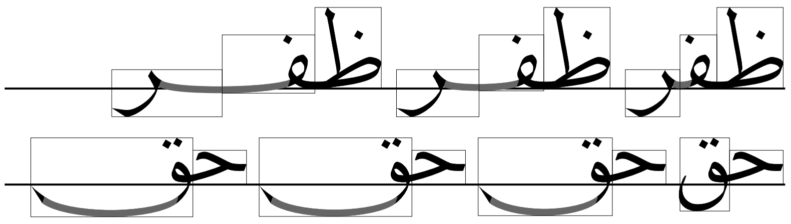

FIG. 4.1 – Inter-curvilinear and intro-curvilinear stretching in Arabic calligraphy

4.2.2 Kashida and outlined letters

One goal of this chapter is to provide a support for developing a PostScript font that allows typesetting Arabic texts, in characters represented as outlines, according to the basic calligraphic rules. The existing systems don't offer such support. In the first line, in Figure 4.2 (from right to left), the word ظفر is displayed in outlines, in two options: with and without limits between two consecutive characters. Most of the systems allowing writing Arabic script in outlines, such as Microsoft Word, can produce only the first sample (in the right). Of course, The cursivity is observed better in the second way of writing in Figure 4.2.

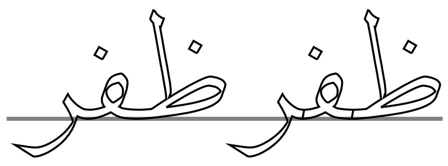

FIG. 4.2 – Possibilities of Outlines writing in Arabic

Managing stretchability is better shown with fonts that provide characters in outlines. An example of stretched outlines is presented in Figure 4.3. This stretchability of characters needs a mathematical formalization. So, the stretching letters model presented in Chapter 2 would be deeply improved.

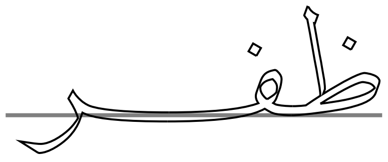

FIG. 4.3 – Stretchable outlines in Arabic

4.2.3 Fixed and variable sized letters

Let us recall that there are *fixed* and *variable sized* letters. Fixed letters have static shapes as the letter د (standalone DEL) and ر (standalone REH) in Figure 4.4(a). Variable sized letters have stretchable parts. The first line of Figure 4.4(b) shows the letter ﺑ (BEH connected after) with three different curvilinear stretching whereas the second line presents the letter ﺮ (REH connected before). Variable sized letters are composed of a static part (in terms of shape; the shape is kept as it is after a shifting transformation) and a dynamic one. In Figure 4.4(b), the dynamic parts are in gray. For the letter ﺑ, this part is used to connect the letter to the following one in the left. In the same context, the stretchable part in the other letter is the preceding connection in the letter ﺮ (this is also used to connect it to the preceding letter in its right).

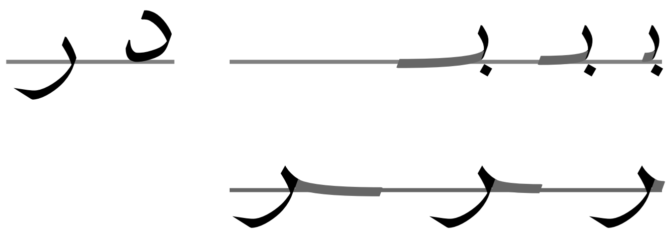
(a) (b)

FIG. 4.4 – a) Examples of static letters, b) Examples of variable sized letters

4.3 Nib's head motion-razing modeling and decomposition

There are different *styles* in Arabic writing. Among the most popular ones, we can mention: Farsi, Koufi, Maghrebi, Naskh, Thuluth, Rouqaa, Dywani... Samples of such styles are presented in Figure 4.5. For the purpose of this thesis, Naskh style is considered (the framed one in Figure 4.5) since it is the most widely used in digital typography.

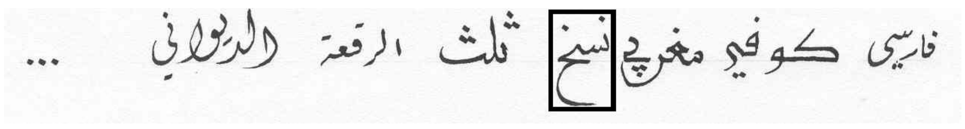

FIG. 4.5 – Most popular Arabic writing styles

So, in this chapter, we'll develop mathematical tools to help in the development of a *stretchable* Naskh font. Of course, this font will support writing in outlines.

Now, let's give some characteristics of nib's head motion under Arabic calligraphy rules.

4.3.1 Qalam's motion modeling

In Naskh style, nib's head or simply the qalam (the pen, calligraphers use to write with) behaves as a rectangle of width l and thickness $e = \frac{l}{6}$. This rectangle moves with a constant inclination angle of about 70° with regard to the baseline. A nib's head with $l = 12$ mm (and $e = 2$ mm) is presented in Figure 2.26.

In the following, let us consider the general case where pen's head motion can be represented by a polygonal shape with $n+1$ vertices, in translation. For more clarity, examples are given with rectangles with sizes and/or inclinations different from Naskh qalam's ones. Let's represent a pen's head with $n+1$ vertices M_0, M_1, …, M_n by $\mathcal{B}(M_0, \mathcal{F})$, where $\mathcal{F} = (\overrightarrow{u_i})_{i=0,\ldots,n}$ is a family of $n+1$ vectors such that $\overrightarrow{u_i} = \overrightarrow{M_0 M_i}$. The point M_0 stands for an *origin* for pen's head in motion. Whenever particular values are associated to that point and to the vectors, all the vertices, and therefore positions, of pen's head can be determined precisely. When the trajectory of the origin vertex is the parametric curve f defined on $[0,1]$, the motion of a pen $\mathcal{B}(M_0, \mathcal{F})$ will be denoted by $\mathcal{M}(\mathcal{B}(M_0, \mathcal{F}), f, [0,1])$. In Figure 4.6, the motion is characterized by the start position of pen's head and a Bézier curve [42] B_0 representing the trajectory of the vertex M_{00}. In the case of the qalam, the origin is the vertex with a circle in Figure 2.26.

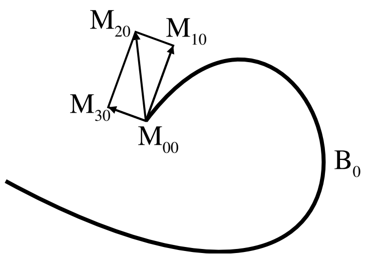

FIG. 4.6 – Example of motion

4.3.2 Motion-razing

The razing of a qalam motion is the set of points darken by the qalam in the plane. Of course, this set is delimited by external curves corresponding to the trajectories of qalam's vertices and some edges of the qalam. Therefore, we need a mean to help comparing the shifting transforms of curves in order to find out the external curves of a razing. For this, it is necessary to define the notion of *the orthogonal range associated to a parametric function according to a vector.*

Consider:

– the affine space $\mathbb{R}^3$ with the orthonormal basis $R\left(O, \overrightarrow{\imath}, \overrightarrow{\jmath}, \overrightarrow{k}\right)$,

- an affine plane $\mathbb{R}^2$ in $\mathbb{R}^3$ containing the origin O with a direct orthonormal basis $(\overrightarrow{\imath}, \overrightarrow{\jmath})$. Let's denote $\overrightarrow{\mathbb{R}}^2$ the vector plane associated to $\mathbb{R}^2$,
- a parametric function f defined on $[0,1]$ with values in $\mathbb{R}^2$. Suppose that f is continuous on $[0,1]$ and differentiable on $]0,1[$ and
- the vectors $\overrightarrow{u}$, $\overrightarrow{u}_1$, $\overrightarrow{u}_2$ in $\overrightarrow{\mathbb{R}}^2$.

Definition 4.3.1 (Direct orthogonal range of a parametric function with respect to a vector)**.** *The direct orthogonal range associated to f, according to the vector $\overrightarrow{u}$, is the scalar function $\mathcal{R}_{(f,\overrightarrow{u})}$ defined from* $[0,1]$ *onto* $\mathbb{R}$ *such that:*
$\mathcal{R}_{\left(f,\overrightarrow{u}\right)}(t) = \left(\overrightarrow{u} \wedge \overrightarrow{Of(t)}\right) \cdot \overrightarrow{k}$.

The derivative verify $\mathcal{R}'_{\left(f,\overrightarrow{u}\right)}(t) = \mathcal{R}_{\left(f',\overrightarrow{u}\right)}(t)$

In Chapter 3, we have defined the direct orthogonal range of a point M in a plane $\mathcal{P}$ directed by a direct base $p = \left(\overrightarrow{P}_1, \overrightarrow{P}_2\right)$ according to the vector $\overrightarrow{u}$ as the amount $\mathcal{R}_{(\overrightarrow{u},p)}(M) = \left(\overrightarrow{u} \wedge \overrightarrow{OM}\right) \cdot \left(\overrightarrow{P_1} \wedge \overrightarrow{P_2}\right)$. In this chapter, we have adopted a more simplified notation of the orthogonal range of a parametric function. The value $\mathcal{R}_{\left(f,\overrightarrow{u}\right)}(t)$ can be written applying the notation in Chapter 3 as $\mathcal{R}_{\left(\overrightarrow{u},(\overrightarrow{\imath},\overrightarrow{\jmath})\right)}(f(t))$. In the notation $\mathcal{R}_{\left(f,\overrightarrow{u}\right)}(t)$, we have neglected the base since it is the particular one $(\overrightarrow{\imath}, \overrightarrow{\jmath})$.

Now, we can define the way to compare the shifting transform of a curve.

Definition 4.3.2 (Comparison of shifted curves)**.** *We have $t_{\overrightarrow{u}_1}(f) \leq t_{\overrightarrow{u}_2}(f)$ on* $[0,1]$ *(respectively $t_{\overrightarrow{u}_1}(f) \geq t_{\overrightarrow{u}_2}(f)$) if and only if $\mathcal{R}'_{(f,\overrightarrow{u}_1-\overrightarrow{u}_2)}(t) \leq 0$ (respectively $\mathcal{R}'_{(f,\overrightarrow{u}_1-\overrightarrow{u}_2)}(t) \geq 0$).*

The comparison between $t_{\overrightarrow{u}_1}(f)$ and $t_{\overrightarrow{u}_2}(f)$ on $[0,1]$ can be done through the study of the monotony of $\mathcal{R}_{(f,\overrightarrow{u}_1-\overrightarrow{u}_2)}$ on $[0,1]$. The determination of this way to compare curves is an application of what has been done in chapter 3.

Now, we have the tools necessary for studying nib's head motions and their associated razings. Before doing that, let's present a particular type of motions, the *normal* motions, and their associated razings.

Consider the motion M such that

- $M = \mathcal{M}\left(\mathcal{B}\left(M_{00}, \mathcal{F}\right), f, [0,1]\right)$ and
- $\mathcal{F} = (\overrightarrow{u_i})_{i=0,\ldots,n}$.

Definition 4.3.3 (Normal Motion)**.** *M is said to be* normal *on* $[0,1]$ *if and only if* $\exists i \in \{0,\ldots,n\}$, $\exists j \in \{0,\ldots,n\}$ *such that:*

$t_{\overrightarrow{u}_k}(f) \leq t_{\overrightarrow{u}_i}(f)$ *and* $t_{\overrightarrow{u}_k}(f) \geq t_{\overrightarrow{u}_j}(f)$ *on* $[0,1]$ $\forall k \in \{0,\ldots,n\}$.

In this definition, $t_{\overrightarrow{u}_i}(f)$ is the *maximal* curve whereas $t_{\overrightarrow{u}_j}(f)$ is the *minimal* one among the set $\left\{t_{\overrightarrow{u}_k}(f), \overrightarrow{u}_k \in \mathcal{F}\right\}$. A normal motion and its corresponding razing are presented in Figure 4.7. The razing is delimited by the two segments $[M_{30}M_{00}]$, $[M_{00}M_{10}]$, the Bézier curve B_1 (maximal curve), the two segments $[M_{11}M_{21}]$, $[M_{21}M_{31}]$ and the Bézier curve B_3 (minimal curve).

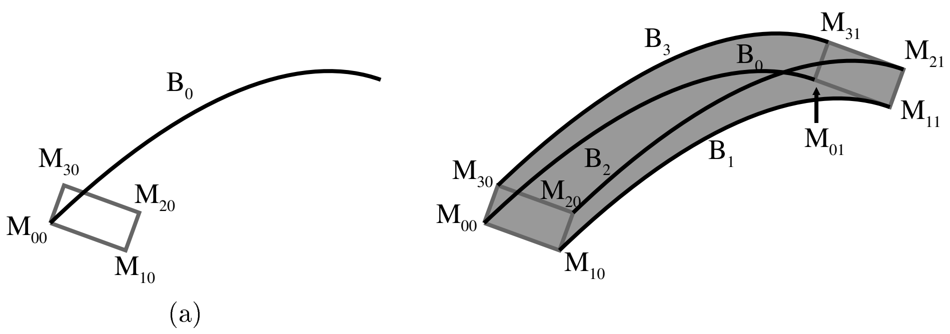

FIG. 4.7 – a) Normal motion, b) The corresponding normal razing

Lemma 4.3.1. *If* $\forall i \in \{0,\dots,n\}$ $\mathcal{R}_{\left(f,\overrightarrow{u}_{(i+1)mod\ n}-\overrightarrow{u}_i\right)}$ *is monotone on* $[0,1]$ *then* M *is a normal motion.*

The lemma can be proved through considering the motion in Figure 4.7. In general, the razing associated to a motion is determined via its decomposition into normal sub-motions. Then, we have the theorem.

Theorem 4.3.1. (Decomposition into normal sub-motions)**.** *The motion* M *can be decomposed into normal sub-motions on* $[0,1]$ *so that the concatenation of their associated razings constitute exactly the global razing associated to* M*.*

Proof

Let $\mathcal{F} = (\overrightarrow{u_i})_{i=0,\dots,n}$ $(\overrightarrow{u_0} = \overrightarrow{0})$
The studying of the sign of $\mathcal{R}'_{(f,\overrightarrow{u}_{(i+1)\text{mod } n}-\overrightarrow{u}_i)}$ on $]0,1[$ for $i \in \{0,\dots,n-1\}$ gives the set of extrema of $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\text{mod } n}-\overrightarrow{u}_i)}$ on $]0,1[$.
Let $E_i = \left\{T_{i,1}, T_{i,2}, \dots, T_{i,\phi(i)}\right\}$ be the set of extrema of $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\text{mod } n}-\overrightarrow{u}_i)}$, ordered in an increasing order ($\phi(i)$ results in the cardinal of E_i).
The restrictions of the direct orthogonal range $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\text{mod } n}-\overrightarrow{u}_i)}$ to the segments $[0,T_{i,1}[$, $[T_{i,1},T_{i,2}[$, $\dots$, $\left[T_{i,\phi(i)-1},T_{i,\phi(i)}\right[$, $\left[T_{i,\phi(i)},1\right]$ are monotone.
Now, let us consider $E = \{T_1, T_2, \dots, T_m\}$ the union of the sets E_i, ordered in an increasing order. According to the previous reasoning, $\mathcal{R}_{(f,\overrightarrow{u}_{(i+1)\text{mod } n}-\overrightarrow{u}_i)}$ is monotone on $[0,T_1[$, $[T_1,T_2[$, $\dots$,$[T_{m-1},T_m[$, $[T_m,1]$ for $i \in \{0,\dots,n-1\}$.
By the lemma 4.3.1, qalam's motion is normal on $[0,T_1[$, $[T_1,T_2[$, $\dots$,$[T_{m-1},T_m[$, $[T_m,1]$.
The restrictions considered together constitute the exact trajectories of the vertices of qalam's head. So, the concatenation of the razings gives the exact razing of the pen's head. ■

In order to illustrate the study of motions, we consider a concrete example. Let us consider the motion of a qalam of extremities M_{00}, M_{10}, M_{20} and M_{30} (See Figure 4.6). The trajectory of the origin M_{00} is the cubic Bézier curve B_0. The point M_0 coincides with M_{00}. The control points of B_0 are $M_0 = (135, 80)$, $M_1 = (180, 140)$, $M_2 = (220, 0)$ and $M_3 = (100, 65)$. The family of the motion vectors is $\mathcal{F} = (\overrightarrow{u}_i)_{i=0,\dots,3}$ such that $\overrightarrow{u}_0 = \overrightarrow{0}$, $\overrightarrow{u}_1 = \overrightarrow{M_{00}M_{10}} = \overrightarrow{(6.83116, 18.7972)}$, $\overrightarrow{u}_2 = \overrightarrow{M_{00}M_{20}} = \overrightarrow{(-2.5696, 22.2069)}$ and $\overrightarrow{u}_3 = \overrightarrow{M_{00}M_{30}} = \overrightarrow{(-9.40076, 3.40964)}$.

Studying $\mathcal{R}'_{(B_0, \overrightarrow{u}_{(i+1) \bmod 3} - \overrightarrow{u}_i)}$ on $]0, 1[$ for $i \in \{0, 1, 2, 3\}$ shows that $\mathcal{R}'_{(B_0, \overrightarrow{u}_1 - \overrightarrow{u}_0)}$ vanishes at $t_0 = 0.580246$ and the sign changes in the neighborhood of t_0. Whereas, $\mathcal{R}'_{(B_0, \overrightarrow{u}_2 - \overrightarrow{u}_1)}$ vanishes at $t_1 = 0.238061$ and $t_2 = 0.919186$ and changes in sign near these two real numbers.

$\mathcal{R}'_{(B_0, \overrightarrow{u}_3 - \overrightarrow{u}_2)}$ and $\mathcal{R}'_{(B_0, \overrightarrow{u}_0 - \overrightarrow{u}_3)}$ have the same roots and opposite sign as $\mathcal{R}'_{(B_0, \overrightarrow{u}_1 - \overrightarrow{u}_0)}$ and $\mathcal{R}'_{(B_0, \overrightarrow{u}_2 - \overrightarrow{u}_1)}$ respectively.

The function $\mathcal{R}_{(B_0, \overrightarrow{u}_{(i+1) \bmod 3} - \overrightarrow{u}_i)}$ is monotone on $[0, 0.238061]$, $[0.238061, 0.580246]$, $[0.580246, 0.919186]$ and $[0.919186, 1]$.

Then, we get that the motions restricted to the intervals $[0, 0.238061]$, $[0.238061, 0.580246]$, $[0.580246, 0.919186]$ and $[0.919186, 1]$ are normal.

Let B_{0j}, $j \in \{0, 1, 2, 3\}$, be the curves obtained by successive Bézier refinement [14] according to t_1, t_0, t_2, the four curves are presented in Figure 4.8.

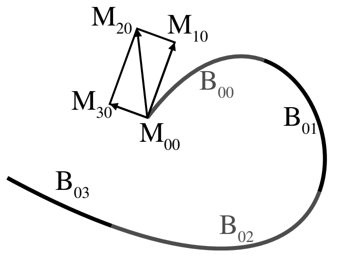

FIG. 4.8 – Decomposition of the origin trajectory.

Let $B_{ij} = t_{\overrightarrow{u}_i}(B_{0j})$, $i \in \{0, 1, 2, 3\}$, $j \in \{0, 1, 2, 3\}$. $t_{\overrightarrow{u}}$ stands for the shifting transformation trough the vector $\overrightarrow{u}$.

- On the interval $[0, 0.238061]$, the maximum is B_{00} and the minimum is B_{20} (See Figure 4.9(a)).
- On $[0.238061, 0.580246]$, the maximum is B_{31} and the minimum is B_{11} (See Figure 4.9(b)).
- On $[0.580246, 0.919186]$, the maximum is B_{22} and the minimum is B_{02} (See Figure 4.9(c)).
- On $[0.919186, 1]$, the maximum is B_{13} and the minimum is B_{33} (See Figure 4.9(d)).

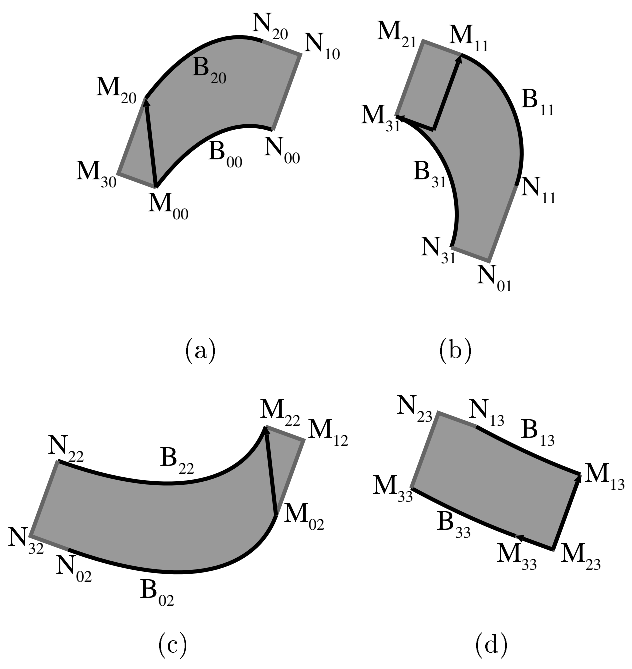

FIG. 4.9 – Normal sub-razings

When the normal sub-razings are superposed, we get exactly the set of points that are really razed with the polygonal pen (See Figure 4.11). Some parts are painted twice. They are in middle gray in Figure 4.10.

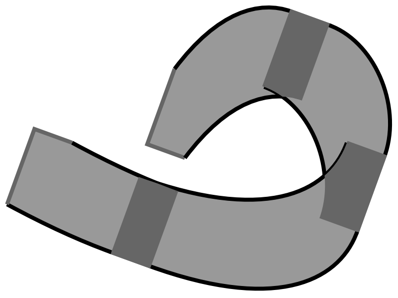

FIG. 4.10 – Superposed normal sub-razings

We can reduce this redundancy through considering the outline with overlapping parts in Figure 4.11. It is obtained by considering the external edges of the polygonal pen in the start position, laying the maximum curves with segments (that are really edges of the pen), doing the same thing for the minimum curves and in the final position, the external edges are considered. Filling in this outline with overlapping with respect to the winding odd algorithm, we get exactly the real razing of the motion.

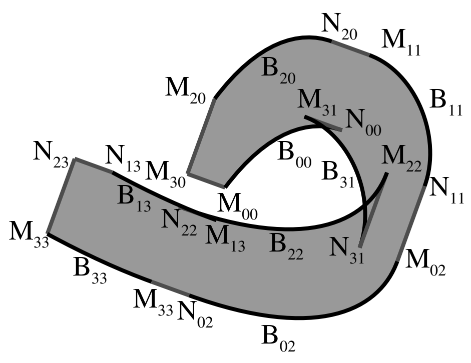

FIG. 4.11 – The razing in outline with overlapping

We can determine the outline that suggests a minimal effort through filling in the interior with determining intersection between B_{00} and B_{31}, the intersection between the curves B_{31} and B_{22} and the intersection between B_{22} and B_{13} (See Figure 4.12).

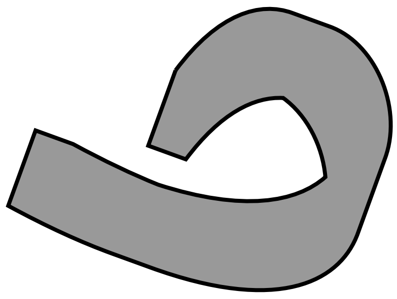

FIG. 4.12 – The razing in true outline

4.4 Stretching model

Now, the way to reconstitute the envelope of the set of the qalam motion is built, we can find out the stretching model. In the following, we present the stretching model based on *cubic* curves. This model has been already developed in Chapter 2. It will be reviewed integrally here in order to facilitate the understanding of the notations. Next, a *quadratic* stretching model will also be presented as improvement of the previous one. A time processing evaluation for the two models will be done through tests on the preceding connection of the letter ر.

4.4.1 Cubic stretching model

As mentioned previously, the trajectory of one of the vertices of the qalam determines the other ones. So, the model will be defined through considering the vertex surrounded with a circle in Figure 2.26. Before presenting the stretching models, some kinds of curves are to be defined.

In the *cubic* stretching model, the dynamic parts in variable sized letters, (for example, the preceding connection in the case of ـر and the following connection in the case of the letter بـ) are *cubic* Bézier specialized curves. The preceding connections are curves belonging to the set $\mathcal{B}_1^c$ and the succeeding ones belong to $\mathcal{B}_2^c$. The sets $\mathcal{B}_1^c$ and $\mathcal{B}_2^c$ are defined as follows. Consider the curve $B = [M_0, M_1, M_2, M_3]$, then:

- The curve $B \in \mathcal{B}_1^c \Leftrightarrow \overrightarrow{M_0M_1} = \lambda \overrightarrow{\imath}$, $\lambda \in \mathbb{R}_*^-$ and $0 \leq \left(\widehat{\overrightarrow{M_3M_2}, \overrightarrow{\imath}}\right) \leq \frac{\pi}{2}$ (See Figure 4.13(a)) where $(\widehat{\overrightarrow{w}_1, \overrightarrow{w}_2})$ stands for the angle between the two vectors $\overrightarrow{w}_1$ and $\overrightarrow{w}_2$. The vector $\overrightarrow{\imath}$ is the axis X vector director.
- The curve $B \in \mathcal{B}_2^c \Leftrightarrow \overrightarrow{M_2M_3} = \lambda \overrightarrow{\imath}$, $\lambda \in \mathbb{R}_*^-$ and $0 \leq \left(\widehat{-\overrightarrow{\imath}, \overrightarrow{M_0M_1}}\right) \leq \frac{\pi}{2}$ (See Figure 4.13(b)).

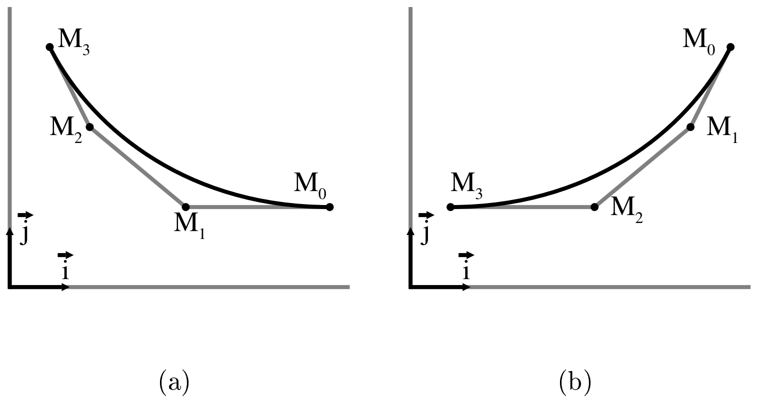

FIG. 4.13 – a) Cubic curve in $\mathcal{B}_1^c$, b) Cubic curve in $\mathcal{B}_2^c$

Now, how can we stretch the curves in $\mathcal{B}_1^c$ and $\mathcal{B}_2^c$. Let $B_1 = [M_{10}, M_{11}, M_{12}, M_{13}]$ be a cubic Bézier curve with control points M_{10}, M_{11}, M_{12} and M_{13} and let $B_2 = [M_{20}, M_{21}, M_{22}, M_{23}]$ be a stretched version of B_1 with an horizontal stretching value h and a vertical one v. We have:

- If $B_1 \in \mathcal{B}_1^c$, then: $M_{20} = M_{10} - (0, v)$, $M_{23} = M_{13} - (h, 0)$, $M_{21} = (1 - c_1)\,M_{20} + c_1 J$, $M_{22} = (1 - c_2)\,J + c_2 M_{23}$ with $c_1, c_2 \in [0, 1]$ satisfying the equations $M_{11} = (1 - c_1)\,M_{10} + c_1 I$, $M_{12} = (1 - c_2)\,I + c_2 M_{13}$ where $\{I\} = (M_{10}M_{11}) \cap (M_{12}M_{13})$ and $\{J\} = \Delta_1 \cap \Delta_2$ such that Δ_1 is the parallel to $(M_{10}M_{11})$ passing through the point M_{20}. Δ_2 is the parallel to $(M_{12}M_{13})$ passing through the point M_{23}.

An illustration of this case is presented in Figure 4.14(a).

- if $B_1 \in \mathcal{B}_2^c$ then: $M_{20} = M_{10}$, $M_{23} = M_{13} - (h, v)$, $M_{21} = (1 - c_1)\,M_{20} + c_1 J$,$M_{22} = (1 - c_2)\,J + c_2 M_{23}$ with $c_1, c_2 \in [0, 1]$ satisfying $M_{11} = (1 - c_1)\,M_{10} + c_1 I$, $M_{12} = (1 - c_2)\,I + c_2 M_{13}$ where $\{I\} = (M_{10}M_{11}) \cap (M_{12}M_{13})$ and $\{J\} = \Delta_1 \cap \Delta_2$ such that Δ_1 is $(M_{10}M_{11})$. Δ_2 is the parallel to $(M_{12}M_{13})$ passing through the point M_{23}.

An illustration of this case is shown in Figure 4.14(b).

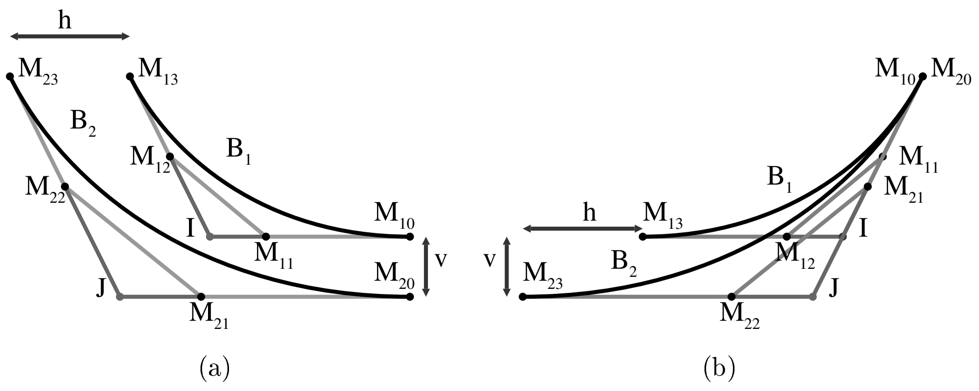

FIG. 4.14 – a) Stretching of a curve belonging to the set $\mathcal{B}_1^c$, b) Stretching of a curve belonging to the set $\mathcal{B}_2^c$

We can remark that the cubic stretching model is based on the conservation of a barycentric relation.

Since the PostScript procedure to produce the dynamic letter should be executed every time the letter has to be drawn, the calculus has to be minimized. The procedure to stretch curves belonging to $\mathcal{B}_1^c$ (respectively $\mathcal{B}_2^c$) is called *ebe* (respectively *eaf*) and has as inputs M_{10}, M_{13}, I, c_1, c_2, h and v. Thus, I, c_1 and c_2 are computed out of the font (they would not be computed by the PostScript interpreter). Whereas, J is determined and so, for M_{20}, M_{21}, M_{22} and M_{23} into the font and every time the procedure to draw the letter is called.

4.4.2 Quadratic stretching model

In the sequel, we present a model based on quadratic curves. The dynamic parts of the letters are now quadratic curves. PostScript doesn't support quadratic curves but every quadratic curve can be presented as a cubic one. When a quadratic curve is a contribution to a drawing, it will be transformed into the corresponding cubic one. In the encoding of the letter ر for example, we determine a quadratic curve to replace the cubic one so that the visual accuracy is satisfied [50].

As in the case of the cubic model, we consider two sets $\mathcal{B}_1^q$ and $\mathcal{B}_2^q$ such that for all $B = [M_0, M_1, M_2]$ we have:

- $B \in \mathcal{B}_1^q \Leftrightarrow \overrightarrow{M_0M_1} = \lambda \overrightarrow{\imath}$, $\lambda \in \mathbb{R}_*^-$ and $0 \leq \left(\widehat{\overrightarrow{M_2M_1}, \overrightarrow{\imath}}\right) \leq \frac{\pi}{2}$ (See Figure 4.15(a)).
- $B \in \mathcal{B}_2^q \Leftrightarrow \overrightarrow{M_1M_2} = \lambda \overrightarrow{\imath}$, $\lambda \in \mathbb{R}_*^-$ and $0 \leq \left(\widehat{-\overrightarrow{\imath}, \overrightarrow{M_0M_1}}\right) \leq \frac{\pi}{2}$ (See Figure 4.15(b)).

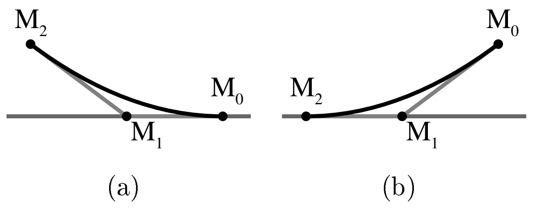

FIG. 4.15 – Quadratic curve in $\mathcal{B}_1^q$, b) Quadratic curve in $\mathcal{B}_2^q$

Now, let's present the way to stretch curves in $\mathcal{B}_1^q$ and $\mathcal{B}_2^q$. Let $B_1 = [M_{10}, M_{11}, M_{12}]$ be the quadratic Bézier curve with control points $M_{10} = (x_{10}, y_{10})$, $M_{11} = (x_{11}, y_{11})$ and $M_{12} = (x_{12}, y_{12})$ and let $B_2 = [M_{20}, M_{21}, M_{22}]$ be a stretched version of B_1 with a horizontal stretching value h and a vertical one v. We have:

- if $B_1 \in \mathcal{B}_1^q$ then
  $M_{20} = (x_{10}, y_{10} - v)$, $M_{22} = (x_{12} - h, y_{12})$
  and $M_{21} = \left(x_{12} - h + \frac{(y_{10}-y_{12}-v)(x_{11}-x_{12})}{y_{11}-y_{12}}, y_{10} - v\right)$.
- if $B_1 \in \mathcal{B}_2^q$ then
  $M_{20} = (x_{10}, y_{10})$, $M_{22} = (x_{12} - h, y_{12} - v)$ and
  $M_{21} = \left(x_{10} + \frac{(y_{12}-y_{10}-v)(x_{11}-x_{10})}{y_{11}-y_{10}}, y_{12} - v\right)$.

An example of stretching for the two types is presented in Figure 4.16. The procedure to perform the stretching in the quadratic case is called *qebe* (in the case of $\mathcal{B}_1^q$) and has as input parameters M_{10}, M_{11}, M_{12}, h and v.

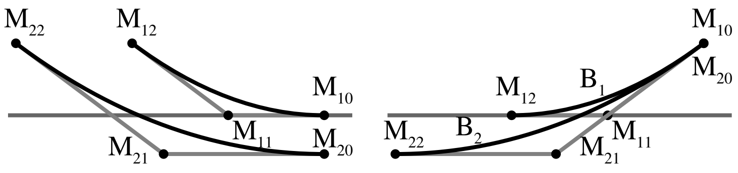

FIG. 4.16 – a) Example of stretching a quadratic curve in $\mathcal{B}_1^q$, b) Example of stretching a quadratic curve in $\mathcal{B}_2^q$

In particular, in the font, the vertical stretching value v depends on h. Let us write $v = \frac{V}{H}h$ where V and H stand respectively for the maximal values of the vertical and horizontal stretching. We can remark that the type of the curve (Type 1 or 2, quadratic or cubic) to stretch is preserved after stretching.

4.4.3 Time processing for each model

It's obvious that the quadratic model requires less time processing than the cubic one, but after stretching a quadratic curve the result can't be used to draw the curve before its transformation into a cubic curve. This transformation is done through a procedure that we call "*quadToCub*". To compare the two models, we have to compare the time necessary to execute "*ebe*" with the time to compute "*qebe*" and "*quadToCub*". The comparison can be done through considering the trajectory of the origin vertex of the preceding connection of the letter ر in both cases of the models. We have programmed a PostScript document named "cubicReh.ps" in which a loop runs n times the *ebe* procedure applied to the connection of the letter ر and strokes this curve where $n =$ 100 000. In the end of the document (just after the loop) the "quit" operator is called to quit the PostScript interpreter and gives the control to the shell process. We get the time necessary to run this document through executing the Linux command "`time -p gs cubicReh.ps`". In the same way, we constituted another document "quadReh.ps" but in the loop body, the procedure "*qebe*" is executed with the same values of stretching followed by "*quadToCub*". The number of iterations is also 100 000 (a big number to solve the problem of rounding values). The observed times for the two cases are presented in Table 4.1 (these values in Table 4.1 can change from a computer to an other but the order will always be kept). So, we can conclude that the quadratic model is better than the cubic one.

TAB. 4.1 – Comparison in time of the cubic and quadratic models

h	v	n
145.207425	12.100619	100000

Procedure	useful time
ebe	6.182 s
qebe + *quadToCub*	5.898 s

4.5 Implementation of static letters

When a letter is static, it is directly implemented in the font with the true outline drawn by the qalam's motion (Outlines without overlapping parts, See Figure 4.17). The true outline is determined with a mean external to the font applying the mathematical notions studied in the paragraph 4.3.

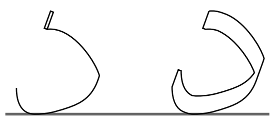

FIG. 4.17 – Qalam's Motion and the corresponding true outlines of the standalone DEL

4.6 Optimized implementation of filled in dynamic letters

4.6.1 Implementation of the static part

In the case of variable sized letters, the encoding can be implemented in PostScript exactly as in the following example of the letter ـر. For the static part of the letter ـر, a non closed outline without overlapping parts is implemented in the font (See Figure 4.18). The outlines are then determined out of the font.

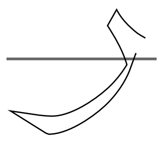

FIG. 4.18 – Non closed outline of the static part of End REH

4.6.2 Implementation of the stretchable part

The stretchable part of a letter implementation can be illustrated once again through considering the preceding connection of the letter ـر. Whether we use a cubic or quadratic curve, the reconstitution of the set of points razed in the plane with the qalam in the preceding connection of the letter ـر is the same. The motion of nib's head is shown in Figure 4.19(a). Let's denote by B^h the trajectory of motion's origin. For all values of h, the scalar function $\mathcal{R}_{\left(B^h,\overrightarrow{u}_3\right)}$ have an extrema on $]0,1[$ (note that $\overrightarrow{u}_2 - \overrightarrow{u}_1 = \overrightarrow{u}_3$). Let t^h be this extrema (h here is not an exponent. It only means that t depends on h). The direct orthogonal ranges $\mathcal{R}_{\left(B,\overrightarrow{u}_1-\overrightarrow{u}_0\right)}$ and $\mathcal{R}_{\left(B,\overrightarrow{u}_3-\overrightarrow{u}_2\right)}$ are strictly monotone over $[0,1]$. So, the motions on $\left[0,t^h\right]$ and $\left[t^h,1\right]$ are normal. Consider the following notations:

- t^h is the extrema of $\mathcal{R}_{\left(B^h,\overrightarrow{u}_3\right)}$ on $]0,1[$,
- B_1^h and B_2^h are the curves obtained by the decomposition of B^h, applying the generalized Bézier algorithm of refinement [14] with respect to t^h (See Figure 4.19(b)),
- $B_{1i}^h = t_{\overrightarrow{u}_i}\left(B_1^h\right)$, $i \in \{0,1,2,3\}$ ($\overrightarrow{u}_0 = \overrightarrow{0}$) and
- $B_{2i}^h = t_{\overrightarrow{u}_i}\left(B_2^h\right)$, $i \in \{0,1,2,3\}$.

Then, we have the results:

- On $\left[0,t^h\right]$ the maximal curve is B_{12}^h and the minimal one is B_{10}^h and
- On $\left[t^h,1\right]$ the maximal curve is B_{21}^h and the minimal one is B_{23}^h.

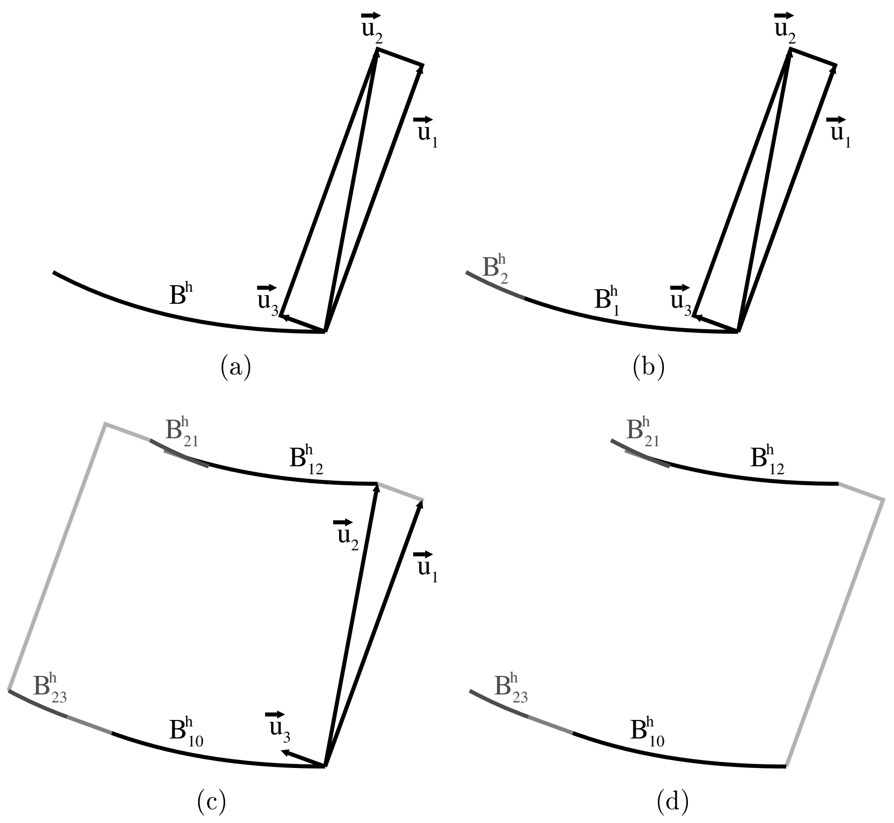

FIG. 4.19 – Motion of the stretchable part of End REH

These results are illustrated in Figure 4.19 with $h = 0$.

Figure 4.19(c) presents the overlapped outline corresponding to the set razed by the qalam in the preceding-connection when filled. But in the font, we use a non closed overlapping outline (See Figure 4.19(d)) to be fitted to the non closed static part of the letter ر (being in Figure 4.18).

As t^h changes as the value of h varies, this parameter must be computed in the font. Consequently, the Bézier refinement process should be incorporated to the procedure of drawing stretchable ر.

Let's recall that the goal here is to optimize the implementation of Kashida in the font. In the sequel, we give the time necessary, in a procedure, to write the stretchable filled in letter REH through applying a quadratic and cubic stretching model.

Denote by t^h_q the parameter t^h in the case of the quadratic model (respectively t^h_c the same parameter in the cubic model) . Then We get:

- $t^h_q = \frac{47.79081298+h}{63.67851945+1.0887286138 \cdot h}$
- $t^h_c = \frac{1412.000904 \cdot h+82586.35211}{.0002 \cdot h+.0125202406} - \frac{22766.53323\sqrt{.003846591162 \cdot h^2+.4499656282 \cdot h+13.15899317}}{.0002 \cdot h+.0125202406}$

So, it is clear that the computing of the quadratic t^h_q requires less time than the cubic t^h_c. Also, the time used to refine a quadratic curve is less than the one we need in the case of a cubic curve. The same comparison holds in the case of shifting curves. In the case of a quadratic model, of course, the quadratic curves are to transform into cubic ones but the time to do so is less than the difference got for the compared previous operations. We have used a daily newspaper to find out the occurrences number of the letter ـر. We've found 5684 occurrence. In a first PostScript document, a font containing only the letter ـر is developed with a preceding-connection based on a quadratic model. In the second, the same is done with a cubic model. The preceding connection is parametrized in h, in the font to support the stretching. So, with a null value of stretching, the computing necessary for the stretching operation is also executed. The evaluation has been done only with null values of stretching. The amount of operations relative to the stretching is independent of the value of h. In the two documents, we've written the letter ـر 5684 times. Then, we have an evaluation of the time to execute in the two cases. The results of such evaluation are presented in Table 4.2.

TAB. 4.2 – Comparison between the complete implementation of the end REH in cubic and quadratic models

h	v	n
0	0	5684

Model	time
Quadratic	0.831 s
Cubic	0.880 s

The daily newspaper was chosen in our study only because we have it in its electronic format what simplified the counting of the number of occurrences of ـر. This daily newspaper produces 90 000 copies (of course in one day) but the saved time 0.049 $(0.880 - 0.831)$ using the quadratic model than the cubic for the letter REH will take place only in the first copy since the result of the first copy would be reused to produce the thousands of copies needed using the offset printing techniques. We should recall that this time is saved only for one letter and there are a lot of stretchable letter others than ـر. So, the quadratic model seems to be better than the cubic one. The case becomes more important when we have to print a big document (in term of pages) in more than one copy with a laser printer (printing some courses at some universities for example with laser copier-printers). The time saved with the quadratic model would be surely very important.

4.7 Optimized implementation of outlined dynamic letters

In this section, we will show how can we get a font allowing writing Arabic with stretchable outlined letters. The letters should be implemented with non closed outlines to get writings like those in Figure 4.3. Samples of letters in this form are showed in Figure 4.20. In the same way, the concepts of implementation are illustrated with the letter ـر. The static part of the letter ـر is encoded in the same way as for the filled in version studied in Section 4.6.

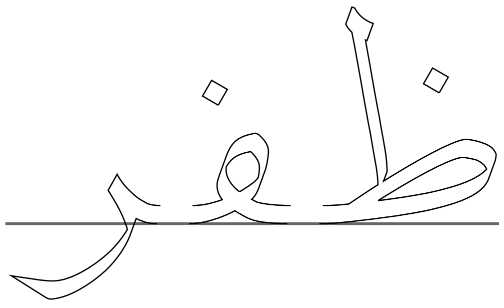

FIG. 4.20 – letters with non closed and non overlapping outlines

Now, what about the preceding connection. In this case, only the top and bottom parts of the connection are considered as it is presented in Figure 4.21. Figure 4.21 is obtained from Figure 4.19(d) through deleting the two edges of the qalam on the right. The connection is not on its final state because we have to remove the overlapping parts representing the intersection between the curves B^h_{12} and B^h_{21} (See Figure 4.21 with a big scaling factor).

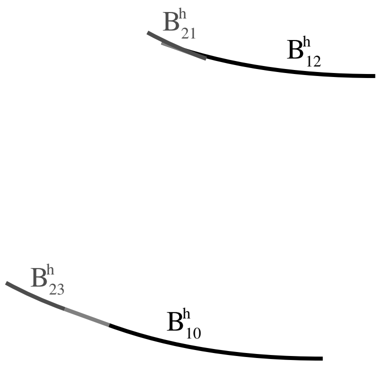

FIG. 4.21 – Opened overlapping of stretchable part of End REH

The non overlapping curves of the preceding connection can't be directly determined because there is an intersection between B^h_{12} and B^h_{21}. Note that this intersection is unique. Let r^h and s^h be the two variables on $[0,1]$ characterizing the intersection coefficients of B_{12}^h and B_{21}^h respectively. Their values are solutions of Equation 4.1.

$$B_{12}^h(r) - B_{21}^h(s) = (0,0) \tag{4.1}$$

The values of t^h, r^h and s^h change as h changes. The values are to be computed dynamically in the font. As B^h is a quadratic curve, the range $\mathcal{R}'_{\left(B^h,\vec{u}_3\right)}(t)$ is a linear function of t. Then, it is easy to determine explicitly t^h in function of h. The formula is given in Equation 4.2. It was in order to get an easy and explicit determination of t^h that we were to represent the stretchable parts in letters with quadratic curves instead of cubic ones. The function t^h is a monotone increasing function of h.

$$t^h = \frac{47.79081298 + h}{63.67851945 + 1.0887286138 \cdot h} \tag{4.2}$$

In general, the determination of the intersection between two quadratic or cubic Bézier curves is done through iterations (there is no explicit solution). Since the PostScript procedure to produce a character, in a dynamic font, is repeated whenever the letter is to draw, the determination of the curve intersections coefficients in the procedure, applying an iterative algorithm is of a high cost. So, we have to find out a way to minimize the time necessary to compute r^h and s^h.

In order to implement a font in stretchable outlines, we use approximations. So, we need to consider some precisions or tolerances. In the following, we define the notions of digital equivalence and tolerance.

Definition 4.7.1 (Digital Equivalence). *Let $M_1 = (x_1, y_1)$ and $M_2 = (x_2, y_2)$ be two points in $\mathbb{Z}^2$. M_1 is digitally equivalent to M_2 if and only if $|x_1 - x_2| \leq 1$ and $|y_1 - y_2| \leq 1$. The digital equivalence is then denoted by $M_1 \equiv M_2$.*

So, for the visual aspect, a point in $\mathbb{Z}^2$ and points in his direct neighbor would then be considered as the same.

To each point in $\mathbb{R}^2$ to draw corresponds an image in $\mathbb{Z}^2$ by a function denoted $\mathcal{N}$. The tolerance of a Bézier curve can then be defined as follows:

Definition 4.7.2 (Tolerance of a Bézier curve). *Let B be a Bézier curve and $\tau = \frac{1}{n}$, $n \in \mathbb{N}^*$. The value τ is a tolerance of B if and only if for all t_1 and t_2 in $[0,1]$ we have:*

$$|t_1 - t_2| \leq \tau \Longrightarrow \mathcal{N}(B(t_1)) \equiv \mathcal{N}(B(t_2))\ .$$

The notions of digital equivalence and the tolerance are tools toward the formalization of the visual accuracy. We can define the notion of the optimal tolerance.

Definition 4.7.3 (Optimal Tolerance). *Let B be a Bézier curve. The optimal tolerance of B is the greater element of the set $\left\{\frac{1}{m}/m \in \mathbb{N}*, \text{ where } \frac{1}{m} \text{ is a tolerance of } B\right\}$*

We have the property:

Property 4.7.1. *Let us consider the Bézier curves B_1, B_2 and B_3. Let τ_1, τ_2 and τ_3 be their optimal tolerances respectively, then we have the properties:*
- *if B_2 is a stretched version of B_1 then $\tau_2 \leq \tau_1$,*
- *if B_2 and B_3 are curves obtained from B_1 by a Bézier refinement according to a parameter t in $[0, 1]$, then we have $\tau_2 \geq \tau_1$ and $\tau_3 \geq \tau_1$.*

We can now continue the presentation of the method to determine he intersection coefficients. Let us consider once again the notations:
- τ_{12}^h the tolerance of B_{12}^h depending on h.
- τ_{21}^h the tolerance of B_{21}^h depending on h.

In Naskh style, the stretching variable h can vary from 0 to H, where H is 6 diacritic points (12 dp for all Kashida and 6 dp for each of its components). Consider a font in 1000 points size, $H = 580.829699$. The determination of the coefficients r^h and s^h would be based on approximations as follows.

The Maple system [38] can be used to get the following properties:

Property 4.7.2.
- *The function r^h is monotone increasing of h on $[0, H]$,* (1)
- *The function s^h has an extrema on $[0, H]$. It is monotone increasing of h on the left edge of $[0, H]$ and monotone decreasing of h on the right edge of $[0, H]$,* (2)
- *The functions τ_{12}^h and τ_{21}^h are monotone decreasing on $[0, H]$,* (3)
- *and $\tau_{12}^h \leq \tau_{21}^h$ on $[0, H]$. This means that for $h \in [0, H]$, τ_{12}^h is an optimal tolerance of B_{12}^h and a tolerance of B_{21}^h.* (4)

Now, we have all the properties necessary to build the function giving the approximated values of r^h and s^h.

Let $r_{min} = r^0$, $r_{max} = r^H$. Consider the sequence $R = (r_i)_{i=0,\ldots,n}$ such that $r_i = r_{min} + \frac{r_{max}-r_{min}}{n} i$, where $n = \left[\frac{r_{max}-r_{min}}{\tau_{12}^H}\right] + 1$. Let $[x]$ be the integer part of x. Let's also consider the two sequences $\mathcal{H} = (h)_{i=0,\ldots,n}$ and $S = (s_i)_{i=0,\ldots,n}$ such that $\{h_i, s_i\}_{i=0,\ldots,n}$ are solutions of Equation 4.1 where h and s are the unknown and r is given taking r_i as value. The system Maple can help to solve all the equations. In some cases, some of these equations can require some minutes when B^h is quadratic and even some hours when B^h is cubic. That's another reason to choose quadratic curves to encode dynamic parts. We get the following results:

- $r_{i+1} - r_i < \tau_{12}^H$ $\forall i = 0, \ldots, n-1$,
- $r_{i+1} - r_i < \tau_{12}^h$ $\forall i = 0, \ldots, n-1$ $\forall h \in [0, H]$ (according to the property 4.7.2(3)). This means that $\mathcal{N}\left(B_{12}^h(r_i)\right) \equiv \mathcal{N}\left(B_{12}^h(r_{i+1})\right)$ $\forall h \in [0, H]$,
- $|s_i - s_{i+1}| < \tau_{21}^h$ $\forall i = 0, \ldots, n-1$ $\forall h \in [0, H]$ (according to the property 4.7.2(2,4)). This means that $\mathcal{N}\left(B_{21}^h(s_i)\right) \equiv \mathcal{N}\left(B_{21}^h(s_{i+1})\right)$ $\forall h \in [0, H]$.

Finally, we have a property useful for the determination of the approximation function, $\forall i \in \{0, \ldots, n-1\}$, $\forall h$, $h_i \leq h \leq h_{i+1}$, we have:

- $r_i \leq r^h \leq r_{i+1}$ and $\mathcal{N}\left(B_{12}^h\left(r^h\right)\right) \equiv \mathcal{N}\left(B_{12}^h(r_i)\right)$ and so, $\mathcal{N}\left(B_{12}^h\left(r^h\right)\right) \equiv \mathcal{N}\left(B_{12}^h(r_{i+1})\right)$.
- $\left(s_{i+1} \leq s^h \leq s_i \text{ or } s_i \leq s^h \leq s_{i+1}\right)$ and $\mathcal{N}\left(B_{21}^h\left(s^h\right)\right) \equiv \mathcal{N}\left(B_{21}^h(s_i)\right)$ and so, $\mathcal{N}\left(B_{21}^h\left(s^h\right)\right) \equiv \mathcal{N}\left(B_{21}^h(s_{i+1})\right)$.

Then, we know very well the tolerant values of $B_{12}^h\left(r^h\right)$ and $B_{21}^h\left(s^h\right)$.

Now, consider the sequence $P = (P_i)_{i=0,\ldots,n} = (r_i, s_i)_{i=0,\ldots,n}$. Let's approximate the set of points P_i by a quadratic Bézier curve A such that $A(0) = P_0 = (r_0, s_0)$ and $A(1) = P_n = (r_n, s_n)$. In order to approximate a set of points $\{P_0, \ldots, P_n\}$ by a curve parametrized on $[0, 1]$, we may associate to every point P_i a coefficient t_i in $[0, 1]$ such that the sequence $(t_i)_{i=0,\ldots,n}$ is strictly monotone increasing and $t_0 = 0$ and $t_n = 1$.

First, let us determine $(t_i)_{i=0,\ldots,n}$. Recall that t^h (defined in Equation 4.2) is a monotone increasing function of h. Since r^h is monotone increasing, the sequence $(h_i)_{i=0,\ldots,n}$ deduced through the sequence $(r_i)_{i=0,\ldots,n}$ is monotone increasing. Consider $(t_i)_{i=0,\ldots,n}$ such that $t_i = \frac{t^{h_i} - t^0}{t^H - t^0}$. $(t_i)_{i=0,\ldots,n}$ is a strictly monotone increasing sequence with $t_0 = 0$ and $t_n = 1$.

Consider that the quadratic Bézier curve to be determined has the control points (r_0, s_0), (x_1, y_1) and (r_n, s_n). We have to determine (x_1, y_1) in order to minimize the value D (See Equation 4.3).

$$D = \sum_{i=0}^{n} \left(P_i - A(t_i)\right)^2 \tag{4.3}$$

D can be minimal if the following mandatory conditions are satisfied:

- $\frac{\partial D}{\partial x_1} = 0$ and
- $\frac{\partial D}{\partial y_1} = 0$.

$\frac{\partial D}{\partial x_1} = 0$ implies that $x_1 = \frac{\sum_{i=0}^{n} t_i(1-t_i)\left(r_i - (1-t_i)^2 r_0 - t_i^2 r_n\right)}{2\sum_{i=0}^{n} t_i^2(1-t_i)^2}$ and $\frac{\partial D}{\partial y_1} = 0$ implies that $y_1 = \frac{\sum_{i=0}^{n} t_i(1-t_i)\left(s_i - (1-t_i)^2 s_0 - t_i^2 s_n\right)}{2\sum_{i=0}^{n} t_i^2(1-t_i)^2}$.

Suppose that $A(t) = (r(t), s(t))$. Then A is a good approximation if the condition in Formula 4.4 is satisfied.

$$|r(t_i) - r_i| < \tau_{12}^H \; \forall i = 0, \ldots, n \tag{4.4}$$

If the condition 4.4 isn't satisfied the sequences $\mathcal{H}$ and P are subdivided in two pairs of sub-sequences $\left\{(h_i)_{i=0,\ldots,k}, (P_i)_{i=0,\ldots,k}\right\}$ and $\left\{(h_i)_{i=k,\ldots,n}, (P_i)_{i=k,\ldots,n}\right\}$ such that k verifies Formula 4.5 and the process is repeated.

$$|r(t_k) - r_k| = \max_{i=0}^{n} |r(t_i) - r_i| \tag{4.5}$$

For a more concrete view, we give in some details the method used to determine the approximated outline of the stretchable part of the letter ر i.e the preceding connection. We recall that the preceding connection is a curve in $\mathcal{B}_1$ that is parametrized (respecting the stretching model given in the chapter) to take into account the stretching. As we know, the fonts in PostScript are designed in the size 1000 pt (PostScript point unit). The basic (the curve with a null stretching) preceding connection curve B has as control points in this size:

- $M_0 = (0, 0)$,
- $M_1 = (-40.4367, 0)$ and.
- $M_2 = (-67.99062, 14.92163)$.

Let h be the horizontal stretching amount and v the vertical corresponding one. v verify that $v = \frac{V}{H}h$ where H and V are maximal amounts of stretching in horizontal and vertical directions respectively. Since H is six diacritic points and V is a half of a diacritic point then $v = \frac{h}{12}$. The control points of the preceding connection of the letter ر B^h when parametrized on h ($B^0 = B$) become:

- $M_0 = \left(0, -\frac{h}{12}\right)$,
- $M_1 = \left(-40.43770052 - .8461247196 \times h, -\frac{h}{12}\right)$ and
- $M_2 = (-67.99062 - h, 14.92163)$.

The vectors defining the motion of the qalam in the size 1000 pt are:

- $\overrightarrow{u}_0 = \overrightarrow{(0,0)}$
- $\overrightarrow{u}_1 = \overrightarrow{(24.2376, 66.5923)}$,
- $\overrightarrow{u}_2 = \overrightarrow{(13.1389, 70.6320)}$ and
- $\overrightarrow{u}_3 = \overrightarrow{(-11.0987, 4.0396)}$.

We determine the value of extrema t^h when $h = 0$ and $h = H$. Then, we have:

- $t^0 = .750501$ and
- $t^H = .903133$,

We get the results concerning the Tolerances of B^h_{12} and B^h_{21} when $h = H$:

- $\tau^H_{12} = 0.00106952$ and
- $\tau^H_{21} = 0.0344828$.

Using Maple or other tools , we solve Equation 4.1 for $h = 0$ and $h = H$. So, we determine the following values:

- $r_{min} = r^0 = .879858$,
- $r_{max} = r^H = .980454$,

We obtain:

- $n = \left[\frac{r_{max}-r_{min}}{\tau^H_{12}}\right] + 1 = 95$,
- $r_i = .879858 + \frac{0.100596}{n} i$.

Getting the sequence $R = (r_i)_{i=0,\ldots,n}$, we use once again a computer algebra system such as Maple to solve the equation with r, taking r_i as value and considering s and h as variables. We get:

- $t_i = \frac{t^{h_i}-t^{h_0}}{t^H-t^{h_0}} = \frac{t^{h_i}-t^0}{t^H-t^0} = \frac{t^{h_i}-.750501}{.152632}$. t_i are real numbers in $[0,1]$ associated to $P_i = (r_i, s_i)_{i=0,\ldots,n}$ for approximation.
- $s(t)$ and $r(t)$ are component functions of a quadratic Bézier curve $A(t)$ where:
  - $r(t) = (1-t)^2 r_0 + 2t(1-t)x_1 + t^2 r_{95}$
  - $s(t) = (1-t)^2 s_0 + 2t(1-t)y_1 + t^2 s_{95}$

To minimize D, we get $(x_1, y_1) = (0.912975, 0.494131)$.

Then we have,

- $\Delta_r = \text{Max}^n_{i=0} |r(t_i) - r_i| = 0.002442$ and
- $\Delta_s = \text{Max}^n_{i=0} |s(t_i) - s_i| = 0.038800$.

We have $\Delta_r > \tau^H_{12}$, so the sequences would be subdivided at index $i = 72$.

The process is repeated with the two sub-sequences. We remark that $\Delta_r < \tau^H_{12}$ and automatically, due to the construction of the sequence R, $\Delta_s < \tau^H_{21}$.

We get then a curve A_1 of control points $M_{10} = (r_0, s_0) = (0.8798580, .361504)$, $M_{11} = (x_{11}, y_{11}) = (0.911133, 0.413212)$ and $M_{12} = (r_{72}, s_{72}) = (0.956099, 0.313817)$. In the same way, we get a curve A_2 with control points $M_{20} = (r_0, s_0) = (0.956099, 0.313817)$, $M_{21} = (x_{21}, y_{21}) = (0.966307, 0.282649)$ and $M_{22} = (r_{23}, s_{23}) = (0.980454, 0.182371)$.

These two curves allow us to approximate the intersection coefficients. For more clarity let's give an abstract algorithm. Let h be a stretching amount, then we have:

```
IF h is in [0,179.595499[ THEN
BEGIN
    $t = \frac{t^h-0.750501}{0.877231-0.750501}$;
```

```
        /* $t^h$ means that t depends
        on $h$ and denote $t$ exponent $h$*/
        $A_1(t)$;
    END
    ELSE
        IF h is in [179.595499,580.818536] THEN
        BEGIN
            $t = \frac{t^h - 0.877231}{0.903132 - 0.877231}$;
            $A_2(t)$;
        END
```

We conclude that for the preceding connection of the letter ـر, the determination of the intersection coefficients are computed with the evaluation of a curve A_1 where h is in $[0, 179.595499[$ and another curve A_2 if h is in $[179.595499, 580.818536]$.

The approximation has been tested with arbitrary values of h and we got intersection coefficients with respect of the curves tolerances.

With this mechanism, we can encode the preceding connection to get the stretchable letter ـر with non closed outline to be used to write words in outlines (See Figure 4.22).

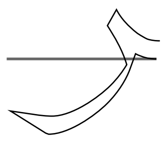

FIG. 4.22 – Opened non overlapping outline of End REH

Now, the process to determine the intersection coefficients can be more improved. Actually, instead of a quadratic approximation of the sequence of points $P = (P_i)_{i=0,\ldots,n} = (r_i, s_i)_{i=0,\ldots,n}$, we would use a cubic one. The reason why we do this is explained in the following.

Let's approximate the set of points P_i respecting the same way and constraints but with a cubic Bézier curve B of control point (r_0, s_0), (x_1, y_1), (x_2, y_2) and (r_n, s_n). We have to determine (x_1, y_1) and (x_2, y_2) in order to minimize D (See equation 4.3).

D can be minimal if the following mandatory conditions are satisfied:

- $\frac{\partial D}{\partial x_1} = 0$,
- $\frac{\partial D}{\partial y_1} = 0$,
- $\frac{\partial D}{\partial x_2} = 0$ and
- $\frac{\partial D}{\partial y_2} = 0$.

We get a system with four equations and x_1, x_2, y_1 and y_2 as unknowns. Solving the system through considering the preceding connection of the letter ـر, the values of the unknown control points are as follows:

- $(x_1, y_1) = (0.910238, 0.328969)$
- $(x_2, y_2) = (0.927859, 0.501073)$

Now, let us consider the Bézier curve B of control points $(r_0, s_0) = (0.879858, 0.361504)$, $(x_1, y_1) = (0.910238, 0.328969)$, $(x_2, y_2) = (0.927859, 0.501073)$ and $(r_{95}, s_{95}) = (0.980454, 0.182371)$. The intersection coefficients are $s(t)$ and $r(t)$ components of $B(t) = (r(t), s(t))$. We have:

- $\Delta_r = \mathrm{Max}_{i=0}^{n} |r(t_i) - r_i| = 0.000501$ and so $\Delta_r < \tau_{12}^{H}$ and
- $\Delta_s = \mathrm{Max}_{i=0}^{n} |s(t_i) - s_i| = 0,011959$ and then $\Delta_s < \tau_{21}^{H}$

We conclude that only one cubic curve approximates the sequences of points $(P_i)_{i=0,\ldots,n} = (r_i, s_i)_{i=0,\ldots,n}$.

A comparison between the quadratic approximation and the cubic one seems to be necessary. It is obvious that the evaluation of a cubic Bézier curve requires more time than a quadratic one. But in our case, the set of points $(P_i)_{i=0,\ldots,n}$ is approximated by two quadratic curves. So, we have to do some alternative tests about the value of h, when it is in $[0, 179.595499[$ or in $[179.595499, 580.818536]$. We know that in algorithmic theory, tests are generally time consuming in CPU especially in PostFix programming languages. So, a comparison of the two approximation models would help to determine the better one. We have developed two PostScript files where we write the letter ـر in opened non overlapping outlines (as in Figure 4.22) 1000 times based on the quadratic approximation in first one and on the cubic model in the second one. The time necessary in the two cases is presented in Table 4.3. We remark that the cubic approximation of the intersection coefficients calculation is better (tests in PostScript are of high cost in CPU).

TAB. 4.3 – Comparison in time of the cubic and quadratic approximations

h	v	n
145.207425	12.100619	1000

approximation	useful time
quadratic	2.18 s
cubic	1.22 s

For the stretchable version of letter ـق in the end of a word, the corresponding generated sequences of points $(P_i)_{i=0,\ldots,n}$ can be approximated only by one quadratic Bézier curve. So, it has been implemented with a quadratic Bézier curve. But for the letter ـر, the approximation process is based on a cubic Bézier curve.

In Section 4.6, we have implemented the letter ـر to be filled in with overlapping outlines. A part will be painted twice when filling in the closed path defining this letter. With reference to Section 4.7, we can encode the letter ـر with true outlines, in other words, with outlines without any overlapping. This is possible because we can determine the intersection and remove the overlapping. What would require less time when we have to draw a filled in ـر, encoding with overlapping outlines or outlines without overlapping. An evaluation of the two methods is shown in Table 4.4.

Tab. 4.4 – Filled End REH, overlapping outlines versus non-overlapping outlines

h	v	n
145.207425	12.100619	10000

outlines	useful time
overlapping	6.74 s
non overlapping	7.03 s

We conclude that when we have to write filled in letters, we don't need to take into account the intersection since that will not optimize the processing.

4.8 Conclusions

Calligraphing stretchable, variable sized outlined or darken Arabic letters needs the development of techniques to optimize the implementation of such letters and PostScript seems not to be directly suitable for dynamic Arabic fonts. On the other hand, to typeset an Arabic document, with all kinds of Arabic letters, a unique font is not enough. The availability of various fonts and the variety of letters inside the same font contribute significantly to improve the rendering of Arabic documents respecting the rules of Arabic calligraphy. Only Naskh style, where nib's head keep in a constant inclination, has been studied here. This approach will be extended next to styles where nib's head can be in translation and/or in rotation. This can also help to compose calligraphic mathematical formulas. So, until now, we only developed some pieces of the dynamic font. The concepts presented here are suitable for other letters because the preceding connections have the same general shape. Nevertheless, developing the whole dynamic font in a reasonable time requires also the development of other specific graphical tools. That will be done in coming works.

Chapitre 5

Conclusions et perspectives

5.1 Conclusions

Ce travail présente un formalisme mathématique pour les fontes arabes dynamiques. Le formalisme a été appliqué concrètement et implanté sur machine pour le développement de fontes qui respectent les règles de la calligraphie arabe. On peut dire que les résultats originaux contenus dedans ont été l'aboutissement du programme suivant :

- Étude de quelques caractéristiques de la calligraphie arabe, en particulier, les types de ligatures et les kashida.
- Développement d'un modèle mathématique pour représenter les caractères en termes de courbes extensibles de manière curvilinéaire.
- Développement d'une méthode d'étude des variations de fonctions planes par rapport à un ensemble fini de vecteurs constants. Ce formalisme a été mis en oeuvre pour déterminer une manière pour comparer les translatés d'une même courbe afin de déterminer l'enveloppe des surfaces balayées par la plume du calligraphe lors de l'écriture.
- Optimisation de l'implantation du modèle d'extension ainsi que celle des méthodes mises en oeuvre, dans des fontes PostScript.

Tout au long du travail, nous avions rencontré un bon nombre de difficultés. En particulier, le manque d'outils graphiques qui supportent nos besoins de manière adéquate a été d'une grande gène au début. La simple préparation d'exemples posait des difficultés. Il fallait ensuite expliciter et présenter le problème. Sans outils, les codages des lettres utilisées dans les exemples prenaient beaucoup de temps.

Le système TEX est un outil performant pour le traitement de documents scientifiques de haute qualité. Ses algorithmes relatifs à la gestion des composantes du document (mot, lignes, paragraphe, ...) sont suffisamment raffinés pour produire des documents de bonne qualité. Une telle qualité n'a été possible que parce que le système TEX et la panoplie d'outils qui l'accompagnent s'appuient sur un effort remarquable en géométrie informatique (Computational Geometry) et en Conception Géométrique Assistée par Ordinateur CGAO (Computer Aided Geometric Design CAGD). Le système TEX avait été développé au moment où ces domaines d'études étaient en plein essor. Deux travaux, entre autres, ont servi de support direct pour TEX : le travail présenté dans la thèse de John Douglas Hobby [26], dirigée par D.E. Knuth lui même. Dans ce travail, la théorie et la pratique des numérisations des plumes sont données. La numérisation des chemins des plumes définies dans METAFONT est une application directe de ce travail. Une deuxième contribution de taille est le travail de Leo Guibas et al. [33] relatif aux fondements mathématiques des sciences informatiques, intitulé "A Kinetic Framework for Computational Geometry". Cette oeuvre présente les tracings qui sont d'un usage de base dans les numérisations du programme METAFONT et autres environnements et langages comme PostScript.

Ces considérations suggèrent que la construction d'un système adapté aux besoins de la calligraphie arabe a fort à voir avec l'étude de formalismes mathématiques des objets de cette calligraphie. Ce n'est pas chose simple mais c'est probablement le prix à payer pour répondre aux exigences d'une écriture aussi raffinée ou, du moins qui n'a rien à envier à l'écriture mathématique.

L'étude et recherche de moyens et outils pour traiter des documents à base du script arabe où seraient respectées les règles de la calligraphie arabe tel était globalement le cadre au début de notre travail. Nous n'avons résolu qu'une partie de la vaste problématique que cela soulève :

- Le modèle d'extension proposé est une contribution au développement d'un environnement d'apprentissage de la calligraphie arabe.
- Ce même modèle d'extension peut être utilisé de manière déterminante dans les outils de reconnaissance automatique des écritures des calligraphes arabes.
- Il offre, par ailleurs, une bonne base pour une réflexion sur le développement de nouveaux logiciels de traitement de texte et de nouveaux langages de codage de fontes.

5.2 Perspectives et problèmes ouverts

Ce travail a ouvert des problèmes et questionnements de diverses natures. Dans l'immédiat, il conviendrait de s'attaquer aux objectifs suivants :

- Développer un logiciel graphique qui assiste un maximum à automatiser la génération de fontes arabes en tenant compte de l'extension curviligne. Nous avons

fait une étude presque complète des mathématiques de fond nécessaires au développement d'un tel logiciel. Il reste à l'appliquer à plusieurs fontes.
- Étudier la morphologie mathématique et essayer de revoir quelques algorithmes et opérateurs mathématiques. Nous nous en servirons pour générer les squelettes des écritures de lettres arabes. Ces squelettes sont les données d'entrée pour la génération automatique de fontes par l'outil graphique.
- Étendre le formalisme mathématique présenté dans ce travail pour offrir un support aux écritures dans certains styles de la calligraphie arabe, entre autres dans les cas où la plume subit des rotations en plus des translations.
- Améliorer et généraliser le modèle d'extension pour offrir le support pour la production et le maniement de symboles mathématiques extensibles tels les accolades, les parenthèses, le signe intégrale ... Ce travail sera une première ossature du développement d'une fonte mathématique des symboles extensibles.
- Mise au point d'environnement de formatage ou de traitement de texte qui se base sur les fontes que nous allons finir. Allons-nous adapter ou étendre des outils déjà existants ou développer de nouveaux ? La réponse n'est pas encore claire mais il n'y a pas d'impasse, nous pourrons à la limite développer des programmes qui permettent de formater une source texte en se basant sur les fontes développées pour générer des documents PostScript qui supportent les composantes dynamiques.

Bibliographie

[1] Abdelouahad Bayar, Khalid Sami, “An Optimal Way to Encode the Outlines of Variable Sized Arabic Letters in a PostScript Font”, *The 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Full Papers Proceedings*, University of West Bohemia, Plzen Czech Republic, pp. 57-64, 2008.

[2] Abdelouahad Bayar, Khalid Sami, “Optimization of the curvilinear stretching in a PostScript font with quadratic Bézier curves”, *The International Arab Conference of e-Technology, Proceedings of IACe-T'2008*, Arab Open University, Amman - Jordan, pp. 97-104, 2008.

[3] Abdelouahad Bayar, Khalid Sami, “How a font can respect rules of Arabic calligraphy”, *The International Arab Journal of e-Technology*, Vol.1, No.1, pp. 1-18, 2009.

[4] Adobe Systems Incorporated, *Adobe Type 1 Font Format*, Part No. LPS0064, 1990.

[5] Adobe Systems Incorporated, *PostScript Language Reference Manual*, MA:Addison-Wesley , Third edition,Massachusetts, 1999.

[6] Adobe Systems Incorporated, *OpenType User Guide for Adobe Fonts*, October 2008.

[7] Ahmed El Houssaini, *The Arabic Calligraphy*, Dar Attaila, 1994.

[8] Ameer M. Sherif et Hossam A. H. Fahmy, “Parameterized Arabic font development for AlQalam”, *TUGboat*, vol. 29, Jan. 2008. Appeared originally in *EuroBachoTEX* 2007: the 17th Annual Meeting of the European TEX Users, Bachotek, Poland.

[9] Ameer M. Sherif et Hossam A. H. Fahmy, “Meta-designing parameterized Arabic fonts for AlQalam”, *TUGboat*, vol. 29, Nov. 2008. Appeared originally in *TUG* 2008: The Annual Meeting of the International TEX Users Group, Cork, Ireland.

[10] Azzeddine Lazrek, “A package for typesetting Arabic mathematical formulas”, *Die TEXnische Komodie*, DANTE e.V., Vol.13, No. 2, pp. 54–66, 2001.

[11] Azzeddine Lazrek, *Vers un système de traitement du document scientifique arabe*, Département des mathématiques, Thèse de Doctorat d'Etat, Université Cadi Ayyad, Faculté des sciences semlalia - Marrakech, février 2002.

[12] Azzeddine Lazrek, “CurExt, Typesetting variable-sized curved symbols”, *EuroTEX 2003 Pre-prints: 14th. European TEX conference,* Brest, France, pp. 47-71, 2003.

[13] Brian A. Barsky, *The Beta-splines: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures*, Department of Computer science, Ph.D. dissertation, University of Utah Salt Lake City, December 1981.

[14] Brian A. Barsky, *Arbitrary Subdivision of Bézier Curves*, Technical Report UCB.CSD 85/265, Computer Science Division, University of California, 1985.

[15] Daniel M. Berry, “Stretching Letter and Slanted-Baseline Formatting for Arabic, Hebrew and Persian with dittroff/fforttid and Dynamic PostScript Fonts”, *Software-Practice and Experience*, vol. 29, no. 15, pp. 1417-1457, 1999.

[16] Donald Ervin Knuth, *The TEXBook, Computers and Typesetting*, Reading MA: Addison-Wesley, Vol.A, 1984.

[17] Donald Ervin Knuth, *The METAFONTbook, Computers and Typesetting,* MA: Addison-Wesley, Vol.C, 1986.

[18] Donald Ervin Knuth and Pirre MacKay, “Mixing right-to-left texts with left-to-right texts”, *TUG-boat 8*, No 1, pp. 14–25, 1987.

[19] D. Weise, D. Adler, *TrueType and Microsoft Windows 3.1*, Technical Report, Microsoft Corporation, Redmond, WA, 1992.

[20] George Williams, *Font forge manual web pages*, http://fontforge.sourceforge.net, 2008.

[21] High-Logic, *Font Creator Manual*, http://www.high-logic.com, 2005 .

[22] Jacques André, Bruno Borghi, “Dynamic Fonts”, *PostScript Language Journal*, Vol 2, no 3, pp. 4-6, 1990.

[23] Jacques André, Irène Vatton, *Contextual Typesetting of Mathematical Symbols Taking Care of Optical Scaling*, Technical report No 1972, INRIA, October 1993.

[24] Jacques André, Irène Vatton, “Dynamic Optical Scaling and Variable-sized Characters”, *Electronic Publishing,* Vol 7, No 4, pp. 231-250, December 1994.

[25] Jeffrey M. Lane, Richard F. Riesenfelf, “Bounds on Polynomial”, *BIT*, Vol 21, No 1, pp. 112-117, 1981.

[26] John Douglas Hobby , *Digitized Brush Trajectories*, Ph.D. Thesis, Department of Computer sciences, Stanford University, August 1985.

[27] John Douglas Hobby, *A User's manual for MetaPost*, Technical Reports Report, AT& T Bell Laboratories Computing Science 162, 1992.

[28] Joris Van Der Hoeven, *The TEXmacs User Manual,* http://www.texmacs.org, Version 1.0.6.6, 2003.

[29] Klaus Lagally, ArabTEX - Typesetting Arabic with Vowels and Ligatures, EUROTEX'92, Prague, 1992.

[30] Klaus Lagally, *ArabTEX User Manual,* Universitat Stuttgart, Fakultat Informatik, Version 4.00, 2004.

[31] KOffice. '*Kontour: Koffice drawing tool*, *http://www.koffice.org/Kontour*, 2001.

[32] Leslie Lamport, *LATEX-A Document Preparation System*, Reading MA: Addison Wesley, 1985.

[33] Leo Guibas, Lyle Ramshaw, Jorge Stolfi, "A Kinetic Framework for computational Grometry", *24th Annual Symposium on Foundations on Computer Science,* IEEE Computer Society Press, 1983.

[34] LYX Team, *The LYX User Manual*, *http://www.LYX.org*, Version 1.3.4, 2004.

[35] Mahdi Essaid Mahmoud, *Learning Arabic Calligraphy: Naskh, Roqaa, Farsi, Thuluth, Diwany*, Ibn Sina publisher, Cairo- Egypt, 1994.

[36] Mamoru Hosaka, Fumihiko Kimura, "A Theory and Methods for Free Form Shape Construction", *Journal of Information Processing,* Vol 3, No 3, pp. 140-151, 1980.

[37] Mandrake Soft, *Mandrake Linux 8.2*, *http://www.mandrakeSoft.com*, 2002.

[38] Maple Soft, *Maple 7.0*, *http://www.maplesoft.com*, 2001.

[39] Mohammed Hachem El Khattat , *Arabic Calligraphy Rules, A Calligraphic set of Arabic Calligraphy Styles*, Books Univers, Beyrouth- Lebanon, 1986.

[40] Mohamed Jamal Eddine Benatia, Mohamed Elyaakoubi, Azzeddine Lazrek, "Arabic Text Justification", *TUG 2006 Conference Proceedings*, Volume 27, No 2, pp. 137-146, 2006.

[41] Mostafa Banouni, Mohamed Elyaakoubi, Azzeddine Lazrek, "Dynamic Arabic Mathematical Fonts", *Lecture Notes in Computer Science (LNCS)*, Springer Verlag, Volume 3130, pp. 158-168, 2004.

[42] Pierre E. Bézier, *Essai de Définition Numérique des Courbes et des Surfaces Expérimentales*, PhD dissertation, Université Pierre et Marie Curie, Paris, 1977.

[43] Richard J. Kinch, "MetaFog: Converting METAFONT Shapes to Contours", *Preprint: Proceedings of the Annual Meeting*, pp. 1001-1011, June 1996.

[44] Sun, *Java 2D Programmer's Guide,* http://java.sun.com/products/jdk/1.2/docs/guide/2d/spec/j2d-title.fm.HTML,2005.

[45] Ronald N. Goldman, "Using Degenerate Bézier Triangles and Tetrahedra to Subdivide Bézier Curves", *Computer-Aided Design*, Vol 14, No 6, pp. 307-311, November 1982.

[46] Thomas Milo, "Authentic Arabic: A case study. Right-to-left font structure, font design, and typography", *Manuscripta Orientalia*, vol. 8, pp. 49-61, Mar. 2002.

[47] Thomas Milo, “ALI-BABA and the 4.0 Unicode characters”, *TUGboat*, vol. 24, no. 3, pp. 502 - 511, 2003.

[48] Yannis Haralambous, “The Traditional Arabic Type-case Extended to the Unicode Set Of Glyphs”, *Electronic Publishing*, Vol. 8, No. 2-3, pp. 111-123, 1995.

[49] Yannis Haralambous, John Plaice, “Multilingual Typesetting with Ω, a Case Study: Arabic”, *International Symposium on Multilingual Information Processing’97* , Tsukuba (Japon), pp. 137-154, 1997.

[50] William T. Reeves, *Quantitative Representations of Complex Dynamic Shapes for Motion Analysis*, PhD thesis, University of Toronto, 1981.

[51] The Unicode Consortium, *The Unicode Standard, Version 5.1.0*, defined by: *The Unicode Standard, Version 5.0* (Boston, MA, Addison-Wesley, 2007. ISBN 0-321-48091-0), as amended by Unicode 5.1.0 (http://www.unicode.org/versions/Unicode5.1.0/).

[52] اياد حسين عبدالله الحسيني، التكوين الفني للخط العربي وفق أسس التصميم ، دار صاد - بيروت، 1424 هـ - 2003 م.